CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

PHD THESIS

Supervised learning is a popular machine learning paradigm that requires labeled data. Data Labeling refers to the process of annotating data to be used for supervised learning. Implementing a robust and efficient process can be difficult for many reasons. In-house and third-party data labeling have their pros and cons. This thesis addresses the primary problems faced in the industry. It outlines the challenges and mitigation strategies in the industry and provides new and improved mitigation strategies based on Active Learning and Semi-Supervised Learning. Active learning is a machine learning paradigm that selects the instances to be labeled based on a query strategy. Semi-supervised learning utilizes both labeled and unlabeled instances to train a classifier. The thesis provides practitioners with essential guidelines for developing a data labeling process based on empirical simulation studies utilizing the most commonly used semi-supervised algorithms and benchmark datasets. The thesis also outlines ways that practitioners can increase supervised classification accuracy using semi-supervised learning. First, it provides the optimal graph-based semi-supervised learning for automatic labeling and optimal semi-supervised algorithms to increase the performance of supervised learning. The optimality of the algorithms is based on the dataset's characteristics and with respect to number of manually labeled instances. Second, it tells whether practitioners should expect the algorithms to perform equally well on real-world datasets as on benchmarks. Third, it provides practitioners with datasets for evaluating semi-supervised learning algorithms. Fourth, the thesis presents a case study where deep semi-supervised learning was applied to a real-world dataset and outperformed supervised learning in terms of accuracy at the cost of added computation time.

CHALMERS
UNIVERSITY OF TECHNOLOGY

TEODOR FREDRIKSSON • On Semi-Supervised Learning: Evaluation, Challenges and Mitigation Strategies • 2025

On Semi-Supervised Learning: Evaluation, Challenges and Mitigation Strategies

TEODOR FREDRIKSSON

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

On Semi-Supervised Learning: Evaluation, Challenges and Mitigation Strategies

Teodor Fredriksson

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2025

On Semi-Supervised Learning: Evaluation, Challenges and Mitigation Strategies

Teodor Fredriksson

Technical Report No. 5664
ISBN:9978-91-8103-206-2
ISSN 0346-718X
This thesis has been prepared using LaTeX.

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, May 2025

To my parents, Ileana, Maria and Anders.

Abstract

Context: Supervised learning requires labeled data but in many real-world datasets there are few or no labeled instances available. Therefore companies may need to allocate resources to obtain labels. However, labeling is not always trivial and companies need people with domain knowledge to perform labeling. Acquiring suitable personnel for labeling may be expensive and time-consuming if new personnel needs to be hired and trained for labeling.
Objective: The objective of this thesis is to investigate current challenges and mitigation strategies for data labeling. After challenges and weaknesses of current mitigation strategies have been identified, the goal is to identify solutions and improve current mitigation strategies.
Method: This thesis employs multiple methods. The first study is a systematic mapping study that presents the most commonly utilized AI-based algorithms for data labeling related problems. In addition, the most common applications of these algorithms and the datasets utilized for evaluating these algorithms are presented. The second study reports on data collected during a case study in industry and interviews with company practitioners from two companies. Based on the data, three data labeling related challenges where formulated together with a mitigation strategy for each challenge. Statistical methods play an important role in the rest of the studies and are utilized to analyze algorithms. In two studies, the Bayesian Bradley-Terry model is utilized to rank graph-based and deep semi-supervised learning algorithms respectively. In both studies Bayesian generalized linear mixed models are utilized to analyze the probabilities of algorithms reaching a certain performance with and without noise added. In two other studies, Bayesian item response theory is utilized to assess how suitable the datasets are for evaluating graph-based and deep semi-supervised learning algorithms. Lastly, Bayesian linear regression is utilized to analyze the performance of a deep semi-supervised learning algorithm and its relative improvement over supervised learning on a real-world dataset provided by Saab.
Results: *First* the most common AI-based algorithms for data labeling are presented along with the application domains and the datasets utilized to evaluate algorithms. *Second*, challenges and mitigation strategies are presented as well as currently available algorithms. *Third*, the optimal graph-based and deep semi-supervised learning algorithms are presented based on performance on each datatype. In addition manual effort is analyzed to demonstrate how

many labeled instances are required to obtain a certain accuracy. *Fourth*, optimal datasets for evaluating graph-based amd deep semi-supervised learning algorithms are presented. *Finally*, proof demonstrating that deep semi-supervised learning may outperform supervised learning on real-world data collected from industry is presented.

Conclusions: Many AI-based algorithms may help mitigate problems regarding data labeling. Active learning allows practitioners to reduce manual labeling and improve performance of supervised learning by choosing the most informative instances to be labeled. Graph-based algorithms are inductive learning algorithms that will automatically label data by learning from already labeled data. Deep semi-supervised learning algorithms are transductive algorithms that utilize unlabeled data to improve the performance of supervised learning by adding a loss term incorporating the loss function. Empirical evidence indicate that active learning outperforms passive learning where instances to be labeled are chosen at random. Theoretical studies demonstrate that machine learning algorithms utilizing unlabeled data may improve the performance over supervised learning. On the other hand, there are studies indicating that unlabeled data by degrade performance. These observations may be the cause as to why global companies have yet to incorporate semi-supervised learning and why there is a lack of research where semi-supervised learning is applied to real-world data. Deep semi-supervised learning has increased in popularity due to its many advantages such as robustness. The recently developed deep semi-supervised learning algorithms outperform supervised learning. Graph-based semi-supervised learning has the ability to label data with an accuracy above 90%. In addition to performing well on benchmark datasets, both algorithms have proven to perform well when noise is present in the dataset, indicating that the algorithms are expected to perform well on real-world datasets. Noise may even increase the accuracy. On the other hand, the datasets utilized when evaluating algorithms may be inappropriate in the sense that they may be to easy for the algorithms to learn. This will cause a false sense of security as the algorithms may perform worse on real-world datasets that are more difficult to learn. Finally, it is demonstrated that deep semi-supervised learning algorithms based on pseudo-labeling and data augmentation have the ability to outperform supervised learning on real-world data from industry.

Keywords: Data Labeling, Semi-Supervised Learning, Active Learning, Item-

Response Theory, Active Learning, Bradley-Terry Model, Bayesian Data Analysis, Bayesian Linear Regression, Deep Semi-Supervised Learning, Pseudo-Labeling, Consistency Regularization.

List of Publications

This thesis is based on the following publications:

[A] **Teodor Fredriksson**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, "Machine Learning Algorithms for Labeling: Where and How They are Used?". Published in 2022 IEEE 16th International Systems Conference (SYSCON).

[B] **Teodor Fredriksson**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, "Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies". Published in 2020 21st International Conference on Product-Focused Software Process Improvement (PROFES) p. 202-206.

[C] **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson, "An Empirical Evaluation of Graph-Based Semi-Supervised Learning Algorithms". Submitted to Elsevier Computers in Industry Journal.

[D] **Teodor Fredriksson**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, "Assessing the Suitability of Semi-Supervised Learning Datasets using Item Response Theory". Published in 2021 47th Euromicro Conference on Software Engineering and Advanced Applications, (SEAA), p. 326-333.

[E] **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson, "An Empirical Evaluation of Deep Semi-Supervised Learning". Published in International Journal of Data Science and Analytics (2025): 1-22..

[F] **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson, "Assessing the Sustainability of Deep Semi-Supervised Learning Datasets using Item Response Theory". In Submission at the Journal of Supercomputing.

[G] **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson, "Classification of Complex-Valued Radar Data using Semi-Supervised Learning: a Case Study". Published in 2023 49th Euromicro Conference on Software Engineering and Advanced Applications, (SEAA), p. 102-107.

Other publications by the author, not included in this thesis, are:

[H] **Teodor Fredriksson**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, "An Empirical Evaluation of Algorithms for Data Labeling". Published in IEEE 45th Annual Computers, Software, and Applications Conference, (COMPSAC) p. 201-209.

[I] **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson, "Machine Learning Models for Automatic Labeling: A systematic literature review". Published in 2020 15th International Conference on Software Technologies (ICSOFT), p. 552-566.

Individual Contributions

Teodor Fredriksson is the main author of the studies presented in this thesis. Teodor is responsible for the planning and execution, analysis and reporting with David Issa Mattos assisted with Paper A,B,D and H. An overview of the specific contribution is specified below according to to *Contributor Role Taxonomy* (Credit).

Conceptualisation: Teodor is responsible for formulating the research goals in **Paper B,C,E-G**. David and Teodor set up the research goals together in **Papers A,D** and **H**

Methodology: Teodor formulated and developed the research methodology in **Papers B,C,E-G**. David and Teodor developed the research methodology together in **Papers A,D** and **H**.

Software Programming: Teodor was the sole contributor of the AI models and the statistical models in all papers except for **Papers C** and **D** where David performed the statistical analysis.

Formal analysis: Teodor was responsible for the formal analysis in all papers. David assisted Teodor in **Papers B** and **C**.

Investigation: Data collection from the screening of papers, interviews and experiments was performed by Teodor except for **Paprs C-D** where David colllected data from the statistical analysis.

Data Curating David managed the data in **Papers C** and **H**. Teodor managed the data for the other studies.

Writing: David wrote the sections where the statistical models were described in **Papers C** and **D**. Teodor wrote the remainder of **Papers C** and **D** and wrote all other papers alone.

Project administration: All contact with the industry collaborators was administered through Teodor.

Acknowledgments

My biggest thanks go to my supervisors, Professor Jan Bosch and Professor Helena Holmström Olsson. Despite many setbacks, disagreements, views and values, you never gave up on me, and I'm eternally grateful. I have learned many valuable things from you.

I want to thank my industry partners, Ericsson and Saab, for letting me collect valuable data for my research.

I want to thank David Issa Mattos for guiding me through the first part of my PhD. I learned many things from you and am grateful for our continuing friendship.

I want to thank all the Interaction Design and Software Engineering Division for being such nice people to be around for such a long time.

I want to thank Richard Torkar, who provided me with much help and advice. It's always nice to see you and chat.

I'd like to give special thanks to Richard Berntsson Svensson, who has helped me a lot. It's always a pleasure talking to them.

I want to thank my examiner, Miroslav Staron, for all the positive encouragement. It was nice to meet someone with the same humor.

I want to give special thanks to my friend, Hanna Kvist, for always being invested in my well-being and ensuring I don't overload.

Thank you to all my fellow PhD students and PostDoc for an enjoyable time at the office: Sushant Pandey, Krishna Ronanki, Amna Pir, Habib Khan, Mazen Mohammad, Ranim Kojha, Beatriz Cabrero-Daniel, Malsha Malwatta, Vladislav Indykov, Sabina Akbarova, Aiswarya Raj Munappy, Hina Saeda, Muhammed Cagri Kaya, Tayssir Bouraffa, Weixing Zhang, Ziming Wang, Afonso Fontes, Hamdy Michael Ayas, Razan Ghzouli and Babu Md Abu Ahammed and Meenu Mary John, you are all fantastic. You could never ask for better colleagues. From the bottom of my heart, I wish you all the best and hope that we will be friends long after this.

I want to give special thanks to my bestie, Wardah Mahmood, who has become one of my closest friends. I can always share anything with you, and you always make me see things differently! Thank you for always listening and calming my nerves!

My greatest thanks go to my friend and mentor, Sakib Sistek. Thank you for everything! Thanks for making me see the world differently and for teaching me to love myself. Thanks for teaching me that I can do anything as long as

I put my mind to it. I would have given up on getting a PhD a long time ago if it wasn't for you.

Thank you for to my parents Maria and Anders Fredriksson for always making sure I got all my schoolwork done so that I could eventually end up where I am today. I owe everything to you!

Finally, I want to thank the woman who bore me, Illeana Mogorgea. Thank you for having the strength to give me up. I know it wasn't an easy choice to give up a child, but if it wasn't for your strength, I would not be where I am today! I hope to meet you one day to thank you in person.

Acronyms

ML:	Machine Learning
DL:	Deep Learning
SL:	Supervised Learning
SSL:	Semi-Supervised Learning
DSSL:	Deep Semi-Supervised Learning
GBSSL:	Graph-based Semi-Supervised Learning
AL:	Active Learning
BDA:	Bayesian Data Analysis
IRT:	Item Response Theory
USB:	Universal Semi-Supervised Benchmark

Contents

CHAPTER 1

Introduction

Artificial intelligence (AI) is an attempt to invent machines that possess human intelligence [1]. The theory of AI is based on mathematical theories that were developed in the early 20th century such as *probability theory* [2], *decision theory* [3], *statistical theory* [4], *information theory* [5] and *control theory* [6]. AI is therefore, an old concept, but it has recently gained attention due to access to modern computational resources.

Reinforcement learning [7] and *machine learning* [8] are two types of AI that learn differently. Reinforcement learning learns through reward [7], and machine learning learns with data [8]. Machine learning is divided into *unsupervised* and *supervised* learning. Unsupervised learning finds patterns in unlabeled data, while supervised learning requires labeled data. Supervised learning is divided into supervised regression and supervised classification problems. An example of supervised regression is house price prediction. Given features of a house, train a model that predicts the house price of any house given input features. On the other hand, training a system that predicts if an image contains a cat or a dog is a supervised classification problem.

Supervised classification requires labeled data, and the performance of a supervised classifier increases the more data is available. Therefore, it is rele-

vant to obtain as many labeled data points as possible. In industry, datasets may lack labels, and empirical evidence suggests that 80% of time spent in ML projects is allocated to data labeling [9]. Data Labeling refers to the task of annotating data for *supervised learning* and is important in applications such as autonomous driving [10], and medical applications [11]. Autonomous vehicles must detect objects with high accuracy to guarantee the safety of passengers and pedestrians. Medical data annotation is required to enable DL models to fro example detect cancer cells in patients and save lives [12], [13].

While labeling images that contain cats and dogs may seem trivial, other labeling tasks are less obvious, such as object detection and scene perception in self-driving vehicles [14]. Localization is a requirement for autonomous vehicles to enter public roads. This means that the vehicle knows where it is always located with centimeter precision. To achieve localization, unmanned aerial vehicles and satellites are utilized to map sideways, and this data needs to be labeled for supervised learning. Medical data based on images and text may be complex and have a wide variety. There are many abnormalities in the data, and it is difficult to train labelers to label medical data correctly. A small mistake may have serious consequences for patient health [15].

There are many options for labeling. The first option is in-house labeling where companies allocate the labeling task to in-house personnel [16]. This option allows control over labeling so that high-quality labels are guaranteed. The next question becomes, "Who should perform the labeling?". The natural choice is to let data scientists and software engineers perform labeling, as they are data experts. However, their expertise is needed to perform other more specialized tasks, such as programming models and performing advanced data analysis. If data scientists and software engineers are unavailable for labeling, then training other personnel to perform labeling is necessary. Then the ultimate question became, "How much will it cost?". Training and recruitment of labelers may be costly in terms of time and resources.

Crowdsourcing is another way to obtain labels [17], by having people from all over the world label data [18], [19] through platforms such as Amazon Mechanical Turk [20]. Crowdsourcing doesn't require companies to hire personnel, but on the downside, it is difficult to guarantee the quality of the labels due to a lack of domain-experts. Many companies have sensitive data and require detailed background checks on their personnel prior to allowing

them access to sensitive data. Performing background checks on foreign nationals may be difficult and time-consuming. Therefore, crowdsourcing is not an option for many companies, and in-house labeling is utilized [9].

The optimal method is to label data automatically, but this may be difficult. *Synthetic labeling* [21] means generating new data based on the current dataset, but then a model that generates data of high quality may be difficult and require expensive computing power.

One step towards automatic labeling is a *Human-in-the-loop* [22] approach such as *active learning* [23], where human and machine intelligence is combined to achieve high-performing models. Another HITL approach is *programmatic labeling* [21] where a script is utilized to annotate data but then a human is required to check the quality of the label. Programmatic labeling is possible in text classification tasks, where text is classified based on its contents.

In a traditional setting, a labeler would randomly select instances to be labeled, but such a strategy is inefficient because non-informative instances may be labeled [23]. Instead, practitioners are encouraged to utilize *active learning* (AL) [23], where instances to be labeled are chosen according to a *query-strategy* that selects instances based on how informative they are. After the informative instances are labeled, the ML algorithm is trained, and if the performance is sufficient, no additional labeling is required. If the performance is insufficient, additional instances are chosen for labeling, and old labels may be re-labeled. This procedure is repeated until sufficient performance has been reached [23]. In real-world scenarios, the labeling cost may be different for different labels [24]. Reducing the number of labels may not lead to reduced labeling costs [24]. Active learning approaches that do not account for different labeling costs may not outperform passive learning [25].

Active learning that accounts for different labeling costs is called *cost-sensitive active learning* [26], [27]. A direct way to reduce the cost is to assign pseudo-labels with the help of the trained model [28]. This does not account for the costs themselves, and pseudo-labels may be low quality due to a lack of data.

Another approach is the value of information [29] that considers both labeling and miss classification costs. Yet another active learning approach is when allocation costs are known and depend on the annotation time [30]. The empirical evaluation [30] demonstrates that the labeling cost varies depending on the labeler because of errors such as *jitter* [30]. Jitter refers to the error

caused by fatigue in the labeler [30].

Semi-supervised learning (SSL) algorithms are ML algorithms that utilize both labeled and unlabeled data [31]. There are two types of SSL: *Inductive SSL* and *transductive SSL*. Inductive SSL performs supervised classification but incorporates unlabeled data to outperform the supervised counterpart [31]. Transductive SSL takes an input of labeled and unlabeled data and then infers labels on the unlabeled data [31]. In other words, inductive algorithms are trained on labeled and unlabeled data to generalize well on incoming new data. In contrast, transductive algorithms only label the unlabeled data provided in the input [31].

A wide range of research has been produced in AL and SSL. Many earlier studies are theoretical and prove that SSL has advantages over SL [32], [33], which has also been demonstrated empirically. Many studies indicate that SSL may worsen performance and that AL outperforms PL. Despite the research available, few studies provide guidelines for practitioners wishing to utilize AL and SSL. ML algorithms rely on several assumptions, such as *independent and identically distributed*, *conditional independence between* and a *linear relationship between feature and target variable* [8]. In addition, semi-supervised learning relies on the *cluster, low-density* and *manifold* assumptions [31]. ML algorithms are developed with these assumptions in mind, and while many benchmark datasets satisfy the assumptions, real-world datasets may be more complicated for benchmark algorithms to learn. Benchmark studies may still be helpful as they provide insight into available algorithms that suit their needs. However, practitioners may need to tweak the algorithm to learn their dataset better [34]. In addition, if the practitioners aim to develop a generic algorithm, they must construct a benchmark test of appropriate datasets that fairly evaluate the algorithm [34].

This thesis aims to assist practitioners from software engineering and AI domain in reducing the costs and risks associated with preparing a labeling infrastructure. This thesis analyzes optimal practices for utilizing unlabeled data to achieve optimal performance in machine learning models for classification.

There exist many challenges with data labeling for supervised classification. Practitioners need to create a systematic approach for labeling, perform exploratory data analysis to identify the correlation between labels and features and choose a model that may be reused on new data. Another problem is

selecting annotators and minimizing labeling errors. Finally, there are problems regarding the label distribution. The best practices for mitigating these challenges are active learning and semi-supervised learning. Active learning can be used to help select the optimal instances to label and include in the training data, enabling a supervised learning algorithm to achieve optimal accuracy. Transductive semi-supervised learning labels data automatically, and inductive semi-supervised learning eliminates labeling by using unlabeled data to improve the classification performance of supervised learning. This thesis discusses AL and SSL algorithms and their potential applications in mitigating the challenges associated with data labeling in industry. Challenges and mitigation strategies are identified from the industry's perspective. Through computer simulations, we evaluate which SSL algorithms are optimal in different scenarios and identify suitable datasets for algorithm evaluation. Selecting suitable datasets for evaluating algorithms is a prerequisite for obtaining reliable and generalizable results. Lastly, we present a case study from industry where we developed an SSL algorithm for a Doppler radar dataset provided by our industry collaborator, Saab.

The research contained in this thesis was conducted in the context of the Wallenberg Autonomous Systems Program (WASP) and the Software Center (SC). We conducted research with three companies, two of whom are affiliated with both WASP and SC. The research was conducted in collaboration with industry to address problems relevant to industry practitioners, and the results may also be applied in academia. However, due to the lack of relevant case studies, we had to include computer simulations. The remainder of this chapter is organized as follows:

Chapter 2 presents a background section that outlines the key concepts and algorithms employed in this thesis. Chapter 3 outlines the thesis's objective, presents the research questions, and describes the research methods and data analysis tools used to answer these questions. Chapter 4 provides a summary of each publication. Chapters 5-11 each contain the included publications. Chapter 12 discusses how the included papers address the research questions and outline the main contributions as well as future research directions.

CHAPTER 2

Background

2.1 Algorithms in Software Engineering

Algorithms and data structures are the foundations of computer applications [35]. The systematic development of efficient algorithms is crucial for developing resource-conserving technologies [35]. Therefore, developing high-performing and fast algorithms plays an integral part in software engineering, especially in modern times when computationally expensive AI algorithms have become popular tools in the industry to enable intelligent systems of systems [36].

For industry practitioners to develop high-performing supervised learning algorithms, datasets need to be fully labeled. However, in industrial scenarios, datasets may lack labeled instances or suffer from other problems, such as imbalanced data distribution, which makes it difficult for the algorithms to generalize to other data [16].

This thesis aims to present the challenges associated with labeling in the industry and provide mitigation strategies by presenting AI-based algorithms that reduce manual data labeling. To study AI-based algorithms for solving label-related issues, practitioners must understand the mathematical founda-

tions of AI. The chapter presents the fundamental concepts necessary to understand the algorithms discussed throughout this thesis. Section 2.2 presents concepts from probability, statistics and decision theory. Section 2.3 presents different types of supervised learning algorithms. Section 2.4 discusses how industry practitioners achieve labels through manual effort. Section 2.5 presents active learning strategies to choose the most informative instance for labeling, thus reducing manual labeling costs. Section 2.6 demonstrates how to derive semi-supervised learning theoretically and how incorporating unlabeled data differs from supervised learning. Section 2.7 presents different techniques utilized in inductive deep semi-supervised learning. These semi-supervised learning algorithms are classification algorithms that incorporate unlabeled data to increase the performance achieved by supervised classifiers. Finally, section 2.8 presents transductive graph-based semi-supervised learning algorithms that serve as automatic labeling algorithms.

2.2 Artificial Intelligence

The theory of AI is built on modern Mathematical theories such as *measure theory* [37], *probability theory* [2], *statistics* [4],*decision theory* [6], and *control theory* [3]. Therefore, AI inherits all the strengths and weaknesses of all these disciplines. From measure theory comes the notion of measure and Lebesgue integral which leads to concepts in probability theory such as probability measure and probability distributions. Thanks to probability theory, a notion of randomness and uncertainty is defined. These are necessary because AI models utilize uncertainty. The Bayesian paradigm becomes a valuable tool for modeling uncertainty, and algorithms utilize the Bayes theorem. Inspiration is taken from decision theory to allow machines to make intelligent decisions.

Measure Theory, Probability Theory and the Theory of Statistical Inference

Machine Learning is a subfield of AI that learns from data, and this thesis studies data labeling for machine learning. Probability is required to study uncertainty in. From measure theory, concepts such as measure and Lebesgue integral are defined. A probability measure is a type of measure μ, such that $0 \leq \mu \leq 1$ and is σ-finite. Thanks to the integral, it is possible to define

concepts such as probability distribution, density, and expectation.

Machine learning was traditionally called statistical learning because it draws inspiration from statistics. If the data is collected from a random phenomenon, statistical theory provides inference on the probability distribution of the phenomenon. The inference is based on probabilistic modeling of the underlying phenomenon, and statistics is an interpretation of the phenomenon.

The data is assumed to be distributed according to a parametric probability distribution with density $x \sim f(x|\theta)$ where the parameter θ is unknown. If θ has a probability distribution called prior distribution, then the model is called a Bayesian model [4].

After the data have been observed, $\ell(\theta|x) = f(x|\theta)$ depends on θ, which is called the likelihood function. The *likelihood principle* states that all information extracted from observed data is entirely contained in the likelihood function.

Decision Theory

The purpose of AI is to achieve human intelligence. Therefore, machine learning is inspired by decision theory. Decision theory is the study of making decisions under uncertainty. If data is involved, then the decision is based on the information collected from the data. Therefore, decision making depends on the uncertainty of the parameter θ. Sometimes, decisions are referred to as actions, especially in reinforcement learning [7]. In addition to the information collected from data, it is also necessary to know the consequences of making certain decisions, which are measured with the loss function. This means a loss function must be defined for every value of θ and every action.

Definition: A *loss function* is any function $L\colon \Theta \times \mathcal{D} \to \mathbb{R}$ where Θ is the parameter space and $\mathcal{D}$ is the *design space*.

The loss function is meant to evaluate the *penalty/error* $L(\theta, d)$ associated with the decision d when the parameter takes value θ.

When making the decision, it is unlikely that the loss function is known, so it needs to be chosen by trial and error. To make decisions based on data, the concept of decision rule is utilized:

Definition: A *decision rule* δ, is a function $\delta\colon \mathcal{X} \to \mathcal{A}$ from the sample

space $\mathcal{X}$ to the space of actions $\mathcal{A}$:

The risk associated with a decision rule is the expected value of the loss function with respect to the distribution of the data:

$$R(\theta, \delta) = \mathbb{E}^X \Big[L(\theta, \delta(X)) \Big], \tag{2.1}$$

and if no data is involved then $R(\delta, \theta) = L(\delta, \theta)$. It is necessary to compare different decision rules according to their risk. A decision rule δ_1 is better than another decision rule δ_2 if δ_1 has lower risk than δ_2:

$$R(\theta, \delta_1) \leq R(\theta, \delta_2). \tag{2.2}$$

A decision rule δ_1 is called *R-better* than δ_1. If a decision rule has no R-better decision rule, then the decision rule is *admissible*.

$$r(\pi, \delta_1) = \mathbb{E}^X \Big[R(\theta, \delta) \Big] \tag{2.3}$$

Another important concept is the expected risk of a decision, defined as the expected value of the risk:

$$r(\pi, \delta_1) = \mathbb{E}^\pi [R(\theta, \delta)], \tag{2.4}$$

where π is the prior distribution. A decision rule may be stochastic, when that's the case, it is considered a probability distribution on the set of actions. If x is observed, then the interpretation is that the randomized decision rule is the probability that action a is taken given the observed data x. Non-randomized decision rules are considered special cases of randomized decision rules. All properties of non-randomized decision rules also hold for randomized decision rules.

There are many principles for making decisions, such as *Bayes risk principle*, the *minimax principle*, and the *invariance principle*:

The Bayes Risk Principle: A decision rule δ_1 is preferred to a rule δ_2

if:

$$r(\pi, \delta_1) < r(\pi, \delta_2), \tag{2.5}$$

A decision rule which minimizes the expected risk is optimal as is called a *Bayes rule* and will be denoted δ^π. The quantity $r(\pi) = r(\pi, \delta^\pi)$ is called the Bayes risk for π.

The Minimax Principle: A randomized decision rule δ_1 is preferred to a rule δ_2 if:

$$\sup_\theta R(\theta, \delta_1) < \sup_\theta R(\theta, \delta_2). \tag{2.6}$$

Furthermore δ^M is a *minimax-decision* rule if:

$$\sup_\theta R(\theta, \delta^M) = \inf_\delta \sup_\theta R(\theta, \delta). \tag{2.7}$$

The Invariance principle: If two problems have identical formal structures, then the same decision should be used in each problem:

The frequentist perspective

In the frequentist perspective no prior distribution is imposed on θ. Consider a decision rule δ and a loss function L. The goal is to pick a $\overline{R}$ s.t δ yields an average performance of $\overline{R}$. Consider the problem of estimating θ. The loss function is defined as:

$$L(\theta, \delta(x)) = 1_{\delta(x)}(\theta) = \begin{cases} 1 & \text{if } \theta = \delta(x) \\ 0 & \text{otherwise} \end{cases} . \tag{2.8}$$

The risk is calculated to be:

$$R(\theta, \delta) = \mathbb{E}^X \Big[L(\theta, \delta(x)) \Big] = P(\delta(x) = \theta). \tag{2.9}$$

If δ is applied to the multiple normal distributed i.i.d samples $X^{(i)}$, then according to the law of large numbers:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} L(\theta_i, \delta(X^{(i)})) = \overline{R}. \tag{2.10}$$

The frequentist perspective is utilized to define structural risk minimization, see section 2.3.

2.3 Supervised Learning

Supervised Learning is divided into regression and classification problems. The goal of supervised learning is to find a function $h\colon \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ is the *feature space* and $\mathcal{Y}$ is the *target space*.

An example of a regression problem is house price prediction. Based on training data:

$$S = \{(x_i, y_i) \ : \ i = 1, ..., n\}, \tag{2.11}$$

the objective is to obtain a function $h\colon \mathcal{X} \to \mathcal{Y}$ that predict house prices for unseen data.

In the example above, the output is a real-value number $y \geq 0$. In supervised classification, the outputs are categorical and take values $\mathcal{C} = \{1, ..., m\}$. This means that a classifier $h\colon \mathcal{X} \to \mathcal{Y}$ is sought, where $\mathcal{Y}$ contains the vectors of probabilities $P_\theta(c|x) = P(c|x;\theta)$. If there are M different class labels then $P_\theta(c|x)$ is a M-dimensional vector whose elements sum to 1:

$$\sum_{i=1}^{M} P(i|x) = 1. \tag{2.12}$$

The label is then found by:

$$\widehat{c} = \arg\max_{c} P_\theta(c|x). \tag{2.13}$$

An example of binary classification is the task of determining whether there is a cat or dog in an image.

Different types of algorithms

There are three types of algorithms. *Generative models* calculates the joint probability by selecting priors $p(c)$ and calculates $p(x, c) = p(x|c)p(c)$. Next, the posterior distribution is calculated using Bayes theorem:

$$p(c|x) = \frac{p(x, c)}{p(x)}. \tag{2.14}$$

where $p(x) = \sum_i p(x|i)p(i)$. Discriminative models obtain the posterior directly without knowing the joint distribution, and *deterministic* models obtains a labeling function $h(x)$ without utilizing probabilities.

Structural Risk Minimization

Supervised learning algorithms are derived from *structural risk minimization* (SRM) [38]. SRM provides a theoretical framework for deriving supervised learning algorithms. It is assumed that the data features $x_1, ..., x_n$ are outputs from a random variable X, where $X \sim P(x)$. Similarly, the output labels $c_1, ..., c_l$ are outputs of the random variable C and the distribution of the labels is the posterior distribution $C \sim P(c|x)$. The labeled training data $\mathcal{L} = \{(x_i, y_i),\ i = 1, 2, ..., l\}$ are outputs of the random variable (X, C) where $(X, C) \sim P(x, c)$.

SRM aims to find a hypothesis h from the hypothesis space $\mathcal{H}$, that produces the labels $c = h(x)$. The approach is based on decision theory, particularly the frequentist perspective. If $\theta = h(x)$ represents the true labels, then SRM seeks to obtain a decision rule as a classifier $\delta(x) = \widehat{h}(x)$.

The estimate $\widehat{h}$ of h is found by minimizing the risk:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} R(h, \widehat{h}), \tag{2.15}$$

where the risk is defined as

$$R(h, \widehat{h}) = \int_{\mathcal{X}} L(h, \widehat{h})\, dP_{X,C}(x, c). \tag{2.16}$$

If $L(h, \widehat{h})$ is defined as:

$$L(h, \widehat{h}) = 1_{\widehat{h}(x)}(x) = \begin{cases} 1 & \text{if } c = \widehat{h}(x) \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Then the risk may be simplified to:

$$R(c, \widehat{h}) = P(\widehat{h}(x) = c). \tag{2.18}$$

To minimize R empirically utilizing training data, R is replaced with:

$$R_{\text{emp}} = \frac{1}{m} \sum_{i=1}^{m} L(c_i, \widehat{h}(x_i)). \tag{2.19}$$

2.4 Data Labeling

SL algorithms are sometimes challenging to use as labeled data might be hard to collect automatically, and a human labeler (*or supervisor*) might be needed to collect labels [39]. ML and DL models are known to perform better on large datasets [40], so obtaining large labeled datasets is of grave importance. Companies might have massive datasets that contain many features. However, it is often the case that these datasets are incomplete because they are missing labels. Because of this, companies must allocate in-house personnel to perform the data labeling manually. Data scientists and software engineers are natural choices because they usually have good knowledge about data. On the downside, they often perform other specialized tasks, so having them label data would be a waste of resources.

One solution is to use *crowdsourcing*. Crowdsourcing in machine learning entails the hiring of annotators at a large scale. Many crowdsourcing platforms such as *Scale, Clickworker, Lionbridge AI, Isahit, MarsCrowd, Amazon Mechanical Turk* and *Cloud Factory* offer data labeling for different tasks such as *sentiment analysis, entity extraction, text classification*, *text translation*, *image classification, object detection* and *image segmentation* among others. Crowdsourcing offers a 24/7 workforce, and it is easy and affordable. Some cons of crowdsourcing are that quality control is not guaranteed, and it is challenging to maintain consistent results over time. Companies that need to label sensitive and confidential data can not share their data with

third-party organizations, so many companies still utilize in-house labeling. In-house labeling allows companies to set up consistent annotation processes that yield long-term reliability and success [16]. However, building the annotation process is expensive and time-consuming if the company is small. Both crowdsourcing and in-house labeling have their pros and cons. To the best of the author's knowledge and experience with the industry, most companies are not interested in utilizing any of these approaches. Companies are more interested in investigating automated approaches to reduce manual effort or to automate the process entirely.

2.5 Active Learning

One way to reduce manual effort is to use *active learning* (AL). Machine Learning systems usually implement *passive learning* (PL) in which the learner randomly chooses instances to be labeled [23]. Selecting instances randomly is problematic as these instances are not guaranteed to be *informative* [23]. Training a model on non-informative instances may lead to low performance. An active learning system selects the most *informative* instances to be labeled according to a *query strategy*. The learner would then train a classifier on the training data and if the classifier does not reach the desired accuracy the learner will query more instances to be labeled or re-labeled. This procedure would iterate until the desired accuracy is reached. In active learning it is assumed that large amounts of unlabeled instances are available and easy to collect.

Stream-based selective sampling

This scenario draws one unlabeled instance [41], [42] at a time from the actual distribution, and then decides whether to query it or not [43]. Although this query type is cheap regarding computational cost, querying only one instance at a time does not consider the entire distribution in the decision process. Another disadvantage is that the learner needs to set a decision boundary that decides if an instance should be queried. The learner might discard valuable instances if the decision boundary is set poorly.

Query Synthesis

Query synthesis generates the most informative instances from a synthetic distribution [44] and queries these to be labeled by the oracle [45]. Like query-based selective sampling, query synthesis may not query labels from the entire distribution. In [46], a DL model was trained to recognize handwritten characters and query synthesis would generate artificial hybrid characters that do not make sense.

Pool-based sampling.

Pool-based sampling is the most popular of the scenarios and has been applied in many different applications [47]–[49]. It is assumed that a large pool of unlabeled data and a small pool of labeled samples are available. Unlike stream-based selective sampling and query synthesis, pool-based selective sampling queries the entire distribution and is optimal for choosing the most informative sample [23]. On the downside, pool-based sampling is computationally expensive, so query synthesis and stream-based selective sampling are utilized on devices with limited memory and processing power [23].

Measuring informativeness:

Once the query type has been selected, the learner needs to decide on what measure to use to measure informativeness. The most popular techniques are *uncertainty sampling, query-by-committee* and *error and variance reduction* [50].

Uncertainty sampling:

The motivation for sampling the most uncertain instances is that the more uncertain instances are, the more they can reveal the ground truth. The simplest is the *least confident* instance, which is found by calculating:

$$x^*_{LC} = \arg\max_x \Big(1 - P_\theta(\widehat{y}|x)\Big), \tag{2.20}$$

where $\widehat{y} = \arg\max_y P_\theta(y|x)$ is the instance with the highest posterior probability. Then x^*_{LC} will be the instance with the lowest probability, hence the most uncertain instance. The problem with the LC estimate is that it only

considers the label with the highest probability. To mitigate that, practitioners may use *margin sampling* [23]. Margin sampling considers the two labels with the highest probabilities. The margin estimate is given by the instance that minimizes the difference.

$$x_M^* = \arg\min_x \Big(P_\theta(\widehat{y}_1|x) - P_\theta(\widehat{y}_2|x) \Big). \tag{2.21}$$

If the margin is large, then the true label is more certain. Another approach is the *entropy* query strategy defined as:

$$x_E^* = \arg\max_x \Big(-\sum_i P(\widehat{y}_i|x) \log P(\widehat{y}_i|x) \Big). \tag{2.22}$$

Entropy measures the average information of x [51].

Query-by-Committee

A hypothesis h is said to be *consistent* if it fits the training instances perfectly but does not have a perfect generalization error. The subset $\mathcal{V} \subseteq \mathcal{H}$ is defined as the set:

$$\mathcal{V} = \{h : \mathcal{X} \to \mathcal{Y} : \text{consistent with the training data}\},$$

and is called *version space* [52]. Let $|\mathcal{V}|$ denote the size of the versions space. The hope is that the generalization error decreases when the model is retrained after adding new instances to the dataset. Therefore, instances that reduce the version space size quickly should be queried.

First, consider the *Query by Disagreement* (QBD) query strategy [42]. This strategy works with the stream-based sampling query type. If $h_1, h_2 \in \mathcal{V}$ and a new instance x comes in, then x is queried if $h_1(x) \neq h_2(x)$, otherwise x is discarded. There are, however, shortcomings to this approach. If $\|\mathcal{V}\|$ is infinite, then the version space cannot be stored in memory. Query by disagreement assumes the following [51]:

1. Disagreement is measured among all hypothesis $h \in \mathcal{V}$, or two extremes h_S and h_G.

2. Disagreement is a binary measure. No controversial instance matters more than another.

Query-by-committee relaxes these assumptions and makes pool-based active learning possible. QBC refers to all disagreement approaches that use a "committee" or ensembles C of hypotheses. All QBC approaches require a method for obtaining a hypothesis in the committee and a heuristic for measuring disagreement among them. Examples of measures for measuring disagreement include *vote entropy*:

$$x^*_{VE} = \arg\max_x \left(-\sum_y \frac{\text{vote}_\mathcal{C}(y,x)}{|\mathcal{C}|} \log \frac{\text{vote}(y,x)}{|\mathcal{C}|} \right),$$

where y represents all labels and:

$$\text{vote}_C(y,x) = \sum_{\theta \in C} 1_{h_\theta(x)=y},$$

is the number of votes. A second disagreement measure is *soft vote entropy* defined as:

$$x^*_{SVE} = \arg\max_x \left(-\sum_y P_\mathcal{C}(y|x) \log P_\mathcal{C}(y|x) \right),$$

where

$$P_\mathcal{C}(y|x) = \frac{1}{|\mathcal{C}|} \sum_{\theta \in \mathcal{C}} P_\theta(y|x),$$

is the average probability that the committee agrees that y is the correct label. Last, the *Kullback-Leibler divergence* query strategy queries the sample x that maximizes the average Kullback-Leibler divergence:

$$x^*_{KL} = \arg\max \frac{1}{|\mathcal{C}|} \sum_{\theta \in \mathcal{C}} \text{KL}\left(P_\theta(y|x) | P_\mathcal{C}(y|x)\right).$$

Other query strategies for QBC include *Jensen-Shannon divergence*[53] and *F-compliment* [54].

2.6 Semi-supervised learning

Vicinal Risk Minimization

SRM derives many supervised learning algorithms but does not generalize to generative models and semi-supervised learning. Generative algorithms use Bayes theorem to model the joint distribution $P(x, c)$ to calculate $P(c|x)$. Other supervised algorithms derived by SRM calculate $P(c|x)$ directly without calculating $P(x, c)$ since it may be time-consuming. *Vicinial risk minimization* (VRM) [55] bridges the gap between SRM, generative models and semi-supervised learning.

The empirical distribution is defined as:

$$\widehat{P}(x, c) = \frac{1}{n} \sum_{i=1}^{n} 1_{x_i}(x) 1_{c_i}(c). \tag{2.23}$$

and is utilized in SRM. To improve the estimate (2.23), it is possible to replace $1_{x_i}(x)$ with an estimate of the distribution around x_i, denoted $\mu_{x_i}(x)$:

$$\widehat{P}(x, c) = \frac{1}{n} \sum_{i=1}^{n} \mu_{x_i}(x) 1_{c_i}(c). \tag{2.24}$$

After plugging in eq (2.23) into eq (2.16) and simplifying, it becomes:

$$R_{vic}(h, \widehat{h}) = \frac{1}{n} \int_{\mathcal{X}} \sum_{i=1}^{n} L(c, \widehat{c}_i) d\mu_{x_i}. \tag{2.25}$$

Therefore, the optimal h is chosen s.t:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} R_{vic}(h). \tag{2.26}$$

A common choice for μ_{x_i} are the *spherical Gaussian kernel* functions controlled by the σ parameter. SRM is a special case of VRM when $\sigma = 0$. The generalization performance of VRM depends on the quality of $\widehat{P}(x, c)$ and *size/capacity* of $\mathcal{H}$. VRM may still perform well if $\widehat{P}(x, c)$ is chosen poorly and vice versa.

Semi-supervised learning utilizes both labeled and unlabeled data:

$$\mathcal{L} = \{(x_i, y_i),\, i = 1, .., |\mathcal{L}|\}, \tag{2.27}$$
$$\mathcal{U} = \{u_i = x_{l+i},\, i = 1, ..., |\mathcal{U}|\}, \tag{2.28}$$

where the set of unknown labels for the unlabeled data $\mathcal{U}$ is denoted:

$$\mathcal{C}^u = \{c_i^u = c_{|\mathcal{L}|+i},\, i = 1, .., |\mathcal{U}|\}. \tag{2.29}$$

The vicinal risk is split into two components where $L = L_s + L_u$:

$$R_{vic}(h, \widehat{h}) = \frac{1}{|\mathcal{L}|} \int_{\mathcal{X}} \sum_{i=1}^{|\mathcal{L}|} L_s(c, \widehat{c}_i) d\mu_{x_i} + \frac{1}{|\mathcal{U}|} \int_{\mathcal{X}} \sum_{i=1}^{|\mathcal{U}|} L_u(\widehat{h}(x_i), \widehat{h}(u_i)) d\mu_{u_i}. \tag{2.30}$$

It is assumed that there are many unlabeled instances and few labeled instances available. Empirical results indicate that unlabeled data improves performance for spherical gaussian kernels with $\sigma_L \to \infty$ and optimal σ_U [55]. However, unlabeled data may not always improve performance.

Common types of losses and different techniques utilized in SSL are presented below.

Generative semi-supervised learning

In the generative paradigm of semi-supervised learning it is assumed that x and c are parametrized by θ and π respectively. Furthermore, it is assumed that $P(\theta, \pi) = P(\theta)P(\pi)$ and it is possible to write:

$$P(x, c|\theta, \pi) = P(x|c, \theta)P(c|\pi). \tag{2.31}$$

If both labeled and unlabeled data are available, a natural criterion is to consider the joint log-likelihood:

$$\log P(x, c) = \sum_{i=1}^{|\mathcal{L}|} \log P(x_i|c_i; \theta)P(c_i|\pi) + \sum_{i=1}^{|\mathcal{U}|} \log \sum_{m \in \mathcal{C}} P(x_i|m; \theta)P(m|\pi). \tag{2.32}$$

Here, C^u is unknown, hence c is treated as a latent variable which may be solved utilizing the EM-algorithm or gradient descent. The generative approach is not an obvious solution in semi-supervised learning. Estimating the marginal distribution of x is wasteful in classification as these algorithms are interested in the posterior distribution $P(c|x)$. Maximizing $P(x, c)$ may lead to large errors utilizing few labeled samples.

A good likelihood maximization may depend on the quality of $P(x)$ rather than the quality of $P(c|x)$. In addition, the latent variable model takes the sum:

$$P(x|\theta, \pi) = \sum_{c \in \mathcal{C}} P(c|\pi)P(x|c; \theta), \tag{2.33}$$

which may lead to multimodal posteriors.

The diagnostic paradigm

For the diagnostic paradigm, let X_l be the labeled features, X_u be the unlabeled features and C_l be the labels of the labeled features. In the diagnostic paradigm it is assumed that $P(\theta, \mu) = P(\theta)P(\mu)$ and a Bayesian network may be constructed so that:

$$P(C_l, X_l, X_u|\theta, \mu) = P(C_l|X_l, \theta)P(X_l, X_u|\mu). \tag{2.34}$$

The posterior distribution $P(\theta|C_l, X_l, X_u)$ is of interest for unseen data and is derived as:

$$P(\theta|C_l, X_l, X_u) \propto P(C_l|X_l, \theta)P(\theta). \tag{2.35}$$

However, this implies that unlabeled data and μ do not affect the posterior distribution of θ. Suppose instead that $P(\mu, \theta) = P(\theta|\mu)P(\mu)$ and that $P(X_u, \mu) = P(\mu|X_i)P(X_i)$. Then it is possible to calculate the distribution:

$$P(\theta|X_u) = \int P(\theta|\mu)P(\mu|X_u)d\mu. \tag{2.36}$$

$P(\theta|X_u)$ may have lower entropy that $P(\theta)$, implying that $P(\theta|X_l, X_u)$ is much narrower than $P(\theta|X_l)$. This means that unlabeled data may or may not be useful in improving classification performance.

Likelihood Estimation

According to the likelihood principle, all information about data is contained in the likelihood. However, below it is demonstrated that unlabeled data does not improve likelihood estimation. Let θ be a parameter and suppose that $X \sim P(x|\theta)$ with probability density function $p(x|\theta)$. Given a sample $x_1, .., x_l$ from X, the *maximum likelihood estimator* (*MLE*) of θ is defined as:

$$\widehat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i=1}^{l} \ell(\theta; x_i) = \arg\max_{\theta} \prod_{i=1}^{l} p(x_i|\theta). \tag{2.37}$$

The fact that unlabeled data does not help improve *maximum likelihood estimation* is demonstrated below. Define the random variable:

$$Z(c) = \begin{cases} 1 & c \text{ is hidden} \\ 0 & c \text{ is observed} \end{cases} . \tag{2.38}$$

Assume that the *missing at random* assumption holds, then the log-likelihood function for labeled and unlabeled data is simplified to:

$$\log \ell(\theta; \mathcal{L}, \mathcal{U}) = \log \sum_{i=1}^{|\mathcal{L}|} p(c_i|x_i; \theta) + \log \sum_{i=1}^{|\mathcal{U}|} p(z_i|u_i; \theta). \tag{2.39}$$

The first term of equation (2.39) is known as the *conditional log-likelihood*. This equation demonstrates that unlabeled data is useless under these assumptions as it does not affect the distribution of $C|X$. Furthermore, it is necessary to assume how X affects C.

Entropy Minimization

Unlabeled data does not improve performance in MLE as seen above. *Entropy regularization* [56] is a means to benefit from unlabeled data and *maximum a posteriori* (MAP) estimation. The belief that unlabeled data is informative is encoded in the prior distribution under the assumption that $P(c|x)$ has high entropy.

Maximum a Posteriori

Given the same assumptions regarding θ and X from the likelihood estimation problem, the MAP estimate may be obtained by maximizing the log posterior [57], [58]:

$$\widehat{\theta}_{MAP} = \arg\max_{\theta} \sum_{i=1}^{n} \log p(x_i, \theta). \tag{2.40}$$

To utilize MAP, it is necessary to specify the prior distribution $p(\theta)$. In semi-supervised learning, the assumptions regarding how C influences X are encoded in the prior $p(c)$.

Measuring class overlap

Class overlap is utilized to derive the prior distribution. Class overlap is caused by regions in the data where the prior distribution of two classes is approximately equal. Here, class overlap is measured utilizing *Shannons conditional entropy* [59], but other measurements may be utilized. The conditional entropy is defined as:

$$H(c|x, z = 1) = -\mathbb{E}_{x,c}\Big[\log P(c|x, h = 1)\Big]. \tag{2.41}$$

The parameter λ specifies the average expected entropy.

$$\mathbb{E}_{\theta}\Big[H(c|x, z = 1)\Big] = \lambda, \tag{2.42}$$

where λ is the positive Lagrange multiplier. According to the *principle of maximum entropy* [60], [61] the optimal choice of probability distribution is [62]:

$$p(c) \propto \exp\left(-\lambda H(c|x, h = 1)\right). \tag{2.43}$$

An expression for H may be found as:

$$H(c|x, z = 1) = -\int\int \log P(c|x, z = 1) p(c|x, z = 1) dc\, dP_{X|z=1}. \tag{2.44}$$

Utilizing the fact that c is latent then applying the *plug-in* principle on $P_{X|Z=1}$ leads to:

$$H(c|x, z = 1) = -\frac{1}{|\mathcal{U}|} \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(c|x_i, z_i = 1) \log P(c|x_i, z_i = 1). \quad (2.45)$$

Lastly, since $P(c|x, z) = P(c|x)$ the final simplification is:

$$H_{\text{emp}}(c|x, z) = -\frac{1}{|\mathcal{U}|} \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(c|x_i) \log P(c|x_i). \quad (2.46)$$

Plugging equation (2.46) into equation (2.43) yields:

$$p(c) \propto \exp\left(-\frac{\lambda}{|\mathcal{L}|} \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(c|x_i) \log P(c|x_i) \right). \quad (2.47)$$

The final MAP estimate is calculated using equation (2.40):

$$\widehat{c}_{\text{MAP}} = \arg\max_{\theta} \left(\sum_{i=1}^{|\mathcal{L}|} \log P(c|x_i; \theta) + \lambda \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(c|x_i; \theta) \log P(c|x; \theta) \right), \quad (2.48)$$

and the loss function utilized in semi-supervised learning is:

$$L = \sum_{i=1}^{|\mathcal{L}|} \log P(c|x_i; \theta) + \lambda \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(c|x_i; \theta) \log P(c|x; \theta) \quad (2.49)$$

where the supervised and unsupervised losses are defined as:

$$L_s = \sum_{i=1}^{|\mathcal{L}|} \log P(c|x_i; \theta), \quad (2.50)$$

and:

$$L_u = \sum_{i=|\mathcal{L}|+1}^{n} \sum_{c \in \mathcal{C}} P(y = c|x_i; \theta) \log P(y = c|x; \theta), \tag{2.51}$$

respectively.

Mutual information criterion to assure fairness.

A classifier is said to be *fair* if the model outputs the classes with equal frequency. The *mutual information* [63] measurement is defined as:

$$I(c; x) := \int \int p(x, c) \log \frac{p(x, c)}{p(c)p(x)} \, dxdc. \tag{2.52}$$

Maximizing equation (2.52) ensures the model outputs fair and decisive predictions. Moreover equation (2.52) may be rewritten as:

$$I = H(\mathbb{E}[p(c|x)]) - \mathbb{E}[H(p(c|x))]. \tag{2.53}$$

It is relevant to point out that fairness is achieved by maximizing $H(\mathbb{E}[p(c|x)])$ and decisiveness is achieved by maximizing $-\mathbb{E}[H(p(c|x)]$.

2.7 Deep Semi-Supervised Learning

As demonstrated above, semi-supervised learning attempts to use unlabeled data by minimizing entropy and guarantee fairness by maximizing the mutual information. Different techniques for improving entropy minimization and fairness are incorporated into DSSL algorithms and are presented below.

For notation, x represents labeled instances, and u represents unlabeled instances. Augmented instances are denoted:

$$\widehat{x} = \text{Augment}(x) \tag{2.54}$$

$$\widehat{u} = \text{Augment}(u) \tag{2.55}$$

and weak and strong augmentations for labeled and unlabeled instances are

denoted:

$$\widehat{x}^{\text{weak}} = \text{WeakAugment}(x) \tag{2.56}$$
$$\widehat{u}^{\text{weak}} = \text{WeakAugment}(u), \tag{2.57}$$

and

$$\widehat{x}^{\text{strong}} = \text{WeakAugment}(x), \tag{2.58}$$
$$\widehat{u}^{\text{strong}} = \text{WeakAugment}(u). \tag{2.59}$$

Pseudo-Labeling

Pseudo-labeling [28] is a semi-supervised learning technique utilized in deep learning that utilizes *the conditional cross-entropy*:

$$H_{\text{cross}}(c, p) = -c \log p - (1 - c) \log(1 - p), \tag{2.60}$$

where c is the 1-of-K code of the label and $h(x) = P_\theta(c|x)$. In other words, $h(x)$ is a vector where $h_j(x) = P_\theta(j|x)$ is the probability that the network outputs j. First, a supervised classifier is trained using training data, and the classifier is utilized to classify pseudo-labels for the unlabeled data. The classifier is then retrained using both training data and pseudo-labels as if the pseudo-labels are the true labels for the unlabeled instances. The pseudo-labels are recalculated every time the model parameters are updated. The label is assigned the class that has the maximum predicted probability:

$$c_i^{\text{pseudo}} = \begin{cases} 1 & \text{if } i = \arg\max_j P_\theta(j|u) \\ 0 & \text{otherwise.} \end{cases}, \tag{2.61}$$

Let $p_j^{\text{pseudo}} = P_\theta(j|u)$, and the total loss function becomes:

$$L = \frac{1}{|\mathcal{L}|} \sum_{i=1}^{|\mathcal{L}|} H_{\text{cross}}(c_i, p_i) + \frac{1}{|\mathcal{U}|} \sum_{j=|\mathcal{L}|+1}^{|\mathcal{L}|+|\mathcal{U}|} H_{\text{cross}}(c_j^{\text{pseudo}}, p_j^{\text{pseudo}}). \tag{2.62}$$

The first and second term of equation (2.62) represents equation (2.50) and equation (2.51) respectively. Therefore, pseudo-labeling is equivalent to entropy regularization. Obtaining the MAP estimate involves maximizing equa-

tion 2.51, therefore, it is equivalent to minimizing entropy.

Sharpening

Sharpening is a technique for minimizing the entropy of the label distribution and was originally utilized to reduce the temperature of categorical distributions [39]. The sharpening function is defined by:

$$\text{Sharpen}(p, T) := \frac{p_i^{\frac{1}{T}}}{\sum_i^n p_i^{\frac{1}{T}}}, \tag{2.63}$$

where $p = (p_1, .., p_n)$ is a vector that represents the probabilities and T is the desired temperature.

Embeddings

Embeddings improve generalization performance in supervised learning by utilizing unlabeled data [64]. Embeddings are data, projected to a low-dimensional space. Given data $u_1, .., u_n$, an embedding, $f(u_i)$ is found by minimizing:

$$\sum_i \sum_j L(f(u_i; \theta), f(u_j; \theta)), \tag{2.64}$$

with respect to a *balancing constraint*. L is a loss function and W is a matrix that specifies the similarity between x_i and y_i.

A deep neural network is built up by an *input layer*, N *hidden layers* and an *output layer*:

$$\text{Input}(x) = \text{Activation}\bigg(\sum_j w_j^{1,i} x_j + b\bigg), \tag{2.65}$$

$$\text{Hidden}^k(x) = \text{Activation}\bigg(\sum_j w^{k,i} \text{Hidden}^{k-1}(x_j) + b^{k,1}\bigg), \quad k > 1, \tag{2.66}$$

$$\text{Output}(x) = \sum_{j=1}^{d} w_j^{O,i} \text{Hidden}_j^N(x) + b^{O,i}. \tag{2.67}$$

There are three ways to invoke embeddings on a regularized network.

1. Add the unsupervised loss to the entire network

2. Add the unsupervised loss only up to the kth layer.

3. Create a copy of the network up to the kth layer, but initiate other weights to obtain another set of final weights. The *embedding functions* are calculated as:

$$g_k(x) = \sum_j w_j^{g,i} \text{Hidden}_j^k(x) + b^i. \tag{2.68}$$

   The copied network is known as *embedding* network and is trained simultaneously as the original network.

Embeddings are utilized in CRMatch [65], SimMatch [66] and CoMatch [67].

Data Augmentation

Data augmentation is commonly used in SSL to increase the number of training samples and improve generalization performance. Data augmentation techniques includes stochastic transformations such as Gaussian noise and other augmentations [68] such as *RandAugment* [69], *AutoAugment* [70] *Back-translation* [71], [72], for image classification and text classification. Π-model [73] and Mean-Teacher [74], utilize one augmentation. Most algorithms utilize two augmentations called weak and strong augmentations. FixMatch [75], Dash [76], AdaMatch [77], FlexMatch [78], CoMatch [67], CRMatch [65], SimMatch [66], SoftMatch [79], and FreeMatch [80]. MixMatch [81] and ReMixMatch are the only algorithms that utilize $K \geq 2$ different augmentations.

Mixup

Mixup is a form of data augmentation derived as a special case of vicinal distribution [82]. Mixup enforces linearity in a model which reduces uncertainty in the predictions and is favored by Occam's razor. To create mixup samples, draw samples (x_j, c_j) from the training data $\mathcal{L}$ and mix the samples with another sample (x_i, c_i): The *mixup samples* $x^{\text{mix}}, c^{\text{mix}}$ are then defined

as:

$$x^{\text{mix}} = \lambda x_j + (1-\lambda)x_i, \tag{2.69}$$
$$c^{\text{mix}} = \lambda c_j + (1-\lambda)c_i, \tag{2.70}$$

where the mixup distribution is:

$$\mu^{\text{mixup}}(x^{\text{mix}}, c^{\text{mix}}|x_i, c_i) = \sum_{j=1}^{J} \mathbb{E}_\lambda\big[1_{\lambda x_i+(1-\lambda)x_j} \cdot 1_{\lambda c_i+(1-\lambda)c_j}\big], \tag{2.71}$$

and $\lambda \sim \text{Beta}(\alpha, \alpha)$. MixUp is applied to the training data in the MixMatch [81] and ReMixMatch [83] algorithms.

Consistency Regularization

Many deep learning algorithms alter the training data utilizing random operations, such as dropout layers and random pooling schemes. This may lead to the same instances being assigned to different classes during multiple passes. However, each instance belongs to one class only. In *consistency regularization*, this issue is mitigated by regularizing the different random outputs utilizing the mean squared error. This technique is popular in DSSL where it is utilized to regularize different augmentations of the same instances so that both instances receive the same label. If the data is passed through the network E times, the loss function is:

$$\sum_{i=i}^{n} \sum_{j=1}^{E-1} \sum_{k=j+1}^{E} \|P_{C|X}(c|\text{Augment}_j(u_i);\theta) - P_{C|X}(c|\text{Augment}_k(u_i);\theta)\|^2. \tag{2.72}$$

In other words, the consistency loss is the absolute value of the difference between the model outputs at pass $t+1$ and t squared. Consistency regularization is utilized with the mean squared error in Π-model [73], Mean-Teacher [74], and MixMatch [81]. However, in ReMixMatch [83] it is discovered that the conditional cross-entropy is superior and is utilized in FixMatch [75], Dash [76], AdaMatch [77], FlexMatch [78], CoMatch [67], CRMatch [65], SimMatch [66], SoftMatch [79], and FreeMatch [80].

Loss functions

All the loss functions are decomposed into a supervised loss and an upervised loss:

$$L = L_s + wL_u. \tag{2.73}$$

In the supervised loss, the instances may be subjected to weak augmentation, and the loss function is always chosen to be the conditional cross-entropy [84]. For the mean-teacher and the MixUp algorithm, the unsupervised loss is chosen to be the mean squared error [74], and for ReMixMatch onward, all the unsupervised loss functions are the conditional cross-entropy [84]. There are always at least a supervised and an unsupervised loss but many algorithms add additional losses on unlabeled data. SimMatch includes instance pseudo-labels and therefore utilizes an additional unlabeled loss for the instance pseudo-labels L_{instance} [66]. CoMatch adds a loss on the graph-embeddings, L_{ctr} [67]. In the CRMatch **crmatch**and the ReMixMatch [83] an additional unsupervised loss is utilized called the *rotation loss*:

$$L_{\text{rot}} = \frac{1}{|\mathcal{U}|} \sum_{\mathcal{U}} H_{\text{cross}}(r, P_\theta(r|\text{Rotate}(u, r)), \tag{2.74}$$

where $r \in \{0, 90, 180, 270, 360\}$. The function $\text{Rotate}(u, r)$ rotates the unlabeled instances $u \in \mathcal{U}$ by r degrees. CRMatch also introduces a distance loss:

$$L_{\text{dist}} = d(\cdot, \cdot), \tag{2.75}$$

on the embeddings, where d is a distance function. The FreeMatch algorithm utilizes a fairness loss L_f to increase convergence, generalization and fairness by maximizing the mutual information criterion [80].

Ramp-up functions

It is common practice to replace the constant weights w with a ramp-up function to control the impact of unlabeled data. For example in the Π-model,

the *gaussian ramp-up* function:

$$w(t) = \begin{cases} \exp\Big(-5(t-1)^2 \Big), & t \leq 80, \\ 0, & \text{otherwise}, \end{cases} \tag{2.76}$$

makes the algorithm only utilize unlabeled data from training step 80 [77]. In AdaMatch, $w(t) = \frac{1}{2} - \frac{1}{2}\cos\big(\min(\pi, \frac{2\pi t}{T}\big)$.

Distribution Alignment

Distribution alignment [63] is another technique to guarantee fair predictors. Suppose that $\overline{P}$ is the running overage of predictions on unlabeled data during the training. Given a prediction $P_\theta(c|u)$, the prediction is rescaled and normalized to obtain a new label distribution that matches another distribution $P_\theta(c)$:

$$P_\theta^{\text{scaled}}(c|u) = \text{Normalize}\bigg(P_\theta(c|u) \cdot \frac{P_\theta(c)}{\overline{P_\theta}(c|u)} \bigg), \tag{2.77}$$

where for any vector $x = (x_1, ..., x_n)$:

$$\text{Normalize}(x)_i = \frac{x_i}{\sum_j^n x_j}. \tag{2.78}$$

Distribution alignment is utilized in ReMixMatch [83] and AdaMatch [77].

Contrastive learning

Contrastive learning is a self-supervised technique that has achieved success in semi-supervised learning [67]. It is a form of consistency regularization that enforces differently augmented instances to have similar normalized low-dimensional embeddings. The technique is utilized in CoMatch [67].

Histogram distribution

A k-Histogram distribution is an approximation of a discrete distribution P, taking values $\{1, 2, ..., M\}$, where H is a piece-wise constant function taking values on k different intervals [85]. The histogram is constructed by utilizing

data $X(\omega) = x \in \mathcal{X}$ where $X \sim P$. The histogram is denoted $\text{Hist}_{\mathcal{X}}(P(\cdot))$. The histogram distribution is utilized in the FreeNatch [80] algorithm.

The quantity-quality tradeoff for pseudo-labels

The *quantity-quality tradeoff* states that if a large amount of pseudo-labels are obtained, the quality of pseudo-labels may be low and vice versa [79]. It is demonstrated in [79], that Pseudo-Labeling achieves high quantity but low quality. This is because pseudo-labeling does not discard pseudo-labels of low quality due to weak performance of the underlying supervised model.

Fixed Threshold

The FixMatch algorithm introduces the idea of utilizing a threshold and improves the quality of pseudo-labels by discarding the less confident pseudo labels by specifying a constant threshold τ. In other words, a pseudo-label is only used in the training if its probability exceeds τ. Fixed thresholds are utilized in [66], [67].

Dynamic threshold

Dash [76] provides a dynamic threshold that changes during training:

$$\tau_t = C\gamma^{1-\tau}\rho, \tag{2.79}$$

where $\tau > 0, C > 1, \gamma > 1$ are chosen to be fixed parameters and ρ is estimated during training.

AdaMatch introduces the *relative threshold*:

$$\tau(u_j) = \begin{cases} 1, & \text{if } \max_{i\in[1,\dots,M]} P_\theta(i|u_j^{\text{weak}}) > c_\tau \\ 0, & \text{otherwise} \end{cases}, \tag{2.80}$$

where

$$c_\tau = \frac{\tau}{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} \max_{i\in[1,..,M]} P(i|x_j^{\text{weak}};\theta). \tag{2.81}$$

FlexMatch [78] proposes *curriculum pseudo-labeling* (CPL), a dynamic thresh-

old that changes during training and takes the learning difficulty of different labels into account. The learning difficulty of class label m is defined as:

$$\sigma_t(m) = \sum_{u \in \mathcal{U}} 1(\max P_{t,\theta}(c|u) > \tau_t(m)) \cdot 1(\max P_{t,\theta}(c|u) = m) \tag{2.82}$$

Large $\sigma_t(m)$ indicates better learning effect. $\sigma_t(m)$ is then normalized and written as:

$$\beta_t(m) := \frac{\sigma_t(m)}{\max_c \sigma_t(m)}, \tag{2.83}$$

and the threshold is defined as:

$$\tau_t(m) = \mathcal{M}(\beta_t(m)) \cdot \tau_0, \tag{2.84}$$

where $\mathcal{M}$ is chosen to be a monotone increasing convex function that provides increasing thresholds. Here $\beta_t(m)$ takes values $0 \leq \beta_t(m) \leq 1$ and τ_0 is the initial threshold. The function $\mathcal{M}$ is chosen to be monotonically increasing and have a maximum $\geq 1/\tau_0$.

SoftMatch introduces a dynamic threshold that depends on the distribution of pseudo labels $p = p^{\text{pseudo}} = P_\theta(c|u^{\text{weak}})$. Let $\lambda_t(p)$ denote the distribution of p at training step t. where:

$$\lambda_t(p) = \begin{cases} \tau \cdot \exp\Big(- \frac{(\max(p) - \mu_t)^2}{2\sigma_2^2} \Big), & \text{if } \max(p) < \mu_t, \\ \tau & \text{otherwise.} \end{cases} \tag{2.85}$$

where μ_t and σ_t are computed from historical predictions [79]. First compute:

$$\widehat{\mu}_b = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} \max(p_i), \tag{2.86}$$

$$\widehat{\sigma}_b^2 = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} (\max(p_i) - \widehat{\mu}_0)^2, \tag{2.87}$$

then for $t = 1, .., T$:

$$\widehat{\mu}_t = m\widehat{\mu}_{t-1} + (1-m)\widehat{\mu}_b, \tag{2.88}$$

$$\widehat{\sigma}_t = m\widehat{\sigma}^2_{t-1} + (1-m)\frac{|\mathcal{U}|}{|\mathcal{U}|-1}\widehat{\sigma}^2_b. \tag{2.89}$$

$\widehat{\mu}_0 = 1/M$, and M is the number of classes and $\widehat{\sigma}^2_0 = 1$. SoftMatch provides both high-quality and high quantity labels [79].

Finally, FreeMatch introduces the *self-adaptive threshold* (SAT). The threshold starts low and then increases as training continues. The threshold is class-dependent and is represented by a local and a global threshold τ_t^{local} and τ_t^{global} respectively. Note that the local threshold is independent of class label, but the global threshold is class dependent. The local threshold is defined as:

$$\tau_t^{\text{local}} = \begin{cases} \frac{1}{M}, & t = 0 \\ \lambda\tau_{t-1}^{\text{local}} + (1-\lambda)\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}\max(P_\theta(c|u_i^{\text{weak}})), & \text{otherwise} \end{cases}. \tag{2.90}$$

where $\lambda \in (0, 1)$. Similarly, the global threshold is defined as:

$$\tau_t^{\text{global}}(m) = \begin{cases} \frac{1}{M}, & t = 0 \\ \lambda\tau_{t-1}^{\text{global}}(m) + (1-\lambda)\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}P_\theta(c|u_i^{\text{weak}}), & \text{otherwise} \end{cases}. \tag{2.91}$$

The SAT is then defined as:

$$\tau_t(m) = \frac{\tau_t^{\text{global}}(m)}{\max_m \tau_t^{\text{global}}(m)} \cdot \tau_t^{\text{local}}. \tag{2.92}$$

2.8 Graph-Based Semi-Supervised Learning

Graph-based Semi-Supervised Learning (GBSSL) algorithms are transductive SSL algorithms utilized to label data automatically. GBSSL assumes that labeled and unlabeled data are embedded on a low-dimensional manifold [62], [86] that can be approximated by a weighted graph. Each instance of the dataset corresponds to a vertex in the weighted graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges and $W = (w_{i,j})_{i,j=1,2,..}$ is the weight matrix. The graph weight w_{ij} measures the similarity between

vertices i and j. Graph-based algorithms are attractive as they provide a natural representation of data, many GBSSL algorithms optimize a convex loss function, and the algorithms are scalable [87] and efficient in practice [86].

Theoretical ideas behind GBSSL are presented below, followed by the GBSSL algorithms utilized in this thesis.

Metric Learning

Metric Learning provides the basics of GBSSL . Let $h\colon S \to \mathcal{C}$, where h is the labeling function, S is the input data S and $\mathcal{C} = \{1, 2, .., m\}$ is the set of class labels. Each $x \in S$ has a likelihood of being assigned label c. This likelihood is the cost of assigning label c to x and is denoted $\text{cost}(x, c)$. The relationship between $x_1, x_2 \in S$ is expressed utilizing the weight w_{x_1,x_2}. Assigning label c_1 to x_1 and c_2 to x_2 costs $w_{x_1x_2} \cdot d(c_1, c_2)$ where $d(\cdot, \cdot)$ is some metric on $\mathcal{C} \times \mathcal{C}$. The labeling function is found by solving the minimization problem:

$$\widehat{h} = \arg\min_{h} \left[\sum_{x \in S} \text{cost}(x, h(x)) + \sum_{e=(x,y)} w_e d(h(x), h(y)) \right]. \tag{2.93}$$

The metric learning problem is connected to the *multiway-cut* problem [88] and *Markov random fields* [89].

Markov Random Fields

Markov random fields assign a probability distribution on the labeling function h, making it a random variable. The random variable h is *Markovian* is the sense that:

$$P(h(x)|\{h(y) : y \neq x\}) = P(h(x)|\{h(y) : (x, y) \in E\}). \tag{2.94}$$

In other words, the state $h(x)$ only depends on the states $h(y)$ where x, y are connected as an edge $e \in E$. If CL_G is the set of cliques of G, then by the *Hammersley-Clifford Theorem* [90], there exists a set of functions

$\{\Gamma_k, \text{for all } k \in \text{CL}\}$ s.t:

$$P(h) = \frac{1}{Z}\exp\left(- \sum_{k \in \text{CL}} \Gamma_k(h_{|_k}) \right). \tag{2.95}$$

Γ_k is a function from k to $\mathbb{R}$, and $h_{|_k}$ is the restriction of h on k to $\mathbb{R}$. Z is the normalizing constant. The cost of labeling x according to $h(x)$ is defined as:

$$\begin{aligned} \text{cost}(x, h(x)) &= -\log P(h(x)) && (2.96) \\ &= \sum_{k \in \text{CL}} \Gamma_k(h_{|_k}) + \log Z. && (2.97) \end{aligned}$$

Given data, h is found by optimizing:

$$\widehat{h} = \arg\min_h \left[\sum_x -\log P(c|h(x)) + \sum_{k \in \text{CL}} \Gamma_k(h_{|_k}) \right]. \tag{2.98}$$

where:

$$E(h) = \sum_x -\log P(c|h(x)) + \sum_{k \in \text{CL}} \Gamma_k(h_{|_k}). \tag{2.99}$$

is called *energy function*. In practice it is assumed that $\Gamma_k \neq 0$ for all edges $k \in \text{CL}$, and $\Gamma_k \colon \mathcal{C} \times \mathcal{C} \to \mathbb{R}_+$. Furthermore, it is assumed that $\Gamma_k = w_k\Gamma$ where Γ is a metric function.

For any metric Γ and $\text{cost}(x, h(x))$, the optimization problem may be written as:

$$\widehat{h} = \arg\min_h \left[\sum_x \text{cost}(x, h(x)) + \sum_{e=(x,y)} w_e \Gamma(h(x), h(y)) \right]. \tag{2.100}$$

Multiway Cuts

For a graph $G = (V, E, W)$ with *terminal set* C, a subset $\mathcal{F} \subset E$ is called a *multiway cut* if the removal of $\mathcal{F}$ from G separates the terminal verities for

each other. The cost $\mathcal{F}$ is defined as:

$$\text{cost}(\mathcal{F}) = \sum_{f \in \mathcal{F}} w_f. \tag{2.101}$$

The task of finding a multiway-cut with minimum cost:

$$\widehat{\mathcal{F}} = \arg\min_{\mathcal{F}} \text{cost}(\mathcal{F}) \tag{2.102}$$

is called the *multiway-cut problem.*

The connection between multiway-cuts and Markov random fields

A special case of metric learning is when:

$$\Gamma\Big(h(x), h(y)\Big) = u_{xy}\Big(1 - \delta_0(h(x) - h(y)\Big). \tag{2.103}$$

If $\widehat{h}$ minimizes $E(h)$, then there exist constants $K(x)$ for all x s.t:

$$K(x) > -\sum_{x \in S} \log P(c|h(x)). \tag{2.104}$$

Define the following sets:

$$C_x = \Big\{c \in \mathcal{C} : K(x) > -\sum_{x \in S} \log P(c[h(x))\Big\}, \tag{2.105}$$

and $\overline{C} = C_1 \times ... \times C_{|S|}$. It holds that $\widehat{h} \in \overline{C}$, so the minimization problem may be restricted to only searching $\overline{C}$. This is equivalent to the optimization problem:

$$\widehat{h} = \arg\min_{h} \sum_{(x,y)} 2u_{x,y}\Big(1 - \delta_0(h(x) - h(y))\Big) + \sum_{x \in S} \sum_{c \in C_x,\, c \neq h(x)} \Big[\log P(c|h(x)) + K(x)\Big]. \tag{2.106}$$

Suppose that $G = (V, E, W)$ with $V = S \cup \mathcal{C}$, where $\mathcal{C}$ is the terminal set. Let $\mathcal{F}_N \subset E$ be the set of edges that connect the vertices $v \in V$. These edges

are referred to as *N-links* and are assigned weights $w_f = 2u_f$. The set $\mathcal{F}_T = \{(x, c) \in E : x \in S,\, c \in \mathcal{C}\}$ is the set of edges that connects the terminals with points $x \in S$. The edges $e \in \mathcal{F}_T$ are called *T-links* and a cut is called *feasible* if there is only one T-link for each $x \in S$. The feasible cut $\mathcal{F}$ corresponds to a labeling $h^{\mathcal{F}} \in \overline{C}$. Furthermore, mathematical theory [91] demonstrates that a minimum cost cut on G for terminals C is feasible. Furthermore $h^{\mathcal{F}}$ minimizes the energy function, and equivalently finds a multiway-cut.

Notation

The theory above illustrates the theoretical concepts that motivates the construction of GBSSL algorithms. The purpose of GSSL is to find a *labeling function* $h\colon S \to \mathcal{C}$. The terminal set $\mathcal{C} = \{1, 2, ..., m\}$, is the set of class-labels and $S = \{x_1, ..., x_n, u_1, ..., u_m\}$ is the set of labeled and unlabeled instances. The data constitutes a graph $G = (V, E, W)$, where $V = S \cup \mathcal{C}$. The labeled points $x_1, .., x_n$ are already connected to one terminal vertex each and the feasibility condition guarantees that each unlabeled instance is assigned only one label. The purpose of GBSSL is to connect the unlabeled vertices to a label. In other words, if $S = \mathcal{L} \cup \mathcal{U}$, it is known that $h(x) = h_{\mathcal{L}}(x),\ x \in \mathcal{L}$ and GBSSL algorithms extend $h(x)$ to $h_{\mathcal{U}}\colon \mathcal{U} \to \mathcal{C}$.

The GBSSL algorithms utilized in this thesis are presented below. Many algorithms are presented for the binary classification problem where $\mathcal{C} = \{-1, 1\}$, but may be extended to multi-class classification.

Label Propagation

Label propagation [92] considers a directed graph $\overrightarrow{G}$ with transition kernel:

$$K(x, y) = \frac{w_{xy}}{d^+(x)},\ \text{where } d^+(x) = \sum_{x \leftarrow y} w_{yx}, \tag{2.107}$$

where K satisfies the balance equation:

$$\pi(y) = \sum_{x \to y} \pi(x) K(x, y), \forall y > 0, \quad \text{where } \pi(y) > 0,\ \forall y \in V. \tag{2.108}$$

The transition kernel defines a random walk on the graph $\overrightarrow{G}$. The label propagation algorithm utilizes the energy functional:

$$E(h) = \frac{1}{2} \sum_{(x,y)\in E} \pi(x) K(x,y) \Big(\frac{h(x)}{\sqrt{\pi(x)}} - \frac{h(y)}{\sqrt{\pi(y)}} \Big)^2, \tag{2.109}$$

and the optimal h is:

$$\widehat{h} = \arg\min_{h} \Big\{ E(h) + \mu \|h - c\|^2 \Big\}, \tag{2.110}$$

where c is the ground-truth labels for the labeled instances and $c = 0$ for the unlabeled data. The energy functional may be solved by utilizing the Θ operator:

$$\Theta h = \frac{1}{2} \Big(\sum_{x \to y} \frac{\pi(x) K(x,y) h(y)}{\sqrt{\pi(x)\pi(y)}} + \sum_{x \to y} \frac{\pi(x) K(y,x) h(y)}{\sqrt{\pi(x)\pi(y)}} \Big), \tag{2.111}$$

where $\pi(x) = d_{xx}$ for all $x \in V$. It is possible to write:

$$\Theta = \frac{D^{-1/2} K D^{-1/2} + D^{-1/2} K^T D^{-1/2}}{2}, \tag{2.112}$$

and the optimal labeling function $\widehat{h}$ is calculated by:

$$\widehat{h} = (I - \alpha\Theta)^{-1} c, \tag{2.113}$$

and the labels are calculated as $\widehat{c} = \operatorname{sign} h(x),\ \forall x \in \mathcal{U}$.

Label Spreading

Label spreading [93] finds $\widehat{h}$ by solving:

$$\frac{\partial E(h)}{\partial h} = 0, \tag{2.114}$$

where:

$$E(h) = \frac{1}{2} \sum_x \sum_u w_{xy} \left\| \frac{h(x)}{\sqrt{\pi(x)}} - \frac{h(y)}{\sqrt{\pi(y)}} \right\|^2. \tag{2.115}$$

The solution may be written as:

$$\widehat{h} = (1 - \alpha P)^{-1} c = (D - \alpha W) c. \tag{2.116}$$

If $S = D^{-1/2} W D^{-1/2}$, then $\widehat{h}$ is obtained by iterating:

$$\widehat{h}_{t+1} = \alpha S \widehat{h}_t + (1 - \alpha) c, \text{ and } c_i = \arg\min_{j<c} \widehat{h}_{ij}. \tag{2.117}$$

Laplace learning

Suppose that $G = (V, E, W)$ is a graph with vertices $v \in V$ and edges $e \in E$. The weight matrix $W = (w_{xy})_{x,y \in V}$ is often on the form:

$$w_{xy} = \exp\left(- \frac{\|x - y\|^2}{\sigma^2} \right), \quad W_{xx} = 0, \ \forall x, y \in V. \tag{2.118}$$

Laplace learning [94] aims to find a labeling function h, that solves the Laplace equation:

$$\begin{cases} \Delta h = 0, \ v \in \mathcal{U} \\ h(v) = h_L(v), \ v \in \mathcal{L}, \end{cases} \tag{2.119}$$

where h_L is the labeling function that labels $v \in \mathcal{L}$. The function h will extend h_L to label the set $\mathcal{U}$. This is equivalent to solving the optimization problem:

$$\arg\min_{h \colon S \to \mathcal{C}} E(h) \quad \text{subject to} \quad h(x) = h_L(x), \ x \in \mathcal{L}, \tag{2.120}$$

where:

$$E(h) = J_2(h) = \sum_{x \in S} \sum_{y \in S} w_{xy} |h(x) - h(y)|^2. \tag{2.121}$$

The solutions h are called *harmonic*. Let $d_x = \sum_y w_{xy}$, then if $D = \operatorname{diag}(d_x),\ x \in S$, then the operator $\Delta = D - W$ is called *graph-Laplacian* and $\Delta h = 0$. Suppose that $h = [h_L, h_U]$ where $h(x) = h_U(x),\ x \in \mathcal{U}$, and write:

$$D = \begin{bmatrix} D_{ll} & D_{lu} \\ D_{ul} & D_{uu} \end{bmatrix}, \quad W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}. \tag{2.122}$$

Then it is possible to find h_U by:

$$h_U = (D_{uu} - W_{uu})^{-1} W_{ul} h_L, \tag{2.123}$$

where W has to be learned utilizing the *average cross entropy*:

$$H_{\text{entropy}}(h) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \Big[-h \log h - (1-h) \log(1-h) \Big]. \tag{2.124}$$

Random Walk

The random walk algorithm [95] learns W and then computes $S = D^{-1/2} W D^{1/2}$. A *lazy random walk* is defined on the graph with transition matrix $K = (1-\alpha)I + \alpha D^{-1} W$ where $\alpha \in (0,1)$ and I is the identity matrix. Moreover there exists a stationary distribution π for the lazy random walk s.t $\pi = \pi K$. where $\pi = \mathbf{1}D/(\text{Vol } G)$. Given labeled $(x, c) \in \mathcal{L}$, compute $h = (I - \alpha S)^{-1} c$, and classify the unlabeled data by $\widehat{c} = \text{sign } h(x)$.

MBO

MBO [96] minimizes the L_2-gradient flow of the *Grindberg-Landlau functional*:

$$\text{GL}(h) = \frac{\varepsilon}{2} \int |\nabla h|^2 \, dx + \frac{1}{2} \int \Phi(h) \, dx, \tag{2.125}$$

when $\varepsilon > 0$ is a constant and $\Phi(h) = \frac{1}{4}(h-1)^2$ is the *double-well potential*. The TV norm of h is defined as:

$$\|h\|_{\text{TV}} = \sum_{x \in V} \sum_{y \in V} \Big| h(x) - h(y) \Big|. \tag{2.126}$$

It has been demonstrated that $\mathrm{GL}(h) \to_\Gamma \|h\|_{\mathrm{TV}}$, where Γ denotes the Γ-convergence. A *fidelity term* is included in the energy function to avoid trivial solutions. The complete energy function is $E(h) = \mathrm{GL}(h) + F(h, c)$, where minimizing $E(h)$ is equivalent to solving the *modified Allen Cahn equation:*

$$\frac{\partial h}{\partial t} = -\frac{\partial \mathrm{GL}}{\partial h} - \mu \frac{\partial F}{\partial h}. \tag{2.127}$$

The GL functional on graphs is defined as:

$$\mathrm{GL}(h) = \frac{\varepsilon}{2}\langle h, L_s h\rangle + \frac{1}{4\varepsilon}\sum_{x\in V}\left[h^2(x) - 1\right]^2, \tag{2.128}$$

and the total energy function becomes:

$$E(h) = \frac{\varepsilon}{2}\langle h, L_s h\rangle + \frac{1}{4\varepsilon}\sum_{x\in V}\left[h^2(x) - 1\right]^2 + \sum_{x\in V}\frac{\mu_x}{2}(h(x) - c), \tag{2.129}$$

where c is the true label, $\mu_x > 0$ if x is a fidelity rate and $\mu_x = 0$ otherwise. Labels are assigned by propagating through:

$$\frac{h^{n+\frac{1}{2}} - h^n}{dt} = -L_s h - \mu(h^n - c), \tag{2.130}$$

and

$$h^{n+1} = \begin{cases} 1 & \text{if} \quad h^{n+\frac{1}{2}} > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{2.131}$$

The scheme may be generalized to multi-label classification.

Weighted non-local Laplacian

Solutions provided by the graph-Laplacian are not continuous at $x \in S$ when $\frac{|\mathcal{L}|}{|\mathcal{U}|}$ is small. Therefore, the Laplace equation is approximated utilizing integral equations on the boundary [97]:

$$h(x) + \mu\frac{\partial h(x)}{\partial \mathbf{n}} = h_L(x), \ x \in \mathcal{L}, \tag{2.132}$$

where $0 < \mu < 1$. This leads to the following optimization problem:

$$\arg\min_{h} \left\{ \sum_{u \in \mathcal{U}} \left(\sum_{x \in S} w_{xu} (h(x) - h(u))^2 \right) + \frac{|S|}{|\mathcal{L}|} \sum_{x \in \mathcal{L}} \left(\sum_{z \in S} w_{xu} (h(x) - h(z)^2 \right) \right\}, \tag{2.133}$$

for $x \in \mathcal{U}$ and $h(x) = h_L(x),\ x \in \mathcal{L}$. As $\frac{|S|}{|\mathcal{L}|} \to \infty$ the weighted non-local Laplacian [97] becomes the graph-Laplacian. Furthermore, it has been demonstrated that the weighted non-local Laplacian converges faster than the graph-Laplacian.

The optimal h is found by solving the linear system:

$$\sum_{z \in S} \Big(w_{zu} + w_{uz} \Big) (h(u) - h(z)) + \Big(\frac{|S|}{|\mathcal{L}|} - 1 \Big) \sum_{x \in \mathcal{L}} w_{xu} \Big(h(u) - h(x) \Big) = 0, \quad u \in \mathcal{U}. \tag{2.134}$$

subject to $h(x) = h_L(x),\ x \in \mathcal{L}$.

Sparse Label Propagation

Sparse label propagation [98] relies on the TV norm rather than graph-Laplacian and that $\overrightarrow{G} = \{V, \overrightarrow{E}\}$ is a directed graph. Suppose that e^- and e^+ represent the tail and head of an edge $e \in \overrightarrow{E}$. If D is the *incidence matrix* of $\overrightarrow{G}$, let $\mathcal{F} = \{C_1, ..., C_{|\mathcal{F}|}\}$ be the partition of disjoint cluster. The *boundary of* $\mathcal{F}$ is defined as:

$$\partial \mathcal{F} := \{ \{e^-, e^+\} \in E : e^- \in C_a,\ e^+ \in C_n, a \neq b \}. \tag{2.135}$$

Assume that:

$$h(x) := 2 \max_{\ell \in \{1,2,...,|\mathcal{F}|\}} |a_\ell| \sum_{e \in \partial F} w_{e^-, e^+}, \tag{2.136}$$

and $\|h\|_{\text{TV}} = \|Dh\|_1$. The optimal h is found by solving the optimization problem:

$$\arg\min_{h} \|Dh\|_1, \quad \text{s.t} \quad h(x) = h_L(x), \quad \text{for all } x \in \mathcal{L}. \tag{2.137}$$

Volume MBO

Given $\Sigma = (\Sigma_1, ..., \Sigma_M)$, where Σ_c, $c = 1, ..., M$, is the set of all x whoose label is c. The purpose of Volume MBO [99] is to minimize:

$$E(\Sigma) = R(\Sigma) + F(\Sigma), \tag{2.138}$$

where $R(\cdot)$ is a functional measuring the robustness, and F is the fidelity function. In addition, Volume MBO requires upper and lower bounds for the number of elements in Σ_i: $B_i \leq |\Sigma_i| \leq U_i$. R is chosen so that the problem is equivalent to the multiway-cut problem:

$$\arg\min_{\Sigma} \frac{1}{2} \sum_{c=1}^{M} \sum_{x \in \Sigma_c} \sum_{y \in \Sigma_c} w_{xy}, \quad \text{s.t } \mathcal{L}_i \subset \Sigma_i, \quad B_i \leq |\Sigma_i| \leq U_i. \tag{2.139}$$

To solve the minimization problem, it is linearized into:

$$\arg\min_{h\colon \mathcal{U} \to \mathcal{K}^n} \sum_{c=1}^{M} \sum_{x \in \mathcal{U}} \psi_c(x) h_c(c), \quad \text{s.t } B_i - |\Sigma_i| \leq \sum_{x \in \mathcal{U}} h_c(x) \leq U_i - |\Sigma_i|, \tag{2.140}$$

where $\psi_i(x) = \sum_{y \in \mathcal{L}_i} w_{xy}$.

p-Laplace Learning

Laplacian regularization techniques such as the graph-Laplacian is ill-posed in problems with few labels. This means the learned function will be constant on the entire graph except for near the labeled data. To mitigate the issue, L_p-based Laplacian [100] regularizes:

$$J_p := \frac{1}{2p} \sum_{x \in S} \sum_{y \in S} w_{xy} |h(x) - h(y)|^p, \tag{2.141}$$

are recommended as they prevent h from developing sharp spikes. The h minimizing J_p satisfies the *variational graph-Laplacian* $\Delta_p^G h(x) = 0$ where:

$$\Delta_p^G h(x) = \frac{1}{2p} \sum_{x \in S} \sum_{y \in S} w_{xy} |h(x) - h(y)|^{p-2} (h(y) - h(x)). \tag{2.142}$$

The function h will transition more smoothly between labeled and unlabeled data as p increases. If $p \to \infty$, then p-Laplace learning is called *Lipschitz*-learning with smoothness functional:

$$J_\infty(h) = \max_{x,y \in S} w_{xy}|h(x) - h(y)|. \tag{2.143}$$

The h's that minimize J_∞ are found utilizing the *Lex minimizer* and satisfy:

$$\Delta_\infty^G h(x) := \min_{y \in S} w_{xy}(h(y) - h(x)) + \max_{y \in S} w_{xy}(h(y) - h(x)). \tag{2.144}$$

It has been demonstrated that Lipschitz learning is well-posed with access to few labels. However, Lipschitz learning does not incorporate the data distribution and is therefore unsuitable for semi-supervised learning. Therefore, the *game-theoretic p-Lacplacian* is introduced as:

$$L_p^G(h) = \frac{1}{d_x p}\Delta_2^G h(x) + \lambda\Big(1 - \frac{2}{p}\Big)\Delta_\infty^G h(x), \tag{2.145}$$

where $\lambda > 0$.

Both the variational and the game-theoretic graph p-Laplacian satisfy the *p-Laplace equation*:

$$\Delta_p h := \operatorname{div}\Big(|\nabla h|^{p-2}\nabla h\Big) = 0, \tag{2.146}$$

To find a semi-supervised classifier utilizing p-Laplace Learning the following optimization problem must be solved:

$$\arg\min_h J_p(h) \quad \text{subject to } h(x) = h_L(x),\ x \in \mathcal{L}. \tag{2.147}$$

The function h will satisfy the *variational p-Laplace equation*:

$$\begin{cases} -\Delta_p^G h(x) = 0, & \text{if } x \in \mathcal{U}, \\ h(x) = h_L(x), & \text{if } x \in \mathcal{L}, \end{cases} \tag{2.148}$$

and in Lipchitz learning h will satisfy the *Lipchitz equation:*

$$\begin{cases} -L_p^G h(x) = 0, & \text{if } x \in \mathcal{U}, \\ h(x) = h_L(x), & \text{if } x \in \mathcal{L}. \end{cases} . \tag{2.149}$$

Poisson Learning

While Laplace learning solves the Laplace equation utilizing the graph-Laplacian, Poisson learning [101] solves the Poisson equation by replacing the graph Laplacian with:

$$\Delta h(x) = \sum_{c \in \mathcal{L}} (c - \overline{c}) \cdot 1(h(x) = c), \ x \in \mathcal{U} \quad \text{subject to } \sum_{x \in S} d_x h(x) = 0, \tag{2.150}$$

where the average of all labels is:

$$\overline{c} = \frac{1}{|\mathcal{L}|} \sum_{c \in \mathcal{L}} c. \tag{2.151}$$

Poisson MBO

The Poisson MBO [101] algorithm applies Poisson learning to propagate the labels and then applies graph-cut to adjust the decision boundary. This is done by solving the optimization problem:

$$\arg\max_h \Big\{ \frac{1}{4} \sum_{x \in S} \sum_{y \in S} [w_{xy}(h(y) - h(x))]^2 - \mu \sum_{x \in S} (c - \overline{c}) h(x) \Big\}, \tag{2.152}$$

$$\text{subject to } \frac{1}{|S|} \sum_{x \in S} h(x) = b. \tag{2.153}$$

where b incorporates prior knowledge about the class distribution.

Conclusion

This chapter has outlined the theoretical concepts utilized to derive SL, AL and DSSL techniques.

Probability and statistics help model data and decision theory allows AI to

make intelligent decisions based on data. Decision theory derives structural risk minimization, which defines SL algorithms. SL utilizes labeled training data to find a classifier to label future incoming unlabeled data points. Structural risk minimization does not apply to generative models and does not incorporate unlabeled data. Vicinal risk minimization generalizes structural risk minimization to incorporate unlabeled data to define SSL. There are transductive and inductive SSL algorithms. Inductive learning has the same goal as supervised classification but includes unlabeled data in the training set. Transductive learning takes input data that consists of both labeled and unlabeled data. The algorithm learns to label the unlabeled input data based on the labeled input data. These algorithms do not generalize labeling to new unseen instances. Entropy regularization is utilized to derive inductive SSL algorithms. Metric learning, and Markov random fields are utilized to define trnsductive GBSSL. Both SL and inductive SSL are usually trained passively. In other words, the unlabeled instances to be labeled are chosen randomly. In AL the instances to be labeled are chosen according to a query strategy and have proven to outperform random sampling in many cases.

Both Active and Semi-Supervised learning are tools to minimize data labeling. The studies in this thesis discuss how different tools and techniques based on AL and SSL can be used in industry and the pros and cons of utilizing these methods.

CHAPTER 3

Research objective and method

The objective of this research is to present AI-based solutions to help practitioners automate data labeling or improve the performance of supervised learning by incorporating unlabeled data. Many AI-based algorithms are available, but a lack of research provides practitioners with guidelines regarding what algorithms are optimal for their use-case. In addition, there is a lack of research demonstrating what datasets are optimal to include in evaluations of SSL algorithms. Lastly, there are a few studies that evaluate SSL on real-world datasets from industry.

To fill the research gap, the following studies are presented. First, a systematic mapping study was conducted to present available AL and SSL algorithms and application domains and datasets utilized to evaluate the algorithms. Second, a case study presenting challenges faced by the industry. For each challenge, a mitigation strategy is presented based on the algorithms found in the systematic mapping study. The remainder of the studies utilize Bayesian models to analyze data from simulations evaluating algorithms and datasets. Third and fourth studies evaluate GBSSL and DSSL algorithms. The algorithms are ranked utilizing the Bradley-Terry model, and the probability of the algorithms reaching a specific performance is calculated utilizing

a GLMM. The fifth and sixth studies utilize IRT to find the optimal datasets for evaluating GBSSL and DSSL algorithms. The last study utilizes linear regression to evaluate the improved performance of a DSSL algorithm compared to a DSL algorithm on a real-world dataset collected from industry. At the end of this chapter, threats to validity are presented.

3.1 Objective

Data Labeling is an essential step in preprocessing data to obtain a high-performance supervised learning model. However, labeling presents many challenges. Companies need to allocate the labeling task to the appropriate personnel and consider the security and privacy issues regarding sharing data. In addition, companies need to guarantee that labels are of sufficient quality. Therefore, companies are interested in methods that reduce or eliminate data labeling. While manual data labeling is necessary for companies working with applications such as medical images for diagnostics [15], they may still want to reduce manual labeling to reduce costs. Based on input from two companies, many companies have no interest in manual data labeling and prefer fully automatic labeling.

This doctoral thesis aims to provide mitigation strategies for challenges that practitioners from industry and academia face when performing labeling. These challenges include but are not limited to obtaining high-quality labels with minimal costs, choosing and developing the optimal algorithm for obtaining labels or improving the performance of supervised learning by incorporating unlabeled data. Practitioners who may not rely on benchmark algorithms must develop their algorithms. To evaluate an algorithm fairly, the appropriate datasets must be chosen. To illustrate the benefits of DSSL in industrial scenarios, a case study applying a DSL algorithm to an industrial dataset is provided. To achieve the research goals above, the following research questions are formulated.

RQ1: What data labelling challenges exist in industry, and how can these be mitigated utilizing AI-based algorithms?

Data labeling is a key component in training supervised learning algorithms. Therefore, companies are required to optimize the labeling procedures. However, there is a lack of research providing mitigation strategies to academic

and industrial practitioners. Many algorithms exist, but many are only evaluated theoretically or based on benchmark studies. Therefore, many solutions may not work in practice, so practitioners need precise guidelines to label data or utilize SSL to improve the performance of SL by utilizing unlabeled data. This research question aims to identify challenges labelers face in industry and academia and formulate mitigation strategies.

RQ2: What existing algorithms exist that may be utilized to solve the labelling challenges?

The optimal scenario for practitioners is when labels may be obtained automatically or if high classification performance is reachable without additional manual labelling. This is possible with AL and SSL, respectively. However, developing and choosing the optimal AL and SSL algorithms for each use-case may be time-consuming. There is a lack of research to help practitioners choose algorithms based on use-case. Therefore, a systematic mapping study is performed to identify algorithms, the applications they are utilized for, and the datasets utilized to evaluate the algorithms.

RQ3: What GBSSL algorithms are optimal for automatic labelling, and what DSSL algorithms are optimal for outperforming DSL utilizing unlabeled data with respect to datatype and manual effort?

Based on the systematic mapping study, the most popular transductive SSL algorithms for automatic data labeling are GBSSL algorithms. GBSSL algorithms such as Laplace Learning, and Poisson Learning are contained in the GraphLearning package [102]. In recent years, DSSL algorithms has become popular due to the success of DL. Popular state-of-the-art DSSL algorithms are contained in the USB package in Python [84]. Utilizing Bayesian data analysis tools, the algorithms contained in GraphLearning and USB are evaluated. Many datasets across different datatypes are utilized, and the number of labels is varied to determine how many labels are required for optimal accuracy.

RQ4: What datasets are optimal for evaluating GBSSL and DSSL algorithms?

For industry practitioners to develop optimal generic algorithms, it is essential to utilize appropriate datasets when evaluating the algorithms. Otherwise, algorithms that perform well on benchmark datasets are not guaranteed to perform well on real-world datasets. Item response theory (IRT) is utilized to help practitioners construct benchmark tests consisting of appropriate datasets for evaluating GBSSL and DSSL algorithms. IRT determines what datasets are appropriate by assessing their difficulty, discrimination, and the ability of algorithms to learn the datasets.

RQ5: What are the pros and cons of utilizing SSL for drone classifcation on real-world Doppler radar dataset?

Practitioners must know the benefits of DSSL before they consider using it. Perhaps utilizing SSL given a limited number of labels yields an insufficient increase in performance at the cost of computation time. To demonstrate the advantages and disadvantages of utilizing DSSL in real-world scenarios, DSSL is applied to an industrial dataset used for classifying Doppler-Radar data. Performance and computation time are measured as the number of labelled instances varies. Bayesian analysis is utilized to calculate the relative improvement of DSSL over DSL.

3.2 Research Context

The research presented in this thesis was conducted in the context of the Wallenberg AI, Autonomous Systems Program (WASP) [36] and Software Center (SC) [103].

The main goal of WASP is to conduct research within AI and Autonomous Systems to create intelligent systems-of-systems software for the benefit of industry. The research is conducted with respect to two dimensions: The *strategic* and the *thematic* dimension. The strategic dimension is divided into three strategic areas: *AI, Autonomous Systems* and *Software*. The thematic dimension is divided into seven themes: *Perception and Sensing, Control and Decision Making, Machine Learning and Knowledge Representation, Interaction and Collaboration, Software Technologies and Methods*, and *Mathematical*

Foundations and Theory.

WASP conducts research collaborations with the industry and eight Swedish universities: *Chalmers University of Technology, KTH Royal Institute of Technology, Linköping University, Lund University, Umeå University, Örebro University, Uppsala University* and *Luleå University of Technology.*

SC conducts close and long-term research projects in collaboration with academia and industry. The SC research is divided four research themes, "*Continuous and Automated Quality Assurance", "Continuous Safety, Security and Architecture*", "*Data-driven Digital Transformation*", and "*AI-supported Engineering*" and five communities *AI-engineering*, *Product Management*, *Senior Leaders*, *Software Engineering* and *Systems Engineering.*

There are five SC universities: *Chalmers University of Technology, Gothenburg University, Malmö University, Linköping University, Mälardalen University*, and fourteen SC companies: *Axis Communications, DEIF, Ericson, Tetra Pak, Advenica, Volvo Group, Volvo Cars, Jeppesen, Grundfors, Siemens, Zenseact, Bosch, Saab*, and *Scania.*

Company A and C are both associated with WASP and SC.

3.3 Method

Multiple research methods were utilized to answer the research questions. Table 3.2 provides an overview of the research methods used for each paper.

In Paper A, a systematic mapping study was performed to identify algorithms to solve issues regarding labeling for machine learning. It was discovered that despite the wide range of algorithm research, there is a lack of research providing practitioners insight into what algorithm works for specific applications and what datasets are used in benchmark tests to evaluate algorithms, [25]. Paper A demonstrates, that GBSSL algorithms are the most popular algorithms to label data automatically.

Paper B reports on a case study conducted in parallel with Paper A. The case study was conducted through an internship with the industry, and interviews were held with industry practitioners from two companies in the embedded systems domain. Paper B identified current challenges and mitigation strategies related to labeling in the industry. After identifying current challenges and mitigation strategies, the data collected from Paper A is used to formulate new mitigation strategies.

Papers C-F evaluate the benchmark algorithms and datasets identified from the mapping study. *Simulation-based studies* [104] were utilized to evaluate the performance of algorithms and datasets in different scenarios. Developing and evaluating algorithms is a time-consuming task. Therefore, guidelines that help practitioners design algorithms and benchmark tests are appreciated [34].

Finally, Paper G presents a new semi-supervised learning algorithm developed for a Doppler radar dataset from industry. First, the study fits a supervised learning algorithm to the data and then improves its performance by incorporating unlabeled data using deep semi-supervised learning. Therefore, Paper G is an exploratory and improving case study [105]. The paper demonstrates the pros and cons of utilizing DSSL on a real-world Doppler radar dataset from industry.

Systematic Mapping Study

Systematic mapping studies are popular tools for identifying current research within a specified topic and have been used in many disciplines, such as *medical research* and *software engineering* [106]. Therefore, systematic mapping studies are appropriate for identifying research on algorithms for labeling. A systematic mapping study was utilized to discover what AL and SSL algorithms are available, the application domains they are utilized in, and what datasets are utilized for evaluating the algorithms.

Google Scholar was utilized to search for papers as it is considered an unbiased source [107] and contains many databases such as arXiv. Textbooks on AL and SSL were read to identify different types of AL and DSSL algorithms. Then, for each $x = "AL, SSL"$ and each type y of algorithm x, the search string $"x+y"$ was performed. The string "active machine learning" had to be specified for active learning because there is a field within pedagogical learning called active learning. Papers published between 2000-2020 were included in the screening because most studies performed from 1980-1999 were expected to be mostly theoretical. In addition, not many simulations may have been performed due to the lack of computing infrastructure during that time-period.

Case Study

Case studies are utilized in fields such as psychology, sociology, political science and software engineering [105]. The purpose of case studies is to study people, organizations, and different phenomena [105]. However, case studies can have multiple purposes and share characteristics from all four types. Software engineering studies how practitioners in academia and industry perform software development, software operations and software maintenance. Therefore, a case study is suitable for studying how practitioners perform labeling in the industry [105]. There are four types of case studies: *exploratory, descriptive, explanatory* or *improving*.

Papers B and G reports on case studies. The companies that participated in the studies are referred to as Company A, B, and C. Company A and B participated in the first case study, and Company C participated in the second case study.

Paper B investigates how companies perform data labeling and seeks insight into the challenges industry practitioners face and what mitigation strategies exist to solve these issues. Therefore, Paper B is considered an *exploratory* case study. The study is divided into two phases called exploration and validation phases. The exploration phase consisted of an internship at Company A, where the authors spent 2-3 times/week at the company. Data was collected by observing how data scientists work with labeling. In addition, the authors attended meetings and workshops and analyzed datasets. Next came the validation phase, where interviews were conducted with participants from both Company A and B. During the interviews, the participants were asked data labeling-related questions. At the end of the study, three challenges, together with mitigation strategies were formulated based on Paper A.

It may be argued that the case study is *improving*, since new mitigation strategies are proposed. However, these mitigation strategies are not implemented to see if they solve the challenges.

Paper G is a case study performed with company C. Company C provided a dataset for which they requested an algorithm to label complex-valued radar data. The authors developed an algorithm that outperformed DSL by incorporating unlabeled data.

Case study companies

All the companies involved in the case studies are based in Sweden. Company A and B have asked to be anonymous, so only a short description is provided for them.

Company A is a global telecommunications provider with more than 10000 employees worldwide

Company B is a data labelling company with more than 100 employees. The company maintains a labelling platform for autonomous vehicle data.

Company C is the Swedish defense company Saab. Saab was founded in 1937 to provide Sweden with military aircraft. Today, Saab provides military defense solutions and operates on every continent, with more than 24500 employees around the world.

Five industry practitioners were interviewed during the validation phase of Paper B. The practitioners held the positions of "data scientist" or "senior data scientist". Their work experience varied between 2 and 8 years in industry. However, the participant from Company B had more specific experience with labeling.

Table 3.1: Overview of participants in the interview study

Company	**Participant Nr**	**Title/Role**	**Experience**
A	I	Data Scientists	4 years
A	II	Senior Data Scientist	8 years
A	III	Data Scientist	3 years
A	IV	Senior Data Scientist	2 years
B	V	Senior Data Scientist	7 years

Simulation Studies

Papers C-F are simulation-based studies [108]. Simulation-based studies have been utilized in software engineering since the 1980s [108]. Simulation-based studies generate data to simulate a phenomenon. In mathematical terms,

simulation means generating random numbers $x_1, x_2, \ldots, x_n$ from a random variable X [104]. In Papers C-F, the generated random numbers correspond to the performance metric obtained from running algorithms on datasets.

A *benchmark study* [34] is a simulation study that evaluates the performance of algorithms in different scenarios. Generally, two questions are answered: How well does a particular algorithm perform on a given problem? and why does an algorithm succeed/fail on a specific test problem? [34]. However, the goals of benchmark studies vary depending on the problem [34]. Papers C-F aim to compare algorithms (Paper C, D) and datasets (Paper E, F) in different scenarios. Therefore, a benchmark study is a suitable choice for Papers C-F [34]. According to Paper A, GBSSL algorithms are the most popular automatic labeling algorithms. Paper A also identifies the most popular datasets for evaluating labeling algorithms. These algorithms are utilized in Papers C and E and are available in the GraphLearning package [102]. DSSL algorithms are not included in Paper A to reduce the scope of algorithms. However, it was later decided to evaluate DSSL algorithms due to their recent popularity. The DSSL algorithms and datasets are from the *universal semi-supervised benchmark* (USB). The algorithms in USB have previously been evaluated [84], but in Paper D, more research questions and algorithms are added, and a superior research methodology is utilized.

Experimental Setup

All AI models for Papers C, E and G, as well as the data generation model for Papers E and F were implemented in Python. All statistical models for Papers C-G are implemented in R. Papers C and F evaluate algorithms across three dimensions: *performance, datatype* and *manual effort*. For Paper C, the performance accuracy was calculated by:

$$\varepsilon = \frac{100}{n-m} \max\left(\sum_{i=1}^{n} I(y_i = \widehat{y}_i) - m, 0\right),$$

where $I(x)$ is *indicator function* defined as:

$$I(x) = \begin{cases} 1 \text{ if } x \text{ is true} \\ 0 \text{ otherwise} \end{cases},$$

where n is the total number of instances and m is the number of labeled instances. In Paper E, the performance is measured by error rate [84].

The datatype dimension is assessed by comparing the performance of algorithms on datasets of different datatypes. The datasets are equally distributed among three datatypes. Papers C utilizes 12 of the most common datasets for evaluating GBSSL algorithms [109]. The different datatypes are image, text and numeric. In Paper D and F, 15 datasets are utilized. These datasets were chosen because they were already available in USB. The datatypes are image, text, and sound.

In all simulation studies, Bayesian data analysis is performed due to the advantages of Bayesian statistics over frequentist statistics [110]–[112]. In Papers C and E, the Bradley-Terry model ranks the algorithms by the highest performance. A generalized linear mixed model calculates the probability of achieving a certain performance. These simulations will assist practitioners in selecting the optimal algorithm for their use-case. In Papers D and F, item response theory determines the optimal datasets for evaluating algorithms. This will help practitioners choose the optimal datasets to include in benchmark tests. Selecting the right datasets to evaluate algorithms is essential. If algorithms are evaluated on datasets that are too easy to learn, then benchmark studies provide a false sense of security that algorithms will perform well on real-world datasets. Therefore, algorithms must be evaluated on several datasets of different levels of difficulty. Paper G utilizes Bayesian linear regression to calculate the relative improvement of DSSL over DSL. The results demonstrate the pros of utilizing DSSL and inspire practitioners to utilize it.

The Who, What and How of Software Engineering research

The *Who-What-How* framework proposed by [113] may be used to describe who benefits from the research contained in this thesis.

The results presented in this thesis benefit both human stakeholders and technical systems, as well as researchers. The human stakeholders are data scientists, software engineers, and machine learning engineers, as they are typically responsible for labeling and possess expertise in data and machine learning models. A technical system would be any software containing an automatic labelling algorithm based on graph-based semi-supervised learning or deep semi-supervised learning. Paper G develops a technical system that utilizes a semi-supervised learning algorithm to classify radar data. Practi-

tioners from both industry and academia who may utilize the results of this thesis to select algorithms and develop new algorithms for technical systems.

The contributions of this thesis are both descriptive and solution based. Paper A is a mapping study that outlines various algorithms for automatic labeling and enhancing the performance of supervised learning by incorporating unlabeled data. It also presents the algorithms used for different applications. The paper's contribution is both descriptive and solution-based, as it describes the labeling problem and presents various solutions that guide practitioners in designing solutions to address the issue of missing labels in machine learning. Paper B is also descriptive knowledge and solution-based because it investigates current challenges and mitigation strategies for labeling problems and provides improved solutions based on data collected from Paper A. Papers C and D are empirical evaluations of graph-based algorithms for automatic labeling and deep semi-supervised learning algorithms for comparing the performance of these algorithms in various scenarios. The contributions are therefore solution-based, as they help practitioners choose the optimal algorithm for their specific use case. Papers E and F are empirical evaluations of datasets used for evaluating graph- Paper G is a solution-based approach, as it proposes a deep semi-supervised learning algorithm for a specific industry dataset.

A *research strategy* is a collection of research methods [113]. A variety of research strategies were used to collect the data utilized in this thesis and all the studies contained in this thesis are based on empirical data. There are four quadrants of research strategies provided by [113]: *lab strategies, field strategies, respondent strategies* and *data strategies*. All the research strategies have their strengths and weaknesses. The research quality criteria are: *generalizability* of the evidence over the population of human system actors studied, *realism* of the context where the evidence was collected and needs to apply, *control* of extraneous human behavior variables that may impact the evidence being collected, and *precision* of the system data that is collected as evidence.

Paper A studies algorithms used by practitioners from many application domains and is therefore considered a respondent strategy. It provides high generalizability because many algorithms are studied across multiple application domains, but it lacks realism because the information provided by practitioners may be influenced by factors that are not observable.

Paper B reports on an interview study and an internship. An interview study is a type of respondent strategy known as a *sample survey*, and an internship is a type of field strategy, called a *field study*. To increase generalizability, the questions were formulated to be domain-independent. Many participants who work on labeling problems were included, and they came from companies with varying levels of labeling experience. The sample survey lacks realism for the same reason as Paper A.

Papers C-G utilize data strategies since they involve *computer simulation* [113]. More specifically, data is generated or simulated from an experimental simulation. Papers C, D, and G utilize data that the authors themselves generated from algorithms evaluated on various datasets. Paper E and F, on the other hand, simulate data based on results obtained by researchers from Microsoft. Papers C and E provide high generalization because many datasets from different domains are used. At the same time, Papers D and F may lack generalizability because the results may differ for other algorithms. Paper G proposes an algorithm that is only evaluated on one dataset and is therefore not guaranteed to be generalized to other datasets. Human behavior is not accounted for in computer simulations, and the precision of the data provided by the algorithms is assumed to be optimal. The Bayesian tools that evaluate the choice of prior distributions guarantee that realism is high in all simulations.

Summary

The research methods utilized in this thesis are chosen to complement each other. In Paper B, a case study with industry practitioners is performed to identify data labeling challenges and current mitigation strategies. In parallel with Paper B, a systematic mapping study was conducted in Paper A to identify algorithms that solve labeling-related problems faced by practitioners. The results of Paper A helped formulate mitigation strategies for the challenges identified in Paper B. The most popular algorithms and datasets found during the mapping study were evaluated in Papers C and D, respectively. DSSL algorithms and datasets were evaluated in Paper E and F. DSSL algorithms were not included in Paper A to reduce the scope but were still evaluated due to the recent interest in DSSL. The benchmark studies show practitioners what algorithms to select for their labeling scenario. In addition, practitioners know what datasets are optimal for evaluating algorithms. In

other words, practitioners know what datasets to include in a benchmark study so that the results generalize to other datasets. The results demonstrate that DSSL algorithms are more generalizable than GBSSL. Lastly, a case study comparing the performance of DSSL to DSL on a real-world Doppler Radar dataset was performed to illustrate the advantages of utilizing DSSL in a real-world application with a dataset collected from industry. The DSSL algorithm was developed based on the algorithms in USB which demonstrates that benchmark DSSL algorithms outperform DSL in Drone detection.

Figure 3.1 illustrates how each paper addresses the research questions, and Figure 3.2 illustrates the complete timeline of the research activities and their outputs.

Paper	Research Method	Comments
A	Systematic Mapping Study	Reviewing literature to find algorithms, datasets and application domains.
B	Case Study	Data collected from an internship with Company A and interviews with Company A and B.
C	Benchmark Simulation Study	Benchmark simulations to find optimal algorithms based on three dimensions.
D	Benchmark Simulation Study	Benchmark simulations to investigate what datasets are suitable for the evaluation of GSSL algorithms
E	Benchmark Simulation Study	Benchmark simulations to find optimal algorithms based on three dimensions.
F	Benchmark Simulation Study	Benchmark simulations to investigate what datasets are suitable for evaluation of DSSL algorithms
G	Case Study	Develop and deploy DSSL algorithm to demonstrate the pros and cons of utilizing unlabeled data on a real-world Doppler-Radar dataset.

Table 3.2: Summary of the research strategies used in the included papers.

3.4 Data Analysis

Thematic Analysis

Thematic analysis is a flexible method for identifying, analyzing and reporting patterns within data [114]. Thematic Analysis is therefore used in Paper B to

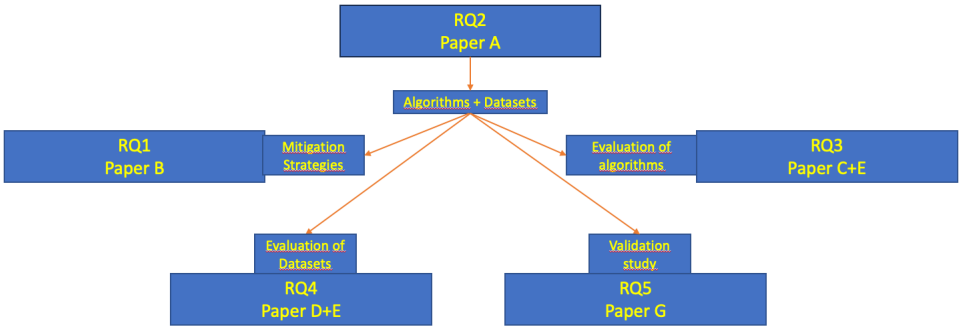
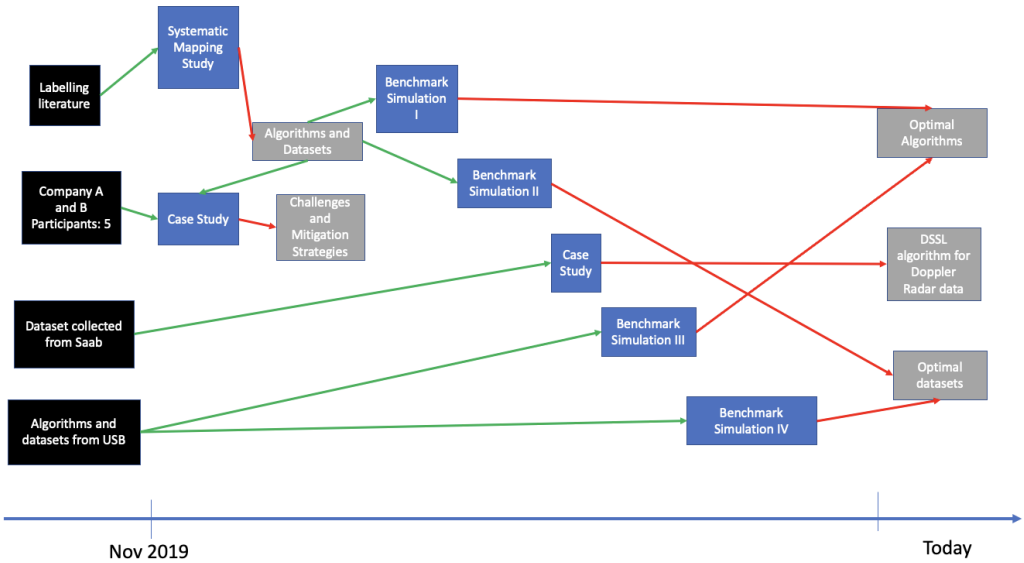


Figure 3.1: An overview of how each paper contributed to the research questions

Figure 3.2: Timeline for research activities. The black boxes contain the input data and the grey boxes illustrates the outputs. Green arrows point the study where the the data serves as input. Red arrows point to the outputs of the studies.



identify and analyze patterns in the data collected from the exploration and validation phases of the study.

In [114], thematic analysis is presented in six phases. In Phase 1 the researchers get familiar with the data. Data from interviews must be transcribed to perform a thematic analysis. In Phase 2 codes are found from the data by identifying interesting key features. The coding can be done manually by analyzing text using pens to highlight as many patterns as possible. In phase 3 the researchers start analyzing the codes and investigate how these codes combine to shape a theme. Codes form main themes and others will make sub-themes. Once candidate themes and sub-themes are defined, phase 4 will be to review the themes. Some candidate themes will be discarded, and some will be merged or divided into separate themes. If the themes form a coherent pattern, consider the validity of individual themes in relation to the dataset. Reread the dataset to make sure the themes work with respect to the dataset and perform re-coding. In Phase 5, the themes are defined and named by identifying the essence of each theme by deciding what viewpoint of the data the theme considers. Be careful not to make a theme too wide, complex, or overlapping. In phase 6 the researchers report results, perform the final analysis and write the report.

The analysis is not linear, and the phases are to be processed in a recursive process where it is allowed to jump back and forward between the phases. Furthermore, the phases are to be viewed as guidelines rather than rules.

Bayesian Data Analysis

Statistical inference allows for the interpretation and analysis of past and future of phenomena using probabilistic modeling. In classical statistics, known as frequentist statistics, observed data is viewed as a random variable X with a distribution $P_\theta(x)$.

Observations from X are independent and identically distributed (i.i.d), meaning that $x_1, \ldots, x_n \sim P_\theta(x)$, and $\mathrm{Cov}(X_i, X_j) = 0$ for all $i \neq j$. The data is represented by the density $f(x|\theta)$ and after the data is observed, the density is a function of θ and is called the *likelihood* function $\ell(\theta|x)$. Utilizing statistical inference, it is possible to obtain estimators of parameters θ. A *Point estimator* is an estimator that may be expressed as a function of X:

$$\widehat{\theta} = g(X). \tag{3.1}$$

Another type of estimator is the *maximum likelihood estimator* (MLE), which is defined as the parameter value that maximizes the likelihood function:

$$\widehat{\theta} = \arg\max_{\theta} \ell(\theta|x). \tag{3.2}$$

The frequentist paradigm treats the parameter as one fixed number in the parameter space. However in the Bayesian paradigm, all unknowns are treated as random variables. In particular, the distribution of θ is called the *prior distribution*, and while frequentist paradigm relies on the likelihood function, the Bayesian paradigm relies the posterior probability. The posterior probability density is calculated utilizing Bayes' formula [115]:

$$f(\theta|x) = \frac{f(x|\theta)f(\theta)}{f(x)}, \tag{3.3}$$

where $f(x)$ is the marginal density:

$$f(x) = \int f(x|\theta)f(\theta)\,d\theta. \tag{3.4}$$

The *maximum a posteriori estimator* (MAP) is defined as :

$$\widehat{\theta}_{MAP} = \arg\max_{\theta} f(\theta|x), \tag{3.5}$$

The MAP estimator is the parameter value θ that maximizes the posterior distribution. MAP estimators are utilized to derive Bayesian estimators.

Bayesian modeling calculates posterior distributions utilizing Markov chain Monte Carlo (MCMC) simulations. In order for MCMC to converge, the appropriate prior distribution must be selected. *Prior* and *posterior predictive checks* may be utilized to ensure that the appropriate priors are selected and that the posterior is calculated correctly. In addition, practitioners may utilize *the number of effective samples, the number of divergent iterations* and $\widehat{R}$ to assess the validity of the model [116].

Bayesian data analysis is utilized due to its many advantages [110], [111] and is recommended for empirical software engineering [112]. In addition, Bayesian tools for evaluating results are more intuitive than p-values and test statistics [110]–[112].

Generalized Linear Model

Generalized linear models (GLM) [117] generalize linear models such as linear regression to account for non-normal responses. A GLM consists of three components: a *random component*, a *systematic component* and a *link function*. A GLM may be constructed from an exponential family:

$$f(x|\theta) = c(\theta)h(x)\exp\Big(x \cdot R(\theta)\Big), \tag{3.6}$$

for any response variable. The parameter $\eta = R(\theta)$ is called the *natural* parameter. The link function $g(\cdot)$ is monotonic, differentiable and links the random and systematic components:

$$\eta_i = g(\mu_i), \tag{3.7}$$
$$\mu_i = E(X_i). \tag{3.8}$$

For the case of linear regression, $g(\mu) = \mu$ and $\eta = \sum_j \beta_j x_j$.

Generalized Linear Mixed Model

Generalized Linear mixed models (GLMM) [117] are extensions of GLMs that include random effects. It is assumed that the data may be divided into M clusters where β denotes the fixed effects parameters and u_i denotes the random effects for cluster i. A GLMM conditional on the random effects u_i is defined as:

$$g(\mu_{ic}) = x_{i,c}^T\beta + z_{i,c}^T u_i. \tag{3.9}$$

In Papers D and F, a GLMM is utilized to calculate the probability of achieving a certain accuracy. The data is Binomially distributed and is clustered with respect to the datasets.

In Paper G, a linear regression model is utilized to calculate the relative improvement of SSL to SL. All the techniques above have previously been utilized to evaluate other algorithms [118].

The Bradley-Terry model

The Bradley-Terry model compares binary data [119], [120]. For example, it may be used to determine which team will win a basketball game. Let the probability that a is preferred over b be denoted $P(a \text{ over } b) = \pi_{ab}$. Ties are not allowed so $\pi_{ab} + \pi_{ba} = 1$. The Bradley-Terry model is defined as:

$$\log \frac{\pi_{ab}}{\pi_{ba}} = \beta_a - \beta_a, \tag{3.10}$$

or equivalently:

$$\pi_{ab} = \frac{\exp(\beta_a)}{\exp(\beta_a) + \exp(\beta_b)}. \tag{3.11}$$

The parameters β are called *strength parameters* and are utilized for ranking. In Paper C and E, the Bradley Terry ranks the algorithms according to highest strength parameter. The Bradley-Terry model have previously been utilized for ranking of other types of algorithms [116].

Item-Response Theory

The development of test theories began in 1904 with Spearman [121] and Binet [122]. IRT consists of techniques that evaluate how well latent variables are estimated. The development of IRT started when the city of Paris asked Binet to invent a method to evaluate whether students had mental illnesses. Binet achieved this task by designing a test that measured students' intelligence. Thurston later proved empirically that the scale for intelligence is normally distributed [123]. The 2PL model provided by Lord [124] and Rasch [125] assumes that the responses from the test takers are normally distributed and depend on the ability of the test taker to answer the items. The probability of correctly answering an item depends on the item difficulty and discrimination. Discrimination measures the association between items and latent variables. Difficulty measures how many students will answer the item incorrectly. Lord and Rasch consider discrete responses, but Paper D and F require continuous responses. The Jöreskog model assumes continuous responses and at least one latent variable [126]. The responses follow a normal distribution and are

conditionally independent [126]:

$$P(U_{i1} \cdot \ldots \cdot U_{ip}|\Phi) = P(U_{i1}|\Phi) \cdot \ldots \cdot P(U_{ip}|\Phi), \tag{3.12}$$

and:

$$\mathbb{E}(U_{pi}|\Phi) = \tau_{pi} = a^T \omega_p. \tag{3.13}$$

The Jöreskog model is utilized in Papers D and F to investigate the ability of the algorithms to learn the datasets.

3.5 Threats to Validity

Four threats to validity are presented for case studies [105] and simulation studies [104]. These are called: *construction validity, external validity, internal validity* and *conclusion validity.*

Construction Validity

Construction validity refers to how well the study is constructed to answer the research questions. For Paper B, construction validity was achieved by introducing the participants to the interview questions before the interviews. This way, they were able to prepare in advance and ask the authors to clarify certain questions. In addition, a lecture where the participants were introduced to data labeling-related terminology and algorithms was held.

External Validity

This type of validity assesses how general the results are and whether they may be extended to other domains. In Paper B, none of the interview questions are domain-specific to ensure that the study focused on solving general data labeling problems. Papers C-F included many datasets to ensure the results are generalizable. Paper G demonstrates that SSL may outperform SL on Doppler radar data and the algorithm was not designed to be generalizable to other datasets.

Internal Validity

Internal Validity is damaged when researchers neglect to account for factors that unexpectedly affect the outcome. In Paper B, data triangulation [127] is utilized to validate the responses from different participants, guaranteeing that all relevant data were collected and the correct conclusions were made.

Conclusion Validity

Conclusion validity refers to the choices and assumptions made in statistical models. As previously mentioned, all statistical models were constructed to address the problems based on empirical evidence. Many statistical models can be extended, for example, the Bradley-Terry model may be modified to account for ties, and the Jöreskog model may be adjusted to include more latent variables. This does not imply that more advanced models are superior and including more variables means that simulations will be more time-consuming and may not yield better results. The advantage of the Bayesian paradigm is that it offers tools such as prior and posterior checks, trace plots, and descriptive statistics like $\widehat{R}$, and number of effective samples. Papers C-G utilized these tools to validate the statistical models.

The Who-What-How framework proposed by [1] may be used to describe who benefits from the research, what the research contribution is, and how the results were obtained.

Table 3.3: Summary table for the datasets. From left to right, the columns contain the datatype, the name of the dataset, the number of labels utilized for each class, the size of the training dataset, the size of the testing dataset and the number of classes in the dataset.

Datatype	Dataset	Labels (Small/Large)	Training data	Test data	Classes
Image	Cifar-100	2 / 4	50000	10000	100
	STL-10	2 / 4	50000	10000	100
	EuroSat	4 / 10	5000 / 10000	8000	10
	TissueMNIST	10 / 50	165466	47280	8
	Semi-Aves	15/53	5959 / 26640	4000	200
Text	IMDB	10/50	23000	25000	2
	Amazon Review	50 / 200	250000	65000	5
	Yelp Review	50/ 200	250000	50000	5
	AG News	10 / 50	100000	7600	4
	Yahoo! Answers	50 / 200	500000	60000	10
Sound	Keyword Spotting	5 / 20	18538	2567	10
	ESC-50	5 / 10	1200	400	50
	UrbanSound8K	10 / 40	7079	837	10
	FZDnoisy	52/171	1772 / 15813	947	20
	GTZAN	10 / 40	7000	1500	10

CHAPTER 4

Summary of included papers

This chapter provides a summary of the included papers.

4.1 Paper A

Teodor Fredriksson, David Issa Mattos, Jan Bosch, Helena Holmström Olsson
Machine Learning Algorithms for Labeling: Where and How They are Used?
.

Supervised learning is a popular tool for solving classification tasks but require labeled data. In industry, datasets may contain few labels or no labels at all. Therefore data labeling is essential and many companies still perform manual labeling. Manual labeling is an expensive and time-consuming task. Therefore companies are looking to find automated tool to obtain labels such as active learning and semi-supervised learning. Active learning help practitioners reduce manual labeling by selecting the most informative instances to be labeled. Transductive semi-supervised learning will label data auto-

matically and inductive semi-supervised learning many improve supervised learning utilizing unlabeled data and therefore eliminates the need to label data altogether. This study presents a systematic mapping study to investigate state-of-the-art active and semi-supervised learning utilized in practice. In addition to algorithms, the study investigates the application domains for the algorithms and the datasets utilized to evaluate the algorithms. Based on the identified algorithms, application domains and datasets, a taxonomy is created to help practitioners to choose the optimal algorithm for their use-case and select datasets for evaluating their algorithm.

4.2 Paper B

Teodor Fredriksson, David Issa Mattos, Jan Bosch, Helena Holmström Olsson
Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies
.

Planning the allocation of data labeling is essential in order to save time and resources. Data labeling is possible through crowdsourcing and in-house labeling. In-house labeling allows companies to let qualified personnel perform labeling. Data scientists and software engineers have in many cases good knowledge about company data and are therefore good candidates for labeling. Their strong knowledge of data guarantees high-quality labels. On the other hand data scientists and software engineers have other more specialized task to perform and having them perform labeling is a waste of resources. The alternative is to find new personnel and train them in order to produce high-quality labels. This approach may be costly in terms of time and resources. Crowdsourcing allows anyone to perform labeling through platforms such as Amazon Mechanical Turk. Thereby companies need not hire personnel for labeling. However, it is difficult to guarantee the quality of the labels, and companies that need to label confidential data are unable to share such data through labeling services. Therefore, to guarantee quality and safety of data companies still practice in-house labeling. This study reports data collected from an internship with industry and semi-structured interviews performed with practitioners from two companies. The collected data is summarized into three challenges that companies face during labeling. Based on the data

collected and the results of **Paper A**, mitigation strategies are formulated for every challenge.

4.3 Paper C

> **Teodor Fredriksson**, Jan Bosch, Helena Holmström Olsson
> An Empirical Evaluation of Graph-Based Semi-Supervised Learning Algorithms
> .

According to **Paper A**, graph-based semi-supervised learning algorithms are popular tools for automatic labeling of data. This study reports on a simulation study where thirteen graph-based semi-supervised learning algorithms are compared across three datatypes, image, text and numeric. The purpose of the simulations is to help practitioners find the optimal algorithm for their use-case. The results of the simulations are analyzed with respect to three dimensions. *Performance*: To measure how accurately the algorithms label data. *Effort:* To measure how much manual effort is required to achieve optimal accuracy. *Datatype:* To investigate if algorithms perform differently in different datatypes. In addition, noise is added to better reflect real-world scenarios. The Bayesian Bradley-Terry model is utilized to rank the algorithms and the generalized linear mixed model is utilized to calculate the probability of reaching a certain accuracy.

4.4 Paper D

> **Teodor Fredriksson**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson
> Assessing the Suitability of Semi-Supervised Learning Datasets using Item Response Theory
> .

A key observation from **Paper C** is that the algorithms achieve high performance in the presence on many datasets, especially image datasets. This observation indicate that the datasets are unsuitable for evaluating graph-based semi-supervised learning algorithms. This study utilizes *Item Response*

Theory to assess the *ability* of the algorithms to learn a dataset, the *difficulty* of learning each dataset, and the *discrimination* of the dataset.

4.5 Paper E

Teodor Fredriksson, Jan Bosch, Helena Holmström Olsson
An Empirical Evaluation of Deep Semi-Supervised Learning
.

Papers A, C and **D** does not consider deep learning. The research interest in deep semi-supervised learning has increased in the past years due to the success of FixMatch. This study evaluates the universal semi-supervised learning benchmark (USB). USB is an open-source Python library containing 16 deep semi-supervised learning algorithms evaluated on 15 datasets across three datatypes, images, text and audio. This study analyzes the same dimensions as **Paper C** and utilizes the same methodology to answer the same research questions as **Paper C**.

4.6 Paper F

Teodor Fredriksson, Jan Bosch, Helena Holmström Olsson
Assessing the Sustainability of Deep Semi-Supervised Learning Datasets using Item Response Theory
.

Based on the results of **Paper E**, many USB algorithms achieve low errors (although not lower than 10%). The results of **Papers C** and **D** demonstrate that GBSSL algorithms reach accuracy above 90%. Still, many datasets in the evaluation are unsuitable because they are too easy for the algorithms to learn. Therefore, these algorithms may perform differently on real-world datasets. Therefore, it is important to use the same methodology as **Paper D** and apply it to the datasets utilized in **Paper E** to investigate whether the USB algorithms are fairly evaluated.

4.7 Paper G

Teodor Fredriksson, Jan Bosch, Helena Holmström Olsson
Classification of Complex-Valued Radar Data using Semi-Supervised Learning: a Case Study
.

This paper, investigates whether a DSSL algorithm improves the accuracy of a DSL algorithm applied to a real-world radar dataset from our industry collaborator Saab. The dataset is utilized to detect drones based on radar signals. The algorithm utilized in this study is based on FixMatch, pseudo-labeling and consistency regularization wtih strong and weak augmentation. The performance and computation speeds are compared between supervised and semi-supervised learning to investigate the pros and cons of utilizing unlabeled data to improve performance.

CHAPTER 5

Machine Learning Algorithms for Labeling: Where and How They are Used?

5.1 Introduction

In software-intensive companies in the online and the embedded systems domain, vast sets of data are being processed and labeled manually [128]. Manual labeling is an expensive approach for a company, but it allows easy maintenance of the data's quality. One of the downsides is that the task is tedious and time-consuming for highly qualified professionals, often leading to prohibitively expensive costs. Data labeling is a way of annotating data depending on the data's content [128]. The labels each data instance receives are decided after information about the entry has been processed. Research in machine learning and artificial intelligence led to the development of multiple algorithms for fully automating (semi-supervised learning) or assisting humans in labeling the data (active learning).

Semi-supervised learning is a set of machine learning algorithms used when the majority of instances are unlabeled. The semi-supervised classification objective is to train a classifier on both unlabeled and labeled data. This

classifier is used to label the instances of the dataset that are not labeled [129].

Active learning is a machine learning framework that utilizes several labeled instances to query an oracle (often a human) to label some desired instances. Active learning is used to assist humans in selecting a smaller subset of the best instances to label [23].

In this paper, we conducted a systematic mapping study [106] to identify the state-of-the-art literature on machine learning algorithms that are used for assisted or automatically labeling and where they are used.

This paper provides three main contributions. First, we identify the machine learning algorithms for labeling. We present a taxonomy of the algorithms. Second, we identify the datasets that are used to evaluate the algorithms. We create a classification scheme for the datasets based on the type of data and the application area. Third, we present guidance to industry practitioners on optimally getting their data labeled and using labeled datasets for machine learning and data labeling practices. The results presented in this paper can be used by both researchers and practitioners to find missing labels with the aid of machine algorithms or to select appropriated datasets to compare new state-of-the-art algorithms in their respective application areas.

The remainder of this paper is organized as follows. Section 5.2 provides an overview of related work and presents key concepts used in semi-supervised and active machine learning algorithms. In section 5.4 we provide a concise description of the problem that we seek to address in this paper, followed by an overview of our research method. We present the systematic literature mapping results in section 5.5 and discuss these in section 5.6. Finally, we end the paper with an overview of open research questions and a conclusion in section 5.7.

Many studies were included in this paper. To preserve space, the references to the papers containing the algorithms and datasets were moved from the paper into a separate reference list that uses a different referencing style. The list of references for algorithms and datasets can be found in an online appendix: https://github.com/teodorf-bit/Systematic-Mapping-Study/blob/main/SMS_SYSCON.pdf.

5.2 Background

Supervised learning is used for classification and regression tasks. Supervised classification requires that each instance in a dataset is associated with a label. In practice, this becomes a problem as they have large amounts of data, but the data is often incomplete as labels are missing partially or entirely. If labels are available, these can still be of lousy quality, affecting a model's performance. Hence, acquiring labels of high quality is paramount to train high-accuracy models.

The most popular strategy for achieving labels with human supervision is crowdsourcing [130]. However, a problem with crowdsourcing is that it requires third-party companies to access sensitive and confidential data. Another problem is when the label distribution is skewed. Imbalanced label distributions tend to make machine learning perform poorly [131].

The distribution independent model of concept learning (supervised learning), known as PAC-learning, a framework for mathematical analysis machine learning, was introduced by Leslie Valiant in [132]. The PAC learning framework was later extended for semi-supervised learning in [129].

The survey [133] introduce researchers and practitioners to the main semi-supervised learning algorithm, such as Graph-based algorithms, Mixture models and EM, self-training, co-training, and multi-view learning. Surveys on active learning such as [134] introduce the three main approaches to active learning. Membership queries, as well as pool-based and stream-based active learning. Furthermore, they define query strategies such as uncertainty sampling, query-by committee, error reduction, and variance strategies. Issues with regards to skewed label distributions, unreliable oracles, and costs associated with labeling are discussed as well.

Semi-supervised learning

Semi-supervised learning is a set of machine learning algorithms that can be used if most instances are unlabeled, but a small subset of them has labels. In technical terms, we have access to a set of data points that can be divided into two disjoint subsets, one containing the labeled instances and the other containing the unlabeled instances. The objective of semi-supervised classification is to train a classifier on both unlabeled and labeled data so that it is better than a supervised classifier trained only on the labeled data.

Co-training and multi-view learning Co-training was first used to classify web pages [135], [136], and it was shown that co-training could improve classification accuracy. Furthermore, a PAC generalization bound was created in [137]. Co-training assumes that the features can be divided into two sets. We say that the features have at least two views. The corresponding labels are then predicted using both of the views using the co-training algorithm. The algorithm takes both labeled and unlabeled data as input, as well as a learning speed k. The goal is to train two classifiers. The first classifier is trained on view 1 and the second classifier on view 2. Furthermore, we must assume that the classifiers alone have a high classification accuracy and that the two views must be conditionally independent given the class label. Both training sets consist initially of the same labeled data, $L_1 = L_2$. Then train the first classifier on L_1 and the second on L_2. Classify the remaining unlabeled data with classifier one and classifier two separately. Take the k most-confident predictions of classifier 1 and add them to L_2 and k most-confident predictions of the second classifier and add them to L_1. Remove these instances from the unlabeled data. Repeat this procedure until we are out of unlabeled data. Multi-view learning was first used in [138] and generalized co-training to n number of views.

Mixture models and the EM-algorithm A (Generative) Mixture Model (MM) is a weighted sum of densities from M components

In the supervised setting, the parameters of the densities are calculated using maximum likelihood estimation (MLE). Using MLE in the supervised setting is straightforward. However, in the semi-supervised setting, we must utilize the unlabeled data as well. Hence, we have to solve a different optimization problem to find the parameters of the densities.

The missing labels are referred to as hidden variables and make the log-likelihood difficult to optimize. To optimize the log-likelihood, the Expectation-Maximization (EM) [139] algorithm is used.

Theoretical aspects of mixture models have been justified in [140], [141]. The only assumption needed for mixture models is that data comes from a mixture model. This assumption is difficult to assess if the labels are scarce. However, it is usually assessed by domain knowledge or mathematical convenience. If the assumption is violated, then the unlabeled data will worsen the accuracy of the predicted labels [142]. Another issue with generative mixture

models is that the model that describes the unlabeled data must be unique.

Semi-supervised Support Vector Machines (S3VM) models Supervised support vector machines (SVM) strive to classify instances by creating a decision boundary. Such a decision boundary is found by solving an optimization problem. If we have unlabeled instances, then there is no way of knowing whether the unlabeled instance is put on the right side of the decision boundary. S3VM strives to mitigate this issue by incorporating a loss function on the unlabeled data to the SVM objective function. Furthermore, S3VM assumes that there is a low-density region that separates the labels. If such a region does not exist, S3VMs might not perform as well as expected. S3VMs or TSVM (Transductive SVMs), as these were originally called, were first introduced in [143].

Graph-based models Graph-based methods construct graphs from the training data that consist of labeled and unlabeled instances. These instances constitute the graphs' vertices, which means that the more unlabeled data, the bigger the graph will become. The learning procedure will then result in labels assigned to the vertices. There are two types of graph constructing algorithms.

Task-independent algorithms do not use labeled data and are hence unsupervised. Popular methods include k-NN,ϵ-neighborhood, b-matching [144] and, hard and soft α-graphs [145].

Task-dependent algorithms do use labeled data such as Inference-drive Metric Learning [146] and Kernel-alignment based spectral kernel design [147].

The next step is to inject seeds and infer labels on the unlabeled data. These algorithms are divided into Transductive and Inductive methods. The goal of transductive algorithms is to predict labels only for the unlabeled data. These algorithms include Graph cut [148], Gaussian random fields (GRF)[94], Local and Global consistency (LGC)[95], Adsorption [149], Modified Adsorption (MAD)[150], Quadratic Criteria (QC) [92], Transduction and Confidence (TACO) [151], Information Regularization [152]–[154] and Measurement Propagation (MP)[86]. Inductive learning estimates a function that can be applied to new data instances. Inductive algorithms are few [130], an example of an inductive algorithm is Manifold regularization. [155].

Active learning

Historically machine learning algorithms usually try to fit a model according to currently labeled data, and we refer to these models as "passive" learning models. Active learning systems, on the other hand, create new models as they iterative learn. Similar to how a scientist plans several experiments to conclude a hypothesis, an active learning method imposes query strategies to help select the most informative examples to be labeled by an oracle.

In some cases, an active learning system might not be optimal if the model does not require a considerable number of labels. Instead, use it when there is a massive set of unlabeled examples, and there is a need to label a massive amount of data to train the system.

If active learning is appropriate, then we need to specify in what way we want to query the examples [23]. The three most common scenarios are:

1. *Query synthesis:* This scenario allows the learner to request labels for any unlabeled example. Query synthesis also works for instances the learner itself has generated. All that is required is knowledge about how the inputs are constructed. Query synthesis is sometimes helpful, but in some cases, it is not reasonable to use. For example, in natural language processing, one might generate a text string that is in-comprehensive [23].

2. *Stream-based selective sampling:* Also known as stream-based sampling, this scenario involves sampling one sample at a time from the actual distribution, and then the learner should decide whether to query it or not[23].

3. *Pool-based sampling:* In many scenarios, an extensive set of unlabeled data must be processed at once, and this is where pool-based sampling is appropriate. The scenario involves having a large set of unlabeled examples and a small pool of labeled examples as well[23].

Uncertainty sampling and Density weighted methods models The uncertainty sampling approach [156] aims to select the instances that we are least certain about and label these. Labels that we are certain about will probably not contribute to informativeness. Two common strategies include *entropy* [59] and *least confident*[157]. These strategies are among the most popular

and work especially well for probabilistic models [27], [90], [157]–[159] but has also been successfully applied to non-probabilistic models [157], [160]–[163]. The downside is that they only consider information about the one prediction and not the entire distribution. Density-weighted methods strive to model the input distribution in the query strategy. Thus, not only querying by uncertainty but also by how representative each instance is. *Information density*[158] does exactly that, other similar strategies have also been proposed [160], [164]–[166].

Query-based models Query by Committee [167] involves a committee of classifiers. Each classifier is trained on the same training data. The *Version Space* is the set of classifiers that are consistent with the labeled training data. The smaller the version space is, the more confident we are about the version space classifiers. Therefore, a smaller version space means that we do not need to test many different classifiers to find the most accurate model. The goal of QBC is to minimize the version space. To produce a QBC algorithm, we first have to construct the committee of models representing the entire version space. Secondly, we need a measurement to determine disagreement between the committee members. Constructing the committee can be done by sampling a committee of random hypotheses [167]. When using generative models, this can be done by sampling models from some posterior distribution [43], [164]. For discriminate and non-probabilistic models, query-by-boosting and query-by-bagging are proposed [168]. Two common measurements for disagreement between committee members are *vote entropy*[43] and *KL divergence*[164].

In practice, QBC is relatively simple to implement and works with any basic model. The downsides are that these are difficult to maintain, and just like uncertainty sampling, it only looks at one instance at a time and does not consider the entire distribution.

5.3 Algorithms

This section describes the main sub-categories of algorithms based on the two categories: semi-supervised and active learning.

Semi-supervised learning

Semi-supervised learning is a set of machine learning algorithms that can be used if most instances are unlabeled, but a small subset of them has labels. In technical terms, we have access to a set of data points that can be divided into two disjoint subsets, one containing the labeled instances and the other containing the unlabeled instances. The objective of semi-supervised classification is to train a classifier on both unlabeled and labeled data so that it is better than a supervised classifier trained only on the labeled data.

Co-training and multi-view learning Co-training was first used to classify web pages [135], [136], and it was shown that co-training could improve classification accuracy. Furthermore, a PAC generalization bound was created in [137]. Co-training assumes that the features can be divided into two sets. We say that the features have at least two views. The corresponding labels are then predicted using both of the views using the co-training Arithm. The Arithm takes both labeled and unlabeled data as input, as well as a learning speed k. The goal is to train two classifiers. The first classifier is trained on view 1 and the second classifier on view 2. Furthermore, we must assume that the classifiers alone have a high classification accuracy and that the two views must be conditionally independent given the class label. Both training sets consist initially of the same labeled data, $L_1 = L_2$. Then train the first classifier on L_1 and the second on L_2. Classify the remaining unlabeled data with classifier one and classifier two separately. Take the k most-confident predictions of classifier 1 and add them to L_2 and k most-confident predictions of the second classifier and add them to L_1. Remove these instances from the unlabeled data. Repeat this procedure until we are out of unlabeled data. Multi-view learning was first used in [138] and generalized co-training to n number of views.

Mixture models and the EM-Arithm A (Generative) Mixture Model (MM) is a weighted sum of M component densities.

In the supervised setting, the parameters of the densities are calculated using maximum likelihood estimation (MLE). Using MLE in the supervised setting is straightforward. However, in the semi-supervised setting, we must utilize the unlabeled data as well. Hence, we have to solve a different optimization problem to find the parameters of the densities.

The missing labels are referred to as hidden variables and make the log-likelihood difficult to optimize. To optimize the log-likelihood, the Expectation-Maximization (EM) [139] Arithm is used.

Theoretical aspects of mixture models have been justified in [140], [141]. The only assumption needed for mixture models is that data comes from a mixture model. This assumption is difficult to assess if the labels are scarce. However, it is usually assessed by domain knowledge or mathematical convenience. If the assumption is violated, then the unlabeled data will worsen the accuracy of the predicted labels [142]. Another issue with generative mixture models is that the model that describes the unlabeled data must be unique.

Semi-supervised Support Vector Machines (S3VM) models Supervised support vector machines (SVM) strive to classify instances by creating a decision boundary. Such a decision boundary is found by solving an optimization problem. If we have unlabeled instances, then there is no way of knowing whether the unlabeled instance is put on the right side of the decision boundary. S3VM strives to mitigate this issue by incorporating a loss function on the unlabeled data to the SVM objective function. Furthermore, S3VM assumes that there is a low-density region that separates the labels. If such a region does not exist, S3VMs might not perform as well as expected. S3VMs or TSVM (Transductive SVMs), as these were originally called, were first introduced in [143].

Graph-based models Graph-based methods construct graphs from the training data that consist of labeled and unlabeled instances. These instances constitute the graphs' vertices, which means that the more unlabeled data, the bigger the graph will become. The learning procedure will then result in labels assigned to the vertices. There are two types of graph constructing algorithms.

Task-independent algorithms do not use labeled data and are hence unsupervised. Popular methods include k-NN,ϵ-neighborhood, b-matching [144] and, hard and soft α-graphs [145].

Task-dependent algorithms do use labeled data such as Inference-drive Metric Learning [146] and Kernel-alignment based spectral kernel design [147].

The next step is to inject seeds and infer labels on the unlabeled data. These algorithms are divided into Transductive and Inductive methods. The

goal of transductive algorithms is to predict labels only for the unlabeled data. These algorithms include Graph cut [148], Gaussian random fields (GRF)[94], Local and Global consistency (LGC)[95], Adsorption [149], Modified Adsorption (MAD)[150], Quadratic Criteria (QC) [92], Transduction and Confidence (TACO) [151], Information Regularization [152]–[154] and Measurement Propagation (MP)[86]. Inductive learning estimates a function that can be applied to new data instances. Inductive algorithms are few [130], an example of an inductive Arithm is Manifold regularization. [155].

Active learning

Historically machine learning algorithms usually try to fit a model according to currently labeled data, and we refer to these models as "passive" learning models. Active learning systems, on the other hand, create new models as they iterative learn. Similar to how a scientist plans several experiments to conclude a hypothesis, an active learning method imposes query strategies to help select the most informative examples to be labeled by an oracle.

In some cases, e.g., an active learning system might not be optimal if the model does not require a considerable number of labels. Instead, use it when there is a massive set of unlabeled examples, and there is a need to label a massive amount of data to train the system.

If active learning is appropriate, then we need to specify in what way we want to query the examples [23]. The three most common scenarios are:

1. *Query synthesis:* This scenario allows the learner to request labels for any unlabeled example. Query synthesis also works for instances the learner itself has generated. All that is required is knowledge about how the inputs are constructed. Query synthesis is sometimes helpful, but in some cases, it is not reasonable to use. For example, in natural language processing, one might generate a text string that is in-comprehensive [23].

2. *Stream-based selective sampling:* Also known as stream-based sampling, this scenario involves sampling one sample at a time from the actual distribution, and then the learner should decide whether to query it or not[23].

3. *Pool-based sampling:* In many scenarios, an extensive set of unlabeled data must be processed at once, and this is where pool-based sampling

is appropriate. The scenario involves having a large set of unlabeled examples and a small pool of labeled examples as well[23].

Uncertainty sampling and Density weighted methods models The uncertainty sampling approach [156] aims to select the instances that we are least certain about and label these. Labels that we are certain about will probably not contribute to informativeness. Two common strategies include *entropy* [59] and *least confident*[157]. These strategies are among the most popular and work especially well for probabilistic models [27], [90], [157]–[159] but has also been successfully applied to non-probabilistic models [157], [160]–[163]. The downside is that they only consider information about the one prediction and not the entire distribution. Density-weighted methods strive to model the input distribution in the query strategy. Thus, not only querying by uncertainty but also by how representative each instance is. *Information density*[158] does exactly that, other similar strategies have also been proposed [160], [164]–[166].

Query-based models Query by Committee [167] involves a committee of classifiers. Each classifier is trained on the same training data. The *Version Space* is the set of classifiers that are consistent with the labeled training data. The smaller the version space is, the more confident we are about the version space classifiers. Therefore, a smaller version space means that we do not need to test many different classifiers to find the most accurate model. The goal of QBC is to minimize the version space. To produce a QBC Arithm, we first have to construct the committee of models representing the entire version space. Secondly, we need a measurement to determine disagreement between the committee members. Constructing the committee can be done by sampling a committee of random hypotheses [167]. When using generative models, this can be done by sampling models from some posterior distribution [43], [164]. For discriminate and non-probabilistic models, query-by-boosting and query-by-bagging are proposed [168]. Two common measurements for disagreement between committee members are *vote entropy*[43] and *KL divergence*[164].

In practice, QBC is relatively simple to implement and works with any basic model. The downsides are that these are difficult to maintain, and just like uncertainty sampling, it only looks at one instance at a time and does not

consider the entire distribution.

5.4 Research Method

In order to reach the objectives of this paper, we conducted a systematic mapping study. Systematic mapping studies seek to identify, analyze and interpret all relevant research on a particular topic [106]. In this study, the topic of interest is automatic labeling in machine learning, and thus the purpose of this SMS is to find and analyze relevant literature on automatic labeling. We conducted this systematic mapping study according to [169]. The procedure consists of four steps:

1. Definition of research questions.
2. Identification of search terms and searching for papers.
3. Screen of papers based on inclusion and exclusion criteria.
4. Data extraction and mapping.

We detail each step below.

Definition of research questions

The purpose of this paper is to provide a systematic overview of the existing research on automatic labeling of data using machine learning algorithms. This paper aims to examine previous research and explore the possibility of contributing to new research.

Three research questions are defined below:

RQ1: What types of machine learning algorithms are used for assisted or automatically labeling?

RQ2: What are the datasets used to evaluate these algorithms?

RQ3: What algorithm(s) should be used based on application?

Identification of search terms and conducting search

To source relevant studies, we utilized a keyword-based database search. The main search string was constructed iteratively. At first, we used keywords such as "automatic labeling," but it gave a too large variety of algorithms specific to a particular type of application. We then changed the keywords to methods based on active machine learning and semi-supervised learning since these algorithms are applied to various applications.

To find papers based on active learning, we searched for "active machine learning" + "*<CATEGORY OF ACTIVE LEARNING>*". If we dismissed the "machine" in the string, we would get results related to "education". Similarly, for semi-supervised learning," we searched for "semi-supervised learning " + "*<CATEGORY OF SEMI-SUPERVISED LEARNING>*". The active learning categories found were Uncertainty, Query by committee, Error and Variance Reduction, and Density-weighted algorithms. The semi-supervised categories found were Co-training and Multi-view learning, EM-algorithms and Mixture models, Semi-supervised Support Vector Machines, and Graph-based semi-supervised learning.

We used Google Scholar (https://scholar.google.com) as the source where we applied our search string. There are three reasons as to why. The first reason is that we expected many relevant articles from the search because the search terms are so general. Secondly, Google Scholar is perceived as an unbiased source [107]. The third reason is that Google scholar includes papers from arxiv, where many research papers on machine learning are submitted.

The theoretical considerations of machine learning started in the 1980s and the first computations on machine learning algorithms did not start until the 2000s. Therefore all papers between 1980 and 1999 should concern theoretical aspect of machine learning and are unnecessary to include in our study. Hence, only paper between 2000 and 2020 will be included.

The search strings were applied in December 2020 to the selected electronic database to retrieve articles that include the keywords in their title, abstracts, and instructions. The retrieval stopped after the abstracts and introductions became less relevant to avoid an infinite number of papers. In the end, 312 articles were retrieved for further screening and processing of inclusion and exclusion criteria.

Screening of papers on the basis of inclusion and exclusion criteria.

All retrieved studies were examined for inclusion and exclusion based on pre-established criteria. The exclusion and inclusion criteria considered in our study are presented below:

Inclusion Criteria

- Papers that includes AL/SSL techniques for labeling unlabeled and or partially unlabeled data form the industry.
- Papers that compare several AL/SSL techniques with each other.
- Papers that include a hybrid between AL/SSL learning.
- Papers that compare AL/SSL techniques with other non-AL/SSL methods.
- Papers that has a title that describes the application.

Exclusion Criteria

- Papers concerning theoretical proofs of AL/SSL methods.
- Papers concerning simulation studies.
- Absence of industrial validation.

Data Extraction and Analysis

Data extraction involved the collection of information related to the RQs of the study. For each paper, we identified the research field, what kind of datatypes, and what method the paper focused on.

5.5 Results

In this section, we provide the results of the systematic mapping study concerning the research questions proposed in section 5.4. First, we provide a list of the algorithms we found from analyzing the papers. Second, we provide a classification of the datasets used in the algorithm evaluation. Finally, we present what type of algorithm should be used for specific applications.

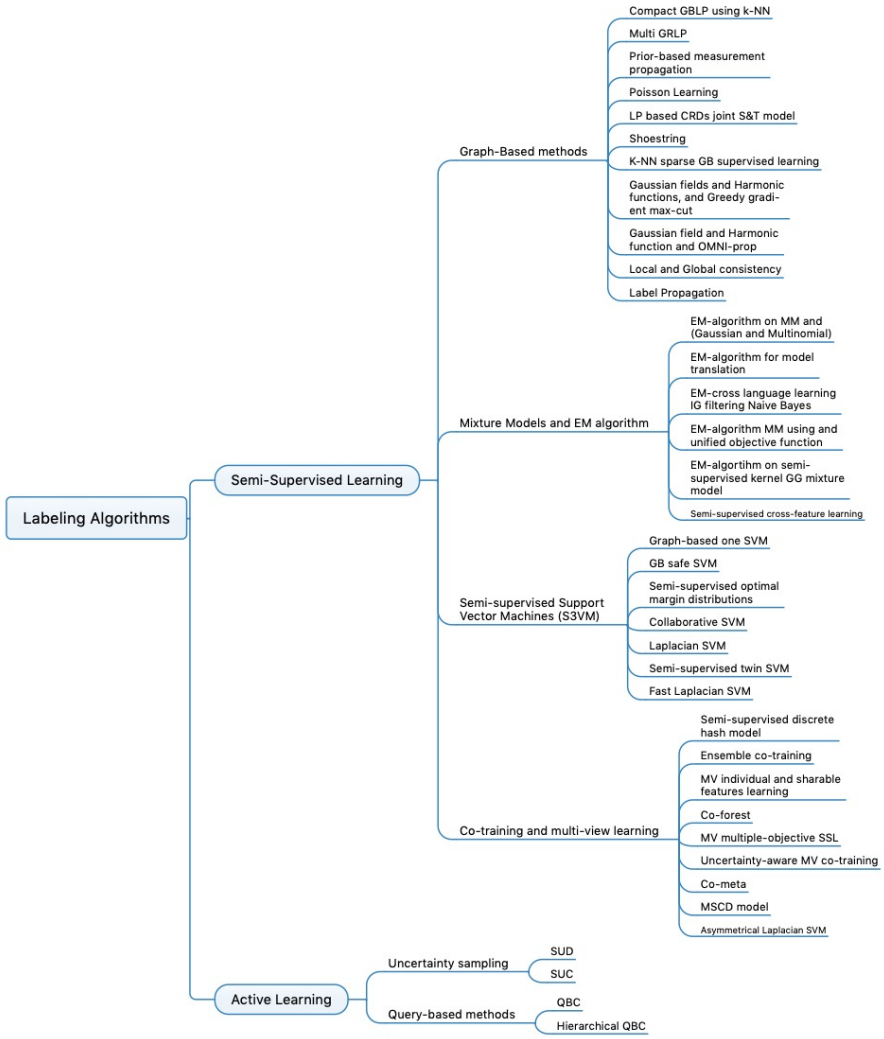


Figure 5.1: Tree diagram illustrating the different types of algorithms in our taxonomy.

RQ1: What types of machine learning algorithms are used for assisted for autimatic labeling?

Co-training and multi-view learning The following were algorithms found: Semi-supervised discrete hash model (SSNDH) [170], Ensemble co-training (En-Co-training) [171], Multi-view individual and shareable features learning [172], Co-forest [173], Multi-view multiple-objective SSL [174], Uncertainty-aware multi-view co-training [175], Co-meta [176], the MSCD model [177] and Semi-supervised cross-feature learning [178].

Mixture models and the EM-algorithm The following algorithms were found in the literature: EM-algorithms on mixture modes (Gaussian, Multinomial), [179], [180], EM-algorithm for model translation [181], EM-cross language learning using IG filering Naive Bayes [181], EM-algorithm on mixture model using an unified objective function [182], EM-algorithm on semi-supervised kernel GG mixture model [183].

Semi-Supervised Support Vector Machines (S3VM) models We found the following algorithms: Graph-based one class support vector machine (OC-SSVM)[184], Graph-based safe support vector machines [185], Semi-supervised optimal margin distributions (ssODM)[186], Collaborative Support Vector Machines (ColSVM) [187], Laplacian Support Vector Machine (LapSVM) [188], Semi-Supervised Twin Support Vector Machine (TWSVM) [189], Fast Laplacian twin Support Vector Machines (FLapTWSVM)[190] and Asymmetric Laplacian Support Vector Machines (AsyLapSVM)[191].

Graph-based models The following algorithms were found: Compact graph-based label propagation using k-NN [192], Multi graph-based label propagation [193], k-NN sparse graph-based supervised learning [194], Prior-base measurement propagation (pMP)[195], Label propagation based CRFs joint S&T model [196], Spectral graph transducer and Gaussian fields [197], Poisson learning [101], Shoestring [198], Sentiment value propagation [199], Label Propagation [195], Local and Global consistency [200], [201], Gaussian fields and Harmonic functions [197], [201], and OMNI-Prop [200], Greedy gradient max-cut [201], ntegrated Graph-based Semi-supervised Multiple/SingleInstance Learning [202].

Table 5.1: Categories for datasets

Datatype	Name	Application	Reference
Image Input	Brazilian Coffee Scenes	Image classification	[200]
	Cats vs Dogs	Cats or dogs classification	[200]
	CIFAR10	Image classification	[170],[101]
	COREL	Image classification, Image retreival	[193],[163]
	CORAL	Image classification	[202]
	Coil-20	Image classification	[191], [185]
	Coil-100	Image classification	[192]
	CMU	face recognition	[185]
	Diabetic Retinopath	Image segmentation, Image sequence recognition	[183]
	Digestive-Tract-Cancer	Image segmentation, Image sequence recognition	[183]
	Digit	Image classification	[192]
	ETH80	Image classification	[192]
	EMNIST	Handwritten digits classification	[200]
	FashionMNIST	Image classification	[101], [200]
	Flickr	Image annotation	[194]
	LiTS (Liver Tumour dataset)	Medical image segmentation	[175]
	MNIST	Handwritten digits classification, Image segmentation, Image sequence recognition	[183],[192],[101]
	MF (Multiple Features)	Image classification	[191]
	miniImageNet	Image classification	[198]
	NUS-WIDE-OBJECT	Image classification	[191]
	Scene	Face recognition	[191]
	tieredImageNet	Image classification	[198]
	TRECVID	Semantic concept detection in videos	[178]
	USPS	Image classification	[191],[192],[185],[188]
	UMIST	Face recognition	[185]
	WIKI	Image classification	[170]
	YaleB	Face recognition	[191], [185]
Text Inputs	AD	Webpage classification	[172]
	Comp2	Text classification	[203]
	DBWorld	E-mail classification	[188]
	Interest	Word Sense Disambiguation	[203]
	Reuters	Text classification	[204]
	Spambase	Email spam classification	[188]
	WebKB	Text classification	[203]
	Yahoo RSS News	Cross language text classification	[181]
	CTB-7	Part-of-speech tagging	[196]
	MSR	Part-of-speech tagging	[196]
	Wall Street Journal text from PTB	Part-of-speech tagging	[205]
	English poetry from BNC	Part-of-speech tagging	[205]
	Universal Dependency 2.3	Part-of-speech tagging	[176]
Sound inputs	TIMIT	Phone and segment classification, Phonetic classification	[195],[182]
Numeric Inputs	Bupa	Liver-disorder classification	[190], [192]
	Adult	Salary over 50k/yr classification	[188]
	Austra	credit lard application classification	[186],[188]
	Banknote	authentication of banknotes	[188]
	Cellcycle	Optimal caching in edge networks	[206]
	DFT database	Power factor prediction in diamond-like thermo	[207]
	Derisi	Optimal caching in edge networks	[206]
	Echodiagram	heart attack survival classification	[186]
	ECG5000	Dynamic network classification	[177]
	ECGFiveDay	Dynamic network classification	[177]
	Eisen	Optimal caching in edge networks	[206]
	Enron	Optimal caching in edge networks	[206]
	Expr	Optimal caching in edge networks	[206]
	Exprindiv_ara	Optimal caching in edge networks	[206]
	Fertility	Fertility classification	[188]
	Gasch1	Optimal caching in edge networks	[206]
	Gasch2	Optimal caching in edge networks	[206]
	German	Credit risk classification	[186]
	House-Votes	U.S. Senate and House of Representatives votes classification	[186]
	Heart-statlog	Heart decease classification	[186], [192]
	Haberman	Survival status classification of breast surgery patitents	[186],[188]
	interpro_ara	Optimal caching in edge networks	[206]
	IONOSPHERE	Radar returns classification	[186], [188], [190]–[192]
	Iris	Iris plan classification	[192]
	liver-dicorders	Presense of liver-disorder classification	[186]
	LSVT	Voice rehabilitation classification	[188]
	Mushroom	Edibility of muchrooms classification	[188]
	KDD	Network intrusion detection	[197]
	krvskp	win or lose classification in chess	[186]
	MUSK	Musk classification	[188]
	Pima	Diabetes prediction	[188], [190], [191]
	QSAR	Bioconcentration classification	[191]
	Scop_ara	Optimal caching in edge network	[206]
	Seq	Optimal caching in edge network	[206]
	Seq_ara	Optimal caching in edge network	[206]
	Sonar	Sonar signal classification	[188], [190]
	SpectHeart	Single Proton Emission Computed Tomography (SPECT) diagnosis classification	[188]
	Spo	Optimal caching in edge network	[206]
	Vote	US election vote classification	[191]
	WCBC	Breast cancer classification	[188]
	WDBC	Breast cancer diagnostics classification	[186], [188], [190]
	WDBC	Breast cancer prognostics classification	[190]
	Wine	Dynamic Network Classification	[192]
	Yoga	Dynamic network classification	[177]
	waveform	Phonetic classification	[182]

Active learning

Uncertainty sampling and Density weighted methods models The algorithms that we identified were, Sampling by uncertainty and density (SUD) and Sampling by clustering (SBC) [203].

Query-based models The algorithms found were: Query by committee [204], [205], [207]–[209] and Hierarchical Query by committee [206]. The different types of algorithms in our taxonomy are illustrated in Fig. 5.1

RQ2: What are the datasets used to evaluate these algorithms?

It is crucial to select a suitable dataset when evaluating machine learning methods. Datasets need to be real, reflecting real-life scenarios. The datasets must be well-studied and have documentation containing information regarding their features. For classification tasks, it is vital to have the same number of labels from each class. Otherwise, one has to deal with the "class imbalance" problem [210]. This is a problem because many of the basic machine learning algorithms assume that the same number of instances of each class is available [211].

The datasets were categorized based on two main characteristics, the application area and the type of input. Below we present, a definition of our classification for the type of input.

Image-based datasets

(Digital) images consists of pixels that are represented by a two-dimensional numerical array [212].

Text-based datasets

Text-based data contains ore more feature columns that contain text.

Sound-based datasets

Sound-based data contains features that come from an audio file (e.g, .wav).

Numerical datasets

Numerical data contains features that are numerical.

Every dataset and its application can be found in table 5.1.

RQ3: What algorithm(s) should be used based on application?

Practitioners from the industry need to know what labeling algorithms are optimal for their application data. Based on the papers included in this study, we found several applications that utilized the algorithms discussed. Co-training and multi-view learning can be used for fast image search, medical image segmentation, activity recognition, classification of dynamic networks, webpage classification, question classification and, natural language processing. Graph-based algorithms can be used for image annotation, data augmentation, network intrusion detection, natural language processing, text classification, document classification, and speech recognition. The EM-algorithm and mixture models can be applied for web images and text classification, microalgae classification, anomaly detection in medical images, cross-language text classification, phonetic classification, and data-driven structural health monitoring. Semi-supervised support vector machines are used for face recognition, object detection, human facial emotion detection, human activity recognition, malware detection, and lung diagnostics classification from lung sounds. AL with uncertainty sampling is used in image sequence recognition, content-based information retrieval, word sense disambiguation, text classification, part-of-speech tagging, and named entity recognition. Finally, AL using QBC is applied to part-of-speech tagging, sentiment classification, document classification, power-factor prediction, and edge caching in mobile data traffic.

Based on the findings of this study, we formulate the following guidelines for choosing the optimal labeling algorithm.

- If automatic labeling is possible, choose a semi-supervised learning algorithm. Otherwise, if the only possible choice is manual labeling, choose an active learning algorithm.

- Examine whether your application is the same or similar to the applications above. Choose your algorithm based on this information.

- Since each algorithm has different assumptions to perform optimally, evaluate your data to see what assumptions are fulfilled. It is important if two types of algorithms can work for the same application.

5.6 Discussion

Several algorithms for data labeling were identified during this study. These algorithms are based on two classes i.e, semi-supervised learning and active learning. More specifically, the semi-supervised algorithms were based on graph-based, mixture models, multi-view learning and S3VMs. Active learning algorithms were based on uncertainty sampling, density weighted methods, expected error and variance reduction and QBC. The details regarding these algorithm are summarized in section 5.2.

Theoretically, these different semi-supervised algorithms have different assumptions in order to perform in the best possible way [133]. This fact is most likely known to practitioners as none of the papers compared semi-supervised learning algorithms of different types.

On the other hand, active learning algorithms can be compared to each other, which we have also seen in the papers. We cannot draw conclusions regarding which algorithm works best as the results vary. One thing that can be said for sure is that all algorithms outperform random sampling.

Eighty-seven datasets were found and categorized into four categories, image, text, sound, and numerical inputs. Each dataset was then labeled according to its application domain. A list of applications domains for each datatype can be found in Table 5.1.

Co-training and multi-view learning has often been used on image, text, and numerical data. We found no application to sound-based data. For most cases, graph-based semi-supervised learning has been used on image data, but it has also been used on text and numerical data. Only one application to sound data has been found. Models based on the EM-algorithm and mixture models have been used primarily on image and text data. Only one application was found for each numeric and sound data. Semi-supervised support vector machines have been widely used on image data, but we only found one application for text, numeric, and sound-based datasets. We only found applications from image and text data for Active Learning using Uncertainty and Density weighted algorithms. No application was found to numerical and sound-based

data. For Active learning based on QBC, the majority of the applications were based on text data, a few on numerical datasets. No applications were found from sound-based data.

5.7 Conclusion

This study aims to provide a detailed overview of what machine learning algorithms exist for labeling and present common datasets that are used to evaluate these algorithms. The study also presents applications where semi-supervised learning and active learning algorithms are applied and a procedure for determining what algorithm should be used for what application.

Semi-supervised learning is a tool to use for automatic labeling. When automatic labeling is impossible, active learning is useful to determine which instances are most informative and best to label manually. This is useful when manual labeling is costly. Furthermore, we found a total of 87 datasets that are used to evaluate labeling algorithms. The datasets are distributed across four datatypes. The majority of the applications were based on image data and numerical data. There were fewer applications based on text data and only two applications were found for sound-based data. The results of this paper help researchers and practitioners to choose data labeling algorithms based on popularity, datatype and application. Furthermore, these results will help select the best dataset for evaluating newly developed algorithms based on what data and application labels are needed. In future investigations, we intend to evaluate more specific algorithms on both simulated and real-world data and investigate what datasets are better to use for evaluating certain algorithms and applications.

CHAPTER 6

Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies

6.1 Introduction

Current research estimates that over 80% of engineering tasks in a machine-learning ML project concern data preparation and labeling. The third-party data labeling market is expected to almost triple by 2024 [12], [13]. This massive effort spent in data preparation and labeling often happens because, in industry, datasets are often incomplete. After all, some or all instances are missing labels. Also, the available labels are of low quality in some cases, meaning that the label associated with a data entry is incorrect or only partially correct. Labels of sufficient quality are a prerequisite to perform supervised machine learning as the performance of the model in operations is directly influenced by the quality of the training data [128].

Crowdsourcing has been a common strategy for acquiring quality labels with human supervision [18], [19], particularly for computer vision and natural language processing applications. However, crowdsourcing has several limitations for other industrial applications, such as allowing unknown third-

party access to company data, lack of people with an in-depth understanding of the problem, or the business to create quality labels. In-house labeling can be half as expensive as crowdsourced labels while providing higher quality [9]. Due to these factors, companies still perform in-house labeling. Despite the large body of research on crowdsourcing and machine learning systems that can overcome different label quality problems, to the best of our knowledge, no research investigates the challenges faced and strategies adopted by data scientists and human labelers in the labeling process of company-specific applications. In particular, we focus on the problems seen in applications where labeling is non-trivial and requires an understanding of the problem domain.

Utilizing case study research based on semi-structured interviews with practitioners in two companies, one of which has extensive labeling experience, we study the challenges and the adopted mitigation strategies in the data labeling process that these companies employ. The contribution of this paper is twofold. First, we identify the key challenges that these companies experience concerning labeling data. Second, we present an overview of the mitigation strategies that companies employ regularly or potential solutions to address these challenges.

The remainder of the paper is organized as follows. In the next section, we provide a more in-depth overview of the background of our research. Subsequently, in section 6.3, we present the research method that we employed in the paper and an overview of the case companies. Section 6.4 presents the challenges that we identified during the case study, observations, and interviews at the company, the results from the expert interviews to validate the challenges as well as the mitigation strategies. Finally, the paper is concluded in section 6.6.

6.2 Background

Crowdsourcing is defined as a process of acquiring required information or results by request of assistance from a group of many people available through online communities. Thus crowdsourcing is a way of dividing and distributing a large project among people. After each process is completed, the people involved in the process are rewarded [213]. According to [13], crowdsourcing is the primary way of achieving labels. In the context of machine learning, crowdsourcing has its own set of problems. The primary problem is annotators

that produce bad labels. An annotator might not be able to label instances correctly. Even if an annotator is an expert, the labels' quality will potentially decrease over time due to the human factor [128]. Examples of crowdsourcing platforms are the *Amazon Mechanical Turk* and the *Lionbridge AI* [214].

Allowing a third-party company to label your data has its benefits, such as not developing your annotation tools and labeling infrastructure. In-house labeling also requires investing time training your annotators, which is not optimal if you don't have enough time and resources. A downside is that sensitive and confidential company data has to be shared with the crowdsourcing platforms. Before selecting crowdsourcing platforms, there are essential factors, such as how many and what kind of projects has the platform been successful with previously? Does the platform have high-quality labeling technologies so that high-quality labels can be obtained? How does the platform ensure that the annotators can produce labels of sufficient quality? What are the security measures taken to ensure the safety of your data?

A tool to be used in crowdsourcing when noisy labels are cheap to obtain is *repeated-labeling*. According to [215] repeated labeling should be exercised if labeling can be repeated and the labels are noisy. This approach can improve the quality of the labels which leads to improved quality in the machine learning model. This seems to work especially well when the repeated-labeling is done selectively, taking into account label uncertainty and machine learning model uncertainty. However, this approach does not guarantee that the quality is improved. Sheshadri and Lease [216] provides an empirical evaluation study that compares different algorithms that computes the crowd consensus on benchmark crowdsourced data sets using the *Statistical Quality Assurance Robustness Evaluation* (SQUARE) benchmark [216]. The conclusions of [216] is that no matter what algorithm you choose, there is no significant difference in accuracy. These algorithms includes *majority voting* (MV), *ZenCrowd* (ZC), *David and Skene (DS)/ Naive Bayes (NB)* [215]. There are also other ways to handle noisy labels. For example, in [130], they improve accuracy when training a deep neural network with noisy labels by incorporating a noise layer. So rather than correcting noisy labels, there are ways to change the machine learning models to handle noisy labels. The downside to this approach is that you need to know which instances are clean and which instances are noisy. This can be difficult with industrial data. Another strategy to detect noisy labels is *confident learning* which can be used to identify noisy

labels and learn from noisy labels. [217].

6.3 Research Method

In this paper, we report on case study research. We explored the challenges of labeling data for machine learning and what strategies can be employed to mitigate them. This section will present the data we collected and how we analyzed it to identify the challenges.

A case study is a research method that investigates real-world phenomena through empirical investigations. These studies aim to identify challenges and find mitigation strategies through action, reflection, theory, and practice, [105], [218], [219].

A case study suits our purpose well because of its exploratory nature, and we are trying to learn more about specific processes at Company A and B. The two main research questions we have are:

- **RQ1:** *What are the key challenges that practitioners face in the process of labeling data?*
- **RQ2** *What are the mitigation strategies that practitioners use to overcome these challenges?*

Data Collection

Our case study was conducted in collaboration with two companies. Company A is a worldwide telecommunication provider and one of the leading providers in Information and Communication Technology (ICT). Company B is a company specialized in labeling. They have developed an annotation platform to provide the autonomous vehicles industry with labeled training data of top quality. Their clients include software companies and research institutes.

- **Phase I: Exploration** - The empirical data collected during this phase is based on an internship from November 18 2019 to February 28 2020 in which the first author spent time at Company As office two-three days a week. The data was collected from the data scientist by observing how the they were working with machine learning and how they deal with data where labels are missing as well as having access to data sets. We held discussions with each of the data scientist working with each

Table 6.1: List of the interview participants of phase II

Company	Participant Nr	Title/role	Experience
A	I	Data Scientist	4 years
A	II	Senior Data Scientist	8 years
A	III	Data Scientist	3 years.
A	IV	Senior Data Scientist	2 years
B	V	Senior Data Scientist	7 years

particular dataset to collect data regarding the origin of the data, what they wish to use it for in the future, and how often it is updated. Using Python we could investigate how skew the label distribution is of the label distribution as well as examine the data to potentially find any clustering structure in the labels. The datasets studied in phase I came from participant I and II.

- **Phase II: Validation** - After the challenges had been identified during phase I, both internal and external confirmation interviews were conducted to validate if the previous phase's challenges were general. Four participants in the interviews where from company A and one participant was from company B. Company A had several data scientists, but we only included scientists that had issues with labeling. Each participant was interviewed separately, and the interviews lasted between 25-55 minutes. All but one interview was conducted in English. The one interview was conducted in Swedish and then translated to English by the first author. During the interview, we asked questions such as *What is the purpose of your labels?*, *How do you get annotated data?* and *How do you assess the quality of the data/labels?*

Based on meetings and interviews, we managed to evaluate and plan strategies to mitigate the challenges we observed during our study.

Data analysis

The interviews were analyzed by taking notes during the interviews and internship. We then performed a *thematic analysis* [114]. Thematic analysis is

defined as "a method for identifying, analyzing and reporting patterns" and was used to identify the different themes and patterns in the data we collected. From the analysis, we were able to identify themes and define the industrial challenges based on the notes. For each interview, we identified different themes, such as topics that came up during the interviews. Several of these themes were present in more than one interview, so we combined the data for each of the interviews, and based on that, we could draw conclusions based on the information on the same theme.

Threats to Validity

According to [105] there are four different concepts of validity to consider, *construct validity*, *internal validity*, *external validity* and *reliability*. To achieve construct validity, we provided every participant of company A with an e-mail containing all the definitions of concepts and some sample questions to be asked during the interview. We also provided a lecture on how to use machine learning to label data before the interviews so that the participant's could reflect and prepare for the interview. We can argue that we achieved internal validity through data triangulation since we interviewed every person at Company A that had experience with labels. Therefore it is implausible that we missed any necessary information when collecting data.

6.4 Results

In this section, we shall present the results of our study. We begin by listing the fundamental problems found from phase I of the study. Coming up next, we state the problems we encountered from Phase II. The interview we held with participant V was then used as an inspiration for formulating mitigation strategies for the data scientist's problems from Company A.

Phase I: Exploration

Here we list the problems that we found during Phase I of the case study.

1. **Lack of a systematic approach to labeling data for specific features:** It was clear that automated labeling processes was needed. The data scientists working at Company A had all kinds of needs for

automatic labeling. Currently, they have no idea how to approach the problem.

2. **Unclear responsibility for labeling:** Data scientists do not have the time to label instances manually. Their stakeholders can label the data by hand, but they do not want to do it either. Thus the data scientist is expected to come up with a way to do the labeling.

3. **Noisy labels:** Participant I has a small subset of his data labeled. These labels come from experiments conducted in a lab. The label noise seems to be negligible, but that is not the case. There is a difference between the generated data and the true data. The generated data will have features that are continuous, while the generated data will be discrete. Participant II works on a data set that contains tens of thousands of rows and columns. The column of interest includes two class labels, "Yes" and "No". The first problem with the labels is that they are noisy. The "Yes" is dependent on two errors, I and II. Only "Yes" based on error I is of interest. If the "Yes" is based on error II. then it should be relabeled as a "No". Furthermore, the stakeholders do not know if the "Yes" instances are due to error I or error II.

4. **Difficulty to find a correlation between labels and features:** Participant I works with a dataset whose label distribution contains five classes that describe grades from "best" to "worst". Where 1 is "best" and 5 is "worst". Cluster analysis reveals that there is no particular cluster structure for some of the labels. Labels of grade 5 seem to be in one cluster, but the other 1-4 seem to be randomly scattered in one cluster. Analysis of the data from participant II reveals no way of telling whether the "Yes" is based on error I or error II. This means that many of the "Yes" are mislabeled. .

5. **Skewed label distributions:** The label distribution from both datasets is highly skewed. The dataset from participant I has fewer instances that has a high grade compared to low grades. For participant II the number of instances labeled "No" is greater than the number of labels set as

"Yes". When training a model on this data, it will overfit.

6. **Time dependence:** Due to the nature of participant IIs data, it is possible that some of the "No" can become "Yes" in the future and so the "No" labels are possibly incorrect too.

7. **Difficulty to predict future uses for datasets.** The purpose of the labels in both datasets was to predict new labels for future instances provided by the stakeholder on an irregular basis. For participant I, the labels might be used for other purposes later. There are no current plans to use the label for different machine learning purposes.

Phase II: Validation

The problems that appeared during the interviews can be categorized as follows:

1. *Label distribution related.* Question regarding the distribution.
2. *Multiple-task related.* Questions regarding the purpose of the labels.
3. *Annotation related.* Questions regarding the oracle and noisy labels.
4. *Model and data reuse related.* Questions regarding reuse of trained model on new data.

   Below we discuss each category in more detail.

1. **Label Distribution:** We found several issues related to the label distribution. Participant Is data has an unknown label distribution. The current labels are measured in percentages and need to be translated into at least two classes, but if more labels are needed, that can be done. Participant II has a label distribution that contains two classes, "Yes" and "No". Participant IIIs data has a label distribution that includes at least three labels. Participants IV has more than three-thousand labels, so it is hard to get a clear picture of its distribution. Participant I-III all have skewed label distributions. If a dataset has a skew label distribution, then the machine learning model will overfit. This means that

if you have a binary classification problem and you have 80% of class A and 20% of class B, the model might predict A most of the time even when an actual case is labeled as B [220].

2. **Multiple tasks:** Participant I, II,and III say that that for now, the only purpose of their labels is to find labels for new data, but the chances are that it will be reused for something else later on. Participant IV does not use its labels for machine learning purposes but other practical reasons. If you do not plan ahead and only train a model concerning one task, then if you need to use the labels for something else later, you will have to relabel the instances for each new task.

3. **Annotation:** Participant I has some labeled data that comes from laboratory experiments. However, these labels are only used to help label new instances to be labeled manually. Participant II has its labels coming from the stakeholders, but these instances need to be relabeled since they are noisy. Participant III has labeled data coming from stakeholders, and these are expected to be 100% correct. Participant IV defines all labels by itself and does not consult the stakeholders at all. The problem here is that the data scientists are often tasked to do labeling on their own. Even if the data scientists get instances from the stakeholders, the amount of labels are often of insufficient quantity and/or quality.

4. **Data Reuse:** Participant III has had problems with reusing a model. First the data was labeled into two classes "Yes" and "No. Later the "Yes" category would be divided into sub-categories "YesA" and "YesB". When running the model on this new data, it would predict the old "Yes" instance as "No" instance. Participant III has no idea as to why this happens.

Summary from Company B

Participant V of Company B has earlier experience with automatic labeling. Therefore interview V was used to verify some actual labeling issues from the industry. According to participant V, Company B has worked and studied

automatic labeling for at least seven years. Company B uses crowdsourcing to label data using 1000 people. Participant V confirms that the labeling task takes 200 times less time thanks to active learning than if active learning was not used. The main problem company B has with the labeling is that it is hard to evaluate the quality labels and access the human annotator's quality. A final remark from Company B is that they have experienced a correlation between automation and quality. The more automation included in the process, the less accurate will the labels be. Three of the authors of this paper performed a systematic literature review on automated labeling using machine learning [25]. Thanks to that paper, we can conclude that active learning and semi-supervised learning can be used to label instances.

Machine Learning methods for Data Labeling

Here we present and discuss Active Learning and Semi-supervised learning methods in terms of how they can be used in practice with labeling problems.

Active Learning:

Traditionally labels would be chosen randomly to be labeled and used with machine learning. However, choosing instances to be labeled randomly could lead to a model with low predictive accuracy since non-informative instances could be selected for labeling. To mitigate the issue of choosing non-informative instances, active learning (AL) is proposed. Active learning queries instances by informativeness and then labels them. The different methods used to pose queries are known as *query strategies* [23]. According to [25] the most commonly used query strategies are *uncertainty sampling, error/variance reduction, query-by-committee (QBC)* and *query-by-disagreement (QBD)*. After instances are queried and labeled, they are added to the training set. A machine learning algorithm is then trained and evaluated. If the learner is not happy with the results, more instances will be queried, and the model will be retrained and evaluated. This iterative procedure will proceed until the learner decides it is time to stop learning. Active learning has proven to outperform passive learning if the query strategy is properly selected based on the learning algorithm [23]. Most importantly, active learning is a great way to make sure that time is not wasted on labeling non-informative instances, thus saving time and money in crowdsourcing [13].

Semi-supervised learning:

Semi-supervised learning (SSL) is concerned with algorithms used in the scenario where most of the data is unlabeled, but a small subset of it is labeled. Semi-supervised learning is mainly divided into *semi-supervised classification* and *constrained clustering* [62].

Constrained clustering is an extension to unsupervised clustering. Constrained clustering requires unlabeled instances as well as some supervised information about the clusters. The objective of constrained clustering is to improve upon unsupervised clustering[221]. The most popular semi-supervised classification methods are *mixture models using the EM-algorithm*, *co-training/multi-view learning*, *graph-based SSL* and *semi-supervised support vector machines (S3VM)* [25].

Below we list eight practical considerations of Active Learning.

1. **Data exploration to determine which algorithm is best.** When starting on a new project involving machine learning, it is hard to know which algorithm will yield the best result. Often there is no way of knowing beforehand what the best choice is. There are empirical studies on which one to choose, but the results are relatively mixed [158], [222], [223]. Since the selection of algorithms varies so much, it is essential to understand the problem beforehand. If it is interesting to reduce the error, then expected error or variance reduction is the best query strategies to choose from [23]. If the sample's density is easy to use and there is strong evidence that support correlation between cluster structure to the labels, then use density-weighted methods [23]. If using extensive probabilistic models, uncertainty sampling is the only viable option [23]. If there is no time testing out different query strategies, it is best to use the more simple approaches based on uncertainty [23]. From our investigation, it is clear that company A needs labels in their projects. However, since they have never implemented an automatic labeling process before, it is important to do right from the beginning. The data scientists must carefully examine the distribution of data set, check whether there are any cluster structures and if there are any relationships between the clusters and the labels. If the data exploration is done in a detailed, correct way, then finding the correct machine learning approach is easy, and we don't need to spend time testing different

machine learning algorithms.

2. **Alternative query types:** A traditional active learner queries instances to be labeled by an oracle. However, there are other querying ways, e.g *human domain knowledge*, incorporated into machine learning algorithms. This means the learner builds models based on human advice, such as rules and constraints, and labeled and unlabeled data. An example of domain knowledge with active learning is to use information about the features. This approach is referred to as *tandem learning* and incorporates feature feedback in traditional classification problems. *Active dual supervision* is an area of active learning where features are labeled. Here oracles label features that are judged to be good predictors of one or more classes. The big question is how to query these feature labels actively.

3. **Multi-task active learning:** From our interview we can see that there are cases where labels are needed to predict labels for future instances. In other cases the labels aren't even needed for machine learning. In one case the data scientist thinks that the labels will be used for other prediction task but is unsure. The most basic way in which active learning operates is that a machine learner is trying to solve a single task. From the interviews it is clear the same data needs to annotated in several ways for several future tasks. This means that the data scientist will have to spend even more time annotating at least one time for each task. It would be more economical to label a single instance for all sub-tasks simultaneously. This can be done with the help of multi-task active learning [224].

4. **Data reuse and the unknown model class:** The labeled training set collected after performing active learning always has a bias distribution. The bias is connected to the class of the model used to select the queries. If it is necessary to switch learners to a more improved learner, it might be troublesome to reuse the training data with models of a different class. This is an essential issue in practical use for active learning. If you know the best model class and feature set beforehand,

then active learning can safely be used. Otherwise, active learning will be outperformed by passive learning.

5. **Unreliable oracles:** It is essential to have access to top-quality labeled data. If the labels come from some experiments, there is almost always some noise present. In one of the data sets from company A, a small subset of the data was labeled. The labels of that particular data set come from experiments conducted in a lab. The label noise seems to be negligible, but that is not the case. There is a difference between the generated data and the actual data. The actual data will have continuous features, while the generated data will have discrete features. Another dataset that we studied has labels that came from customer data. The labels were coded "Yes" and "No". However, the "Yes" was due to factors A and B. So the problem here is to find a model that can predict the labels, but we are only interested in the "Yes" that is due to factor A. The "Yes" due to factor B needs to be relabeled to a "No". Since the customer data does not provide whether the "Yes" are due to factor A or B. The second problem was that some of the "No" could develop into a "Yes" over time. It was up to the data scientist to find a way to relabel the data correctly. The data scientist had a solution to the problem but realized that it was faulty and asked us for help. We took a look at the data and the current solution. We saw two large clusters, but no significant relationship existed between the different labels and the features. We found two clusters, but both contained almost equally many "Yes" and "No". Let's say that the first cluster contained about 60% "Yes" and 40% "No" and in the second cluster we had 60% "No" and 40% "Yes". After doing this, all of the first cluster instances were relabeled as "Yes" and all instances in the second cluster were relabeled as "No". We conclude that this is an approach that will yield noisy labels. The same goes if the labels come from a human annotator because some of the instances might be difficult to label. People can easily be distracted and tired over time, so the labels' quality will vary over time. Thanks to crowdsourcing, several people can annotate the same data, and that it is easier to determine which label is the correct one and produce "gold-standard quality training sets". This approach can also be used to evaluate learning algorithms on training sets that

are non-gold-standard. The big question is: How do we use noisy oracles in active learning? When should the learner query new unlabeled instances rather than update currently labeled instances if we suspect an error. Studies, where estimates of both oracle and model uncertainty were taken into account, show that data can be improved by selectively repeated labeling. How do we evaluate the annotators? How might the effect of payment influence annotation quality? What to do if some instances are noisy no matter what oracle you use and repeated labeling does not improve the situation?

6. **Skewed label distributions:** In two of the data sets we studied, the distributions of the labels are skewed. That is, there is more of one label than there is of another. In the "Yes" and "No" labeled example, there are way more "No" instances. When the label distribution is skewed, active learning might not give much better results than passive learning. If the labels are not balanced, active learning might query more of one label than another. The skewed distribution is a problem, but the lack of labeled data is also a problem. In one of the datasets, we have instances labeled from an experiment. Very few labels are labeled from the beginning, and new unlabeled data is coming every fifteen minutes. "Guided learning" is proposed to mitigate the slowness problem. Guided learning allows the human annotator to search for class-representative instances in addition to just querying for labels. Empirical studies indicate that guided learning performs better than active learning as long as it's annotation costs are less than eight times more expensive than labeling queries.

7. **Real labeling costs and cost reduction:** From observing the data scientists at Company A, we would say that they will spend about 80% of the time they spend on data science prepossessing the data. Therefore we recognize that they do not have time to label too many instances, and it is crucial to reduce the time it takes to label things manually. If the possibility exists, avoid manual labeling. Assume that the *cost* of labeling is uniform. The smaller the training set used, the lower will the associated costs be. However, in some applications, the cost

might be varying, so simply reducing the labeled instances in the training data does not necessarily reduce the cost. This problem is studied within *cost-sensitive active learning*. To reduce the effort in active learning, *automatic pre-annotation* can help. In automatic pre-annotation the current model predictions helps to query the labels [26], [27]. This can often help the laboring efforts of the learner. If the models make many classification mistakes, then there will be extra work for the human annotator to correct them. To mitigate these problems *correlation propagation* can be used. In correlation propagation, the local edits are used to update the prediction interactively. In general automatic pre-annotation and correction propagation do not deal with labeling costs themselves. However, they do try to reduce the costs indirectly by minimizing the number of labeling actions performed by human oracle. Other cost-sensitive active learning methods take varying labeling costs into account. The learner can incorporate both current labeling costs and expected future errors in classification costs [29]. The costs might not even be deterministic but stochastic. In many applications, the costs are not known beforehand. However, they might be able to be described as a function over annotation time [51]. To find such a function, train a regression cost-model that predicts the annotation costs. Studies involving real human annotation cost shows the following results.

- Annotation costs are not constant across instances [225]–[228].
- Active learners that ignore costs might not perform better than passive learners [23].
- The annotations costs may vary on the person doing the annotation [225], [229].
- The annotation costs can include stochastic components. *Jitter* and *pause* are two types of noise that affect the annotation speed.
- Annotation can be predicted after seeing only a few labeled instances. [227], [228].

8. **Stopping criteria:** Since active learning is an iterative process, it is relevant to know when to stop learning. Based on our empirical findings, the data scientists have no interest in doing any manual labeling, and if they have to, they want to do it as little as possible. So when

the cost of gathering more training data is higher than the cost of the current system's errors, then it is time to stop extending the training set and hence stop training the machine learning algorithm. From our experience at company A the data scientist have so little time free from doing other tasks than data prepossessing, so time is the most common stopping factor.

Challenges and Mitigation Strategies:

Many of the problems identified during phase I and phase II overlap to a certain degree, so we took all the problems and summarized them into three challenges (C1-C3) that were later mapped to three mitigation strategies (MS1-MS3). These mitigation strategies are derived from the practical consideration above. Finally, we map MS1 to C1, MS2 to C2, and MS3 to C3.

C1: **Pre-processing:** This challenge represents all that needs to be done during the planning stage of the labeling procedure. This would include creating a systematic approach for labeling (problem 1 of phase I), doing an exploratory data analysis to find the correlation between labels and features (problem 4 of phase I), as well as choosing a model that can be reused on new data (problem 6 of phase I) and label instances concerning multiple tasks (problem 7 of phase I, problem 4 of phase II).

MS1: **Planning**: This strategy contains all the solution frameworks from practical consideration 1, 2, 3, 4, 7 and 8 as they all involve the steps necessary to plan an active learning strategy for labeling.

C2: **Annotation:** This challenge represents the problems concerning choosing an annotator as well as evaluating and reduce the label noise (problems 2,3 from phase I and problem 3 from phase II).

MS2: **Oracle selection**: This strategy contains only solution frameworks from practical consideration 5. It describes how we can choose oracles to produce top quality labels.

C3: **Label Distribution:** This challenge represents all the problems concerning the symmetry of the label distributions such as learning with a skew label distribution (problem 5 of Phase I and problem 1 o Phase II).

MS3: **Label distribution**: This strategy contains solution frameworks from practical consideration 6. It describes how we can do labeling when the label distribution is skew.

6.5 Discussion

We learned that active learning is a popular tool for acquiring labels from our verification interview with Company B. Thanks to active learning, the labeling task takes 200 times less than if active learning was not used.

In the background, we presented some current practices that can help with labeling. The most popular practice being crowdsourcing. However, crowdsourcing has its own set of problems. The primary concern is those bad annotators will produce noisy labels due to inexperience or human factors. Secondly, The benefit of allowing third-company to label data is that you don't have to spend time training your employees to do the job, nor do you need to develop your own annotation tools and infrastructure. The big downside is that you have to share confidential company data with the crowdsourcing platform. Repeated labeling can improve the quality of the labels, but there are no guarantees that this will enhance the quality. Rather than correcting noisy labels, there are ways in which you can change the machine learning models to handle noisy labels. The downside to this is that you need to know which instances are bad, and this can be difficult in an industrial setting.

None of the techniques discussed in the background utilizes automated labeling using machine learning. Thanks to our efforts, we formulated three labeling challenges and provided mitigation strategies based on active machine learning. These challenges are related to questions such as, How can labeling processes be structured?, who and how do we label the instances? Can the correlation between labels and features be found, so that labels can be determined from the features? Both manual and automatic labeling involves some noise in the labels. How should these noisy labels be used? What do we do if the distribution of the labels is skewed? How do we consider the fact that some of the labels might change over time, due to the nature of the

data? How do we label instances so that the labels can be useful for several future tasks?

Three mitigation strategies that could possibly solve the three challenges were presented.

6.6 Conclusion

This study aims to provide a detailed overview of the challenges that the industry faces with labeling and outline mitigation strategies for these challenges.

To the best of our knowledge 95%of all the machine learning algorithms deployed in the industry are supervised. Therefore, every dataset must be complete with labeled instances. Otherwise, the data would be insufficient, and supervised learning would not be possible.

It proves to be challenging to find and structure a labeling process. You need to define a systematic approach for labeling and examine the data to choose the optimal model. Finally, you need to select an oracle to produce top-quality labels as well as plan how to handle skewed label distributions.

The contribution of this paper is twofold. First, based on a case study involving two companies, we identified problems that companies experience in relation to labeling data. We validated these problems using interviews at both companies and summarized all problems into challenges. Second, we present an overview of the mitigation strategies that companies employ (or could employ) to address the challenges.

In our future work, we aim to further develop the challenges and mitigation strategies with more companies. In addition, we intend to develop solutions to simplify the use of automated labeling in industrial contexts.

CHAPTER 7

An Empirical Evaluation of Graph-based Semi-Supervised Learning for Data Labeling

7.1 Introduction

The need for labeled data has increased with the increased use of supervised learning for classification tasks in industry. Supervised learning algorithms require labeled data, and the larger the datasets, the better the accuracy the algorithm will yield [16], [128]. Companies may have access to large datasets, but these datasets are often incomplete because there is a lack of labeled instances. Practitioners can achieve labels in two ways [16]. First, labeling can be outsourced to third-party personnel through crowdsourcing platforms [18], [19] such as Amazon Mechanical Turk [214]. The problem here is that it is difficult for the customers to guarantee the quality of the labels as the labelers may lack the experience and expertise to perform the labeling [128]. In addition, companies that own sensitive data and require authorized personnel to handle it. These companies requires deep background checks on their employees and are therefore usually limited to employing citizens as it is easier to do background checks for them. The second approach is in-house labeling,

which allows companies to train authorized labelers to do the labeling and guarantee high-quality labels [9]. In-house labeling is usually the best choice for companies that cannot share their data with third-party organizations. On the downside, in-house labeling requires the companies to develop their labeling infrastructure, which may be expensive. Another issue that arises is the question of who should be responsible for the labeling. Data Scientists and Software Engineers, with their extensive domain knowledge about the dataset and the problem they aim to solve, are typically the most qualified labelers. However, these professionals are usually experts in programming languages such as Python, R, and SQL. Their time is best spent on more specialized tasks such as building machine learning models in Python, performing data analysis in R, and manipulating data using SQL.

Given the challenges associated with crowdsourcing and in-house labeling, practitioners are increasingly turning to automatic labeling as a promising solution. According to a systematic mapping study, Graph-based semi-supervised learning (GBSSL) emerges as the most popular automatic labeling algorithm. Despite the availability of numerous automatic labeling algorithms, the lack of studies guiding practitioners in choosing the right algorithm for their use case has hindered their widespread adoption in the industry.

To fill this research gap and help practitioners, we provide guidelines to help practitioners find the optimal algorithm for their scenario. This is done by constructing a benchmark test consisting of the 13 GBSSL algorithms found in the GraphLearning package and taking 12 datasets equally distributed among three data types: image, text and numerical. We analyze the algorithms in two dimensions: *Performance*: Which algorithms rank the highest w.r.t highest accuracy? and *Datatype:* Do the algorithms rank differently w.r.t datatype? We also examine the probability of the algorithms reaching at least 90% accuracy and whether the presence of noise will affect the accuracy. We add noise to the accuracy to simulate a real-world scenario where real-world datasets are more difficult to solve than benchmark datasets that can be too easy to learn [230]. The results of this study provide the optimal algorithm for practitioners to use for labeling their data, and it lets them know whether the algorithm has a high probability of achieving an accuracy above 90%.

The paper is organized in the following manner. Section 7.2 describes the theory behind semi-supervised learning. Section 7.3 outlines the research method, how the simulations were set up, what software packages were used

and how the algorithms were evaluated. Section 7.4 presents the results, and section 7.5 discusses the results. Finally, the paper is concluded in section 7.6. A replication package together with an in-depth analysis of the validity of the Bayesian Data Analysis can be found at the online repository [1].

7.2 Background

This section discusses related work and provides an overview of the algorithms and statistical tools utilized in this study.

Semi-Supervised Learning

Semi-supervised learning is a combination of supervised and unsupervised learning because it uses both labeled data and unlabeled data [62]. There are two types of semi-supervised learning: *transductive* and *inductive* learning. Inductive SSL works just like supervised learning but uses both labeled and unlabeled data to improve the accuracy of a classifier that utilizes only labeled data. Transductive learning algorithms take labeled and unlabeled data as input and then only label the unlabeled data provided as input, unlike inductive learning, which labels all new incoming data points [62].

SSL relies on three different assumptions [62]: First is the *smoothness assumption*, which states that if two instances x_1 and x_2 lie in a high-density region, then so does their corresponding labels y_1 and y_2. Second is the cluster assumption, which states that instances in the same cluster most likely belong to the same class. Lastly, we have the *manifold assumption* that states that high-dimensional data lie in a low-dimensional manifold.

Graph-based Semi-Supervised Learning

A graph-based algorithm aims to build a graph $G = (V, E, W)$ representing the input dataset. Here, V is the set of vertices, E is the edges between the nodes, and W is the set of weights measuring the correlation between the nodes i and j. If $W_{ij} = 0$, there is no correlation between node i and j [231]. Graph algorithms are very popular due to their many advantages [231]:

1. Most algorithms are convex optimization problems.

[1] https://github.com/teodorf-bit/emprical-evaluation-of-graphlearning

2. Many datasets are naturally represented by graphs.

3. The algorithms are scalable and work well in practice.

Graph-based SSL algorithms all rely on the manifold assumption and some algorithms also rely on the cluster assumption [231]. Each instance in the dataset is represented by a vertex in the graph, and each vertex is connected to another through weights that indicate the strong correlation between two nodes. In addition, the graph must be undirected, and there must be no self-loops [231].

Models for matched pairs

Matched pairs are the comparison of two samples $\left(y^{(1)}, y^{(2)}\right)$ where both $y^{(i)},\ i = 1, 2$ come from the same distribution and are therefore dependent samples. For a sample of pairs $\left(y_1^{(1)}, y_2^{(2)}\right), ..., \left(y_n^{(1)}, y_n^{(2)}\right)$ we denote $\pi_{ab} = P\left(y^{(1)} = a, y^{(2)} = b\right)$. If $n_{ab} = \#\left\{i : y_i^{(1)} = y_i^{(2)}\right\}$ then $\pi_{ab} = n_{ab}/n$ [117].

If $y^{(1)},\ y^{(2)}$come from categorical outcomes, we can rank the outcomes according to some preference. Ranking can be done using the Bradley-Terry model. Suppose that $\pi_{ab} = P\,(a \text{ over } b)$, and $\pi_{ba} = P\,(b \text{ over } a)$. The Bradley-Terry model [119], [120] is defined as:

$$\log \frac{\pi_{ab}}{\pi_{ba}} = \beta_a - \beta_b, \tag{7.1}$$

or:

$$\pi_{ab} = \frac{\exp(\beta_a)}{\exp(\beta_a) + \exp(\beta_b)}. \tag{7.2}$$

This paper will use the Bradley-Terry model to rank the algorithms with the highest accuracy.

Generalized Linear Models

Generalized Linear Models (GLMs) model functions $g(\mu_i)$ where $\mu_i = E(Y_i)$ using response variables Y_i that are not normally distributed. A GLM consists of three components:

- A *random component:* the response variable Y where $y_1, ..., y_N$ are response observations.
- A *systematic component* $\eta_i = \sum_j \beta_j x_{ij},\ i = 1, ..., N.$
- A *link function* $\eta_i = g(\mu_i)$, where $\mu_i = E(Y_i)$ and g is a monotone differentiable.

Furthermore, the distribution of Y comes from an *exponential family* so its probability density function can be written in the form:

$$f(y_i|\theta) = a(\theta_i)b(y_i)\exp[y_i Q(\theta_i)], \tag{7.3}$$

where $Q(\theta_i)$ is called *natural parameter*function [117].

Binomial Logit Models

If the response variable is binary: $Y \in \{0, 1\}$, then $Y \sim$ Bernoulli(p) and has probability density function:

$$f(y|p) = (1-p)\exp\left(y \log \frac{p}{1-p}\right). \tag{7.4}$$

which is an exponential family: $a(p) = 1-p, b(y) = 1$ and $Q(p) = \log \frac{p}{1-p} = g(\mu)$ is the log-odds. We call this model the Binomial Logit Model [117]:

$$\log \frac{p}{1-p} = \sum_j \beta_j x_{ij}. \tag{7.5}$$

In this study, we shall later use the binomial logit to model the probability of success for an algorithm to exceed 90%.

Random effects modeling

Suppose that the response can be clustered and that y_{it} represents the ith instance of cluster t. We assume that each cluster is affected by a random effect. Suppose that x_{it} and z_{it} are vectors containing the exploratory variables for the fixed and random effects, respectively. Then, we include random effects

into a GLM. A GLM with random effects is called Generalized Linear Mixed Model (GLMM) [117] and is of the form:

$$g(\mu_{it}) = \mathbf{x}_{it}^T\beta + \mathbf{z}_{it}^T\mathbf{u}_i, \tag{7.6}$$

where $g(\cdot)$ is the link function and $\mathbf{u} \sim \text{Normal}(\mathbf{0}, \mathbf{\Sigma})$. In this paper, we will use the binomial logit mixed model to investigate the random effects of each benchmark dataset.

7.3 Research Method and Data Analysis

Developing the optimal data labeling algorithm is time-consuming and complicated. Industry practitioners usually rely on benchmark studies to choose algorithms depending on their use case to avoid spending time on algorithm development. Benchmark studies are used in algorithm development to evaluate how algorithms perform in different scenarios and are optimal for helping practitioners choose algorithms based on their use case [34].

Conversely, empirical evidence shows that algorithms may perform differently on real-world datasets because they are more difficult for the algorithms to learn, as has been shown with supervised learning [232], [233]. Item Response Theory [121], [122] has shown that many benchmark datasets used in benchmark evaluations are too easy for the algorithms to learn, which has been shown with supervised learning and [234] and GBSSL [230]. To investigate what algorithms are likely to perform equally well on benchmark and real-world datasets, we add noise to the data and compare the performance of the algorithms with and without noise.

This study aims to answer the following research questions:

RQ1: *What algorithms are optimal for automatic labeling of data? Do the algorithms perform better on specific datatypes?*

RQ2: *What algorithms are optimal for each datatype?*

RQ1 and RQ2 are required to determine which algorithms practitioners should use depending on the scenario. We first compare the algorithms

across all the datasets, and then each datatype is compared separately.

RQ3: *What algorithms have high probability of achieving an accuracy above 90% ?*

This RQ is important to answer as evaluating algorithms using simulations is a time-consuming task. Practitioners want to know whether it is likely for the algorithm to reach an accuracy above 90% without running simulations.

RQ4: *What is the impact of noise in the performance on the algorithms?*

This RQ is important to determine whether the benchmark test is reliable. Practitioners need to know if the algorithms will perform equally well on real-world datasets as they do on benchmark datasets.

To answer RQ1 and RQ2, we utilized the Bradley-Terry model for ranking and a GLMM, respectively. We use Bayesian thinking rather than frequentist due to its many advantages [115].

Bayesian Data Analysis

Frequentistic statistics rely on p-values and summary statistics to analyze parameters. On the downside, p-values can be misleading; for example, in hypothesis testing, the p-value is the probability of test variables T exceeding a specific value, assuming that the null hypothesis is true and is not the probability that the null hypothesis is true which seems more intuitive. In addition, hypothesis testing depends on what confidence level α is chosen, and there is no intuitive way to choose that confidence level [112], [118].

The Bayesian view treats each parameter θ as a random variable that can be modeled by imposing a prior distribution on the variable θ [115]. We then impose a likelihood $f(x|\theta) = \ell(\theta|x)$ to model the data. We then calculate the posterior distribution by using the Bayes formula:

$$f(\theta|x) = \frac{\ell(\theta|x)f(\theta)}{f(x)}, \tag{7.7}$$

where $f(x)$ is the marginal density defined as:

$$f(x) = \int_{\Theta} f(x, \theta)\, d\theta = \int_{\Theta} \ell(\theta|x) f(\theta)\, d\theta. \tag{7.8}$$

However, the marginal distribution is difficult or sometimes impossible to compute, so we must use Markov Chain Monte Carlo (MCMC) sampling to simulate the posterior distribution. In this study, we always simulate four MCMC parallel chains using 2000 iterations and 200 warmup iterations.

Assessing the Convergence of MCMC

For the MCMC simulation to be considered valid, we require that all chains converge and examine the *Gelman-Rubin Potential Scale Reduction*($\widehat{R}$) and number of efficient samples n_{eff} [235]. As a rule of thumb, we want $\widehat{R} < 1.01$ and $n_{eff} \geq 200$ [118].

Posterior Predictive Checks

Bayesian modeling allows us to evaluate the model based on the posterior distribution using posterior predictive checks. Posterior predictive checks are performed by simulating from the posterior predictive distribution:

$$f(\widehat{x}|x) = \int_{\Theta} \ell(\theta|\widehat{x}) f(\theta|x)\, d\theta, \tag{7.9}$$

The simulated values are then compared to the observed values. The posterior is valid if the simulated distribution matches the observed distribution. To do this, we use the last 1000 iterations from each chain for a total of 4000 iterations $\theta^{(1)}, ..., \theta^{(4000)}$ from the posterior distribution and calculate the posterior predictive distribution by averaging:

$$f(\widehat{x}|x) \approx \frac{1}{M} \sum_{m=1}^{M} \ell(\theta^{(m)}|\widehat{x}),\ M = 4000. \tag{7.10}$$

We can then simulate values from $f(\widehat{x}|x)$. The calculation of the posterior distributions is highly sensitive to the choice of prior, especially when a limited amount of data is available.

High Posterior Density Intervals

Suppose we want to perform inference on the parameter θ. Subsets C^x of the parameter space Θ where θ should lie with high probability are called *confidence regions* [115]. These intervals are important for analyzing where predicted variables will be located with high probability. Confidence regions are also important in the Bayesian paradigm since it is important to consider the quantities $P\left(\theta \in \ C^x|x\right)$.

For a prior distribution π, a set C^x is said to be an α-*credible set* [115] if:

$$P^{\pi}(\theta \in C_x|x) \geq 1 - \alpha. \tag{7.11}$$

This region is called a *High Posterior Density (HPD)* α-*credible region* [115] if it can be written under the form:

$$\{\theta; \pi(\theta|x) > k_{\alpha}\} \subset C_x^{\pi} \subset \{\theta; \pi(\theta|x) \geq k_{\alpha}\}, \tag{7.12}$$

where k_{α} is the largest bound such that:

$$P^{\pi}\left(\theta \in C_x^{\alpha}|x\right) \geq 1 - \alpha. \tag{7.13}$$

HPD-regions are used because they minimize the volume of α-credible region and are therefore optimal in a decision scenario [115].

Datasets

The systematic mapping study [50] contains the 79 most popular datasets for evaluating semi-supervised learning algorithms. We chose the twelve most popular datasets that were available to find online. The datasets were equally distributed across three datatypes: image, text, and numeric. Details about about the datasets can be found in Table 7.1

Algorithms

We are comparing 13 graph-based semi-supervised learning algorithms. All these algorithms can be found in the GraphLearning package[2]. The algorithms are based on Laplace learning [94], lazy random walks [95], multiclass

[2]https://github.com/jwcalder/GraphLearning/blob/master/graphlearning/graphlearning.py

Table 7.1: Datasets contained in the empirical evaluation

Type	Dataset	Total Instances	Number of Classes
Image	Cifar-10	60000	10
	Digits	1797	10
	MNIST	60000	10
	FashionMNIST	60000	10
Text	Fake News	44594	2
	20 Newsgroup	2034	4
	Ohsumed	5379	4
	Reuters	???	6
Numeric	Iris	450	3
	Wine	178	3
	MUSK	476	2
	German	522	2
	GTZAN	10000	10

MBO [96], weighted non-local Laplacian [97], volume contrained MBO [99], Centered kernel method [236], Sparse Label Propagation[98], Poisson learning and Poisson learning using MBO [101]. Figure 7.2 contains details of the algorithms. From left to right, the first column contains the algorithm's name, and the second contains the abbreviation for each algorithm we use in the figures of the result section. The last column contains relevant referenes.

Experiments

We utilize a Python script that runs each of the 13 algorithms on 12 datasets. The script records the accuracy defined as:

$$accuracy = \frac{100}{n-m} \max\left(\sum_{i=1}^{n-m} I\left(y_i = \hat{y}_i\right), 0\right), \tag{7.14}$$

wehre n is the number of labeled and unlabeled samples, m is the number of labeled instances, y_i is the true label and $\hat{y}_i$ is the predicted label. The

Algorithm	Abbreviation	Reference
Laplace Learning	laplace	[94]
Mean Shifted Laplace	mean_shifted_laplace	[101]
Centered Kernel Method	centeredkernel	[236]
Poisson Learning	poisson	[101]
Balanced Poisson Learning	poissonbalanced	[101]
Poisson MBO	poissonmbo	[101]
Poisson with volume contraints	poissonvolume	[101]
Poisson MBO with volume contraints	poissonvolumembo	[101]
Random Walk	randomwalk	[95]
Sparse Label Propagation	sparselabelpropagation	[98]
Weighted non-local Laplacian	wnll	[97]
Poisson Learning (alternate version)	poisson2	[101]

Table 7.2: The algorithms in the GraphLearning package

function $I\left(y_i = \hat{y}_i\right)$ is interpreted as:

$$I(y_i = \hat{y}_i) = \begin{cases} 1 & \text{if } y_i = \hat{y}_i \text{ is true} \\ 0 & \text{otherwise} \end{cases}. \tag{7.15}$$

In other words, $I(y_i = \hat{y}_i)$ is 1 if the correct label is predicted and 0 if not. Since randomness is involved in the algorithms, we run each iteration ten times using different seeds.

The Bradley Terry Model

Prepare Dataset

We want to count the number of times algo_0 beats algo_1. For each pair of combination $(\text{algo}_0, \text{algo}_1)$, suppose $accuracy_{\text{algo}_0,\text{seed},\text{dataset}}$ and $accuracy_{\text{algo}_1,\text{seed},\text{dataset}}$ are the accuracies of algo_0 and algo_1 for each seed respectively. For each seed $= 1, 2, ..., 10$ we compute:

$$y_{\text{algo0,algo1,seed,dataset}} = \begin{cases} 1, & \text{if } accuracy_{\text{algo0,seed,dataset}} - accuracy_{\text{algo1,seed,dataset}} < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7.16}$$

$y_{\text{algo0,algo1,seed,dataset}}$ will be 1 if algorithm algo0_1 beats algo1_1. We then summarise and calculate the total number of times algo0 beat algo1:

$$y = \sum_{\text{seed}} y_{\text{algo0,algo1,dataset,seed}}. \tag{7.17}$$

The input data for the Bradley Terry model are the total number of instances N_{total}, the success vector, y, the number of algorithms $N_{\text{algorithm}}$, and the number of benchmark datasets N_{bm}. We choose to use conjugate priors:

$$a_i \sim \text{Normal}(0, 2), \qquad \text{for all } i = 1, ..., N_{\text{algorithm}}, \tag{7.18}$$

$$s \sim \text{Exponential}(0.1), \tag{7.19}$$

$$u_{i,j} \sim \text{Normal}(0, 1) \qquad \text{for all } i = 1, ..., N_{\text{algorithm}},\ j = 1, ..., N_{\text{bm}}. \tag{7.20}$$

Then for $i = 1, 2, ...N_{\text{total}}$:

$$p_i = a_{\text{algo1}_i} + su_{\text{algo1}_i,\text{bm\_id}_i} - a_{\text{algo0}_i} - su_{\text{algo0}_i,\text{bm\_id}_i} \tag{7.21}$$

$$y_i \sim \text{BernoulliLogit}(p_i) \tag{7.22}$$

where $y_i \sim \text{BernoulliLogit}(p_i)$ has density:

$$f(y_i|p_i) = \begin{cases} \text{logit}^{-1}(p_i) & y = 1 \\ 1 - \text{logit}^{-1}(p_i) & \text{otherwise.} \end{cases} \tag{7.23}$$

Probability of success

Create the dataset

We want to investigate whether the algorithms perform differently in noise and do so by adding simulated data. We add a second column called SD to the original data. The SD column indicates whether noise was added to the data. For the original dataset we put $SD = 0$. According to [237], [238] noise

can either improve accuracy or degrade it slightly, therefore the simulated data containing noise is generated from a normal distribution with standard deviation 3. For $j = 1, .., n$:

$$\widehat{accuracy}_j \sim \text{Normal}(accuracy_j, 3), \tag{7.24}$$

and we put $SD = 3$ to indicate the presence of noise. We then concatenate the simulated dataset with the original dataset and get the full dataset. Next for each $i = 1, ..., N_{total}$ we compute:

$$z_i = \begin{cases} 1 & \text{if } accuracy_i > 90 \\ 0 & \text{otherwise.} \end{cases} \tag{7.25}$$

We then calculate the number of times each algorithm has an $accuracy > 90$ on each dataset and call it y. Next, we create new columns $algorithmID$ and $datasetID$ containing numerically coded algorithms and datasets. We replace the $algorithm$ and $dataset$ columns with $algoirthmID$ and $datasetID$, respectively. N_{draws} denotes the number of samples in each category where, in this case, the maximum is 10 because that is the number of seeds.

The input data for the model is the total number of instances N_{total}, the number of successes y, the number of draws N_{draws}, the number of algorithms $N_{\text{algorithms}}$, the numerical_id of each algorithm algorithm_id, and the number of benchmark datasets bm_id. We choose the following conjugate priors:

$$a_i \sim \text{Normal}(0, \sigma_1) \qquad \text{for all } i = 1, 2, ...N_{\text{algorithm}} \tag{7.26}$$

$$b_{noise} \sim \text{Normal}(0, \sigma_2) \qquad \text{for all } i = 1, 2, ...N_{\text{algorithm}} \tag{7.27}$$

$$a_{bm\_norm} \sim \text{Normal}(0, s) \qquad \text{for all } i = 1, 2, ..., N_{\text{bm}} \tag{7.28}$$

$$s \sim \text{Exponential}(d) \qquad \text{Standard deviation for the random effects.} \tag{7.29}$$

For each $i = 1, .., N_{\text{total}}$ compute:

$$p_i = a_{\text{algorithm\_id}} + a_{\text{bm\_norm}_i} + b_{\text{noise,algorithm\_id}_i} x_{\text{noise}}, \tag{7.30}$$

$$y_i \sim \text{BinomialLogit}(N_{\text{draw}}, p_i), \tag{7.31}$$

where:

$$y_i \sim \text{BinomialLogit}(N_{\text{draw}}, p_i) \tag{7.32}$$

and $y_i \sim \text{BinomialLogit}(N_{\text{draw}}, p_i)$ has density:

$$f(y_i|p_i, N_{\text{draw}}) = \binom{N_{\text{draw}}}{y} \left(\text{logit}^{-1}(p_i)\right)^{y_i} \left(1 - \text{logit}^{-1}(p_i)\right)^{N_{\text{draw}} - y_i}. \tag{7.33}$$

Generating posterior replications,

The algorithm generates 4 chains $\times 2000$ iterations $= 8000$ replications of a sample of size N_{total}. We only use the last 1000 iterations for each chain, so we end up with a total of 4000 samples. We replicate each y_i, $i = 1, 2, ..., N_{\text{total}}$ store the replications as matrix with elements $y^{\text{rep}}_{j,i}$, $i = 1, 2, ..., N_{\text{total}}$, $j = 1, 2, ..., 4000$ where the ith column contain all the 4000 replications of y_i.

7.4 Results

In this section, we present the results from our data analysis and then summarise the results into guidelines for practitioners.

Strength Parameters and Ranks

Figures 7.1, 7.2, 7.3 and 7.4 show the HPD regions for the strength parameters for aggregated data, image datasets, text datasets and numeric datasets respectively. The ranking of the datasets is based on the highest strength parameters and can be found in Tables 7.3, 7.4, 7.5 and 7.6. Tables 7.3, 7.4, 7.5 and 7.6 show the median rank for aggregated data, image datasets, text datasets and numeric datasets. All figures and tables indicate that the algorithm with the highest ranking is "poisson2" for all scenarios. Regarding the other algorithms, we can see from the HPD regions that many of the algorithms overlap and that the variance of the ranks is high. These two facts indicate high uncertainty in the ranks, that the algorithms perform similarly, and that it is difficult to say which algorithm is better.

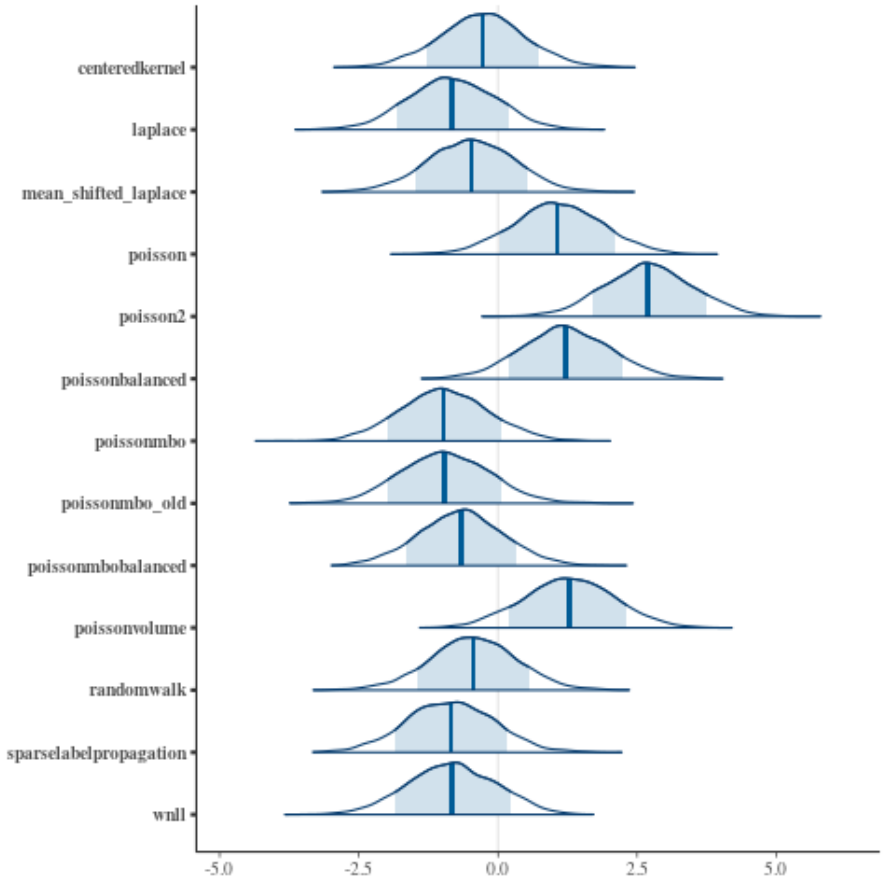


Figure 7.1: HPD region of the strength parameters for each algorithm

Table 7.3: Ranking of the algorithms (aggregated data)

Models	Median Rank	Variance of the Rank
poisson2	1	0.155
poisson	3	1.399
poissonbalanced	3	1.139
poissonvolume	3	1.200
centeredkernel	7	5.532
mean_shifted_laplace	8	6.271
randomwalk	8	6.640
poissonmbobalanced	9	6.523
laplace	10	6.115
poissonmbo	10	6.693
poissonmbo_old	10	6.144
sparselabelpropagation	10	6.459
wnll	10	5.785

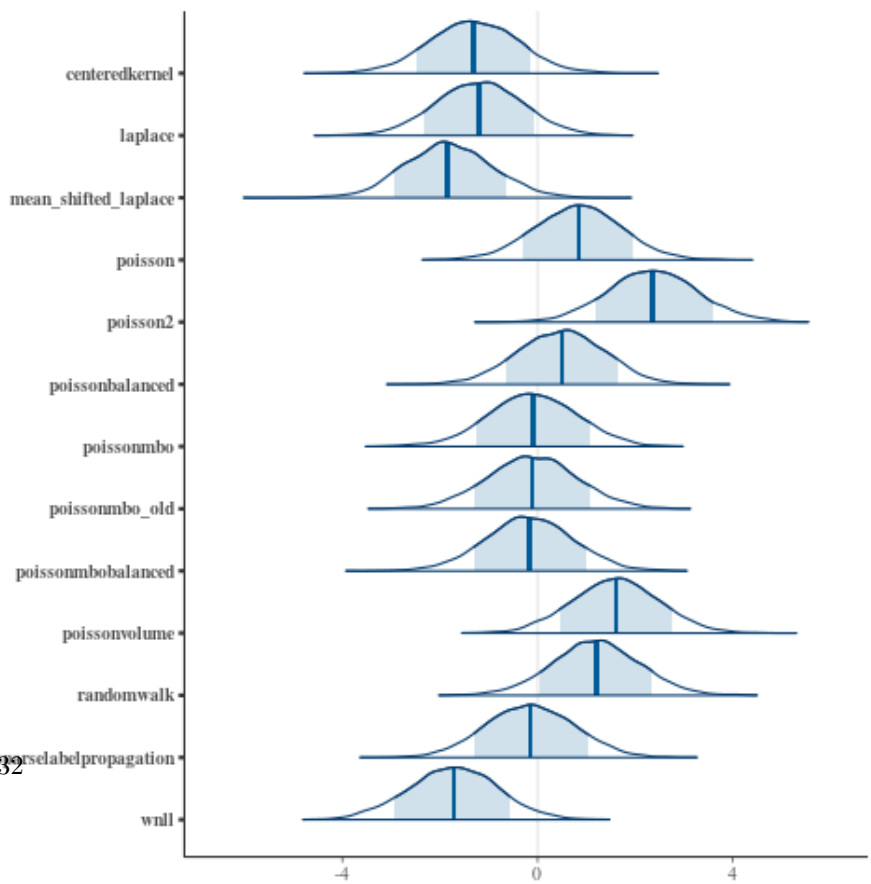


Figure 7.2: HPD region of the strength parameters for each algorithm

Table 7.4: Ranking of the algorithms (image data)

Models	Median Rank	Variance of the Rank
poisson2	1	0.819
poissonvolume	2	2.111
randomwalk	3	2.878
poisson	4	4.020
poissonbalanced	5	4.597
poissonmbo	7	4.897
poissonmbo_old	7	4.983
poissonmbobalanced	7	4.954
sparselabelpropagation	7	4.848
laplace	10	3.444
centeredkernel	11	3.344
mean_shifted_laplace	12	2.074
wnll	12	2.218

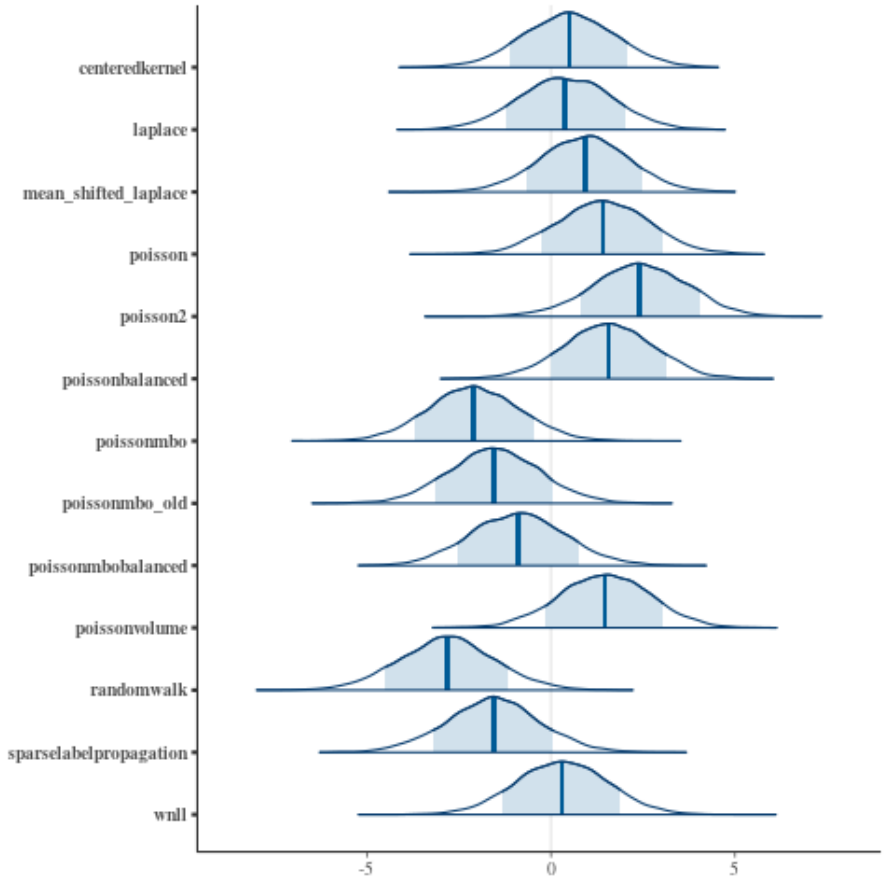


Figure 7.3: HPD region of the strength parameters for each algorithm

Table 7.5: Ranking of the algorithms (text data)

Models	Median Rank	Variance of the Rank
poisson2	2	3.655
poissonbalanced	3	4.911
poisson	4	5.202
poissonvolume	4	5.325
mean_shifted_laplace	5	5.979
centeredkernel	6	7.040
laplace	7	6.579
wnll	7	6.695
poissonmbobalanced	9	5.187
poissonmbo_old	10	3.758
sparselabelpropagation	10	4.460
poissonmbo	11	3.117
randomwalk	13	1.947

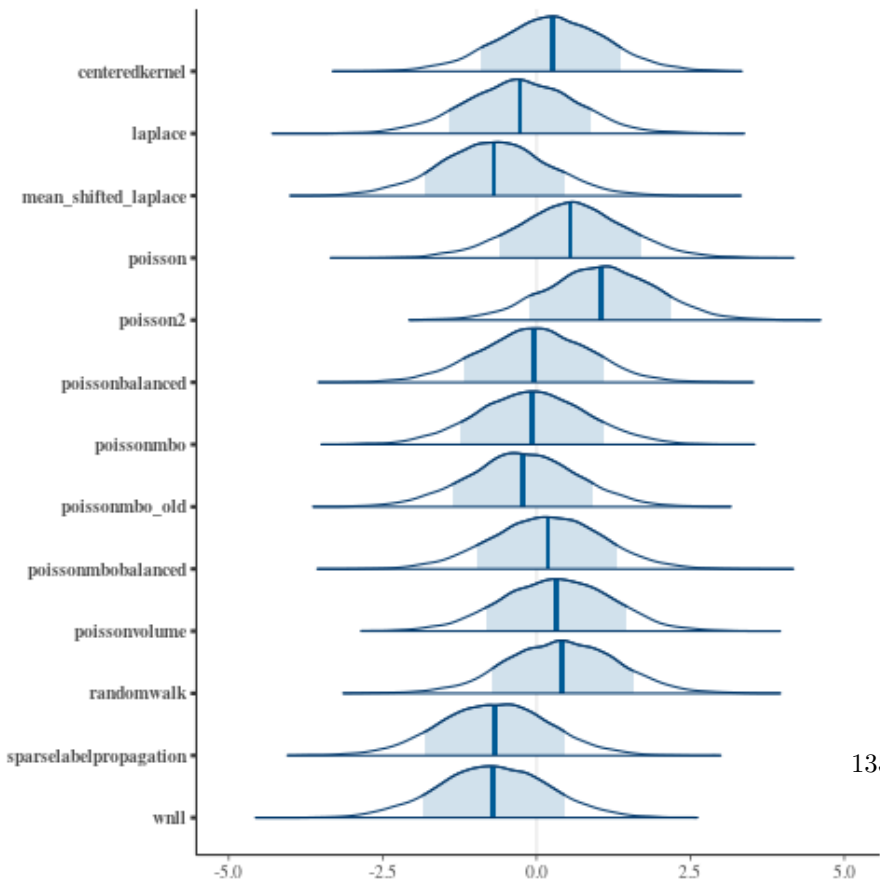


Figure 7.4: HPD region of the strength parameters for each algorithm

Table 7.6: Ranking of the algorithms (numeric data)

Models	Median Rank	Variance of the Rank
poisson2	2.0	5.831
poisson	4.0	8.996
centeredkernel	5.0	10.261
poissonvolume	5.0	10.443
randomwalk	5.0	9.900
poissonmbobalanced	6.0	10.843
poissonbalanced	7.0	10.993
poissonmbo	7.5	10.749
poissonmbo_old	8.0	10.715
laplace	9.0	10.848
sparselabelpropagation	10.5	8.651
mean_shifted_laplace	11.0	7.720
wnll	11.0	8.026

Probability of Success

We draw 4000 posterior samples for each $a_i, b_{\text{noise},i}$ $i = 1, ..., N_{algorithm}$. After that we calculate the average odds ratios e^{a_i} and $e^{b_{\text{noise},i}}$ for a_i and $b_{\text{noise},i}$. The odds ratios measure the relative probability of success compared to the probability of failure. If OR > 1, then the parameter increases the probability of success. If OR $= 1$, the probability is unaffected by the parameter, and if OR < 1, then the parameter decreases the probability of success.

Table 7.7 shows the odds ratios for each scenario, and Table 7.8 and Table 7.9 show the probability of success without and with noise, respectively. For the text datasets, Table 7.7 shows that the a parameter has average OR $= 0$ for all algorithms, so the parameter decreases the probability of success. Table 7.7 also shows that the noise parameter b has OR > 1 for all algorithms except for mean_shifted_laplace and poisson2, so these two algorithms are the only ones that decreases the probability of success in the presence of noise. In addition Table 7.8 shows that the probability of success without noise is less than three percent for all algorithms. Table 7.9 shows that in the presence of noise the probability of success is increased to $\approx 100\%$ except for laplace, poissonbalanced, and wnll, where the probabilities have decreased.

For the image datasets, Table 7.7 shows that the a parameter has average OR > 1 except for Laplace and mean_shifted_laplace so these two algorithms are the only ones to that decreases the probability of success. Table 7.7 also shows that the noise parameter b, has OR < 1 for all algorithms except for laplace, mean_shifte_laplace, and poissonbalanced, so these are the algorithms that decreases the probability of success in the presence of noise. In addition Table 7.8 shows that the probability of success is $\approx 99.9\%$ for all algorithms except for laplace and mean_shifte_laplace whose probability of success are 13.527% and 13.659% respectively. Table 7.9 shows that in the presence of noise, the probability of success have increased to 100% for all algorithms except for laplace and mean_shifte_laplace whose probabilities of success have decreased to 4.047% and 11.136% respectively.

For the numerical datasets, Table 7.7 shows that the a parameter has an average OR > 1 for all algorithms, so they all increase the probability of success. Similarly, all the b parameters have average OR > 1, so the probability of success will increase for all algorithms in the presence of noise. Table 7.8 shows that all algorithms have a probability of success $\approx 50\%$. Table 7.9 shows that the probability of success increases to above 50.1% for all algorithms except for centeredkernel and ranomwalk where the probability of success is increased to 99.997%.

Lastly, for aggregated data Table 7.7 shows that all algorithms have average OR>1 except for laplace and mean_shifted_laplace and indicates that on average the a parameter will increase the probability of success for all algorithms except for laplace and mean_shifted_lapl ace. Similarly, all algorithms except for mean_shifted_laplace, and poisson2 will increase the probability of success in the presence of noise. Table 7.8 shows that the probability of success is approximately 56%-57% for all algorithms except for laplace and mean_shifted_laplace whose probabilities of success are 13.527% and 13.659% respectively Table 7.9 shows that the probability of success increases in the presence of noise for all algorithms except for laplace.

Guidelines for practitioners

The results suggest that practitioners should label their data using the Poisson2 algorithm, which is optimal for all scenarios. Therefore, practitioners need not bother implementing or testing the other algorithms as they all perform similarly and are outranked by Poisson2. All algorithms, except for the

Table 7.7: Odds ratios for fixed effects and noise parameters. The upper part of table contains the Odds rations for the fixed effect and the lower part of the table contains the odds rations for the noise parameter.

parameters	aggregated	image	text	numeric
a_centeredkernel	1.434	69.115	0.000	6.310
a_laplace	0.113	0.006	0.000	4.505
a_mean_shifted_laplace	0.110	0.006	0.000	5.612
a_poisson	1.398	67.777	0.000	4.733
a_poisson2	1.362	70.413	0.000	5.555
a_poissonbalanced	1.369	32.411	0.000	5.440
a_poissonmbo	1.430	75.986	0.000	5.881
a_poissonmbo_old	1.456	71.373	0.000	5.363
a_poissonmbobalanced	1.329	75.451	0.000	4.809
a_poissonvolume	1.390	67.738	0.000	4.438
a_randomwalk	1.400	72.113	0.000	5.061
a_sparselabelpropagation	1.355	68.955	0.000	5.062
a_wnll	1.388	74.405	0.000	5.983
b_centeredkernel	2.349	4.655	5.076	2.790
b_laplace	2.307	0.975	5.339	2.977
b_mean_shifted_laplace	0.985	0.964	0.005	2.880
b_poisson	2.342	4.581	5.362	3.094
b_poisson2	0.994	4.436	0.005	3.189
b_poissonbalanced	1.007	0.057	5.231	3.308
b_poissonmbo	3.295	4.583	7.187	2.735
b_poissonmbo_old	2.317	4.368	5.162	3.041
b_poissonmbobalanced	3.381	4.309	7.445	3.074
b_poissonvolume	2.347	4.330	5.089	3.177
b_randomwalk	3.330	4.428	7.268	3.138
b_sparselabelpropagation	4.135	4.597	8.780	2.994
b_wnll	2.349	4.366	5.173	3.366

lower-ranking ones (wnll, laplace, mean_shifted_laplace), will have a higher chance of improving in the presence of noise. This indicates that practitioners can use the algorithms on more noisy datasets without risking worse performance than the benchmark test we performed in this study suggested. Practitioners can add more noise to the dataset to improve performance fur-

Table 7.8: Probabilities of successfully achieving an accuracy above 90% for each scenario

algorithm	aggregated	image	text	numeric
centeredkernel	57.651	99.927	2.526	50.391
laplace	38.498	13.527	0.439	50.088
mean_shifted_laplace	38.918	13.659	2.712	50.070
poisson	57.763	99.936	2.287	50.097
poisson2	57.715	99.934	2.223	50.077
poissonbalanced	56.893	99.923	0.404	50.081
poissonmbo	57.421	99.933	2.511	50.077
poissonmbo_old	57.579	99.933	2.313	50.097
poissonmbobalanced	57.307	99.934	2.906	50.082
poissonvolume	57.439	99.938	2.500	50.070
randomwalk	57.984	99.941	2.570	50.386
sparselabelpropagation	57.887	99.928	2.517	50.076
wnll	57.057	99.938	0.452	50.098

Table 7.9: Probabilities of successfully achieving an accuracy above 90% for each scenario when noise is added.

algorithm	aggregated_w_noise	image_w_noise	text_w_noise	numeric_w_noise
centeredkernel	96.000	100.000	99.610	99.997
laplace	30.073	4.047	0.000	50.121
mean_shifted_laplace	95.982	11.136	99.930	50.131
poisson	90.805	100.000	99.599	50.128
poisson2	90.860	100.000	99.599	50.131
poissonbalanced	57.877	100.000	0.000	50.118
poissonmbo	90.774	100.000	99.593	50.147
poissonmbo_old	90.771	100.000	99.594	50.114
poissonmbobalanced	97.676	100.000	99.929	50.140
poissonvolume	97.663	100.000	99.929	50.135
randomwalk	97.629	100.000	99.868	99.997
sparselabelpropagation	97.637	100.000	99.929	50.116
wnll	58.407	100.000	0.000	50.168

ther. Adding noise is especially important for text datasets. Without noise, text datasets will not reach an accuracy above 90%. When noise is added, the probability of reaching 90% accuracy goes from 2.5% to 99.9% , so text datasets can reach an accuracy above 90% when noise is added.

7.5 Discussion

Data Labeling is essential in data preparation as large labeled datasets are required to obtain optimal accuracy for supervised machine learning models. However, manual data labeling is expensive in terms of resources and time. In-house data labeling is the most optimal approach, as it allows the company to train certified labelers and ensures that sensitive data can be labeled safely. On the downside, in-house labeling may require the company to train in-house personnel to label the data, which can be time-consuming. The most flexible approach would be to let data scientists or software engineers do the labeling as they possess the most domain knowledge and do not require much training to perform labeling. On the other hand, data scientists and software engineers usually perform other tasks involving fetching data from SQL servers, building machine learning models and performing data analysis using statistical tools. Therefore, using data scientists and software engineers for labeling would be a waste of resources.

Due to the limitations of in-house labeling, practitioners would prefer an automatic labeling approach to save resources on manual labeling. Transductive semi-supervised learning algorithms such as GBSSL are an approach that takes the unlabeled instances of a sample and labels them. There are many GBSSL algorithms to choose from, and deciding which algorithm to choose can be time-consuming as parameter tuning can be complicated. They rely on benchmark studies to help practitioners choose algorithms for their applications. Benchmark studies are tests designed to evaluate the strengths and weaknesses of algorithms in different scenarios and evaluate the different choices of parameters for each algorithm [34]. Benchmark studies contain many algorithms and are essential tools that guide practitioners in choosing among existing algorithms and developing new ones [34]. A benchmark test may be constructed using many algorithms evaluated on many standard datasets. The algorithms are run multiple times using different random seeds as described in Section 7.3 to account for stochastic validity. The data collected from running the simulations are then analyzed using a statistical model such as the Bradley-Terry Model to rank the algorithms and GLMMs to calculate the probability of success of an algorithm. The Bradley-Terry and GLMMs have previously been used to evaluate other algorithms [118].

ML algorithms need to be accurate and *robust*. Robustness is the capacity of an ML model to maintain its predictive performance when the input data

is subject to noise and perturbations [237]. Robustness may be quantified using measurements such as *robustness measure* [239] and *coefficient of variation* [240]. Algorithms rely on assumptions on data called *induction bias*. For example, if we train a model on i.i.d data and then feed the algorithm test data that is c.i.i.d, this will cause errors in the model. Another example is when an adversarial attacker provides a model with malicious input changes to trick the model into misclassifying data [237]. Both SL and SSL rely on empirical risk minimization [241] where each sample instance is assumed to come from the same distribution. When the assumptions are violated, there is no guarantee that the algorithm will perform well on the data. The performance of algorithms on clean and perturbed data has been compared before on image [240] and text datasets [242]. The studies demonstrate that noise improves the robustness of algorithms and increases accuracy up to a certain point. This means that too much noise will degrade the performance. Due to the results of [240], [242], it was necessary to investigate how the probability of achieving an accuracy above 90% depends on perturbed data. The results demonstrated that the probability of achieving an accuracy above 90% increases on perturbed data.

When constructing benchmark tests, it is essential to include appropriate datasets. To the best of the author's expertise, there is no efficient way to know whether a dataset is appropriate before performing and running the algorithms. Once the evaluation of algorithms using the chosen datasets has been completed, we can use Item Response Theory (IRT) [243] to check the evaluation. IRT was initially developed to assess a test used to evaluate students' mental capacity in France [122]. Later, IRT was adopted to show that the datasets used to test algorithms were unsuitable because they were not challenging enough for the algorithms to learn [230], [234]. This means that many algorithms will perform very well on all datasets, and assessing which algorithms are optimal will be difficult. In addition, the algorithms may perform differently on real-world datasets since they may be more complex than those used in the test. IRT was previously used on tests containing supervised ML algorithms [234]. The results show that 90% of the algorithms are too easy and that 60% of datasets need to be more diverse [234]. In another study, IRT is applied to the datasets and algorithms used in this study [230]. The results indicate that 83% of the datasets are unsuitable for evaluating GBSSL algorithms because they are too easy and non-discriminating [230].

In addition, the ability level of the parameters overlaps, which means that it is hard to distinguish which algorithm is optimal, which can be confirmed by this study. According to this study, Poisson2 is the optimal algorithm for all scenarios. However, for the algorithms, we can see from Figures 7.1,7.2,7.3 and 7.4 that the HDP regions are overlapping and from Tables 7.3, 7.4, 7.5 and 7.6 we see that the variance in the ranks is high which mean that its difficult to make out which algorithm is the best. Furthermore, the studies that initially evaluated the algorithms in this study were only evaluated on image datasets, which explains why the algorithms have a high probability of achieving an accuracy above 90%

Based on the results of this study and those of the IRT paper, the algorithms in this study need to be re-evaluated using more difficult and discriminating datasets. The algorithms need to be re-evaluated, especially on image datasets, since they are highly likely to achieve accuracy above 90

Threats to Validity

This study is a simulation study [104], so we discuss four aspects of validity as described in [104]. These four aspects are *construct*, *external*, *internal*, and *conclusion* validity defined according to [104].

Construct validity concerns how appropriate the chosen models are to answer the RQs. In this paper, we need to rank algorithms, which is the purpose of the Bradley-Terry model. To investigate the probability of success for achieving accuracy above 90% using a GLMM is appropriate because they help estimate the odds ratio for binary outcomes and are designed to incorporate noise. Due to these reasons, we have construction validity.

External validity concerns how well the results of this paper can be transferred to other situations. This paper's results are based on how well each algorithm performs on the datasets. As indicated by the IRT study, many of the datasets used to test the ability of the algorithms are too easy and not discriminating, which indicates that in another benchmark study, when other more difficult datasets are used, the algorithms can perform vastly differently on difficult datasets. This means that we cannot generalize the results of this study to other scenarios. For instance, based on this study, we cannot choose the optimal algorithm as it might not be optimal for real-world scenarios.

Internal validity concerns the degree of simplification in our models. It is always possible to extend models, but to the best of the author's knowl-

edge, making things more complicated does not always improve them. In the Bradly-Terry model, we can release the assumption that ties are not allowed. This assumption seems reasonable as many ranks are tied according to the results. In addition, we can include other predictors to predict the strength parameters. For the GLMM used to investigate the probability of success, we can use a Bernoulli distribution instead of Binomal for likelihood, and many other categorical models could have been used [117], [244].

Finally, conclusion validity concerns how well the models were evaluated. To evaluate the Bayesian model, we used appropriate tools per recommendation [118]. We chose our priors so that all MCMC chains would converge. We used descriptive statistics such as $\widehat{R}$, n_{eff} and trace plots to access the MCMC, as well as posterior predictive checks to ensure that the posterior was valid.

7.6 Conclusion

GraphLearning is a Python package containing 13 different Graph-based semi-supervised learning algorithm algorithms for automatic data labeling. We evaluated the algorithms on 12 datasets equally distributed among three datatypes, image, text and numerical. First, we utilized the Bradley-Terry model to rank the algorithms according to the highest algorithms and investigate optimal algorithms for the different data types. Second, we used a generalized linear mixed model to study the probability of successfully achieving an accuracy above 90%. We utilize Bayesian modeling for its many advantages [112].

According to the results, Poisson2 is the highest-ranking algorithm for all scenarios. The other algorithms all have overlapping HPD regions, and the variance in their rankings is high, which indicates that it's difficult to say which algorithms are better than others.

Regarding the probability of success, all algorithms except for Laplace and mean_shifted_laplace have an almost 100% probability of achieving an accuracy above 90%. The numerical datasets have about 50% probability, but the text datasets only have a 0-2% probability of success. We can see that adding noise has positive effects as it increases the probability of achieving an accuracy above 90% for most algorithms. The noise is especially important for text datasets where the probability of success goes from 2% to 99% with noise.

Based on the results, practitioners know poisson2 is the optimal algorithm for the highest accuracy. Still, practitioners should not expect the algorithm to yield an accuracy above 90% on text datasets unless noise was added. Practitioners should add noise to any of the datatypes and expect an increased probability of achieving 90% accuracy. Other studies shows that adding noise can improve algorithms for other types of algorithms [240], [242].

In future work, we intend to evaluate GraphLearning on more difficult datasets and evaluate it on real-world datasets. The motivation for this comes from the fact that there is uncertainty in the rankings of the algorithms. The findings of [230] also support this claim, which shows that the HPD regions of the ability parameter overlap, meaning that the algorithms have the same ability to learn the datasets. It's difficult to tell which is the optimal algorithm. In addition, the datasets used in this evaluation are too easy and do not discriminate. Therefore, the results of this benchmark study can be misleading, and the algorithms may perform differently on real-world datasets.

CHAPTER 8

Assessing the Suitability of Semi-Supervised Learning Datasets using Item Response Theory

8.1 Introduction

In the past ten years, machine learning has increased in usage across companies that have implemented or are in the process of implementing machine learning. Supervised learning is used for classification problems. In order to perform supervised learning, huge amounts of data is required. Each instance in the dataset must be associated with a label. Companies usually have access to large amounts of data, but the data is often incomplete in the sense that the data is partially missing labels [128]. Several labeling issues were identified in a case study performed with industry [16]. Labeling is costly as companies have to spent money on services such as crowdsourcing or in-house labeling [18], [19]. These are costs that they would rather spend on automated approaches. In a systematic literature review [25], several machine learning algorithms for labeling were investigated. One of the learning paradigms found in this study was semi-supervised learning. Graph-based algorithms, Mixture models and EM, Co-training and multi-view learning are the most popular semi-

supervised algorithms [25]. Semi-supervised learning algorithms are trained using both labeled and unlabeled instances to label unlabeled data.

Even if semi-supervised learning algorithms have been around for some time, they are unknown to most companies and as a consequence the usage of such algorithms is rare in industry. In order for practitioners to know what semi-supervised learning algorithms are the best to use, these algorithms needs to be evaluated on the right datasets. To the best of our knowledge, there is no taxonomy of datasets for evaluating semi-supervised learning algorithms.

Utilizing a simulation study we evaluated twelve datasets across thirteen different graph-based semi-supervised learning algorithms. The datasets where equally distributed across three different types, namely numerical, text and image data. The datasets where evaluated using a Bayesian congeneric item response theory model.

The contributions of this paper is two-fold. First, we propose the use of Bayesian congeneric item response theory model to assess the suitability of commonly used datasets. Second, we compare the different SSL algorithms using these datasets. The results show that with except of three datasets, the others have very low discrimination factors and are easily solved by the current algorithms. Additionally, we show that the SSL algorithms perform similarly under a 90% credible interval.

The remainder of this paper is organized as follows. In the following section we provide an overview of how graph-based semi-supervised learning and how the item-response theory works. In section 8.2 we describe the method that was used during this study. What datasets we used, how we performed the simulations and the metrics that was used to evaluate the accuracy of the algorithms and how we implemented the Item response theory. The results of our simulations are presented in section 8.5. The interpretations of these results are presented in section 8.6 the paper is concluded in 12. The online appendix provides the data and the reproducible code for the model fitting, figures and tables presented in this paper. The online appendix can found at: `https://davidissamattos.github.io/congeneric-irt-ssl/`

8.2 Background

In this section, we provide an overview of graph-based semi-supervised learning

Datatypes

In this paper we study datasets of three different types, numeric, image, and text-based datasets. See definitions below.

- **Image-based:** Datasets where each instance is represented by two-dimensional numerical arrays, known as pixels. Applications include face recognition, image retreival and image segmentation.
- **Text-based:** Datasets where the feature columns contain text. Applications include named entity recognition, information extraction and word sense disambiguation.
- **Numerical datasets:** Datasets where the features are numerical. Applications include activity recognition, network intrusion detection and structural health monitoring.

Semi-Supervised learning

Supervised learning algorithms only utilize labeled data. Semi-supervised learning utilizes both labeled and unlabeled data. In some cases a semi-supervised learning algorithm can outperform supervised learning algorithms. For more information on semi-supervised learning algorithms we refer the reader to [62]. We decided to study graph-based semi-supervised learning algorithms as these are the most popular according to [25].

The graph-based semi-supervised learning procedure can be summarized into three steps. First, a graph is constructed, secondly, seed labels are injected on a subset of nodes, the last step is to infer labels on the unlabeled nodes.

Given a set of labeled instances $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n_u}$ and unlabeled instance $\mathcal{U} = \{\mathbf{x}\}_{i=1}^{n_l}$. The objective of the graph construction step is to find a graph $G = (V, E, W)$, where V are the verticies, E are the edges and $W = \{w_{ij}\}$ are the weights. The weights can be calculated using different algorithms such as kNN and ϵN methods as well as b-matching methods such as Binary, Gaussian Kernel and Localy Linear Reconstruction [144].

The second step is the seed injection step, several different algorithms has been proposed for this such as, graph cut [245], Gaussian fields [94], local and global consistency[95], adsorption[149], modified adsorption[150], quadratic criteria[92] and measurement propagation[86] are examples of different seed inferring methods.

8.3 Item Response Theory

Item response theory (IRT) corresponds to a family of statistical models developed to evaluate how latent traits of students (such as intelligence) when evaluated by a set of items (an exam). The foundations of item response theory comes from the idea of utilizing latent variables in education research from Binet (1905) and Thurstone (1925) and further developed by Lord (1952) and Rasch (1960) [246]. Since then, item response theory has been standard practice in the development of psychometric scales, national exams such as the Programme for International Student Assessment (PISA) and the National Assessment of Educational Progress (NAEP) [247]. Recent research has also suggested its use for evaluating the applicability of IRT for assessing datasets for machine learning research [248].

In this paper, we take the analogy from education to evaluate how SSL algorithms perform in different datasets. We consider the different SSL algorithms as students taking an exam. The exam consists of a selection of datasets. Each dataset corresponds to an item in the exam and each SSL model have a score for each item.

Next, we describe the simple dichotomous two-parameter IRT model to introduce some IRT concepts and then we proceed to introduce the congeneric IRT model used in this paper as well as estimation method.

The two-parameter logistic model

The two-parameter logistic model (2PL) response for dichotonomous items was introduced by Birnbaum (1968) [249] to model students abilities when taking an exam. Each item of the exam accepts only binary responses, correct or wrong.

The model assumes a latent trait variable (the ability θ) that will influence the probability of a test taker (student p) to correctly answer an item (i). In the 2PL, we model each item based on their difficulty level (b_i) and on the discrimination of the item (a_i) [246].

The model takes a logistic regression curve to estimate the probability of the test taker p to correctly answer the item i. The difficulty level shifts the logistic curve either to the left (easy items) or right (hard items). Easier items have higher probability to being correctly answered regardless of the difficulty level. Hard items require a much higher ability level of the respondent p to correctly

answer the question. The discrimination coefficient indicates the maximum slope of the logistic curve. A higher slope allows a shift in probability of correctly differentiating the respondents when their ability matches the item's difficulty.

The 2PL model is represent by the equations below [250]:

$$\pi_{p,i} = \frac{\exp a_i(\theta_p - b_i)}{1 + \exp a_i(\theta_p - b_i)} \tag{8.1a}$$

$$y_{p,i} \sim \text{Bernoulli}(\pi_{p,i}) \tag{8.1b}$$

In this model, we have the following notation:

- $\pi_{p,i}$ is the probability of an item i being correctly answered by test taker p.
- $y_{p,i}$ is the dichotomous response from test taker p on item i. The value of 1 is for a correct answer and 0 for a wrong answer.
- a_i is the discrimination parameter of item i
- b_i is the difficulty level of item i
- θ_p is the latent trait of the test taker p.

Despite the large applicability of this model in including in ML research [248], dichotomous models are not suitable for evaluating the accuracy of ML models since these variables are inherently continuous and any transformation on those can add significant bias to the results.

The congeneric model

To address the problem of using dichotomous variables, we utilize the Jöreskog's model for congeneric measurements, also called the congeneric model [126], [251].

The congeneric model assumes that the regression of the item score is modeled by a linear function on the latent variables. If assumed that only a single latent variable is present (as in the 2PL) this model takes the form of:

$$y_{p,i} \sim \mathcal{N}(\mu_{i,p}, \sigma^2) \tag{8.2a}$$
$$\mu_{i,p} = b_i + a_i\theta_p \tag{8.2b}$$

In this model, we have the following notation and interpretation of the parameters:

- $\mu_{p,i}$ is the average observation of score of an item i being answered by test taker p.
- $y_{p,i}$ is actual observation of the continuous response from test taker p on item i.
- a_i is the discrimination parameter of item i
- b_i is the difficulty level of item i. Constraining the b_i parameter as a positive value shifts the interpretation from difficulty to easiness of the item.
- θ_p is the latent trait of the test taker p.

It is worth noting that the items are modeled with a linear regression and a normal distribution of the errors. While this approach simplifies the interpretation of the model, it does not add constraint bounds on the observed values.

Bayesian estimation

The congeneric model can be both estimated using the maximum likelihood estimator procedure described in [251] or utilizing a Markov Chain Monte Carlo (MCMC) sampler in a Bayesian estimation procedure. In this paper, we utilize a Bayesian estimation method. Bayesian Data Analysis (BDA) has multiple advantages over the frequentist counterpart such as easier interpretation of the credible intervals, transparency of the model assumptions. The benefits of BDA have been widely discussed in research and an in-depth analysis is beyond the scope of this paper [112], [252]–[254].

We utilize as basis for our Bayesian congeneric IRT model, the model presented in equations 8.2. By adding normally distributed and weakly informa-

tive proper priors for all parameters [255] being estimate we arrive at following model:

$$y_{p,i} \sim \mathcal{N}(\mu_{i,p}, \sigma^2) \quad \text{[Likelihood]} \tag{8.3a}$$
$$\mu_{i,p} = b_i + a_i\theta_p \tag{8.3b}$$
$$a_i \sim \text{Half-Normal}(0, 1) \quad \text{[Prior]} \tag{8.3c}$$
$$b_i \sim \text{Half-Normal}(0, 1) \quad \text{[Prior]} \tag{8.3d}$$
$$\theta_p \sim \text{Half-Normal}(0, 3) \quad \text{[Prior]} \tag{8.3e}$$
$$\sigma \sim \text{Half-Normal}(0, 1) \quad \text{[Prior]} \tag{8.3f}$$

In this model, we use the following notation:

- $\mu_{p,i}$ is the average observation of score of an item i being answered by test taker p.
- $y_{p,i}$ is actual observation of the continuous response from test taker p on item i.
- a_i is the discrimination parameter of item i
- b_i is the easiness level of item i. Constraining the b_i parameter as a positive value shifts the interpretation from difficulty to easiness of the item.
- θ_p is the latent trait of the test taker p.

The presented model in equations 8.3 is implemented in Stan [256] and estimated with the No U-Turn Hamiltonian Monte Carlo algorithm [257]. The code related to the implementation and the data used can be found at the online appendix.

Assessment of the convergence and suitability of the model with predictive posterior checks [255] are presented in the online appendix. In section 8.5, we present the results of this model with credible intervals and median to summarize the posterior distribution.

8.4 Experimental Setup

The purpose of this paper to empirically evaluate the suitability of the datasets commonly used to evaluate and compare different SSL algorithms. We performed a simulation study using fifteen datasets of three different datatypes (numerical, text, image) on thirteen different SSL algorithms. The fifteen datasets are equally distributed across three the different types of data.

We explore the following research questions:

- **RQ1:** *What datasets are suitable to compare different graph-based SSL algorithms.*
- **RQ2:** *How can different graph-based SSL algorithms be compared.*

To compare the different graph-based SSL algorithms using IRT, we run the algorithms using a fixed percentage of labels each iteration. We let 10% of the data be labeled and the remaining of the 90% unlabeled.

Simulations

The datasets

We selected our datasets based on a systematic mapping study that is currently in proceedings. In this study the authors listed several data labeling algorithms such as active learning and semi-supervised learning algorithms. The study also contains 79 datasets that were commonly used to evaluate these algorithms. We choose twelve benchmarked datasets from the mapping study to be used in our study. These twelve datasets were the most popular and had the best availability. We choose four datasets of of each type.

- **Image:**
  - **Cifar-10:** The Cifar-10 dataset consists of 60000 images with 32×32 resolution. Each image contains one object that can be divided into ten categories, "airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", and "truck". The dataset has been used in many studies [258]–[260].
  - **Digits:** The digits dataset consists of 1797 images with 8×8 resolution. Each image contains a digit and there is a total of ten different classes representing each digit 0,1,2,...9 [261], [262].

  - **MNIST:** This is a subset of the NIST, a databese for handwritten text-recognition. Each image contains a handrwritten digit of class 0,1,2,...,9. The dataset consists of 60000, 28×28 resolution images. [263], [264].
  - **FashionMNIST:** This dataset contains Zalando's article images. The dataset consists of totally 60000 images with 28×28 resolution. Each image depicts an article from Zalando. There are ten classes of articles, T-shirt, Trouser, Pullover, Dress, Coat, Sneaker, Bag and Ankle boot. This dataset was indendent as a replacement to the classical MNIST dataset [265], [266]. .

- **Text:**
  - **Fake news:** This dataset contains news articles that can be divided into two classes, false and truthful articles. The dataset contains 44594 instances. There are three features, the "title", "subject" and the "text"[267], [268].
  - **20 Newsgroups:** This dataset consists of approximately 18000 instances of twnety classes. In this dataset we have selected only four categories "alt.atheism", "talk.religion.misc", "comp.graphics" and "sci.space" resulting in a dataset consisting of 2034 instances. [269]–[272].
  - **Ohsumed:** This dataset is a subset of the MEDLINE database and contains peer-review medical literature. The are 23 classed of medical subject headings and 50216 medical abstracts. In this paper we use a subset four medical subject headings and 5379 instances.
  - **Reuters:** This dataset consist of news documents divided into 90 categories and 9598 instances. In this paper we use a subset of six categories "Neg-", "Pos-acq", "Pos-coffee", "Pos-earn", "Pos-gold", "Pos-heat" [273]–[276].

- **Numeric:**
  - **Iris:** The purpose of this dataset was to distinguish different Iris flower species. The data consist of 150 instances equally distributed across thre classes, "Iris setosa", "Iris virginica" and "Iris versicolor" [277]–[279].

- **Wine:** This data is used for wine classification. It consist of 1797 instances and 64 features. The labels are divided across seventeen instances [280], [281].
- **MUSK:** This dataset is a subset of the MUSK dataset. It is used to classify molecules as "musks" or "non-musks". There are 476 instances and 166 features [282]–[284].
- **German:** This dataset describes german credit scores. It contains 522 instances and nine features. There are two classes "good" and "bad". The purpose of the dataset is to determine the quality of a persons credit score. [285]–[287].

Table 8.1: Summary table for the datasets.

Datatype	Dataset
Image	Cifar-10 Digits MNIST FashionMNIST
Text	Fake and truthful news 20news Ohsumed Reuters
Numeric	Iris Wine MUSK German

Each dataset was preprocessed so that the 10% of the labels were available. Each algorithm was run on the datasets ten times utilizing different seeds for each iteration. To store the results, a data frame with ten rows was created from running an algorithm on a dataset. The accuracy of the predictive labels were logged in each iteration and stored in a total of 195 data frames.

The SSL algorithms

It is recognized in [25] that graph-based semi-supervised learning is one of the most popular type of semi-supervised learning algorithms. We have included

thirteen different graph-based semi-supervised learning that are listed below. In bold we represent the short name used in the tables and figures.

The algorithms utilized in this study are based on Laplace learning [94], lazy random walks [95], multiclass MBO [96], weighted non-local Laplacian [97], volume contrained MBO [99], Centered kernel method, Sparse Label Propagation[98], Poisson and PoissonMBO algorithms [101]. All algorithms are implemented in Python using the GraphLearning package [1].

- Laplace learning **(laplace)** [94]
- Mean Shifted Laplace **(mean_shifted_laplace)** [101]
- Centered kernel method **(centeredkernel)** [236]
- Poisson learning **(poisson)** [101]
- Poisson learning, alternate version **(poisson2)**
- Balanced Poisson learning **(poissonbalanced)**
- Poisson MBO **(poissonmbo)** [101]
- Poisson MBO with volume contraints **(poissonvolumembo)**
- Balanced Poisson MBO [101]
- Poisson with volume constraints **(poissonvolume)**,[101]
- Random Walk **(randomwalk)** [95]
- Sparse Label Propagation **(sparselabelpropagation)** [98]
- Weighted non-local Laplacian **(wnll)** [97]

To calculate the accuracy of each algorithm we assume that a fixed percentage of the instances are already labeled and the rest is unlabeled. The accuracy is calculated by

$$\varepsilon = \frac{100}{n-m} \max\left(\sum_{i=1}^{n} I(y_i = \widehat{y}_i) - m, 0\right),$$

[1] https://github.com/jwcalder/GraphLearning/blob/master/graphlearning/graphlearning.py

where $I(x)$ is **indicator function** defined as

$$I(x) = \begin{cases} 1 \text{ if } x \text{ is true} \\ 0 \text{ otherwise} \end{cases},$$

n is the number of both labeled and unlabeled instances and m is the number of labeled instances. The purpose of the algorithms is to predict the labels of all of these unlabeled instances.

Threats to Validity

A treat to validity is that the accuracy of our semi-supervised algorithms might be compromised as we have not considered whether the class labels are balanced or not for all datasets [108].

8.5 Results

In this section, we present the results obtained from the parameter estimation of the Bayesian congeneric model represented by equations 8.3. We first present the estimated easiness and the discrimination parameters of the items (i.e. the datasets). Next, we present the ability level of the test taker (i.e. the SSL models)

Items parameters

Utilizing the data collected and discussed in section 8.4, table 8.2 shows the parameters for the easiness level and the discrimination level. The columns CI 5% and CI 95% represent the lower and higher of the 90% credible intervals, respectively. This table is can be visualized in figure 8.1.

We can see from table 8.2 and figure 8.1 that the easiness parameter is almost 1 for most datasets. These datasets are solved by most SSL algorithms. Analogously, most datasets have a discrimination parameter close to zero indicating that the SSL models obtain very similar results under these datasets. These datasets with high level of easiness and very low level of discrimination are not suitable for the comparison between different SSL models.

Table 8.2: Posterior summary values of the discrimination and easiness level parameters for the datasets

Dataset	Median	CI 5%	CI 95%
Discrimination value (a)			
20news	0.015	0.004	0.047
cifar	0.004	0.000	0.018
corpus	0.004	0.000	0.018
digits	0.005	0.000	0.020
fashionmnist	0.004	0.000	0.018
german	0.001	0.000	0.008
iris	0.005	0.001	0.022
mnist	0.011	0.002	0.035
musk	0.157	0.066	0.413
ohsumed	0.186	0.078	0.491
reuters	0.462	0.193	1.201
wine	0.005	0.000	0.020
Easiness level (b)			
20news	0.852	0.837	0.861
cifar	0.996	0.988	1.001
corpus	0.578	0.569	0.582
digits	0.990	0.981	0.995
fashionmnist	0.996	0.988	1.001
german	0.594	0.590	0.598
iris	0.993	0.983	0.998
mnist	0.977	0.964	0.985
musk	0.637	0.546	0.688
ohsumed	0.396	0.289	0.456
reuters	0.312	0.047	0.462
wine	0.993	0.984	0.998

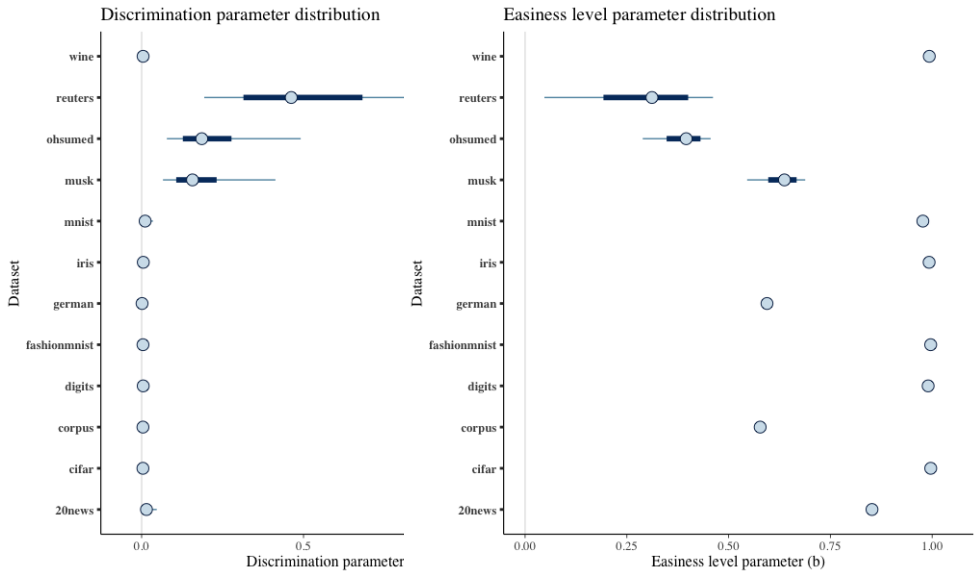


Figure 8.1: The posterior distribution of the discrimination and easiness parameters per dataset. The circles represent median value, the thick dark blue lines represent the 50% probability mass and the thin light blue line represent the 90% probability mass.

The ability parameters

Table 8.3 shows the ability parameters SSL models. The columns CI 5% and CI 95% represent the lower and higher the 90% credible intervals, respectively. This table can be visualized in figure 8.2.

We can see in table 8.3 and figure 8.2 that while there are differences between the ability parameters of the SSL models, they have large overlapping intervals, in particular in the 50% probability mass (thick dark blue line). This overlapping indicates an uncertainty in the difference between the accuracy of these SSL models.

However, it is worth noting that this large uncertainty comes from the fact that the choice of the datasets with high easiness and low discrimination parameters does not help in the proper selection of the correct SSL model.

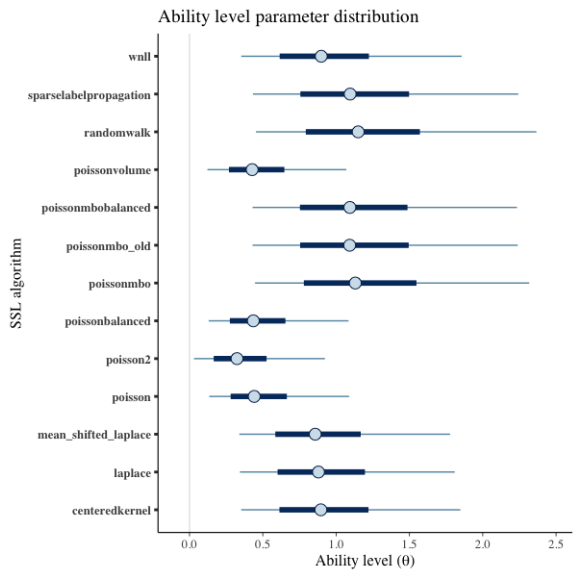


Figure 8.2: The posterior distribution of the ability parameter of the SSL models. The circles represent median value, the thick dark blue lines represent the 50% probability mass and the thin light blue line represent the 90% probability mass.

Table 8.3: Posterior summary values of the ability level of the SSL models

Model	Median	CI 5%	CI 95%
centeredkernel	0.896	0.353	1.848
laplace	0.880	0.344	1.809
mean_shifted_laplace	0.858	0.340	1.777
poisson	0.441	0.134	1.089
poisson2	0.324	0.031	0.924
poissonbalanced	0.435	0.131	1.085
poissonmbo	1.131	0.447	2.317
poissonmbo_old	1.092	0.430	2.239
poissonmbobalanced	1.094	0.430	2.234
poissonvolume	0.427	0.122	1.069
randomwalk	1.150	0.453	2.366
sparselabelpropagation	1.096	0.433	2.242
wnll	0.898	0.353	1.856

8.6 Discussion

As pointed out in section 8.1, companies are in need for automatic labeling algorithms. When performing data exploration to determine what algorithm is the best to use, it is essential to evaluate different algorithms on the the most optimal datasets.

IRT has previously been used to evaluate datasets for supervised learning classifiers [234]. In [234], 60 datasets from the well-known OpenML-CC18 benchmark are evaluated. The results show that 88% of the datasets are easy and 60% are not discriminating, hence not suitable for evaluating supervised learning algorithms.

According to Table 8.1, most of the datasets in our study have a high easiness parameter and a low discrimination parameter. This means that these datasets are not suitable for evaluating semi-supervised learning algorithms. The Corpus and German datasets have low discrimination parameter despite a medium easiness parameter. This means that algorithms with different ability parameters will have the same accuracy on these datasets.

According to Table 8.3, the algorithms with the highest ability parameters are random walk, Poisson MBO and sparse label propagation. In Figure 8.2

we can see that the C.I intervals of the algorithms overlap. This is because the datsets have low discrimination values. If the datasets would have high discrimination value, then it would be easy to differentiate the algorithms by estimating their ability parameter.

8.7 Conclusion

The goal of these simulations is to provide an overview over what datasets should be used to evaluate semi-supervised machine learning algorithms to automatically label data in an industrial setting. According to our results Reuters, Ohsumed and Musk are the only datasets suitable for evaluating semi-supervised learning algorithms. Diverse datasets with good discrimination reduces the uncertainty of the algorithms ability. This means that it is easy to determine what algorithms are better. According to our results we can say that it does not matter what semi-supervised learning algorithm we choose for labeling because our datasets are not very discriminitative.

Among the suitable datasets, Reuters and Ohsumed are text-based datasets and Musk is numerical. We have not identified any suitable image-based datasets. Therefore, in future research we want to investigate more image-based datasets to determine what datasets should be used to evaluate semi-supervised learning algorithms.

CHAPTER 9

An Empirical Evaluation of Deep Semi-Supervised Learning

9.1 Introduction

Many industries have recently started implementing machine learning algorithms for various tasks. Among these tasks, practitioners utilize supervised learning algorithms to solve classification tasks such as classifying images and text. For companies to utilize supervised classification, datasets need to be fully labeled. However, datasets are rarely fully labeled in industry, and it is difficult to obtain labels for many reasons [16]. One reason is that the data needs to be manually labeled, and companies need qualified in-house personnel to obtain high-quality labels. Specialists such as data scientists and software engineers are suitable labelers but are busy with more specialized tasks and do not have time for labeling. If specialized personnel are unable label, the other personnel must undergo training to perform labeling. On the other hand, training in-house personnel is expensive in terms of time and resources. A solution to the problems associated with in-house labeling is crowdsourced labeling [17], [18]. Crowdsourcing allows practitioners to obtain labels by out-

sourcing manual labeling to a group of people through crowdsourcing platforms such as Amazon Mechanical Turk[1]. A problem with crowdsourcing is that it is difficult to guarantee the quality of the labels since the labelers may lack the required expertise and knowledge. In addition, crowdsourcing is expensive, and companies with confidential data are not allowed share such data through labeling services. These two problems make manual labeling through crowdsourcing unappealing, and companies prefer to implement automated labeling approaches [18], [19].

According to [25], semi-supervised learning is a popular tool to reduce manual labeling. Semi-supervised learning applies to image, video, sound, text, and tabular datasets, but industry practitioners rarely utilize it [25]. Theoretical and empirical studies demonstrate that semi-supervised learning is not guaranteed to outperform supervised learning and may even degrade performance [288]–[292]. Due to the success of deep learning, the interest in deep semi-supervised learning [293]–[298] has increased. Development and evaluation of algorithms are time-consuming, so practitioners rely on benchmark studies to choose algorithms [34]. There are four dimensions to consider in semi-supervised learning. *Performance:* Which algorithms have the lowest error rate? *Datatype:* How do the algorithms perform differently depending on the datatype, *Manual Effort:* how many labels are required for an algorithm to reach optimal error rate and, *Robustness:* the capacity of a model to maintain its predictive performance when the input data is subject to noise and perturbations [237]. Recently, Microsoft made the *Unified Semi-Supervised learning Benchmark* (USB) [2] publicly available for practitioners, which lets them experiment and evaluate deep semi-supervised learning algorithms. At the time of this study, USB contains 16 deep semi-supervised learning algorithms evaluated on 15 datasets across three datatypes: image, text and audio. Since the first evaluation of USB [84], two more algorithms, FreeMatch [78] and SoftMatch [79], have been added to USB. All 16 algorithms, including SoftMatch, FreeMatch and supervised learning, are included in this study.

This study reports an empirical evaluation of deep semi-supervised learning algorithms. The study aims to assess the algorithms' performance by measuring the error rate. To access the datatype and manual effort dimensions, the error rate is analyzed separately for different datatypes and different amounts of data, respectively. Last, the study analyses the robustness dimension by

[1] https://www.mturk.com
[2] https://github.com/microsoft/Semi-supervised-learning

simulating how noise in the dataset affects the error rate.

The contribution of this study is multi-fold. First, the study presents the top three highest ranking algorithms with respect to the lowest error rate. Second, it provides the highest-ranking algorithms for each individual datatype. Third, it demonstrates to practitioners how much manual effort is needed to receive sufficient labels for optimal error rate. Fourth, the study presents which algorithms have a high probability of achieving an error rate below 10%. Fifth, it investigates whether noise in the datasets improve the performance of the algorithm and increase the algorithm's probability of reaching an error rate below 10%. The abovementioned aspect is important since many datasets in empirical evaluations are too easy for algorithms to learn. Empirical evaluations that only utilize easy datasets are unreliable. Furthermore, studies [237], [240] demonstrate that adding noise may improve the error rate of algorithms. Last, the study recommends the optimal algorithms for each scenario based on the lowest error rate and their ability to improve or maintain optimal error rate in the presence of noise. The algorithms are ranked with the Bayesian Bradley-Terry model. USB was initially evaluated in [84] by ranking the algorithms according to their *Friedman Ranks* [299]. Friedman ranks only provide point estimates of the ranks, which may lead to misleading conclusions regarding the performance of algorithms. Running each algorithm on each dataset many times utilizing different seeds lead to variations in performance. The Bayesian Bradley-Terry model accounts for that variation by modeling uncertainty [110], [111].

Thanks to the results of this study, practitioners in academia and industry have concrete guidelines on what algorithm is optimal for their labeling scenario. An optimal algorithm is in the top three highest ranking algorithms with respect to the error rate and have an increased probability of achieving an error rate below 10% in the presence of noise. According to the results FreeMatch, SimMatch and SoftMatch are the top three algorithms. None of the algorithms achieve an error rate below 10% without noise. FreeMatch is optimal for all datatypes for small allocation of labels. SimMatch is tied with FreeMatch and is optimal for image datasets and small allocation of labels. Finally, SoftMatch is optimal for text data for both small and large allocations of labels.

The paper is organized in the following manner. Section 9.2 describes the theory behind semi-supervised learning. Section 9.3 outlines the research

method, how the simulations are set up, what software packages are utilized and how the algorithms are evaluated. Section 9.4 presents the results, and section 9.5 discusses the results. Finally, the paper is concluded in section 9.7. A replication package together with an in-depth analysis of the validity of the Bayesian Data Analysis are found at the online repository [3].

9.2 Background

This section discusses related work and presents a theoretical overview of the machine learning and statistical tools utilized in this study.

Labeling challenge in Software Engineering

Many machine learning tasks in the industry are concerned with supervised learning which requires labeled data. Labeling may be time-consuming and 80% of the time spent in a Machine Learning project is allocated to labeling [300]. Since labeling is time-consuming, it is relevant that the right personnel within the company do the labeling. Data scientists and software engineers might have to spend their time on more specialized tasks such as utilizing different programming languages to build machine learning models, perform statistical analysis, and collect data from databases such as SQL [16].

If the company does not have the resources to perform in-house labeling, third-party services such as crowdsourcing [13] are available. Examples of crowdsourcing platforms are *Amazon Mechanical Turk* [20][4] and *Lionbridge AI*[5]. Crowdsourcing encourages different labelers to label data by rewarding them. By utilizing crowdsourcing, companies do not need to develop their labeling infrastructure or hire and train labelers. The issues with this approach are that it is challenging to guarantee high-quality labels and that many companies may not share sensitive data.

A tool that helps reduce the manual labeling is *Active Learning* [23]. Active learning queries what instances to be labeled according to a *query strategy* that selects the instances based on how informative they are. Query strategies ensure that labelers do not waste time labeling random instances that will not reduce the error rate. The training set is updated by adding the newly

[3] https://github.com/teodorf-bit/Bayesian-Data-Analysis-of-Universal-Semi-Supervised-Benchmark
[4] https://www.mturk.com
[5] https://www.lionbridge.com/machine-translation/

labeled instances, and the model is retrained utilizing the updated training set. If the model has not reached the desired error rate, then the training set is updated, and the model is retrained until the error rate of the model is sufficient.

Semi-Supervised Learning

Machine learning and deep learning algorithms require large amounts of data to achieve low error rate. In the industry many datasets are missing labels either entirely or partially. In order to achieve high-performance classification algorithms without utilizing costly tools such as crowdsourcing and active learning for manual labeling, companies utilize *semi-supervised learning* [62]. Semi-supervised learning algorithms have been designed to learn from unlabeled and labeled data to improve the decision boundary acquired by supervised learning. As unlabeled data is often abundant in industrial settings, it is reasonable to utilize semi-supervised learning to improve the error rate.

There are four main assumptions in semi-supervised learning. The main assumption is that few labeled instances and many unlabeled instances are available. The three other assumptions put constraints on the distribution. These are the *smoothness*, *cluster*, and the *manifold* assumptions [231]. The smoothness assumption says that if two features lie close to each other in a high-density region, their output labels also lie close. The cluster tells us that if two features lie in the same cluster, they likely have the same class label. The manifold assumption, often considered a generalization of the two assumptions above, states that each datapoint lies on a manifold [62].

Universal Semi-Supervised Benchmark (USB)

USB is an open-source platform for evaluating semi-supervised learning algorithms. It contains algorithms for various Computer Vision (CV), Natural Language Processing (NLP), and Audio-related tasks. The algorithms are evaluated utilizing 15 datasets equally distributed among the three tasks. Initially, The first evaluation of USB [84] utilizes Friedman Ranks [299] to rank a subset of the algorithms included in this study. Furthermore, this study ranks the algorithms utilizing the Bayesian Bradley-Terry model. Bayesian models are more interpretable, do not rely on p-values and have methods to validate results. In addition, this study provides more evidence to prove that the algo-

rithms will work better on real-world datasets. Other studies [230] utilize *Item Response Theory* to illustrate that several datasets are inappropriate for evaluating algorithms. Real-world datasets contain noise and are more complex for algorithms to learn. This study evaluates whether the algorithm will perform well on real-world datasets by adding noise to the benchmark datasets to investigate if there is a change in the algorithm's performance. USB is an extension of its predecessor, TorchSSL, and evaluates the algorithms utilizing fewer labels. In addition, USB introduces pre-trained transformers to speed up training time for several algorithms. The supervised baselines utilized to evaluate USB in this study are: WRN [301], WRN-Var, Resnet [258], ViT [302], BERT [303], and Wave2vec-v2 [84]. The Semi-Supervised vised algorithms are, Pseudo-Labeling [28], Π-model [73], Mean-Teacher [74], VAT [304], MixMatch [81], ReMixMatch [83], UDA [305], FixMatch [75], Dash [76], CoMatch [67], CRMatch [65], FlexMatch [78], AdaMatch[77], SimMatch [66] and SoftMatch [79].

The Bradley-Terry model

The Bayesian version [116], [118] of the Bradley-Terry model [119], [120] is utilized for ranking and comparison of objects. Each outcome $y_{i,j}$ of the comparisons are binary variables, either taking value 1 with probability $p_{i,j}$ if i beats j or value 0 with probability $1-p_{i,j}$ otherwise. Therefore the outcomes $y_{i,j}$ are Bernoulli distributed:

$$y_{i,j} \sim \text{Bernoulli}(p_{i,j}). \tag{9.1}$$

Furthermore, the Bradley-Terry model assumes that the outcomes are independent. To rank n objects, the first step is to calculate the strength parameter $\mu \in \mathbb{R}$ of each object and then calculate the probability of object i beating object j:

$$p_{i,j} := P(i \text{ over } j) = \text{logit}^{-1}(\mu_i - \mu_j). \tag{9.2}$$

Next, the algorithms are ranked by strength parameter so that the highest ranking algorithm have highest value of strength parameter. The Bradley-Terry model's ability to calculate the probability of objects beating each other and access the reliability of ranks through uncertainty estimation makes it

preferable to other models [116].

Logit Generalized Linear Mix Model for binomial samples

This study utilizes a *Generalized Linear Mixed Model* (GLMM) [117] to account for the random effect on each dataset. The Generalized Linear Mixed Model for Binomial samples [116], [117] calculates the probability of success (an algorithm yields a specific error rate). Let y_i be Bernoulli distributed observations:

$$y_i = \begin{cases} 1 & \text{if success} \\ 0 & \text{if failure} \end{cases}, \tag{9.3}$$

i.e., $y_i \sim \text{Bernoulli}(p)$.

For n samples $y_1, y_2, ..., y_n$, the sum of all outcomes will be binomial distributed:

$$y = \sum_{i=1}^{n} y_i \sim \text{Binomial}(n, p). \tag{9.4}$$

Hence, the binomial distribution is utilized as likelihood. The probability of success will be modeled as:

$$p = \text{logit}(P(y = 1)) = a + bx + u, \tag{9.5}$$

$$u \sim \text{Normal}(0, \sigma^2). \tag{9.6}$$

where a is the fixed effect, b is the log-odds ratio and u is the random effect.

9.3 Research Method and Data Analysis

This section describes the datasets utilized, the data collection approach, and the tools utilized to analyze the results.

- **RQ1:** *What are the the top-3 highest ranking algorithms in terms of lowest error rate.*
- **RQ2:** *How do the algorithms rank differently according to a specific*

datatype?

- **RQ3:** *How do the algorithms rank differently depending on the number of labeled instances in the dataset?*
- **RQ4-a:** *What algorithms have high probability of yielding an error rate* $\varepsilon \leq 0.1$
- **RQ4-b:** *What is the impact of noise in the probability of success of each algorithm's error rate* $\varepsilon \leq 0.1$

RQ1 and **RQ2** have previously been answered in [84], but in this study, more algorithms were studied, and Bayesian Bradley-Terry ranks were utilized. Due to the Bayesian nature of Bradley-Terry ranks, they provide a more fair and accurate data analysis [110], [111] than frequentistic Friedman ranks [299].

Descriptive Statistics

The collected data is assumed to be a sample of instances $x_1, .., x_n$, independent and identically distributed (i.i.d) from a random variable (r.v) X. The following descriptive statistics were utilized to describe the data collected from the simulations. The *sample mean* is defined as:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{9.7}$$

The *sample variance* is defined as:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2, \tag{9.8}$$

Furthermore, the mean represents the average value of the sample. For $\alpha \in (0, 1)$, a real number q_α is called the α-*quantile* if:

$$P(X \leq q_\alpha) \geq \alpha. \tag{9.9}$$

If q_α is the α-quantile of a sample, then $\alpha\%$ of the instances in the sample distribution are greater than q_α. For $\alpha = 0.5$, the quantile $q_{0.50}$ splits the sample dataset into two equal sizes and is called the *median*. The difference

between the 95% quantile and the 5% quantile is called the *interquantile* range:

$$\text{Range} = q_{0.95} - q_{0.05}, \tag{9.10}$$

Moreover, the interquantile range measures the spread of the data.

Bayesian Data Analysis

In this study, *Bayesian Data Analysis* (BDA) was utilized due to the many disadvantages of frequentistic statistics that have been reported [110], [111]. BDA is recommended for empirical software engineering due to its ability to mitigate the shortcomings of the frequentistic approach [112], and has previously been utilized to analyze other benchmarks [116], [306].

The *classical* view of statistics expresses probability in terms of random repeatable events. However, many events are not repeatable, so the classical view of viewing probability becomes useless. The existence of non-repeatable events motivates the *Bayesian* viewpoint to express probability as a measurement of uncertainty. This uncertainty is updated through new evidence. Suppose prior information of the parameter θ is available before observing evidence x. This prior information is expressed in a *prior* probability distribution $p(\theta)$. After observing evidence x, the updated information is expressed through the *posterior probability* $p(x|\theta)$ and is calculated with Bayes formula [115]:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}, \tag{9.11}$$

where $p(x)$ is the marginal distribution. Utilizing Bayesian methods for modeling θ is advantageous because posterior distributions consider all values of θ compared to frequentistic statistics where θ is treated as a scalar. *Prior predictive checks* and *posterior predictive checks* are utilized to evaluate the suitability of the prior distribution and the quality of the resulting posterior distribution. Predictive checks are intuitive methods for evaluating results compared to test statistics and p-values [110]–[112].

Algorithms

The deep semi-supervised learning algorithms in USB are *inductive*. Inductive algorithms work just as supervised learning because the algorithms are

trained with training and testing sets. However, semi-supervised learning utilizes both labeled and unlabeled data in the training set. The deep semi-supervised learning algorithms in USB were chosen for evaluation due to their popularity [307]. Fourteen different deep semi-supervised learning algorithms were evaluated in this study: Π-Model [73] **(pimodel)**, Mean-Teacher [74]**(meanteacher)**, Pseudo-Label [28]**(pseudolabel)**, VAT [304]**(vat)**, MixMatch [81]**(mixmatch)**, ReMixMatch [83]**(remixmatch)**, UDA [305]**(uda)**, FixMatch [75]**(fixmatch)**, FlexMatch [78]**(flexmatch)**, Dash [76]**(dash)**, AdaMatch [77]**(adamatch)**, CRMatch [65]**(crmatch)**, CoMatch [67]**(comatch)**, SimMatch [66]**(simmatch)**, SoftMatch [79]**(softmatch)** and FreeMatch [80]**(freematch)**. Supervised learning **(supervised)** was included in the evaluation to investigate when unlabeled data reduces the error rate.

Datasets

The 15 datasets utilized in this study are found in the list below. There are five datasets for each datatype: *image, text* and *audio*.

- **Image data:**
  - **Cifar-100:** [308]. The dataset contains 32×32 color images divided into 100 classes. Each class contains 600 images each.
  - **STL-10:** [309] The dataset contains 96×96 pixel images divided into ten classes: *airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck*. Each class contains 1300 instances each. The images were collected from ImageNet.
  - **EuroSat:** [310], [311] The dataset contains 64×64 pixel images divided into ten classes, *AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, Permanent Crop, Residental, River and SealLake*. All classes contain 3000 instances each, except *Permanent Crop* and *River* which contain 2500 instances each.
  - **TissueMNIST:** [312], [313] The dataset contains $32 \times 32 \times 7$ greyscale images of kidney cortex cells. There are eight classes and a total of 236386 instances.
  - **Semi-Aves:** [314] The dataset contains images of birds divided into 1000 classes of different Aves bird species. The images are

sampled from the iNat-2018 dataset. There are 12220 images and each class contains 23-250 instances.

- **Text data:**
  - **IMDB:** [315] The dataset contains movie reviews labeled as positive or negative. The dataset is utilized for binary sentiment classification and contains 50000 instances.
  - **AG News:** [316] The dataset contains news articles collected from 2000 online web sources. There are four classes: *world, sports, business*, and *sci-tech*. Each class contains 31900 instances for a total of 127600 instances.
  - **Amazon Review:** [317] This dataset contains reviews from Amazon. It contains 233.1 million instances distributed across 5 classes.
  - **Yahoo! Answers:** [318] The dataset is a sample from the original corpus provided by the Yahoo! Research Alliance Webscope Program. The dataset is a text classification benchmark and contains the ten largest classes from the original dataset: *Security & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relations*, and *Politics & Government*. Each class contains 14600 instances for a total of 146000 instances.
  - **Yelp Review:** [319] The dataset contains reviews from Yelp and is divided into five classes: 1,2,3,4,5. There is a total of 10000 instances [320].
- **Audio**
  - **GTZAN:** The dataset contains 30-second-long audio files. The dataset is divided into ten classes: *blue, classical, country, disco, hip-hop, jazz, metal, pop, reggae,* and *rock*. Each class contains 100 instances for each class.
  - **UrbanSound8K:**[319] The dataset contains 4-second-long audio files in .wav format. There are ten classes: *air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren,* and *street music*. The dataset contains a total of 8732 instances.

- **FZDnoisy18K:** [321] The dataset contains 42.5 hours of audio from Freesound. There are 16860 instances in total, distributed across 20 classes: *Acoustic guitar, Bass guitar, Clapping, Coin, Crash cymbal, Dishes pots and pans, Engine, Fart, Fire, Fireworks, Class, Hi-hat, Piano, Rain, Slam, Sueak, Tearing, Walk footsteps, Wind,* and *Writing.*
- **Keyword Spotting:** [322] The dataset contains audio files of people saying one-word commands. The dataset contains more than 60000 audio files distributed across 30 classes.
- **Esc50:** [323]. The dataset contains 5-second-long environmental audio recordings divided into 50 semantic classes. Each class contains 40 instances.

Data Collection

This study utilized the results from evaluations previously performed by Microsoft. The simulation results are found at the USB GitHub repository [1]. All choices of hyperparameters for the simulations are found in the supplementary material of the original paper [84]. The USB repository contains intervals around the mean error rate for each algorithm on each dataset. The intervals are on the form $(\overline{x} \pm m)$, where x is the error rate. Assuming that $(\overline{x} \pm m)$ is a $(1-\alpha)\%$ CI and that $\overline{x} \sim \text{Normal}(\mu, \sigma)$ the $(1-\alpha)\%$ CI for μ was derived. Data was simulated from the $(1-\alpha)\%$ interval of μ is given by:

$$\overline{x} - \lambda_{\alpha/2}\frac{s}{\sqrt{n}} \leq \mu \leq \overline{x} + \lambda_{\alpha/2}\frac{s}{\sqrt{n}}. \tag{9.12}$$

If $m = \lambda_{\alpha/2}\frac{s}{\sqrt{n}}$ then $s = m\lambda_{\alpha/2}\sqrt{n}$ where $\lambda_{\alpha/2}$ is the $\alpha/2$-quantile. Therefore the true distribution of μ is $\text{Normal}(\overline{x}, m\lambda_{\alpha/2})$. For this study, the parameters were chosen as $n = 1000$ and $\alpha = 0.05$.

Experimental setup

The number of available labels was varied to answer questions regarding manual effort. This paper considers the two cases where the training contains a "small" number of labels and a "large" number of available labels. For "small" the number of labels varies depending on the dataset and for "large", there are

between 2-5 times the "small" number of labels see table 9.1. The simulations were computed for ten iterations utilizing different random seeds, and results were saved in a .csv file called the *master dataset*. The pseudocode for creating the master dataset is found in algorithm 1, and a sample of the master dataset is illustrated in table 9.2.

Table 9.1: Summary table for the datasets. From left to right, the columns contain the datatype, the name of the dataset, the number of labels utilized for each class, the size of the training dataset, the size of the testing dataset and the number of classes in the dataset.

Datatype	Dataset	Labels (Small/Large)	Training data	Test data	Classes
Image	Cifar-100	2 / 4	50000	10000	100
	STL-10	2 / 4	50000	10000	100
	EuroSat	4 / 10	5000 / 10000	8000	10
	TissueMNIST	10 / 50	165466	47280	8
	Semi-Aves	15/53	5959 / 26640	4000	200
Text	IMDB	10/50	23000	25000	2
	Amazon Review	50 / 200	250000	65000	5
	Yelp Review	50/ 200	250000	50000	5
	AG News	10 / 50	100000	7600	4
	Yahoo! Answers	50 / 200	500000	60000	10
Sound	Keyword Spotting	5 / 20	18538	2567	10
	ESC-50	5 / 10	1200	400	50
	UrbanSound8K	10 / 40	7079	837	10
	FZDnoisy	52/171	1772 / 15813	947	20
	GTZAN	10 / 40	7000	1500	10

Algorithm 1 Creating the dataset

Require: n

for $d \in$ dataset **do**
  for $a \in$ algorithm **do**
    for $\ell \in$ available labels **do**
      Draw $(x_1, \cdots, x_n)$ from Normal$(\overline{x}, m\lambda_{\alpha/2})$.
Concatenate into a data frame ε_d with n rows.
Concatenate all $\varepsilon_1, \cdots, \varepsilon_d$ into one data frame see Table 2. 9.1.

Table 9.2: Sample of the generated dataset

error_rate	dataset	algorithm	iteration number	manual effort	datatype
35.788	fsdnoisy	pimodel	5	small	audio
61.232	yahoo_answers	pimodel	5	small	text
37.741	semi_aves	vat	6	large	image
34.721	yahoo_answers	vat	5	large	text
60.018	amazon_review	fixmatch	4	small	text
31.095	yahoo_answers	adamatch	1	large	text
48.716	gtzan	vat	8	large	audio
31.154	fsdnoisy	freematch	7	small	audio
60.009	urbansound8k	dash	9	small	audio
22.335	stl	softmatch	2	small	image

Data analysis

Bradley-Terry Model

The Bradley-Terry model described in section 9.2 was utilized with $\mu = a_{alg} + a_{bm}$. Here, a_{alg} and a_{bm} are the fixed effects of the algorithms and the benchmarks, respectively. The priors were chosen to have the following distributions.

$$a_{alg,i} \sim \text{Normal}(0, 2), \tag{9.13}$$

$$a_{bm,i,j} \sim \text{Normal}(0, s), \tag{9.14}$$

$$s \sim \text{Exponential}(0.1). \tag{9.15}$$

for all scenarios.

Generalized Linear Mixed Model (GLMM)

The model in section 9.2 was utilized with $a = a_{alg} + a_{bm}$ and $b = b_{noise}$. Here, a_{alg} is the fixed effect of the algorithm, and a_{bm} is the fixed effect of

each benchmark. The priors were chosen to have the following distributions:

$$a_{alg,i} \sim \text{Normal}(0, d), \tag{9.16}$$
$$b_{noise,i} \sim \text{Normal}(0, d), \tag{9.17}$$
$$a_{bm,j} \sim \text{Normal}(0, s), \tag{9.18}$$
$$s \sim \text{Exponential}(z). \tag{9.19}$$

where the variables d, z were chosen so that all the MCMC chains converge for each of the scenarios. For aggregated data $d = 5,\ z = 0.1$, for audio $d = 8.5,\ s = 1.9$, for images, text, small allocation, and large allocation of labels $d = 8.5,\ z = 2.2$.

9.4 Results

This section presents the results from the data analysis and summarises the results into guidelines for practitioners. The measurements utilized to describe the results are discussed in section 9.3.

Tables 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, illustrate descriptive statistics for the error rates for each scenario. From left to right, the columns in the tables illustrate the name of the algorithm, the median, 5% quantile, 95% quantile, and the interquantile range.

Table 9.3 illustrates the descriptive statistics of the error rate of the aggregated data. The descriptive statistics demonstrate that AdaMatch is the algorithm with the lowest error rate. Tables 9.4, 9.5 and 9.6 demonstrate that the algorithm with the lowest error rate differs when investigating datatype individually. The algorithms with the lowest error rates for image, text and audio are SimMatch, FreeMatch and SimMatch, respectively. Similarly, tables 9.7 and 9.8 demonstrate that the algorithm with the lowest error rate is different for small and large allocations of labels. For small allocation of labels, the algorithm with the lowest error rate is AdaMatch, and for large allocation of labels, the algorithm with the lowest error rate is SimMatch. In addition, the error rate decreases as the number of labels increases.

Analysis of the ranks

The master dataset was altered differently to answer each research question before applying the models. To obtain the aggregated results, the column that contains manual effort and datatype was dropped from the master dataset. The manual effort column was ignored to obtain the results based on the datatype. Finally, the column containing the datatype was dropped to analyze the results based on manual effort.

Tables 9.9, 9.10, 9.11, 9.12, 9.13, 9.14 illustrate the rankings of the algorithms calculated utilizing the Bradley-Terry model described in section 9.3. The ranks are utilized to answer RQ1-RQ3. From left to right, the columns illustrate the name of the algorithm, the median rank and the sample variance of the ranks. The high variance indicates uncertainty in the estimated ranks and explains why the ranks of many algorithms are tied. Table 9.15 illustrates which algorithms are always in the top-3 highest-ranking algorithms and table 9.16 illustrates the algorithms that are tied.

Table 9.3: Summary statistics for the error rate (aggregated)

Algorithm	Median	5%	95%	Range
adamatch	26.113	3.275	52.843	49.568
crmatch	28.846	2.467	57.829	55.362
comatch	29.818	4.458	56.191	51.733
dash	31.050	3.482	56.297	52.816
fixmatch	28.684	2.522	56.985	54.463
flexmatch	29.995	2.674	68.091	65.417
freematch	26.429	2.874	54.844	51.970
meanteacher	32.472	5.437	63.772	58.335
mixmatch	44.701	10.334	74.372	64.038
pimodel	41.073	11.866	81.752	69.886
pseudolabel	34.859	5.084	60.871	55.786
remixmatch	80.000	8.180	98.000	89.820
simmatch	27.347	2.658	52.637	49.979
softmatch	27.924	2.260	59.830	57.569
supervised	33.863	4.854	59.981	55.127
uda	33.643	7.380	93.393	86.013
vat	33.638	2.853	64.891	62.038

Table 9.4: Summary statistics for the error rate (image data)

Algorithm	Median	5%	95%	Range
adamatch	21.804	3.626	59.667	56.041
crmatch	24.985	10.170	62.449	52.279
comatch	28.231	3.629	65.146	61.517
dash	27.005	5.325	58.286	52.960
fixmatch	29.135	4.584	61.240	56.655
flexmatch	27.413	4.810	81.700	76.890
freematch	23.442	3.323	60.397	57.074
meanteacher	30.730	3.505	60.562	57.057
miixmatch	40.268	21.788	65.746	43.958
pimodel	36.693	10.415	76.959	66.544
pseudolabel	30.067	3.817	65.154	61.338
remixmatch	25.862	2.387	63.580	61.194
simmatch	21.037	4.686	57.470	52.784
softmatch	23.329	2.976	74.020	71.043
supervised	33.287	6.411	60.125	53.714
uda	25.866	6.332	62.387	56.054
vat	27.406	8.467	58.261	49.795

Analysis of the porbability of success

This research question was answered with respect to aggregated results, datatype, and manual effort. To answer RQ4, the following operations were performed on all three of the datasets that were utilized to answer the previous RQs. First, a copy of the dataset was made. A new column called "SD" (for standard deviation) was added to copied and original variants. In the original dataset, $SD = 0$ to indicate the absence of noise. In the copied dataset, $SD = 3$ to indicate noise in the data. To account for noise we simulated accuracy from a normal distribution with mean y and standard deviation 3. After the operations on the copied dataset were finished, both datasets were concatenated by row into a new dataset. When the GLMM was applied, the odds ratio (OR) of each algorithm's intercept (a_{alg}) and noise (b_{noise}) were computed utilizing the new dataset. OR measures the relative probability of success compared to the probability of failure. An OR > 1 means that the

Table 9.5: Summary statistics for the error rate (text data)

Algorithm	Median	5%	95%	Range
adamatch	30.827	6.236	52.806	46.570
crmatch	32.674	4.806	56.839	52.033
comatch	33.274	4.459	54.208	49.749
dash	35.263	3.627	57.921	54.294
fixmatch	32.528	3.055	59.410	56.355
flexmatch	33.466	3.598	53.348	49.751
freematch	29.669	5.889	52.176	46.286
meanteacher	38.513	8.153	63.545	55.391
mixmatch	44.317	8.236	78.126	69.889
pimodel	50.667	15.160	86.742	71.581
pseudolabel	42.917	7.708	58.447	50.740
remixmatch	80.000	50.000	90.000	40.000
simmatch	32.439	5.377	52.301	46.924
softmatch	33.866	4.226	51.319	47.092
supervised	37.000	8.431	63.814	55.384
uda	57.691	21.537	115.187	93.651
vat	38.485	4.520	83.389	78.869

parameter increases the probability of success. If $0 \leq \text{OR} < 1$, the parameter decreases the probability of success and if $\text{OR} = 1$, the probability of success is unchanged.

The ORs are located in tables 9.17 9.18, 9.19, 9.20, 9.21, and 9.22. No algorithm has $\text{OR} > 1$ for the fixed effects in any scenario. A summary of the algorithms that have an $\text{OR} > 1$ for the noise parameters is located in table 9.23.

Guidelines for practitioners

The recommended algorithms are presented in table 9.24 based on the ranks and the probability of success. The recommendations are based on the algorithms in the top-three highest-ranking algorithms and have $\text{OR} > 1$ in the noise parameter. In other words, the algorithms achieve a low error rate and perform well in the presence of noise.

Table 9.6: Summary statistics for the error rate (audio data)

Algorithm	Median	5%	95%	Range
adamatch	25.213	2.491	39.255	36.764
crmatch	25.369	1.325	54.504	53.180
comatch	25.677	9.192	47.807	38.615
dash	31.802	1.852	48.570	46.718
fixmatch	23.316	1.854	49.496	47.641
flexmatch	29.551	2.222	49.338	47.116
freematch	26.333	2.368	57.522	55.155
meanteacher	29.383	5.494	63.724	58.230
mixmatch	50.611	9.969	75.715	65.746
pimodel	39.693	19.141	64.029	44.887
pseudolabel	30.908	4.929	59.690	54.761
remixmatch	93.246	75.096	105.270	30.173
simmatch	22.890	2.157	46.629	44.472
softmatch	26.346	1.366	44.198	42.832
supervised	32.137	1.828	53.046	51.219
uda	27.960	5.619	52.383	46.764
vat	34.829	2.296	51.224	48.929

Practitioners are recommended to try FreeMatch because it is among the top three highest-ranking algorithms for each scenario. It has OR > 1 for the noise parameter in each scenario except for a small allocation of labels. Therefore, FreeMatch is recommended for all datatypes but works better with more labels. If utilizing image datasets and a small allocation of labels and practitioners are not satisfied with FreeMatch, they are recommended to try SimMatch. It shares the second-highest ranking algorithm spot with FreeMatch and has OR > 1 for the noise parameter. For text datasets and a small allocation of labels, practitioners are recommended to utilize SoftMatch if they are not satisfied with FreeMatch. SoftMatch is in the top-three highest ranking algorithms and outranks FreeMatch for text datasets. In addition, SoftMatch has OR > 1 for the noise parameter for text and a small allocation of labels.

Table 9.7: Summary statistics for the error rate (small allocation of labels)

Algorithm	Median	5%	95%	Range
adamatch	27.062	2.652	54.640	51.988
crmatch	31.654	2.834	60.215	57.381
comatch	33.542	5.231	62.719	57.488
dash	34.553	4.164	58.762	54.598
fixmatch	31.501	2.617	55.869	53.252
flexmatch	31.694	2.659	60.062	57.404
freematch	29.162	2.935	58.859	55.925
meanteacher	36.885	5.817	65.507	59.691
mixmatch	46.660	10.088	74.324	64.235
pimodel	49.472	20.681	85.018	64.337
pseudolabel	39.232	6.954	67.484	60.530
remixmatch	80.000	8.451	98.000	89.549
simmatch	29.969	2.401	54.586	52.185
softmatch	29.811	2.692	67.251	64.559
supervised	38.428	5.384	65.532	60.148
uda	40.277	9.917	86.941	77.025
vat	36.938	2.847	81.862	79.014

9.5 Discussion

Semi-supervised learning is a combination of supervised learning and unsupervised learning where unlabeled data is utilized to improve supervised learning [63]. An unsupervised classifier is said to improve supervised learning well if it helps the classifier predict the correct label and if it provides *fairness*. Fairness means that the model outputs each class label with an equal frequency given that the class distribution in the training data is uniform [63]. Fairness is obtained when the mutual information is maximized [63].

The objective of semi-supervised learning algorithms is to minimize the total loss function, defined as the sum of a supervised loss and an unsupervised loss [28], [307]. The supervised loss involves labeled data, and the unsupervised loss involves unlabeled data. Semi-supervised learning algorithms based on other machine learning methods are available. However, previous empirical and theoretical studies demonstrate that semi-supervised machine learning

Table 9.8: Summary statistics for the error rate (large allocation of labels)

Algorithm	Median	5%	95%	Range
adamatch	25.202	4.501	49.681	45.180
crmatch	28.056	1.813	47.874	46.061
comatch	25.651	4.135	48.926	44.791
dash	28.555	2.470	50.240	47.771
fixmatch	22.285	2.500	59.410	56.910
flexmatch	27.987	3.188	84.791	81.602
freematch	24.992	4.049	48.096	44.046
meanteacher	30.143	5.206	54.357	49.151
mixmatch	38.777	11.146	74.636	63.489
pimodel	35.722	8.892	61.902	53.011
pseudolabel	30.195	3.895	52.304	48.409
remixmatch	80.000	9.777	98.000	88.223
simmatch	24.682	3.272	50.560	47.288
softmatch	25.487	1.989	44.927	42.938
supervised	30.293	4.768	50.468	45.700
uda	31.854	5.662	98.290	92.628
vat	29.984	2.987	52.603	49.616

algorithms may degrade performance [142], [324].

The quantity-quality tradeoff

In recent years, deep semi-supervised learning has increased in popularity due to the success of the FixMatch algorithm [75]. Previous algorithms like UDA, MixMatch and ReMixMatch precede FixMatch and are all inferior in many scenarios [75]. FixMatch takes many ideas from the previous techniques and simplifies them yet achieves better performance [75].

Many modern deep semi-supervised learning algorithms including FixMatch, utilize *pseudo-labeling* [28], and *consistency regularization* [305], [325]. Pseudo-labeling is a semi-supervised technique that trains a supervised classifier to classify pseudo-labels for unlabeled instances. These pseudo-labels improve the generalization performance by maximizing the conditional log-likelihood and minimizing the entropy of unlabeled data [28].

Table 9.9: Ranking of the algorithms (aggregated data)

Models	Median Rank	Variance of the Rank
adamatch	2	3.644
simmatch	3	4.649
freematch	4	4.565
softmatch	4	4.972
crmatch	5	4.918
fixmatch	5	5.054
flexmatch	5	5.080
dash	8	3.452
comatch	9	3.587
meanteacher	11	2.763
supervised	11	2.583
vat	11	2.497
uda	12	2.366
pseudolabel	14	1.610
pimodel	15	1.140
mixmatch	16	0.771
remixmatch	17	0.005

A problem with semi-supervised learning is that if labeled data is scarce, the supervised classifier will perform poorly and produce low-quality pseudo-labels [62]. To mitigate this, FixMatch utilizes a threshold that makes sure that only the pseudo-labels of sufficient quality are utilized. The consistency regularization aspect of FixMatch is to minimize the cross-entropy between the predictive distributions of the class labels given weakly augmented and strongly augmented instances. Utilizing pseudo-labeling leads to the quantity-quality trade-off [79]. The quantity-quality trade-off states that higher thresholds lead to fewer pseudo-labels in the training set. Based on the results of this study, the optimal algorithms are FreeMatch, SimMatch and SoftMatch, which all extend upon FixMatch and try to provide both high quality and quantity but in different ways. SimMatch and SoftMatch were both developed in the same year and were originally not compared to each other. FlexMatch was published a year after SoftMatch and SimMatch and consequently was not compared to these either. Therefore, none of the three was explicitly designed

Table 9.10: Ranking of the algorithms (image data)

Models	Median Rank	Variance of the Rank
freematch	3	7.323
simmatch	3	7.169
adamatch	4	8.649
remixmatch	4	8.201
flexmatch	6	10.034
uda	6	10.349
crmatch	7	11.142
dash	7	10.471
fixmatch	8	11.620
softmatch	8	10.730
comatch	12	9.570
supervised	12	8.426
mixmatch	14	6.564
pimodel	14	6.689
vat	14	7.911
pseudolabel	15	5.983
meanteacher	16	3.951

to outperform one another.

The pseudo-labels that FixMatch computes are called semantic pseudo-labels. SimMatch extends FixMatch and strives to include more high-quality pseudo labels by utilizing both semantic and instance pseudo labels. The instance pseudo-labels are calculated utilizing similarity distributions. Semantic and instance pseudo-labels are then matched to belong to the same class. The instance pseudo-labels are inputs in a third loss function called the *instance loss*. Unlike other semi-supervised learning algorithms, SimMatch is unique in this regard. The previous state-of-the-art algorithm was CoMatch which is based on consistency regularization and contrastive learning [67]. CoMatch utilizes similarity matching through label distribution, but SimMatch is faster, more robust and achieves better performance than CoMatch. The FixMatch and SimMatch algorithms utilize a fixed threshold during training to maintain high-quality pseudo-labels. On the downside, the algorithms discard many labels and reduce quality. Other algorithms, such as Dash and AdaMatch, uti-

Table 9.11: Ranking of the algorithms (text data)

Models	Median Rank	Variance of the Rank
adamatch	3	5.406
crmatch	4	5.990
softmatch	4	6.509
comatch	5	6.632
fixmatch	5	6.670
freematch	5	6.802
simmatch	5	6.665
dash	7	6.330
flexmatch	7	5.657
meanteacher	10	2.270
supervised	11	1.719
vat	12	1.795
mixmatch	14	1.585
pimodel	14	1.581
pseudolabel	14	1.490
uda	16	0.220
remixmatch	17	0.003

lize a dynamically increasing threshold to outperform algorithms that utilize a fixed threshold.

Like FixMatch, SoftMatch minimizes a total loss that is decomposed into a supervised and an unsupervised loss. The inputs of these losses are the same as FixMatch, but the unsupervised loss is the weighted cross entropy, which requires a sample weight function. The sample weight function in SoftMatch is assumed to have a Gaussian truncated distribution whose mean and variance are estimated utilizing historical predictions of the model's exponential moving average. Therefore, the threshold varies at each time stamp, and theoretical arguments demonstrate that SoftMatch provides better quantity and quality over UDA, FixMatch and FlexMatch [79].

The FreeMatch algorithm utilizes supervised and unsupervised loss as FixMatch, but also calculates the self-adaptive threshold (SAT) to balance the quantity-quality trade-off by automatically adjusting the threshold during training. The threshold is low at the start of training, but as training con-

Table 9.12: Ranking of the algorithms (audio data)

Models	Median Rank	Variance of the Rank
fixmatch	3	6.201
softmatch	3	5.346
adamatch	4	6.839
simmatch	4	6.404
freematch	6	8.865
uda	6	9.413
crmatch	7	8.640
flexmatch	7	8.571
comatch	10	8.254
dash	11	7.286
meanteacher	11	8.005
vat	11	7.784
supervised	12	6.196
pseudolabel	13	4.673
pimodel	15	0.729
mixmatch	16	0.195
remixmatch	17	0.000

tinues, it becomes more confident and increases. The SAT is calculated by combining two other thresholds, known as global and local thresholds. The global threshold represents the model's confidence in unlabeled data and is computed utilizing EMA. The local threshold is class-specific and, therefore, considers that different class labels are easier than others to predict. Finally, fairness is included to make the label predictions fair. From simulations, it is observed that FreeMatch achieves superior performance on various benchmarks. ReMixMatch and UDA may outperform FreeMatch due to the MixUp property [82]. The results of [80] demonstrate that FreeMatch has a lower threshold in the early learning process than FlexMatch and FixMatch and, therefore, utilizes more data than these [80].

Table 9.13: Ranking of the algorithms (small allocation of labeled data)

Models	Median Rank	Variance of the Rank
adamatch	2	3.071
simmatch	3	3.671
flexmatch	4	4.233
freematch	4	4.108
softmatch	4	4.277
crmatch	5	4.484
fixmatch	6	4.405
dash	8	3.996
comatch	9	3.278
meanteacher	11	2.945
vat	11	3.007
supervised	12	2.661
uda	13	2.376
mixmatch	14	1.972
pseudolabel	15	1.383
pimodel	16	0.480
remixmatch	17	0.070

Benchmarking

Due to the rapid development and publication of deep semi-supervised learning algorithms, continuous benchmarking is necessary. Furthermore, benchmarking is only able to evaluate methods implemented in a current release of the software. New releases of a method may differ in accuracy and runtime, which is why permanent benchmarking efforts are necessary. In addition, datasets utilized to evaluate algorithms need to be updated due to the fact that many datasets are too easy for the algorithms to learn. In [230], *Item Response Theory* demonstrates that many datasets utilized to evaluate graph-based semi-supervised learning algorithms are too easy to learn. Therefore, many algorithms will achieve low error rate on benchmark datasets but may perform differently on real-world datasets because they are more difficult to learn. In particular, only four out of 15 benchmark datasets are suitable for evaluating graph-based semi-supervised learning [230]. Similarly, [234] demonstrates that many supervised machine learning algorithms suffer

Table 9.14: Ranking of the algorithms (large allocation of labeled data)

Models	Median Rank	Variance of the Rank
fixmatch	2.5	4.899
adamatch	3.0	5.053
freematch	4.0	7.280
simmatch	4.0	5.209
softmatch	5.0	7.281
crmatch	6.0	7.472
comatch	6.0	8.069
flexmatch	7.0	8.249
dash	9.0	7.606
supervised	11.0	6.400
uda	11.0	7.358
meanteacher	13.0	4.504
pimodel	13.0	4.629
vat	13.0	5.253
pseudolabel	14.0	3.375
mixmatch	16.0	0.157
remixmatch	17.0	0.089

from the same problem. Furthermore, USB utilizes many benchmark datasets that have been proven too easy for supervised learning and graph-based semi-supervised learning. Therefore, determining whether these datasets are suitable for evaluating the algorithms contained in USB is essential. Datasets must be updated or discarded with time due to their ability to evaluate the algorithms [230] and it is important to include real-world datasets in evaluations. Furthermore, benchmarking is essential for the industry as they must know what algorithms to utilize on their real-world datasets.

Robustness

This study does not utilize real-world datasets in the evaluation. Instead, noise is added to the benchmark datasets to make them more representative of real-world data and make the results more generalizable. When constructing machine learning algorithms, it is essential to consider accuracy and

Table 9.15: Summary of top-three highest ranking algorithms

	Aggrgated	Image	Text	Audio	Small allocation of labels	Large allocation of labels
Π-model	NO	NO	NO	NO	NO	NO
Mean-Teacher	NO	NO	NO	NO	NO	NO
Pseudo-Label	NO	NO	NO	NO	NO	NO
VAT	NO	NO	NO	NO	NO	NO
MixMatch	NO	NO	NO	NO	NO	NO
ReMixMatch	NO	YES	NO	NO	NO	NO
UDA	NO	YES	NO	YES	NO	NO
FixMatch	NO	NO	YES	YES	NO	YES
FlexMatch	NO	YES	NO	NO	YES	NO
Dash	NO	NO	NO	NO	NO	NO
AdaMatch	YES	YES	YES	YES	YES	YES
CRMatch	NO	NO	YES	NO	NO	NO
CoMatch	NO	NO	NO	NO	NO	NO
SimMatch	YES	YES	YES	YES	YES	YES
SoftMatch	YES	NO	YES	YES	YES	NO
FreeMatch	YES	YES	YES	YES	YES	YES
Supervised	NO	NO	YES	NO	YES	YES

Table 9.16: Top-three highest ranking algorithms

Scenario	1st	2nd	3rd
Aggregated	AdaMatch	SimMatch	FreeMatch, SoftMatch
Image	FreeMatch, SimMatch	AdaMatch, ReMixMatch	FlexMatch, UDA
Text	AdaMatch	CRMatch, SoftMatch	CoMatch, FixMatch, FreeMatch, SimMatch
Audio	FixMatch, SoftMatch	AdaMatch, SimMatch	FreeMatch, UDA
Small allocation of labels	AdaMatch	SimMatch	FlexMatch, FreeMatch, SoftMatch
Large allocation of labels	FixMatch	AdaMatch	FreeMatch, SimMatch

robustness. Many machine learning algorithms rely on assumptions on the data. Both supervised and semi-supervised learning rely on empirical risk minimization [241], which means that all instances in the training set is from the same distribution. Therefore there is no guarantee that the trained algorithm generalizes well on data that is out of distribution (OOD). There are many different measurements for evaluating the robustness of an algorithm, such as *robustness measure* [239] and *coefficient of variation* [240]. The results of this study demonstrate that noise may improve the performance of semi-supervised algorithms. More specifically, noise increases the probability of achieving error rate below 10% for all datatypes. Noise has previously been demonstrated to improve the performance of algorithms in other studies [237], [240], [242]. In [240], three deep learning algorithms are evaluated across image datasets and their performance is compared between clean data and perturbed data. The results demonstrate that perturbed data improves robustness, and error rate in many cases [240]. Similarly, [242] compares the error rate of machine learning algorithms evaluated on text datasets where the amount of noise added to the samples varies between 0%-100%. The results

demonstrate that adding noise up to 40% will leave the error rate unchanged, and adding noise up to 70% will only increase the error rate slightly. Since noise may degrade or increase performance, it is necessary to investigate the impact of noise on the probability of successfully obtaining an error rate of 10% or less by simulating accuracies as described in section 9.3. The results of this study demonstrates that noise increases the probability of successfully achieving error rate below 10% for all datatypes, which is confirmed by [237], [240].

Comparison with the original evaluation

This paper evaluates two additional algorithms SoftMatch and FreeMatch that have been added to USB since the original paper evaluation [84]. This paper utilizes Bayesian modeling, which has many advantages and provides a more fair evaluation due to the many benefits of Bayesian analysis.

The Bayesian Bradley-Terry ranks obtained in this study differ from the Friedman Ranks obtained in the original paper [84]. The Bayesian Bradley-Terry model accounts for the uncertainty that is associated with the variation the error rate. Therefore, this study provides less misleading ranks than [84]. The original paper does not compare aggregated data or consider the number of available labels [84]. It only considers datatypes and does not give practitioners an idea of how many labels are required to achieve the lowest possible error rate.

The Friedman ranks obtained from the original study [84] are located in table 9.25. The original study had CRMatch in the top three highest ranking algorithms for images, but UDA replaced it in this study. The results of this study are the same as the original for text and audio. However, the rank order is different and more algorithms are top-ranked in this study because the Bradley-Terry ranks incorporate uncertainty. Therefore there are ties in this study. Friedman ranks do not incorporate uncertainty, so there are no ties in [84].

9.6 Threats to Validity

This section discusses four types of threats to validity for simulation studies described in [104].

First is *Construct Validity*, which refers to how appropriate the statistical model is for answering the research questions. The RQs of this paper are concerned with ranking semi-supervised learning algorithms according to the lowest error rate. The Bayesian Bradley-Terry model was created for ranking and is appropriate for answering our RQs. The Bayesian Linear Regression model is utilized to calculate the probability of exceeding a certain threshold and is therefore appropriate to calculate the probability of an algorithm to achieve an error rate below 10%.

Second is *External Validity*, which refers to how generalizable the results are to other situations. In [84], the algorithms are evaluated on the same datasets. The algorithms have previously been evaluated in a similar way but with Friedman ranks in [84]. Thanks to the replication package, the results may be replicated and there is external validity.

Third is *Internal Validity*, which refers to whether the independent variables cause the outcome because simplification was made in the machine learning model or because some factors were not accounted for. In this study, no factors were ommited or any simplification in the models were made. Thus internal validity is ensured.

The final threat is *Conclusion Validity*, which refers to whether the results were evaluated utilizing appropriate statistical tests. In the study, posterior predictive checks are performed and the number of efficient examples and Gelman-Rubin potential scale reduction are interpreted [252] to evaluate the results.

9.7 Conclusion

This study analyzes deep semi-supervised learning algorithms and presents a framework for what algorithms to utilize in industrial situations to obtain a certain error rate. The study provides an updated evaluation of USB that includes more algorithms added since the first evaluation [84]. In addition, this study investigates the impact of noise to understand how the algorithms will perform on real-world datasets.

According to the results, none of the algorithms have an error rate below 10%. The original simulations [84] were run utilizing different baseline deep learning models, and different hyperparameters were utilized for each task and varied across algorithms. It is possible to achieve a lower error rate if different

supervised baselines are utilized and the hyperparameters are appropriately set for a given dataset. A takeaway is that many of the semi-supervised learning algorithms outperform supervised learning, and therefore, utilizing unlabeled data is relevant to improving the error rate. In addition, the results also demonstrate that the more labels are available, the better the error rate.

Generally, practitioners are recommended to investigate three algorithms: FreeMatch, SimMatch and SoftMatch. FreeMatch is recommended on all datatypes and for large allocations of labels. SimMatch is recommended for utilize on image data and a small allocation of labels. Finally, it is recommended that SoftMatch be utilized for text data types and small allocation of labels. The algorithms are recommended for these scenarios since the results demonstrate the algorithms are the highest-ranking and perform well on real-world data.

The results of this study help machine learning specialists in industry and academia determine which algorithm will have the lowest error rate.

For future simulation studies, it is relevant to examine these algorithms utilizing other statistical models such as Bayesian regression [116] to answer related RQs and evaluate other types of semi-supervised learning algorithms not included in USB. Another interesting study is to utilize the item response theory to investigate if the 15 datasets are appropriate for evaluating USB.

Table 9.17: Odds ratios for fixed effects and noise parameters (aggregated data)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.334	0.067	2.334	2.268
a_crmatch	0.206	0.041	1.436	1.395
a_comatch	0.183	0.036	1.249	1.213
a_dash	0.178	0.035	1.296	1.261
a_fixmatch	0.240	0.047	1.683	1.637
a_flexmatch	0.297	0.058	2.119	2.061
a_freematch	0.175	0.034	1.240	1.206
a_meanteacher	0.092	0.018	0.635	0.617
a_mixmatch	0.021	0.004	0.154	0.150
a_pimodel	0.016	0.003	0.113	0.110
a_pseudolabel	0.098	0.019	0.680	0.660
a_remixmatch	0.034	0.006	0.251	0.245
a_simmatch	0.261	0.051	1.859	1.807
a_softmatch	0.334	0.065	2.353	2.288
a_supervised	0.057	0.011	0.411	0.399
a_uda	0.052	0.010	0.370	0.360
a_vat	0.110	0.022	0.771	0.749
b_adamatch	0.929	0.800	1.078	0.278
b_crmatch	1.106	0.946	1.289	0.343
b_comatch	0.956	0.814	1.120	0.305
b_dash	0.970	0.828	1.139	0.311
b_fixmatch	0.976	0.838	1.139	0.301
b_flexmatch	0.976	0.835	1.139	0.304
b_freematch	1.033	0.879	1.216	0.337
b_meanteacher	0.980	0.829	1.158	0.329
b_mixmatch	1.087	0.864	1.363	0.499
b_pimodel	1.031	0.794	1.333	0.539
b_pseudolabel	1.000	0.852	1.177	0.325
b_remixmatch	0.999	0.815	1.233	0.418
b_simmatch	1.006	0.856	1.180	0.325
b_softmatch	0.973	0.836	1.134	0.298
b_supervised	1.032	0.864	1.238	0.375
b_uda	0.982	0.806	1.193	0.387
b_vat	0.998	0.848	1.170	0.322

Table 9.18: Odds ratios for fixed effects and noise parameters (image data)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.152	0.030	0.940	0.911
a_crmatch	0.011	0.002	0.079	0.077
a_comatch	0.140	0.027	0.867	0.840
a_dash	0.070	0.014	0.437	0.423
a_fixmatch	0.064	0.012	0.413	0.401
a_flexmatch	0.071	0.014	0.465	0.451
a_freematch	0.068	0.014	0.442	0.428
a_meanteacher	0.034	0.006	0.223	0.217
a_mixmatch	0.002	0.000	0.018	0.018
a_pimodel	0.014	0.002	0.095	0.092
a_pseudolabel	0.041	0.008	0.257	0.249
a_remixmatch	0.093	0.018	0.596	0.578
a_simmatch	0.063	0.012	0.413	0.401
a_softmatch	0.153	0.030	0.983	0.952
a_supervised	0.017	0.003	0.117	0.114
a_uda	0.033	0.006	0.219	0.213
a_vat	0.026	0.005	0.170	0.165
b_adamatch	0.952	0.763	1.182	0.420
b_crmatch	1.923	1.394	2.765	1.371
b_comatch	0.924	0.732	1.162	0.430
b_dash	0.936	0.737	1.192	0.455
b_fixmatch	0.995	0.773	1.275	0.502
b_flexmatch	1.031	0.815	1.302	0.487
b_freematch	1.032	0.813	1.309	0.496
b_meanteacher	0.864	0.650	1.141	0.490
b_mixmatch	0.951	0.354	2.450	2.096
b_pimodel	1.002	0.693	1.450	0.758
b_pseudolabel	0.999	0.786	1.273	0.487
b_remixmatch	0.997	0.789	1.256	0.467
b_simmatch	1.086	0.845	1.387	0.542
b_softmatch	0.968	0.770	1.220	0.450
b_supervised	1.055	0.769	1.447	0.678
b_uda	1.020	0.776	1.350	0.574
b_vat	0.999	0.756	1.320	0.564

Table 9.19: Odds ratios for fixed effects and noise parameters (text data)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.054	0.008	0.439	0.430
a_crmatch	0.127	0.019	1.098	1.079
a_comatch	0.055	0.008	0.455	0.447
a_dash	0.058	0.009	0.491	0.482
a_fixmatch	0.082	0.013	0.706	0.693
a_flexmatch	0.088	0.013	0.744	0.731
a_freematch	0.042	0.006	0.354	0.347
a_meanteacher	0.012	0.002	0.106	0.104
a_mixmatch	0.015	0.002	0.129	0.126
a_pimodel	0.006	0.001	0.053	0.052
a_pseudolabel	0.015	0.002	0.125	0.123
a_remixmatch	0.000	0.000	0.001	0.001
a_simmatch	0.060	0.009	0.493	0.484
a_softmatch	0.035	0.005	0.304	0.299
a_supervised	0.011	0.002	0.099	0.097
a_uda	0.001	0.000	0.013	0.013
a_vat	0.024	0.003	0.198	0.195
b_adamatch	0.893	0.688	1.155	0.466
b_crmatch	0.846	0.657	1.089	0.432
b_comatch	1.018	0.791	1.306	0.515
b_dash	0.975	0.753	1.269	0.516
b_fixmatch	0.946	0.734	1.213	0.479
b_flexmatch	0.906	0.694	1.190	0.496
b_freematch	1.011	0.791	1.298	0.507
b_meanteacher	1.172	0.852	1.618	0.766
b_mixmatch	1.066	0.781	1.459	0.678
b_pimodel	0.991	0.645	1.520	0.875
b_pseudolabel	0.996	0.726	1.366	0.639
b_remixmatch	0.003	0.000	3.455	3.455
b_simmatch	0.958	0.732	1.250	0.518
b_softmatch	1.065	0.813	1.391	0.578
b_supervised	1.059	0.768	1.469	0.702
b_uda	0.956	0.375	2.394	2.019
b_vat	1.024	0.774	1.358	0.584

Table 9.20: Odds ratios for fixed effects and noise parameters (audio data)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.469	0.077	3.310	3.233
a_crmatch	0.376	0.063	2.696	2.633
a_comatch	0.008	0.001	0.059	0.057
a_dash	0.039	0.006	0.292	0.285
a_fixmatch	0.206	0.033	1.490	1.457
a_flexmatch	0.585	0.100	4.030	3.929
a_freematch	0.097	0.015	0.727	0.711
a_meanteacher	0.110	0.017	0.825	0.808
a_mixmatch	0.004	0.001	0.027	0.026
a_pimodel	0.001	0.000	0.009	0.009
a_pseudolabel	0.079	0.013	0.586	0.573
a_remixmatch	0.000	0.000	0.001	0.001
a_simmatch	0.700	0.124	4.653	4.529
a_softmatch	0.862	0.156	5.896	5.741
a_supervised	0.034	0.006	0.235	0.229
a_uda	0.036	0.006	0.255	0.249
a_vat	0.210	0.035	1.614	1.579
b_adamatch	0.914	0.618	1.344	0.726
b_crmatch	0.996	0.670	1.491	0.821
b_comatch	0.919	0.636	1.323	0.688
b_dash	1.025	0.659	1.601	0.942
b_fixmatch	0.987	0.634	1.538	0.903
b_flexmatch	0.999	0.698	1.430	0.731
b_freematch	1.126	0.703	1.795	1.093
b_meanteacher	0.894	0.573	1.372	0.798
b_mixmatch	1.178	0.799	1.743	0.944
b_pimodel	1.166	0.664	2.096	1.432
b_pseudolabel	0.993	0.654	1.505	0.852
b_remixmatch	0.003	0.000	3.586	3.586
b_simmatch	0.932	0.660	1.308	0.648
b_softmatch	0.877	0.625	1.220	0.595
b_supervised	1.019	0.688	1.506	0.818
b_uda	0.887	0.589	1.350	0.760
b_vat	0.974	0.627	1.520	0.893

Table 9.21: Odds ratios for fixed effects and noise parameters (small allocation of labels)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.047	0.012	0.196	0.185
a_crmatch	0.036	0.009	0.151	0.142
a_comatch	0.028	0.007	0.121	0.114
a_dash	0.024	0.006	0.104	0.098
a_fixmatch	0.034	0.008	0.142	0.133
a_flexmatch	0.036	0.009	0.146	0.138
a_freematch	0.031	0.008	0.130	0.123
a_meanteacher	0.017	0.004	0.073	0.069
a_mixmatch	0.004	0.001	0.017	0.016
a_pimodel	0.000	0.000	0.000	0.000
a_pseudolabel	0.012	0.003	0.049	0.046
a_remixmatch	0.005	0.001	0.024	0.023
a_simmatch	0.052	0.013	0.223	0.210
a_softmatch	0.061	0.015	0.255	0.240
a_supervised	0.010	0.002	0.044	0.042
a_uda	0.004	0.001	0.021	0.020
a_vat	0.019	0.004	0.079	0.074
b_adamatch	0.912	0.715	1.167	0.452
b_crmatch	0.961	0.749	1.230	0.481
b_comatch	0.940	0.727	1.212	0.485
b_dash	0.970	0.744	1.261	0.517
b_fixmatch	0.965	0.751	1.233	0.482
b_flexmatch	0.995	0.773	1.288	0.515
b_freematch	0.995	0.771	1.281	0.510
b_meanteacher	1.045	0.802	1.367	0.564
b_mixmatch	1.001	0.693	1.459	0.766
b_pimodel	4.007	0.669	53.703	53.034
b_pseudolabel	0.995	0.748	1.331	0.582
b_remixmatch	0.930	0.655	1.312	0.657
b_simmatch	1.044	0.824	1.330	0.507
b_softmatch	1.008	0.794	1.270	0.476
b_supervised	1.031	0.765	1.403	0.638
b_uda	0.994	0.700	1.408	0.709
b_vat	0.897	0.679	1.186	0.507

Table 9.22: Odds ratios for fixed effects and noise parameters (large allocation of labels)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.103	0.027	0.390	0.363
a_crmatch	0.090	0.023	0.339	0.316
a_comatch	0.042	0.011	0.162	0.151
a_dash	0.054	0.014	0.205	0.191
a_fixmatch	0.093	0.025	0.342	0.317
a_flexmatch	0.149	0.039	0.556	0.517
a_freematch	0.048	0.013	0.183	0.170
a_meanteacher	0.019	0.005	0.074	0.069
a_mixmatch	0.005	0.001	0.020	0.019
a_pimodel	0.009	0.002	0.034	0.031
a_pseudolabel	0.035	0.009	0.134	0.125
a_remixmatch	0.006	0.001	0.023	0.022
a_simmatch	0.073	0.020	0.274	0.254
a_softmatch	0.093	0.025	0.350	0.326
a_supervised	0.013	0.003	0.052	0.049
a_uda	0.018	0.005	0.068	0.064
a_vat	0.031	0.008	0.115	0.107
b_adamatch	0.922	0.735	1.158	0.423
b_crmatch	1.053	0.828	1.337	0.510
b_comatch	0.972	0.776	1.217	0.441
b_dash	0.962	0.770	1.205	0.435
b_fixmatch	0.983	0.790	1.227	0.437
b_flexmatch	0.949	0.752	1.201	0.449
b_freematch	1.072	0.858	1.341	0.483
b_meanteacher	0.956	0.746	1.219	0.473
b_mixmatch	1.163	0.853	1.599	0.746
b_pimodel	1.043	0.783	1.393	0.610
b_pseudolabel	1.008	0.814	1.241	0.426
b_remixmatch	1.050	0.775	1.420	0.645
b_simmatch	0.981	0.779	1.229	0.450
b_softmatch	0.959	0.765	1.200	0.435
b_supervised	1.034	0.809	1.320	0.510
b_uda	0.936	0.720	1.217	0.496
b_vat	1.069	0.856	1.330	0.474

Table 9.23: Summary of algorithms with odds ration higher than one.

	Aggrgated	Image	Text	Audio	Small allocation of labels	Large allocation of labels
Π-model	YES	YES	NO	YES	YES	YES
Mean-Teacher	NO	NO	YES	NO	YES	NO
Pseudo-Label	YES	NO	NO	NO	NO	YES
VAT	NO	NO	YES	NO	NO	NO
MixMatch	YES	NO	NO	YES	YES	YES
ReMixMatch	NO	NO	NO	NO	NO	YES
UDA	NO	YES	NO	NO	NO	NO
FixMatch	PN	NO	NO	NO	NO	NO
FlexMatch	NO	YES	NO	NO	NO	NO
Dash	NO	NO	NO	YES	NO	NO
AdaMatch	NO	NO	NO	NO	NO	NO
CRMatch	YES	YES	NO	NO	NO	YES
CoMatch	NO	N0	YES	NO	NO	NO
SimMatch	YES	YES	NO	NO	YES	NO
SoftMatch	NO	NO	YES	NO	YES	NO
FreeMatch	YES	YES	YES	YES	NO	YES
Supervised	YES	YES	YES	NO	YES	YES

Table 9.24: Recommended Algorithms

Algorithm	Datatype	Comment
FreeMatch	All, large allocation of labels	In the top-3 highest ranking algorithms for every scenario Has OR > 1 in the presence of noise for every scenario except for small allocation of labels.
SimMatch	Images, small allocation of labels	Ties spot as highest ranking algorithm with FreeMatch. Has OR > 1 in the presence of noise for images and small allocation of labels.
SoftMatch	Text	Second-highest ranking algorithm Has OR > 1 in the presence of noise for text and small allocation of labels.

Table 9.25: Top-three highest ranking algorithms for the original study [84]

Scenario	1st	2nd	3rd
Image	ReMixMatch	CRMatch	AdaMatch
Text	SimMatch	CRMatch	CoMatch
Audio	AdaMatch	SimMatch	FixMatch

CHAPTER 10

Assessing the Suitability of Deep Semi-Supervised Learning Datasets using Item Response Theory

10.1 Introduction

Data Labeling is an important part of ML since DL models achieve higher accuracy when accessing large labeled datasets. However, labeling data is costly in terms of time and resources. Data labeling needs to be planned thoroughly, *Who should do the labeling?* and *How do we make sure that the labels are of high quality?* are some of the questions practitioners need to ask when planning how to obtain labeled data [16]. To eliminate manual data labeling practitioners can use Semi-Supervised Learning (SSL).

SSL combines supervised and unsupervised learning using labeled and unlabeled data [326]. There are two types of SSL algorithms, *inductive* and *transductive* algorithms [326]. Inductive learning performs classification like supervised learning but tries to improve the performance by using unlabeled instances [326]. Transductive algorithms work differently as they only classify the labels for the unlabeled instances of the input data [326].

As with all machine learning (ML) and deep learning (DL) based AI solu-

tions, finding the optimal model is a time-consuming task [16]. To save time choosing and calibrating models, practitioners often rely on benchmark studies [34] to choose algorithms [16]. Benchmark algorithms answer questions such as *how well does a certain algorithm perform on a given problem?* and *Why does an algorithm succeed/fail on a specific test problem?*. The problem with benchmark studies is that they sometimes use datasets that are too easy for the algorithms to learn. Algorithms evaluated on datasets that are too easy might not perform as well on real-world datasets that are more difficult to learn. Previous studies have shown that many benchmark datasets are unsuitable for evaluating SL [234] and graph-based SSL (GBSSL) [230]. The GBSSL algorithms are ranked on benchmarked datasets in [232] but have different rankings when evaluated on a real-world dataset from industry in [233].

We can use Item Response Theory (IRT) [243] to evaluate the sustainability of datasets used to evaluate a set of algorithms. IRT was originally developed to evaluate the mental health of students [122] and has previously been used to evaluate the sustainability of datasets for evaluating other types of algorithms [230], [234], [327].

Deep semi-supervised learning (DSSL) has recently increased in popularity due to its scalability [307]. The *Universal Semi-Sipervised Benchmark* (USB) is an open-source platform containing many state-of-the-art DSSL algorithms and benchmark datasets. Using USB, users can easily add their datasets and algorithms to evaluate with the other datasets and algorithms contained in USB. USB has previously been used in benchmark studies to illustrate its strengths [84].

To the best of our knowledge, nobody has evaluated the sustainability of datasets used to evaluate USB. To fill this research gap, we take the results of 14 DSSL algorithms evaluated on 15 datasets and use the congeneric model [126] to identify the datasets that are too easy, and we can determine what datasets are suitable for evaluating DSSL algorithms.

Thanks to the results of this study, practitioners will know what datasets are suitable for evaluating the USB algorithms and whether it is worth using the USB algorithms on real-world datasets. If all the datasets in USB are too easy, then practitioners can not know for certain that the algorithms perform equally well on their real-world datasets. By identifying the too easy datasets, we teach practitioners what datasets to replace with more difficult datasets

and create optimal benchmarks. The optimal benchmarks will lead to better empirical evaluation of DSSL algorithms that provide better insight into the performance of USB on real-world datasets. In addition, the new benchmarks can be used when developing new DSSL algorithms to help practitioners understand how the new algorithms will perform on real-world data.

The paper is organized as follows. Section 10.2 describes the theory behind semi-supervised learning and test theory. Section 10.3 outlines the research method, how the simulations were set up, what software packages were used and how the algorithms were evaluated. Section 10.4 presents the results, and section 10.5 discusses the results. Finally, the paper is concluded in section 10.6. A replication package together with an in-depth analysis of the validity of the Bayesian Data Analysis can be found at the repository [1].

10.2 Background

This section discusses related work and provides an overview of the algorithms and statistical tools utilized in this study.

Labeling challenge in Software Engineering

Data Labeling is an essential part of ML projects and takes up 80% of its time [16]. Supervised learning is a standard ML paradigm but requires labeled data. It is commonly known that ML models perform better the more data they have access to [326]. Therefore, practitioners need to have access to large labeled datasets.

If a company plans to use manual labeling, the question becomes, "*Who should perform the labeling?*". There are two options: *crowdsourced* or *in-house labeling*. Both have their pros and cons. Crowdsourcing platforms such as *Amazaon Mechanical Turk* [328] allow anyone to sign up as a labeler. Using crowdsourcing means the company does not need to hire in-house personnel and set up the necessary labeling infrastructure. On the downside, crowdsourcing is not an option for companies with sensitive data. As an example, the Swedish company Saab works in the defense industry and sells products to the Swedish military. Saab handles sensitive data and must do background checks before recruiting personnel, which makes crowdsourcing impossible. In

[1] https://github.com/teodorf-bit/Assessing-the-Sustainability-of-Deep-Semi-Supervised-Learning-Datasets-using-Item-Response-Theory

addition, it is challenging to guarantee the quality of labels and labelers from crowdsourcing [16]. Obtaining high-quality labels is essential as it will affect the validity of the trained model.

For these reasons, many companies still prefer in-house labeling. It allows companies to train the labeler and do quality checks. Even if in-house labeling is more flexible and gives companies more control, hiring labelers can be economically expensive, and training them can take time and effort. Data scientists and software engineers are often the most knowledgeable about the datasets and most suitable for labeling. On the other hand, data scientists and software engineers are experts in other areas, such as programming and building ML models. Data Scientists spend 80% of their time in SQL browsing databases, and the rest of the time they need to build models. Therefore, they do not have time to label data [16].

Semi-Supervised Learning

Practitioners can use semi-supervised learning (SSL) to avoid data labeling altogether. The goal of using SSL is to improve upon SL by utilizing both labeled and unlabeled data [326].

The first SSL algorithm introduced was *Self-Training* in 1960 [329]–[331]. The increased interest in SSL began in the 1970s [32], [332]–[334] and became even more popular in the 1990s due to the big interest in Natural Language Processing [136], [334]–[337].

There exist many SSL ML models that have been used on a variety of applications and datasets [25]. Many of these algorithms are extensions of supervised ML algorithms such as Support Vector Machines, Mixture Models, Multiview Learning and Graph-based algorithms. Although SSL can improve SL accuracy, SSL only increases accuracy if the data distribution matches the classification problem. Therefore, we must impose several assumptions on the distribution [307], [326]. First, the *smoothness assumption* says that if two feature instances x_1, x_2 lie in a high-density region, so do their labels y_1, y_2. Second, the two equivalent assumptions *cluster/low-density assumptions* say that points belonging to the same cluster should belong to the same class or, equivalently, the decision boundary should lie in a low-density region. Lastly, the manifold assumption says that high-dimensional data should lie in a low-dimensional manifold. The manifold assumption is necessary to avoid *the curse of dimensionality* [338].

Many theoretical studies from the early 1990s suggest that unlabeled data should be utilized [33], [141], [339]. These studies assume that it is possible to find the empirical distribution $\widehat{p}(x, y)$ of the true distribution $p(x, y)$ and show that performance will be better the more labeled and unlabeled data is used [33], [141], [339]. On the other hand, there are other studies from the same era that show that unlabeled data can degrade performance [288]–[292].

Deep Semi-Supervised Learning

Deep Semi-Supervised Learning (DSSL) algorithms are very popular to use due to their *scalability* [293]–[298]. As discussed in earlier, data is expensive to collect and therefore it is important to develop DSSL models that can leverage unlabeled data to improve classification performance. In recent years, many DSSL benchmarks have been developed, such as FixMatch [75], FlexMatch [78], CoMatch [67], SoftMatch [79] and SimMatch [66]. Due to the recent popularity of DSSL, we have chosen to evaluate these algorithms in this study. DSSL can be divided into four categories [307]: *Generative methods, Consistency Regularization, Graph-based methods, Pseudo-labeling methods* and *Hybrid methods*. Using Bayes theorem, generative methods predict the label y given x by computing the posterior distribution $p(y|x)$. Examples of generative DL algorithms are Generalized Adversarial Networks (GAN) and Variational Autoencoders (VAE), all of which can be extended to the semi-supervised setting. See [307] for examples of semi-supervised GANs and VAEs

Consistency regularization relies on the smoothness and manifold assumption. The methods impose a regularization term on the loss function so that the output is invariant to perturbations on the input [325]. For examples of algorithms based on consistency regularization, see [307].

Graph-based SSL algorithms build a graph to represent the dataset, assuming that the dataset can be expressed as a graph [231]. Each node of the graph represents an instance in the training set, and the algorithm measures how similar each pair of nodes is to each other to decide whether two nodes should be connected by an edge [231]. Examples of graph-based algorithms can be found in [307].

Pseudo Labeling uses a supervised baseline trained on the labeled training set to predict pseudo labels to the unlabeled data [28]. The pseudo-labeled instances will then be added to the training data, and the model will then

be retrained to improve the training accuracy. Performance improvement is only guaranteed if the pseudo-labels are of high quality. Examples of Pseudo-labeling algorithms can be found in [307].

Lastly, hybrid methods rely on a mixture of pseudo-labeling, consistency regularization and entropy minimization. Examples of such methods can be found in [307].

Universal Semi-Supervised Benchmark (USB)

USB is an open-source Python-based library for DSSL that was developed using Pytorch. USB contains many supervised baselines and 15 datasets divided into three tasks: computer vision, natural language processing and audio. There are 15 SSL algorithms available. Π-Model [73] **(pimodel)**, Mean-Teacher [74]**(meanteacher)**, Pseudo-Label [28]**(pseudolabel)**, VAT [304]**(vat),**, MixMatch [81]**(mixmatch)**, ReMixMatch [83]**(remixmatch)** ,UDA [305]**(uda)**, FixMatch [75]**(fixmatch)**, FlexMatch [78]**(flexmatch)**, Dash [76]**(dash)**, AdaMatch [77]**(adamatch)** , CRMatch [65]**(crmatch)**, CoMatch [67]**(comatch)**, SimMatch [66]**(simmatch)** and DeFixMatch [340]**(defixmatch)**ÚS is also compatible with tools such as Tensorboard, checkpoints, and logging to prevent data loss due to incidents such as computing infrastructure crashes. In addition, users can develop their algorithm using USB and add custom datasets.

Item Response Theory (IRT)

The development of both *Classical Test Theory* (CTT) and *Modern Test Theory*, commonly known as *Item Response Theory* (IRT), were established around the same time. CTT by Charles Spearman in 1904 [121] and IRT by Alfred Binet [122].

In many applications, there are observable variables known as *latent variables* that can only be measured through observable variables. CTT and IRT contain a set of techniques that evaluate how well we estimate the latent variables. For instance, how well do questions of an exam evaluate students' ability.

CTT assumes that the observed score X contains some error E and that the value of the error should be close to zero. Mathematically speaking we

observe the scores $X_i, i = 1, 2, .., N$ of N students where:

$$X_i = T_i + E_i, \quad \mathbb{E}(E_i) = 0. \tag{10.1}$$

Here T is the true score, and we assume that the T and the error E are uncorrelated: $\text{corr}(T, E) = 0$. There are two types of errors, *systematic* and *random*. An example of systematic error is when an item is supposed to measure one thing but then also measures something else. Random errors arise when students make careless mistakes, such as computational errors.

CTT measures *reliability* of a test by comparing two *parallel test*. Tests A and B are said to be parallel if $T_A = T_B$ and $\text{Var}(E_A) = \text{Var}(E_B)$. The reliability is calculated by collecting the test scores of all students and computing the correlation coefficient:

$$\rho_{XY} = \text{corr}(X, Y). \tag{10.2}$$

The higher the correlation, the higher the reliability. In order to fully carry out an analysis, we need to assume that if Test A & B are two parallel tests, then $\text{corr}(T_A, E_B) = \text{corr}(T_B, E_A) = 0$

In addition to reliability, CTT also studies *discrimination* and *difficulty*. Discrimination measures the association between items and the latent variables. An item with high discrimination is strongly associated with the true score. If many students fail to give the correct response to an item, the item is said to be difficult. Items in parallel tests must contain items of the same difficulty.

The problem with CTT is the computation of the reliability coefficient. Using the assumptions above it means the reliability coefficient can be written as:

$$\rho_{X_A, X_B} = \frac{\text{Var}(T)}{\text{Var}(X)}. \tag{10.3}$$

Hence, to compute 10.3, we need to know the true score T. Secondly, the computation of reliability depends on the correlation between two samples. Suppose two samples are collected from two groups of students, and the test scores are different in the two groups. The variance of the two samples will be different, and the reliability cannot be compared among different groups [341].

Many researchers prefer IRT over CCT because of its many advantages [342]. One advantage is that it provides a sample independent measure instead of using reliability [342]. Binet invented IRT when the city of Paris asked him to develop a test that would make it possible for schools to determine whether a student had a mental illness or not.

Binet realized he had to quantify the latent variable, which he did by designing a fully standardized test for the items, where each item was a task designed to measure the students' intelligence. He then used age as a scale to measure their intelligence and the test score of the items. Twenty years later, Thurstone improved upon Binet and showed empirically that the latent scale for the intelligence is normally distributed [123].

After Thurstone came Lord [124] and Rasch[125] who invented the two-parameter model [124], [125] where the probability of correct response is assumed to be normally distributed and depends on the latent *ability* parameter. The probability of correct response is modeled as a function of the latent parameter *ability* and the *difficulty* and *discrimination* parameters. The ability parameter represents the ability of each person to provide the correct response.

Models for continuous responses

Rasch [125] and Lord [124] considered discrete responses to the items, such as multiple-choice questions. This paper studies continuous responses since error rate and accuracy are real numbers.

In order to define the model, we make the following assumptions:

1. There exists at least one latent trait:

$$\mathbf{\Phi} = (\mathbf{\Phi}_1, ..., \mathbf{\Phi}_d),\ d \geq 1.$$

2. The response of test taker p to item i is normally distributed:

$$U_{ip} \sim \text{Normal}(\tau_{pi}, \varphi_{pi}^2).$$

3. The responses of test takers to the n items are conditionally independent of the latent variables:

$$P(U_{i1}U_{i2} \cdot ... \cdot U_{ip}|\mathbf{\Phi}) = P(U_{i1}|\mathbf{\Phi})P(U_{i2}|\mathbf{\Phi}) \cdot ... \cdot P(U_{ip}|\mathbf{\Phi}).$$

4. The responses can be modeled w.r.t the latent traits using regression:

$$\mathbb{E}(U_{pi}|\Phi_1 = \varpi_{p1}, \cdots, \Phi_D = \varpi_{pD}) = \tau_{pi} = b_i + a_{i1}\varpi_{p1} + \cdots + a_{iD}\varpi_{pD}, \tag{10.4}$$

where τ_{pi} is the true item score of test taker p. The parameters $\varpi_{p1}, \cdots, \varpi_{pD}$ are the values of latent traits $1, \cdots, D$ of test taker p. The parameters $a_{i1}, ..., a_{iD}$ represent the slopes of the regression function, and b_i is the intercept representing the discrimination.

In this study, we further simplify by assuming that the variance of the item score is homogenous $\varphi_{pi}^2 = \varphi^2$ and that we only have one latent variable:

$$\mathbb{E}(U_{pi}|\Phi = \varpi_p) = \tau_{pi} = b_i + a_i\varpi_p, \tag{10.5}$$

$$\mathrm{Var}(U_{pi}) = \varphi^2. \tag{10.6}$$

The model defined above is known as the *Jöreskog* or *congeneric* model [126].

10.3 Research Method and Data Analysis

Benchmark datasets are used in algorithm development to evaluate how algorithms perform in certain scenarios [232], [233]. Empirical evidence shows that algorithms that perform well on benchmarks might perform differently on real-world datasets [232], [233]. Therefore, it is important to assess the sustainability of the datasets used to test the ability of the algorithms. Once the unsuitable datasets have been identified, we need to know how to replace them and evaluate the DSSL algorithms. This is done by answering the following RQs:

- **RQ1:** *What datasets are suitable to compare different DSSL algorithms?*
- **RQ2:** *How can different DSSL algorithms be compared?*

To answer the research question we utilize IRT and Bayesian Data Analysis (BDA) instead of frequentist statistics due to its many advantages [115]. The RQs have previously been answered with supervised learning [234] and Graph-based SSL [230]. IRT aims to study how a population responds to items [243]. The internet makes high-quality response data available from public online databases [343]. Data collected from large-scale online surveys often have

a hierarchical structure with complicated dependencies. Differences in hierarchical data are usually difficult to measure and interpret [343]. Bayesian models are helpful as they model uncertainty by imposing a prior distribution on observed data and then calculate the posterior distribution using the Bayes formula. Visualizing the posterior allows for better data analysis than frequentist approaches, where decisions are made based on descriptive statistics and p-values. However p-values can be misleading and problematic to interpret [112], [116]. Choosing the prior can be costly in terms of time, but *prior predictive checks* [344] makes it easy to evaluate the chosen priors. Similarly, *posterior predictive checks* [344] can be used to analyze the validity of the final posterior distribution. BDA is recommended for empirical software engineering [112] and has been used for IRT [343].

Bayesian Data Analysis

In the Bayesian framework, we treat a parameter θ as a random variable drawn from a prior distribution $P_\Theta(\theta)$ containing prior knowledge of θ. We assume the θ can take values inside the parameter space Θ and that X is the random variable representing the observed data $x \in \mathcal{X}$.

Consider the following experiment.

1. We randomly choose $x \in \mathcal{X}$ from the distribution P
2. Then choose $\theta \in \Theta$ from the distribution P_x.

We are interested in modeling the *posterior distribution*, which is the conditional distribution of θ given X, $P_{\Theta|X}(\theta|x)$. The posterior distribution is calculated using 10.7:

$$P_{\Theta|X}(\theta|x) = \frac{\int_B f_{X|\Theta}(x|\theta) f_\Theta(\theta)\, d\theta}{P_X(x)}. \tag{10.7}$$

We introduce the following notation:

- $\ell(\theta|x) = f_{X|\Theta}(x|\theta)$ is the likelihood function.
- $f_\Theta(\theta)$ is the density of the prior distribution.
- $P_X(x) = \int_\Theta f_{X|\Theta}(x|\theta) f_\Theta(\theta)\, d\theta$ is the marginal distribution:

Algorithms and Datasets

At the time of writing this paper, there are 15 algorithms available in USB. We include every algorithm except for DeFixMatch [340] since it is only evaluated on image datasets.

The algorithms are evaluated on 15 equally distributed datasets across the three datatypes: image, text, and audio. Details about the datasets are found in Table 10.1.

Data Collection

We collect the data from the USB Github. Since the DL algorithms are trained in epochs, the data is reported in intervals on the form $(\overline{x}-m, \overline{x}+m)$, where $\overline{x}$ is the average error rate, $\overline{x}-m$ is the lowest error rate and $\overline{x}+m$ is the highest error rate. We assume that the error rate is normally distributed $x \sim \text{Normal}(\mu, \sigma)$ and we need to generate samples from this distribution to use in the BDA. The *maximum likelihood estimator*s (MLE) for μ is given by $\widehat{\mu} = \overline{x}$ and the MLE for σ is given by:

$$\widehat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

We can find $\widehat{\mu} = \overline{x}$ with the given intervals, but the MLE of $\widehat{\sigma}$ is not applicable here since we do not have access to samples $x_1, \cdots, x_n$. We use the test statistic T:

$$T = \frac{\overline{x} - \mu}{\sigma/\sqrt{n}}, \tag{10.8}$$

to derive a confidence interval (CI) for μ and then solve for σ:

$$-\lambda_{\alpha/2} \leq T \leq \lambda_{\alpha/2}, \tag{10.9}$$

$$\Rightarrow -\lambda_{\alpha/2} \leq \frac{\overline{x}-\mu}{\sigma/\sqrt{n}} \leq \Rightarrow \lambda_{\alpha/2}, \tag{10.10}$$

$$\Rightarrow \lambda_{\alpha/2} \geq \frac{\mu-\overline{x}}{\sigma/\sqrt{n}} \geq - \Rightarrow \lambda_{\alpha/2}, \tag{10.11}$$

$$\Rightarrow \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}} \geq \mu - \overline{x} \geq - \Rightarrow \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}}, \tag{10.12}$$

$$\Rightarrow \overline{x} + \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}} \geq \mu \geq \overline{x} - \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}}. \tag{10.13}$$

So $\mu \in \left(\overline{x} - \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}}, \overline{x} + \lambda_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right)$, and we put $m = \sigma\lambda_{\alpha/2}/\sqrt{n}$ and solve for σ to derive the estimate:

$$\widehat{\sigma} = \frac{m\sqrt{n}}{\lambda_{\alpha/2}},$$

where λ_β denotes the β%-quantile. We choose $\alpha = 0.05$ and simulate $n = 1000$ samples from Normal$(\overline{x}, \widehat{\sigma})$.

Data Analysis

Once we have generated the data, we perform the BDA. We start by transforming the error rate to accuracy $U = 1 - \varepsilon$. We use *rstan*, *cmdstan*, *bayesplot* and *posterior* libraries to do the Bayesian modeling and produce necessary plots.

We are using equation (10.7) with $X = U$ and $\theta = (a, b, \varpi)$. The distribution of the accuracy (responses) is normally distributed:

$$U_{ip} \sim \text{Normal}(\tau_{ip}, \varphi), \quad \text{where } \tau_{ip} = b_i + a_i\varpi_p.$$

We use weakly informative priors per recommendation [252]:

$$a \sim \text{Normal}(0, 1), \tag{10.14}$$
$$b \sim \text{Normal}(0, 1), \tag{10.15}$$
$$\varpi \sim \text{Normal}(0, 3), \tag{10.16}$$
$$\varphi \sim \text{Normal}(0, 1). \tag{10.17}$$

We utilize Markov Chain Monte Carlo sampling with four parallel chains, 200 warmup iterations, and 2000 iterations after warmup. The chains are evaluated using traceplots and measurements R-hat ($\widehat{R}$) and the number of effective samples (n_{eff}). The choice of priors and the resulting posterior distribution are evaluated using prior and posterior predictive checks. All the evaluation statistics and checks can be found in the online appendix.

Table 10.1: Datasets contained in USB

Type	Dataset	Total Instances	Number of Classes
Image	Cifar100	60000	100
	STL-10	13000/108000	100
	EuroSat	21600	10
	TissueMNIST	212746	8
	Semi-Aves	9959/30640	200
Text	IMDB	50000	2
	Amazon Review	340000	5
	Yelp Review	325000	5
	AG News	117600	4
	Yahoo! Answers	610000	10
Audio	Keyword Spotting	23682	10
	ESC-50	2000	50
	urbanSound8k	8732	10
	FSDnoisy18k	2719/15760	20
	GTZAN	10000	10

10.4 Results

In this section, we present the results from our data analysis and then summarise the results into guidelines for practitioners.

Discrimination Parameter

The *discrimination* parameter represents the association of the responses (accuracy) of the algorithms to the items (datasets) [243]. The median discrimination parameter of the datasets can be found in the upper part of Table 10.2. In addition, high posterior intervals for the discrimination parameters can be found in Figure 10.2. We can see that the image datasets cifar100, eurosat, semi-aves and tissuemnist have low discrimination values between 0.01 and 0.066. Stl has low discrimination but is slightly higher than the other image datasets. The rest of the datasets have much higher median discrimination, varying between 0.276 and 0.587. The high posterior density intervals for the other datasets are much wider and overlapping, indicating uncertainty in their estimates.

Easiness Parameters

The *easiness* parameter indicates how easy it is for the test takers to respond to an item correctly [243]. In our case, the easiness parameter indicates how easy a dataset is for the algorithms to learn . The median easiness parameter of the datasets can be found in the lower part of Table 10.2. The high posterior density intervals can be found in Figure 10.3. We can see that the four image datasets, stl, semi-aves, eurosat and cifar100, have high easiness level parameters of 0.636-0.827. tissuemnist, aclimbd, and ag_news have mid-range easiness level parameters of 0.212-0.428. The rest of the algorithms have low easiness parameters.

Ability Parameter

The *ability* parameter measures the test takers ability to respond to the items correctly. The algorithms' median ability can be found in Table 10.3. We can see that the ability is between 1 and 2 for all algorithms except for ReMixMatch, which has an ability parameter close to 0. Algorithms with similar abilities will have the same probability of learning the datasets. The high posterior density intervals are wide and overlapping, which indicates uncertainty in their estimates.

Item information curve

The *item characteristic curve (ICC)*, $I_i(\mathbf{\Phi})$ is a plot of the probability of successfully answering item i, depending on the ability parameter [243]. The *item information curve* (IIC) is the derivative of the ICC w.r.t $\mathbf{\Phi}$ [243] and indicates the rate of change of the probability. Figure 10.4 illustrates the item information curves. According to Figure 10.4, cifar100, eurosat, semi-aves, stl and tissuemnist are the only datasets that add no information to the capacity of the test to estimate the ability of algorithms.

Test information curve

The *test information curve* (TIC) is the sum of all the IICs and measures the probability of correctly answering the whole test. Optimally we want the value of the ability in the region where the value of the TIC is high. This means that the test is appropriate for evaluating the ability of the test taker (algorithms). The test information curve can be found in Figure 10.5. The figure shows that all algorithms have ability levels in the region where the test has high measurement accuracy.

Summary

Based on the analysis, all datasets except for the image datasets are suitable for evaluating the benchmarks. We can see that all image datasets have low discrimination and high easiness parameters. The values of the parameters indicate that there is no association between the datasets and the accuracy of the algorithms since the datasets are too easy to learn. Therefore, the datasets do not contribute to the capacity of the test, as can be seen from inspecting the IICs. We can see from the TIC that the test is suitable for evaluating all algorithms since they all have ability levels within the region where the test information is high. However, the ability parameters are estimated to have high uncertainty (except ReMixMatch), as seen from the wide overlapping HPDIs. We recommend replacing the image datasets with more difficult datasets that add information to the test's capacity to estimate the algorithms' ability. The other datasets can be kept in the test as they have high discrimination, low easiness values, and good IICs.

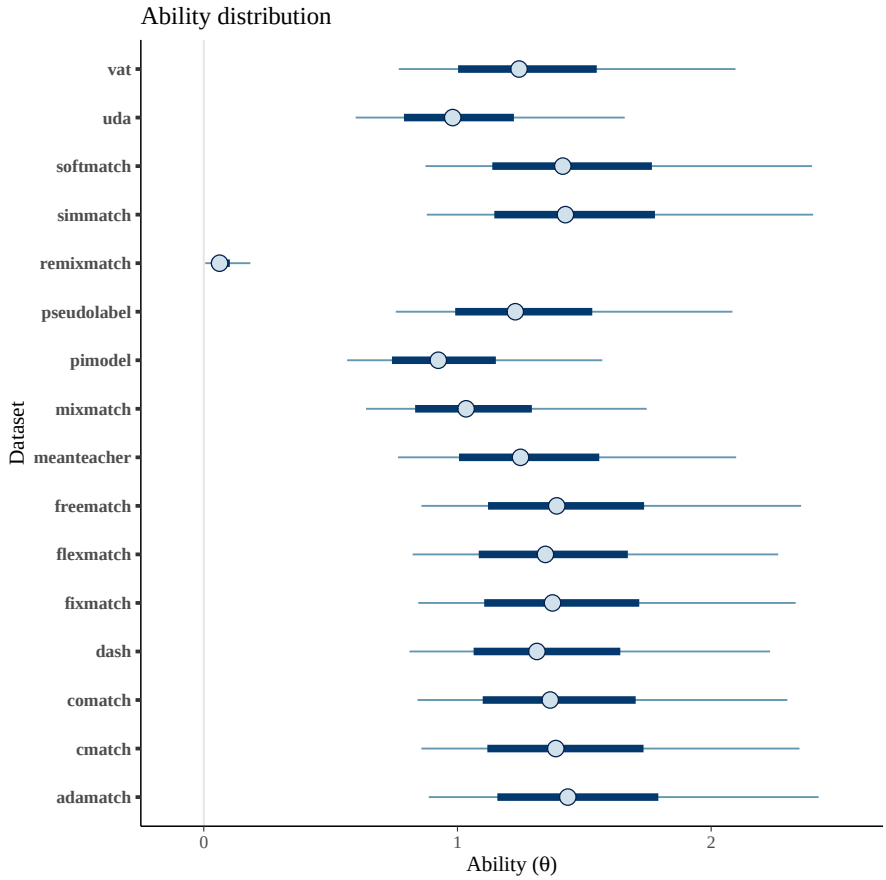


Figure 10.1: Distribution of the ability parameter for each algorithm

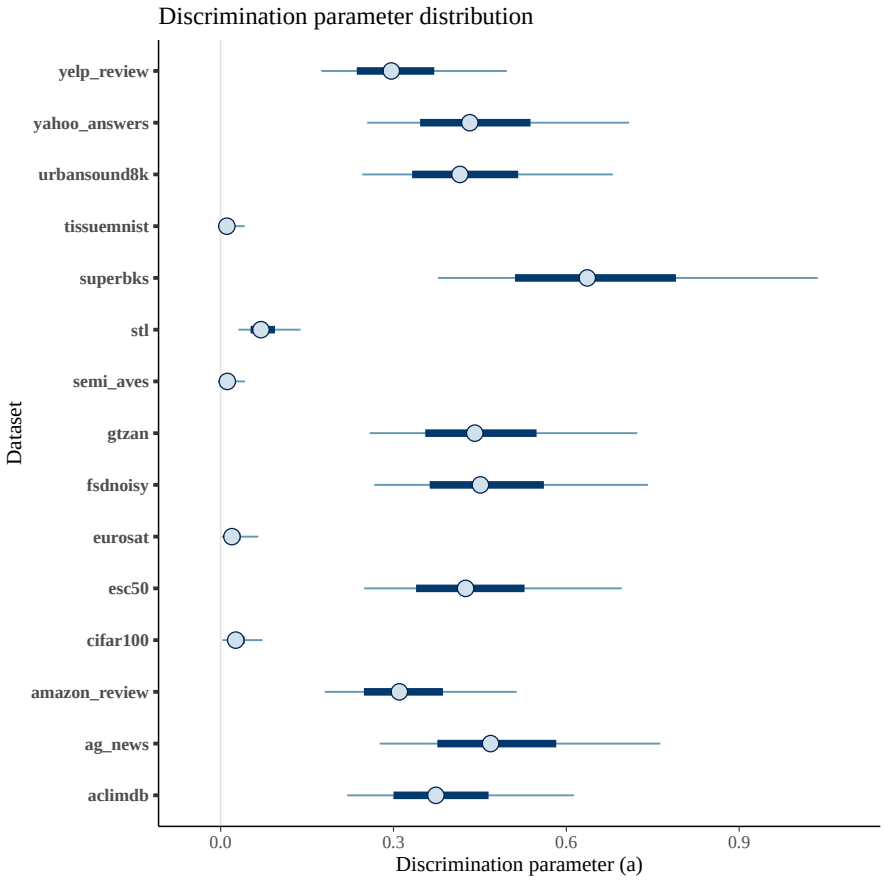


Figure 10.2: Distribution of the discrimination parameter for each dataset

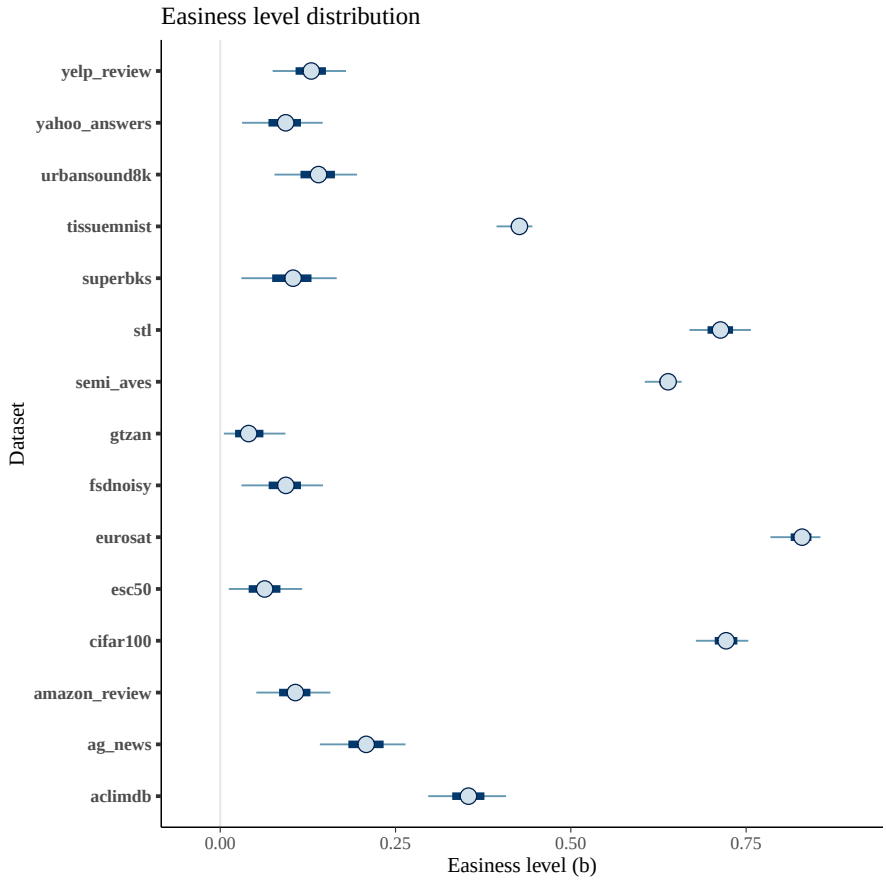


Figure 10.3: Distribution of the difficulty parameter for each dataset

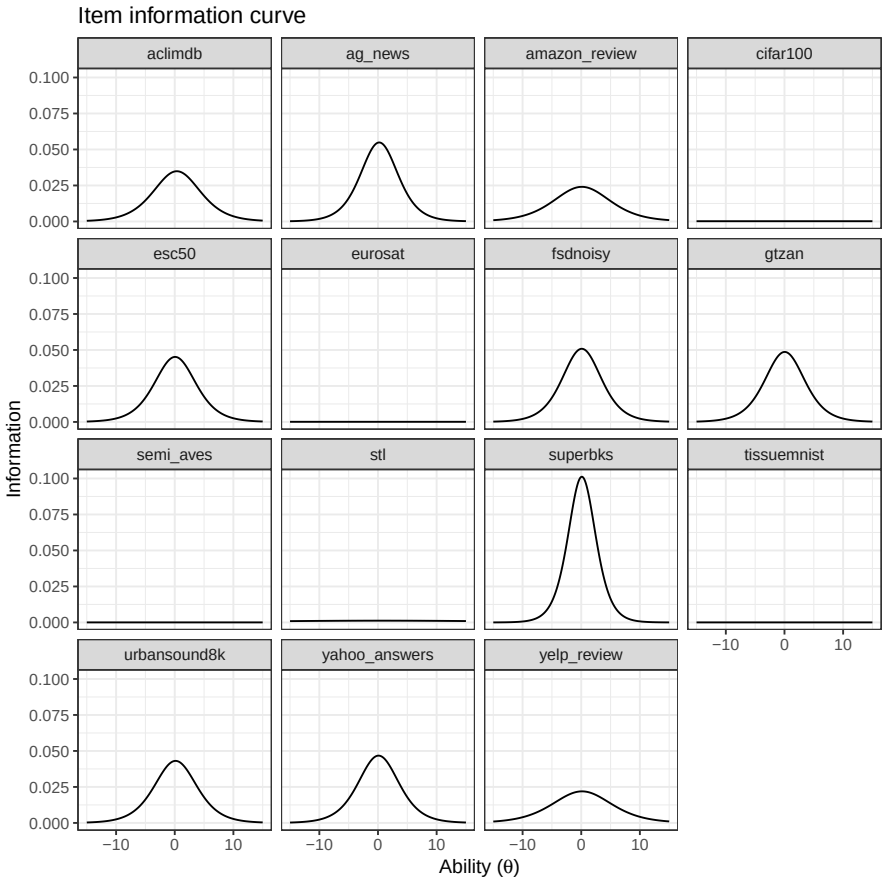


Figure 10.4: Median item information curve for the benchmark datasets.

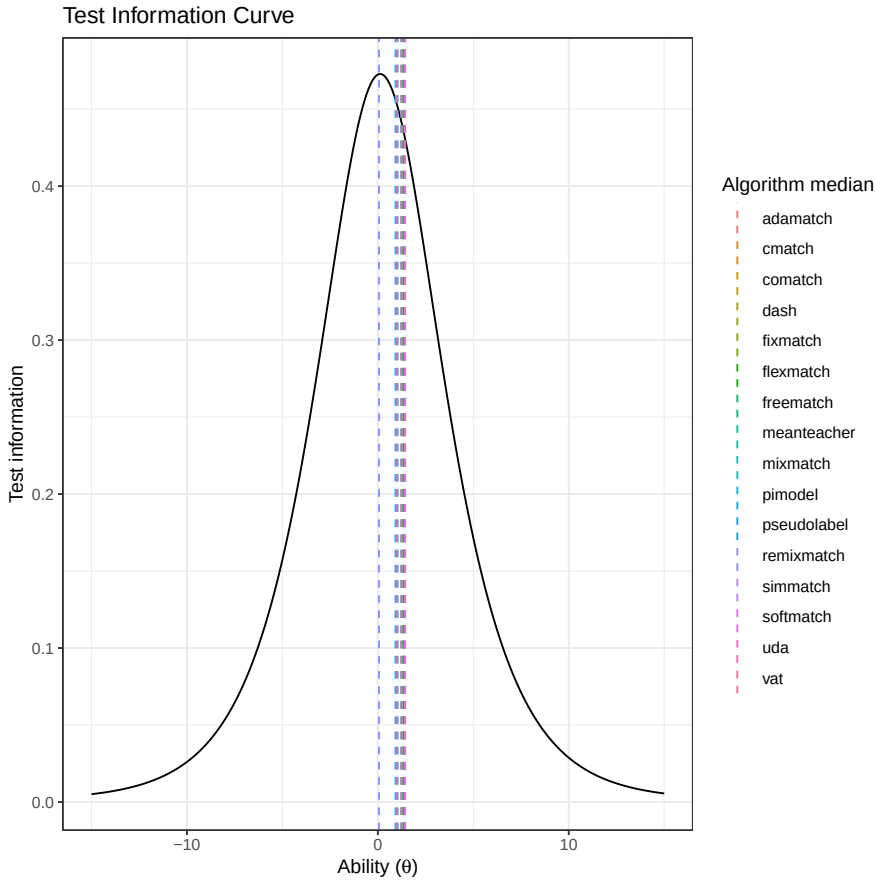


Figure 10.5: Test information curve with the median ability parameters

Table 10.2: Summary values of the discrimination and easiness level parameters for the datasets

Dataset	Median	CI 5%	CI 95%
Discrimination value (a)			
aclimdb	0.374	0.220	0.613
ag_news	0.468	0.276	0.763
amazon_review	0.310	0.181	0.514
cifar100	0.026	0.003	0.073
esc50	0.425	0.249	0.696
eurosat	0.020	0.002	0.065
fsdnoisy	0.451	0.267	0.742
gtzan	0.441	0.259	0.723
semi_aves	0.012	0.001	0.042
stl	0.070	0.031	0.139
superbks	0.636	0.377	1.036
tissuemnist	0.011	0.001	0.042
urbansound8k	0.415	0.246	0.681
yahoo_answers	0.432	0.254	0.709
yelp_review	0.296	0.175	0.497
Easiness level (b)			
aclimdb	0.354	0.297	0.408
ag_news	0.208	0.142	0.264
amazon_review	0.107	0.051	0.157
cifar100	0.721	0.678	0.753
esc50	0.063	0.012	0.117
eurosat	0.830	0.785	0.856
fsdnoisy	0.094	0.030	0.147
gtzan	0.041	0.005	0.093
semi_aves	0.639	0.606	0.658
stl	0.713	0.669	0.757
superbks	0.104	0.030	0.166
tissuemnist	0.427	0.394	0.445
urbansound8k	0.140	0.077	0.195
yahoo_answers	0.093	0.031	0.146
yelp_review	0.130	0.075	0.179

Table 10.3: Summary values of the ability level of the SSL algorithms

Algorithms	Median	CI 5%	CI 95%
adamatch	1.435	0.887	2.424
cmatch	1.388	0.858	2.348
comatch	1.366	0.842	2.301
dash	1.313	0.811	2.232
fixmatch	1.375	0.846	2.333
flexmatch	1.347	0.823	2.264
freematch	1.391	0.858	2.355
meanteacher	1.249	0.765	2.099
mixmatch	1.034	0.640	1.746
pimodel	0.924	0.565	1.571
pseudolabel	1.228	0.757	2.084
remixmatch	0.062	0.006	0.184
simmatch	1.426	0.879	2.402
softmatch	1.415	0.874	2.398
uda	0.981	0.599	1.659
vat	1.243	0.769	2.096

10.5 Discussion

Data Scientists and software engineers are usually the most suitable for labeling but are busy doing other more specialized tasks such as extracting data from databases and programming tasks in programming languages such as Python and R [16]. SSL, which uses both labeled and unlabeled data to improve the classification performance of SL, is a solution to avoid manual data labeling. Even if DSSL can improve classification performance, it is not guaranteed. Choosing and calibrating models can be time-consuming, and practitioners usually rely on benchmark studies to choose algorithms.

The purpose of benchmark studies is to investigate how well a particular algorithm performs on a given problem [34]. Benchmark studies include many algorithms for comparison of their performance [34]. The collected data should then be analyzed using statistical tools to understand the strengths and weaknesses of the algorithms and find the optimal algorithm for different scenario [34]. In addition, benchmark studies are suitable for investigating the optimal

parameter tuning for algorithms and analyzing how sensitive the performance is w.r.t changes in the tuning and why a certain algorithm performs better than another [34]. Lastly, benchmarking is useful to identify weak spots in order to develop better-performing algorithms [34]. Even if benchmark studies seem useful for practitioners, empirical evidence suggests that algorithms that perform well on benchmark datasets might perform poorly on real-world datasets [232], [233]. In that case, the test to evaluate the algorithms needs to be better constructed because the datasets in the test were too easy for the algorithms to learn. If the datasets are too easy, the benchmark study becomes unreliable. Practitioners cannot choose an algorithm based on an unreliable benchmark study since the algorithms can perform differently on a more difficult dataset.

IRT was initially developed as a tool to evaluate the mental capacity of students [122]. In addition, IRT has been widely applied to applications such as psychology [243], evolutionary computing [327], supervised learning [234], and transductive GBSSL algorithms [230]. Therefore, IRT is suitable for investigating how well benchmark datasets evaluate the DSSL algorithms. This study applies IRT to evaluate the datasets' suitability for evaluating the DSSL algorithms in [84]. Choosing the optimal algorithm based on a benchmark study that used unsuitable datasets is bad as there is no guarantee that the algorithm will perform as well on real-world datasets. Data scientists and software engineers have strict deadlines and cannot afford the risk of implementing an algorithm that performs poorly.

IRT has been used to evaluate the sustainability of the 60 datasets contained in the OpenML-CC18 benchmark [345] used for supervised classification algorithms. The results of [345] show that the benchmark contains 10% difficult datasets, 6% average difficult datasets and the rest are too easy. 40% of the datasets are discriminating, but the authors conclude that these datasets are only appropriate for pairwise comparison of algorithms. In [230], IRT is applied to 13 different Graph-based Semi-Supervised Learning to evaluate the sustainability of 12 datasets equally distributed among three datatypes: image, text and numerical. The results show that 10 out of 12 datasets have low discrimination and high easiness and therefore unsuitable for evaluating the algorithms. Compared to the abovementioned studies, most USB datasets evaluated are suitable except for the image datasets. TissueMNIST, Stl, semiaves, eurosat and Cifar100 all have low discrimination and high easiness pa-

rameters. In addition, the item information curves show that these datasets add no information to estimating the ability of an algorithm. Therefore, it is essential to re-evaluate USB using more difficult image datasets with higher discrimination values.

Once practitioners have constructed a better benchmark containing datasets that are more difficult, have high discrimination and add more information to the test, the benchmark can be considered more reliable. The algorithms should perform similarly on real-world datasets. The benchmark can then be used in the algorithm development process to detect weaknesses in new algorithms that are being developed.

Threats to Validity

We discuss four types of threats to validity for simulation studies as described in [104].

First is *Construct Validity*, which refers to how appropriate the statistical model is for answering the research questions. The RQs of this paper are concerned with how appropriate the datasets are for evaluating the DSSL algorithms of USB. The Bayesian congeneric IRT model [126] was used and is appropriate for answering our RQs.

Second is *External Validity*, which refers to how generalizable the results are to other situations. The results of this study cannot be used to conclude that the datasets unsuitable for evaluating USB are unsuitable for evaluating other algorithms.

Third is *Internal Validity*, which refers to whether the independent variables cause the outcome because of simplifications in the model. Regarding the congeneric model, we assume that ability is a latent variable. In reality, there could be more latent variables and parameters of importance.

The final threat is *Conclusion Validity*, which refers to whether the results were evaluated using appropriate statistical tests. In this study, we used descriptive statistics such as $\widehat{R}$, n_{eff} as well as posterior and posterior checks [252] to check the validity of the statistical model.

10.6 Conclusion

The USB is a Python package containing 15 DSSL algorithms evaluated on 15 datasets equally distributed across three datatypes: image, text and sound. Even if these algorithms perform well on benchmark datasets, more research is needed to evaluate USB algorithms using real-world datasets. Real-world datasets can be challenging to collect, so practitioners rely on benchmark studies. Benchmark studies can be misleading if they utilize too easy datasets, resulting in the possibility of algorithms performing worse on real-world datasets. Furthermore, to the best of our knowledge, no study has used IRT to assess benchmark datasets' sustainability for evaluating the USB algorithms. To fill this gap in research, we used the Bayesian congeneric model to investigate how to improve USB. By analyzing the ability of each algorithm, discrimination and easiness for each dataset, and item and test information curves, we conclude that all the image datasets are bad at evaluating the algorithm's ability. The image datasets need to be replaced because they all have low discrimination and high easiness. The datasets should have good discrimination and low easiness in a good benchmark study. The estimated ability of the algorithms is similar, and the high posterior density intervals are broad and overlapping, which indicates uncertainty in their estimates. Since each algorithm has a similar ability parameter, they all have the same probability of learning a dataset, and it does not matter what algorithm is chosen.

Based on the results, practitioners know that image datasets are unreliable for evaluating USB algorithms. To create a new set of benchmark datasets more suitable for evaluating DSSL algorithms, practitioners should add more difficult, discriminative datasets that add more information to the test. Once suitable datasets have been added to the benchmark, practitioners can perform a new empirical evaluation of USB algorithms. The new empirical evaluation will provide reliable insights regarding how the USB algorithms will perform on real-world datasets. In addition, practitioners should use the new benchmark datasets when developing new DSSL algorithms to ensure that new algorithms perform well on real-world datasets.

In future work, we want to improve USB by identifying new datasets that are more suitable for evaluating DSSL algorithms. We also wish to apply IRT to more ML and DL benchmarks to identify unsuitable datasets.

CHAPTER 11

Classification of Complex-Valued Radar Data using Semi-Supervised Learning: a Case Study

11.1 Introduction

For the past years, machine learning (ML) and deep learning (DL) tools have risen in popularity among data scientists in industry. Many companies have or are in the process of introducing ML and DL in their pipelines. ML was defined in the 1950s but not implemented until modern times due to the lack of cheap computational power [346]. In modern times, computational resources have become affordable and abundant, thus allowing more enterprises to utilize ML and DL [347].

One of the most common machine learning tasks is classification. Classification tasks are supervised and require a labeled dataset [348]. Large and labeled datasets generally yield higher performance accuracy, especially for deep learning models [349]. However, fully labeled datasets are rare in practice, as much data from the industry is incomplete. There are two ways for companies to fill in missing labels.

The first solution is in-house labeling, which assigns company workers for

the labeling task [350]. However, using a specialist such as a data scientist for labeling tasks might waste time and money since data scientists are also performing more advanced tasks. Hence, data labeling for a data scientist might waste talent [16]. Data labeling can also be very complicated and tiring over time. Tasks such as labeling an image by whether it contains a cat or dog are straightforward. In contrast, PolSAR images, for instance, require annotation on a pixel level and require attention to detail [351]. Therefore the quality of the labels might decline over time. Labels of quality are essential to training well-trained ML and DL models.

The second approach is to use third-party crowdsourcing services [18], [19]. Sadly, in-house labeling has proven to yield higher-quality labels than crowdsourcing [9]. There is also the issue of confidentiality, as many companies have sensitive and confidential data. Therefore companies cannot share their data with third-party companies.

As an alternate approach to in-house labeling and third-party companies, semi-supervised learning [31], [62] can be utilized to remove manual labeling as it utilizes both labeled and unlabeled data to learn.

This study uses a dataset based on a *Pulse-Doppler Radar*[352] obtained from our industry partner Saab. Saab is a Swedish defense and aerospace company. Supervised and semi-supervised [353], [354], models have been utilized using neural networks [355]–[357] on benchmark PolSAR datasets for labeling [356], [357]. However, to the best of our knowledge, there is a lack of research utilizing real-world data from the industry and radar data other than PolSAR images. Furthermore, no studies help practitioners decide how to build their model and the minimal manual effort required to achieve optimal learning accuracy.

We utilize the FixMatch algorithm to investigate whether it is beneficial to use unlabeled data to improve a supervised baseline. We evaluate the algorithms across three dimensions, *Accuracy*, *Manual Effort* and *Convergence*. Accuracy refers to the validation accuracy obtained during training. Manual Effort refers to how many unlabeled instances are necessary to achieve the desired accuracy. Convergence refers to whether the unlabeled data will help reduce the loss and avoid making the model overfit and underfit.

This paper provides three main contributions. First, we show that the classification accuracy of complex-valued radar data can be improved by utilizing unlabeled data with the FixMatch algorithm. Second, we show that by us-

ing FixMatch, practitioners can save on manual data labeling and still reach a higher accuracy than the supervised baseline. Lastly, we show that FixMatch will promote convergence in the loss curve and prevent the model from overfitting and underfitting.

Based the results of this study, practitioners will know under what circumstances FixMatch will be useful for improving accuracy and reducing manual labeling without risking underfitting or overfitting the model.

The rest of the paper is organized as follows: Section 11.2 describes the challenges concerning data labeling, semi-supervised learning and the Bayesian Linear Regression Model. Section 11.3 describes the procedure we followed to obtain the results. Section 11.4 contains the results obtained from the simulations. Section 11.5 discusses the results. Finally, Section 11.6 presents the paper's contributions.

The details around the supervised baseline and the statistical analysis can be found in the online appendix: https://github.com/teodorf-bit/SEAA-2023-paper.git

11.2 Background

In this section, we present the background to the labeling challenge problem and theory for the semi-supervised learning and modeling approach taken in this paper.

Labeling Challenge in Software Engineering

Machine learning has a vast range of applications, such as self-driving vehicles. When engineering machine learning-based software, training deep learning models for object detection and scene perception in self-driving vehicles faces many challenges [358]. One problem is localization which is solved using maps. Mapping however is a very expensive task [359] that can be addressed using sensors with limited range and coverage. Therefore, unmanned aerial vehicles (UAVs), satellites, and other aerial vehicles have been used to find mappings of sideways. These datasets need to be labeled for the supervised classification of unseen data.

Data Labeling for text data is the process of structuring text by content. Obtaining labels through automated scripts is possible, but then the program

must be able to assign the correct label to each unit of textual data. Humans can have a better understanding of this, but manual labeling by a domain expert can be costly [14]

TexAccording to research [16], 80% of the time spent in a machine learning project is on data labeling. Because of the time it takes to label the data, the task should not be allocated to the data scientist who is busy doing more specialized tasks. A person performing this detailed labeling job might also make mistakes that will lead to low-quality labels, which directly influences the performance of the machine learning algorithm.

Another solution to the labeling problem is to use third-party data labeling services. Due to the high demand for labeling, the data labeling market is expected to triple by 2024 [12], [13]. Crowdsourcing is an example of a third-party labeling service and the primary way of getting labels [13]. Crowdsourcing distributes and divides a task among several parties. The parties involved in this task will be rewarded once said task is completed [213]. Through crowdsourcing, companies can obtain labels by requests from a group of online communities. *Amazon Mechanical Turk* and *Lionbridge AI* are examples of crowdsourcing services [214]. Using crowdsourcing means that the companies do not have to develop their own labeling infrastructure and tools necessary to perform the labeling. The downside of crowdsourcing is that companies cannot share sensitive data. When choosing a crowdsourcing service, one important question is, how can we ensure annotators produce high-quality labels?

Semi-Supervised Learning

Big datasets are the key to well-performed supervised classifiers. A fact that is especially true for deep learning models. Large datasets are available in many companies, but they are often incomplete because they have missing labels. Solutions such as active learning exist, but active learning requires additional instances to be manually labeled by an oracle. These instances are then added to the training set, and the supervised model is re-trained using the updated training set [24]. This procedure is repeated until a stopping criterion is reached. This approach requires much effort as the underlying supervised model must be re-trained often. This approach might also include manual labeling. To the best of the author's experience with companies that utilize machine learning, manual labeling is not preferred, and automated

approaches are preferable [16].

Semi-supervised learning combines supervised and unsupervised learning as it performs classification using labeled and unlabeled data [133]. Semi-supervised learning is a more realistic approach because it is common to have many unlabeled instances in industrial datasets.

FixMatch

Deep Neural Networks are popular partially due to their *scalability*. A powerful approach for training models on datasets that do not contain many labeled instances is semi-supervised learning (SSL). SSL utilizes both labeled and unlabeled to train a classifier. Many such algorithms have been proposed, such as Deep Semi-Supervised Learning (DSSL) [307]. Many DSSL algorithms rely on *Pseudo-Labelling*[28] and *Consistency Training* [325], [360], [361]. An example that utilizes both pseudo labeling and consistency training to create artificial labels is *FixMatch* [75]. We use FixMatch as inspiration for our algorithm as it can achieve high accuracies using few labeled instances and due to its simplicity [75]. Furthermore, FixMatch is easy to implement and be built on any supervised baseline.

The set $\mathcal{O}_{\mathcal{Y}}$ of one-hot labels is a vector of probabilities $o = (p_1, p_2, .., p_n)$ where p_i is the probability that $y = y_i$. Let

$$\mathcal{B}_L = \{(x_b, o_b) \colon b \in (1, ..., B)\}$$

be a batch of labeled examples where x_b are the features and o_b are the on-hot labels of labels y_b.

$$\mathcal{B}_U = \{u_b \colon b \in (1, ..., zB)\}$$

be the unlabeled batch where zB is the size of the unlabeled batch. FixMatch utilizes *strong* and *weak* augmentation $\alpha(\cdot)$ and *strong* augmentation $\mathcal{A}(\cdot)$ to augment the unlabeled instances and use them during training.

Consistency Regularization assumes the model predicts similar outputs when fed perturbed versions of the same image.

Pseudo-Labelling uses the model itself to create artificial labels for unlabeled data.

Our version of FixMatch first pre-trains using the labeled samples by min-

imizing the supervised loss

$$\ell_s = \frac{1}{B}\sum_{b=1}^{B} H(o_b, p_m(y|x)).$$

The FixMatch algorithm then predicts the probability distribution (pdf) for the pseudo labels using the weakly augmented instances $q_b = p_m(y|\alpha(u_b))$ and then incorporates strongly augmented instances into the unlabeled loss function

$$\ell_u = \frac{1}{zB}\sum_{b=1}^{zB} 1\{\max(q_b) \geq \tau\} H(\widehat{q}_b, p_m(y|\mathcal{A}(u_b)),$$

where $\widehat{q}_b = \arg\max(q_b)$, τ is the threshold and H is the cross entropy

$$H(p,q) = -\sum_{x\in\mathcal{X}} p(x)\log q(x).$$

In other words, we only consider instances where its class probabilities exceed the threshold. If the instances with a class probability below the threshold are included FixMatch could worsen the performance.

Bayesian Analysis: Relative Improvement of SSL over SL

To assess whether FixMatch improves the accuracy we utilize a Bayesian Linear Regression Model to assess if there is a relative improvement of SSL over SL. The model is described as

$$\begin{aligned} y &\sim \text{ Normal}(\mu, \sigma), \\ \mu &= a_{ssl}, \\ \sigma &\sim \text{Exponential}(1), \\ a_{ssl} &\sim \text{Normal}(0, 1). \end{aligned}$$

The parameters in the model are: y denoting the *relative improvement*. a_{SSL} is the mean effect of the FixMatch algorithm. μ is the mean of the likelihood modeled by the linear equation and denotes the average improvement of FixMatch.

11.3 Research Method

This section presents the research method, including a description of the dataset, our algorithm, experimental setup and the evaluation metrics. Our study is a case study performed with our industry collaborator Saab. Saab has transitioned from agile software development to DevOps in recent years. Saab has started implementing machine learning algorithms into their pipelines and has yet to create a well-defined labeling infrastructure. The study aims to find a classifier to predict and classify the labels of unseen instances. As the dataset is small, and much of it is unlabeled, we wish to learn whether introducing unlabeled data can help improve model performance by using the FixMatch algorithm. We aim to learn these facts by answering the research questions below. **RQ1:** *What is the relative improvement of FixMatch in terms of accuracy?* **RQ2:** *How does the number of total labels affect the accuracy and to what extent does FixMatch help save time on manual data labeling?* **RQ3:** *By how much will FixMatch worsen the computation time of the supervised baseline?* **RQ4:** *How will FixMatch affect the convergence of the loss function and prevent overfitting and underfitting?*

During the study we were in continuous contact with a representative from Saab who helped us gain access to the dataset and taught us how to read it into Python. The authors of this study built the models and after the simulations were done, the results were reviewed by the Saab representative.

Dataset

Our dataset is complex-valued and obtained from a pulse-doppler radar from Saab, it contains 8601 instances, and there are three different labels, "Drone", "Nothing" and "Probably Drone". 3810 instances are "Probably Drone" and therefore treated as unlabeled. The label distribution is skewed, 931 of the instances are labeled as "Drone," and the other 3860 are labeled as "Nothing." These labels are then encoded to 0("Drone") and 1("Nothing"). Furthermore, the labels are obtained by computer software and augmentation.

Because the label distribution is skewed we include dropout layers and regularization into the CNN we use for the supervised baseline. For the details of the supervised baseline see the appendix.

Algorithm

Using a CNN, we first train a classifier $h\colon \mathcal{X} \to \mathcal{Y}$ on our labeled training data, S_L. The model trains for 100 epochs using a batch size $B = 100$ and the Adam optimizer using cross entropy for the loss function.

Once supervised training has been completed, we use h to predict pseudo-labels using weakly augmented unlabeled images $\alpha(x)$. For weak augmentation, we apply Gaussian Blur to all images and 80% of the images are then rotated by -5 to 5 degrees. The one-hot labels will then be $o^p = h(\alpha(x))$ and pseudo labels will then be calculated and included if $\max(o_y^p) > \tau$ otherwise discarded.

We then continue training the model using strongly augmented features $\mathcal{A}(x)$ and pseudo-labels, i.e, we update $h\colon \mathcal{X} \to \mathcal{Y}$ for 30 more epochs using strong augmentations.

For the semi-supervised training we use $\tau = 0.9$, $\mu = 1$ and $B = 100$. <For strong augmentation, we first apply dropout, which means we sample $p \in (0.01, 0.1)$, and drop p% of all pixels but do it independently per channel in 50% of all images. Then we rotate 80% of the images by -25% to 25%. Scale the images to a value of 80% to 120%. Translate the images by -0.2 to 0.2 pixels. Rotate the images by -25% to 25%. Lastly, we shear the images by 12% to 15%.

Experiments

To evaluate the results of the models, we calculate the accuracy of the evaluation set utilizing tf.keras.metric.Accuracy and the loss is calculated as the cross entropy utilizing tf.keras.metrics.categorical_crossentropy. To investigate how the total number of labels affects accuracy, we varied the amount of upsampled labels. We put five different conditions on the total number of labels. Since label 0 had fewer instances, we sampled each label to 1000, 2000, 3000, 4000 and 5000 to keep a balanced class distribution. The supervised classifier was trained for 100 epochs each time and FixMatch was trained for an additional 30 epochs using the unlabeled instances. We ran each simulation six times using different random seeds.

11.4 Results

Figure 11.1 shows a boxplot describing the accuracy of the FixMatch algorithm and the supervised baseline.

In some cases, the simulations failed because none of the pseudo-labels surpassed the threshold and we got a NaN (Not a Number) for the accuracy and loss values for FixMatch. During the analysis the rows containing NaN were dropped and we applied the Bayesian Linear Model to investigate the relative improvement. Table 11.1 shows the estimated parameters for the Bayesian Linear Regression Model. The estimated mean effect of FixMatch is estimated to be $a_{ssl} = 0.049$ since $a_{ssl} > 0$ it indicates that FixMatch has an average better performance than the supervised baseline. The estimated variance is $\sigma = 0.177$, which is more significant than a_{ssl}, meaning that the data has an unexplained variance. Figure 11.1 illustrates the descriptive statistics of the aggregated accuracy as a boxplot. According to Figure 11.1 FixMatch outperforms the supervised baseline in most cases. There are a few instances in which both algorithms perform similarly. Three instances are severe outliers, meaning unlabeled data will degrade performance.

To answer this research question we separated the dataset into five separate datasets, each containing the results from when we fixed the number of upsampled instances. Figure 11.2 shows the accuracy w.r.t the number of labeled instances. We can see that the accuracy of both FixMatch and the supervised baseline increases as the number of labeled instances increases. Figure 11.2 illustrates that FixMatch trained with 2000 labeled instances can reach an accuracy higher than the supervised baseline using up to 4000 labeled instances. Similarly, training FixMatch using 3000 and 4000 can outperform the supervised baseline trained using 4000 and 5000 instances respectively. To understand the average increase in accuracy w.r.t the number of labeled instances we created a new column in the dataset by subtracting the accuracy of the supervised baseline from the accuracy of FixMatch. Figure 11.3 contains boxplots describing how much the accuracy increased when using unlabelled data. We can see that the improvement is most significant for 2000 labeled instances, and then the improvement decreases as the labeled instances increases. For 2000 labeled instances, the improvement is 8-20 units, and for 5000 labeled instances, the improvement is 2-4 units.

To investigate the difference in computation time we created a boxplot with the computation times measured from the simulations w.r.t number of labeled

instances. Figure 11.4 contains the boxplots describing the computation time with respect to the number of labeled instances. We can see that the computation time for FixMatch and the supervised baseline increases as the number of labeled instances increases. To understand how much longer it takes to run the FixMatch algorithm we created a new column in our dataset containing the ratio calculated by dividing the computation time of FixMatch by the computation time of the supervised baseline. Figure 11.5 shows the time ratio between FixMatch and the supervised baseline. It shows that unlabeled data takes 9-18 times longer than only labeled instances. We can see that the ratio is at its highest for 2000 labeled instances, then decreases and approaches nine as the number of labeled instances increases.

We collected the loss curve from the different manual effort scenarios to determine whether unlabeled data will influence the loss so that the model will not overfit or underfit. Figure 11.6 illustrates the validation losses obtained when training the model. We can see that the loss decreases as the training goes on, and after 100 epochs, we can see that the unlabeled data helps decrease the loss dramatically. This improvement in loss reduction is more evident for few labeled instances and less for more labeled instances. We can see that the loss reaches a plateau at the end of the training phase, meaning that the model does not underfit or overfit.

Table 11.1: HPD Intervals for the estimated parameters of Bayesian Linear Regression.

Parameter	Mean	HPD low	HPD high
sigma	0.1177822	0.0841900	0.1526602
a_unsup_acc	0.0490084	0.0060856	0.0959808

11.5 discussion

The model utilized in this paper is based on FixMatch [75]. Algorithms based on FixMatch can improve the performance of supervised baselines utilizing few labeled instances and many unlabeled instances. The original algorithm reaches an accuracy of 86% when utilizing only four labeled instances per class of the CIFAR10 dataset. Our algorithm is less complicated and requires less

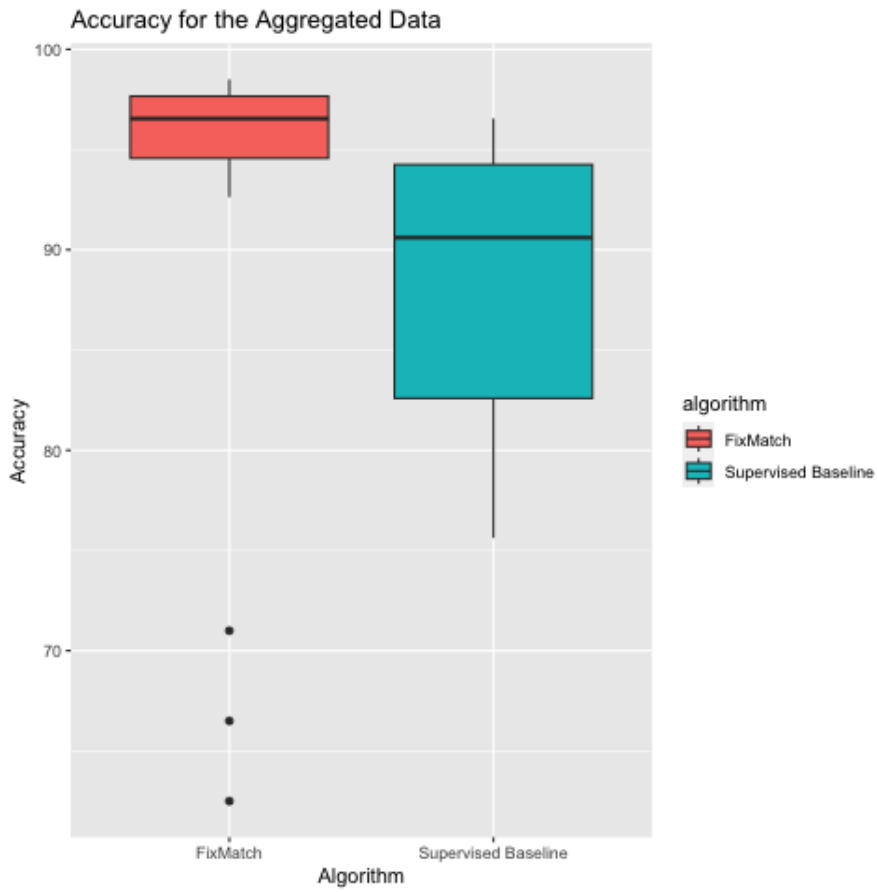


Figure 11.1: Boxplots containing the accuracy for the algorithms of aggregated data.

training time. Our version uses the Adam optimizer, while the original paper proposes SGD as the better choice. We use a constant learning rate while the original uses a cosine learning rate decay. The original FixMtch utilizes an exponential moving average (EMA) while ours do not. Although our algorithm is less complicated and requires less training time, it will outperform the supervised benchmark. Our version of FixMatch is easy to implement and

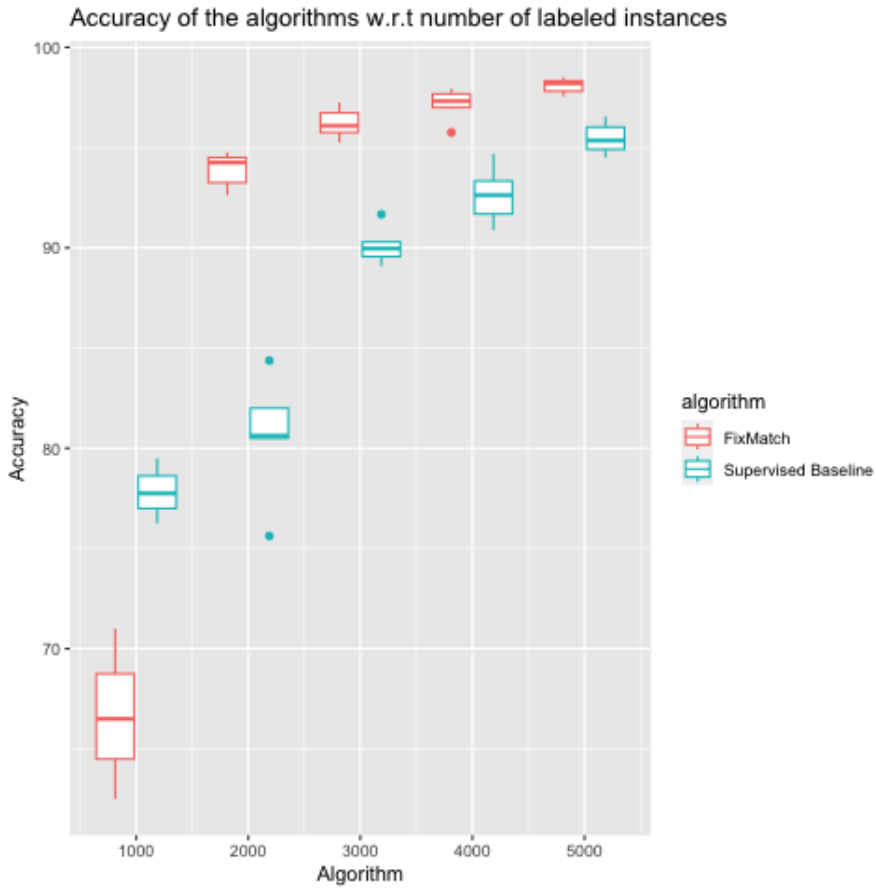


Figure 11.2: Boxplots containing the accuracy for the algorithms w.r.t to the number of labeled instances.

can be applied to any classifier and complex-valued dataset.

There are many ways that we can extend FixMatch. According to [307], algorithms that can outperform FixMatch are UDA [68], ReMixMatch [362] and FlexMatch [78]. FixMatch can be viewed as a simplification of ReMixMatch or UDA as it misses components from these two algorithms. Augmentation Anorching and Distribution Alignment are two techniques that are

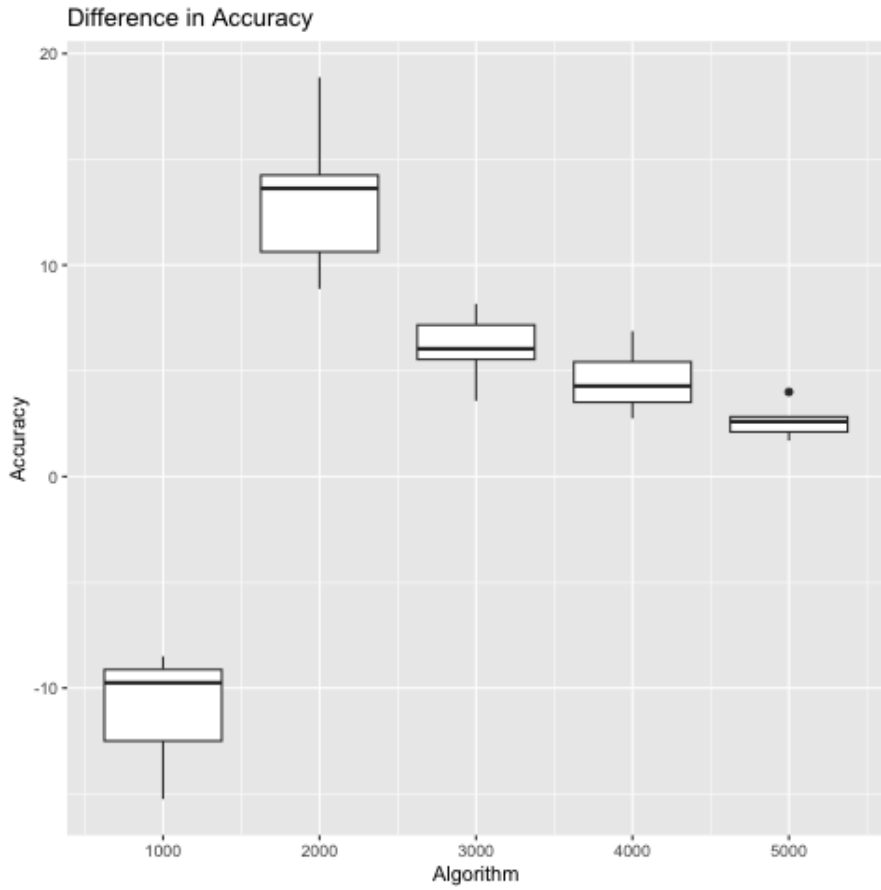


Figure 11.3: Boxplots containing the accuracy for algorithms w.r.t number of labeled instances. The purpose is to illustrate the improvement in accuracy when using FixMatch.

used ReMixMatch that practitioners could try to improve FixMatch. Practitioners can also try different strong augmentations such as MixUp [82] or Adervsarial Perturbations [304] as well as Cutout [363], CTAugment [69] and RandAugment [69]. Another problem with FixMatch is that it uses a constant threshold τ in the pseudo-labeling step and does not account for each class label's learning difficulty. FlexMatch mitigates this by incorporating Curricu-

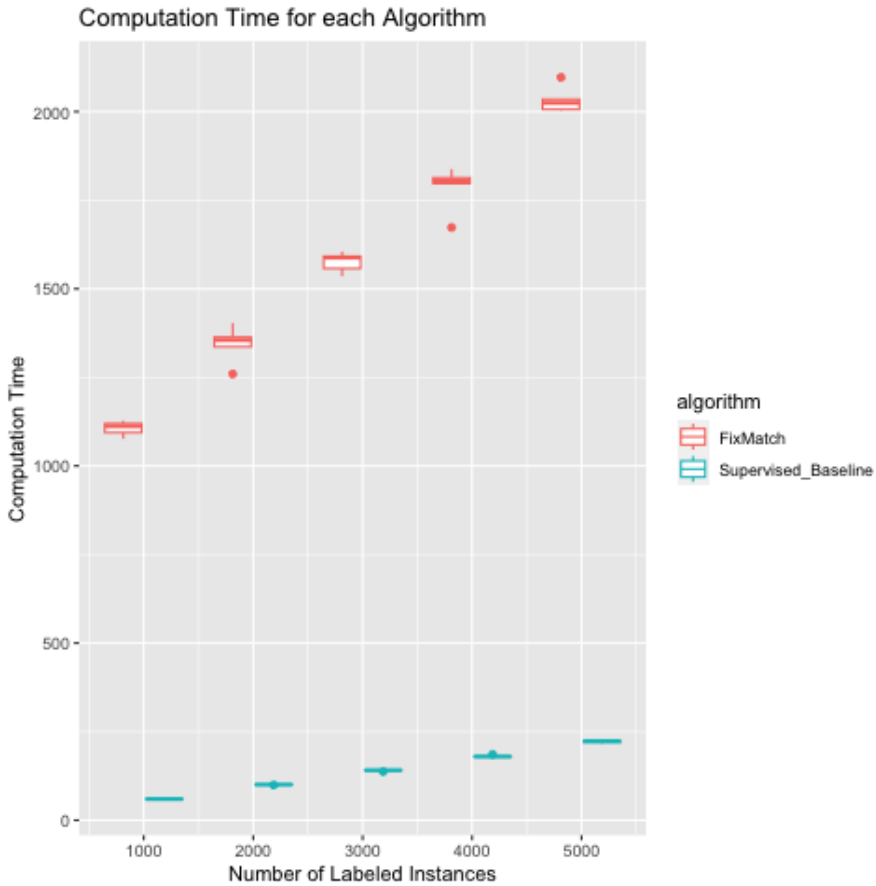


Figure 11.4: Boxplots containing the computation time w.r.t number of labeled instances.

lum Pseudo Labeling to choose unlabeled instances according to the current learning status. FlexMatch has proven to outperform FixMatch in many scenarios and reduces the training time to 20% of the training time required by FixMatch.

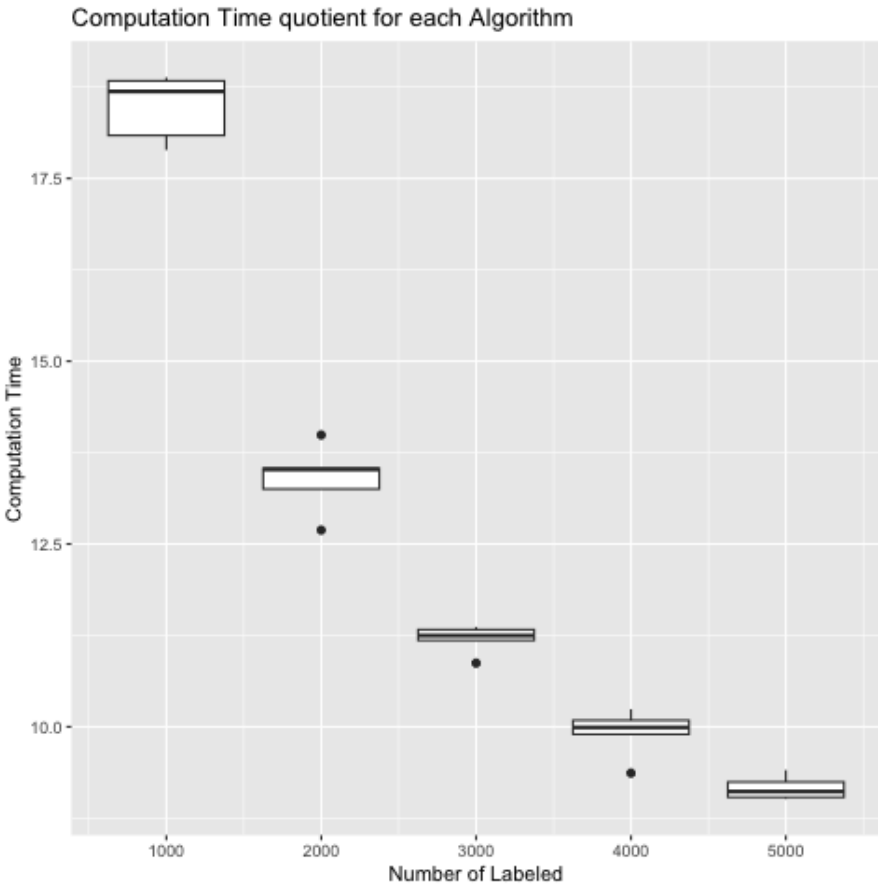


Figure 11.5: Boxplot containing the ratio of computation times. The purpose of the figure is to illustrate how much longer it takes to perform FixMatch compared to the supervised baseline.

Threats to Validity

According to [104], four validity treats are associated with simulation studies. These threats to validity are known as *Construct Validity, External Validity, Internal Validity* and *Conclusion Validity.* Construct validity refers to whether the statistical model is appropriate for answering the research questions. Ex-

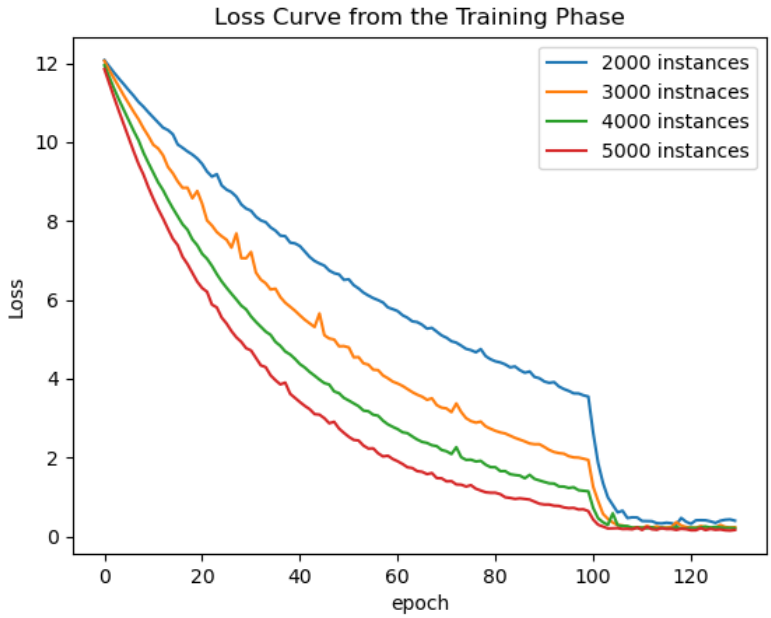


Figure 11.6: The losses obtained during the Training Phase

ternal validity refers to what extent the results apply to other scenarios, such as different datasets. Internal validity refers to whether the treatment variable causes the outcome of the experiments rather than simplifications in the simulation model or because we failed to account for certain factors. Conclusion Validity refers to whether we made the correct assumptions and statistical tests to validate our results.

To account for construct validity, we have calculated the average relative improvement to evaluate how much better SSL is compared to SL. This model is an appropriate model for answering the RQs. The FixMatch has previously been evaluated on many benchmark datasets. The operations performed in this study are similar to the ones in the original paper, and our results are similar, so we have external validity. Regarding internal validity, we did not simplify the model or omit any essential factor. To account for conclusion validity, we made appropriate choices in the Bayesian model and verified the

correctness of the model outputs following the recommendations of [252]. In addition, each simulation was run six times using different seeds to account for randomness.

11.6 Conclusion

This study investigates whether unlabeled data can be utilized to improve the classification of real-world Pulse-Doppler Radar Data using the FixMatch algorithm.

The Bayesian Linear Regression Model shows positive average relative improvement so FixMatch improves on the supervised baseline in most cases. The improvement of FixMatch is guaranteed as long as 2000 labels or more are available. The accuracy and computation time of both FixMatch and the supervised baseline increase as the number of labeled instances increases. It is possible to save manual labeling effort because FixMatch can yield a higher accuracy with fewer labels than the supervised baseline using more labels. The downside is that the computation time for FixMatch will always be higher than the computation time of the supervised baseline, no matter the number of labeled instances. FixMatch is at least times slower than the supervised baseline. Furthermore, using unlabeled data will help the loss decrease faster during the training phase. The loss curve will plateau when the unlabeled data is utilized during the last 30 epochs, so overfitting and underfitting will be avoided.

The results of this study will help practitioners understand the benefit of using FixMatch to utilize unlabeled data when labeled data is expensive to collect. Practitioners will know how to allocate manual labeling to minimize the costs of manual labeling while achieving the highest accuracy. In future studies, we wish to improve our FixMatch algorithm's accuracy using ideas from UDA, ReMixMatch and reduce computation time using Curriculum Pseudo Labeling.

CHAPTER 12

Concluding Remarks and Future Work

This section presents the conclusion and future direction of research. This thesis presents guidelines and mitigation strategies for practitioners in industry and academia that lack labeled instances in datasets. Fully labeled datasets are necessary to train high-performance SL algorithms. However, labeled datasets may lack labels partially or entirely. To label data, companies need to allocate the labeling task to the appropriate personnel and consider all the risks involving the cost of labeling, quality assurance, and privacy concerns of sharing data with third-party organizations. Based on data collected from industry, current challenges regarding labeling in industry were identified. To identify mitigation strategies for practitioners, previous literature was analyzed to collect data regarding available AL and SSL algorithms to assist in labeling or utilizing unlabeled data to improve the performance of SL. The simulation studies evaluate the algorithms and datasets to help practitioners construct benchmark tests and choose optimal automatic labeling and DSSL algorithms. Last, a case study demonstrates the benefits of applying DSSL on real-world datasets.

12.1 Conclusion

This thesis reports exploratory research utilized to discover and analyze challenges and mitigation strategies for data labeling. After formulating challenges and mitigation strategies, a subset of semi-supervised learning algorithms are empirically evaluated. Furthermore, this thesis provides an overview of algorithms applicable to many datasets and suggest suitable datasets for evaluating semi-supervised learning algorithms.

RQ1: What data labeling challenges exist in the industry, and how can they be mitigated using Machine Learning for Data Labeling?

This research question was answered by utilizing a case study in which data was collected from two companies. The data collecting procedure was divided into two phases. First was the exploration phase, in which time was spent with Company A. The second phase was the validation phase where interviews were conducted with company participants. During phase I and II different problems were identified and summarized into three challenges. A mitigation strategy was formulated for each challenge based on active and semi-supervised learning practices. Thanks to the mitigation strategies, companies have been given the tools necessary to solve challenges concerning data labeling. The three different challenges are *Pre-processing Annotation* and *Label Distribution* (C1-C3). The mitigation strategies are called *Planning, Oracle selection* and *Label Distribution* (MS1-MS3). The first challenge involves the planning of the labeling procedure. Practitioners are required to perform suitable exploratory analysis to do labeling with respect to different tasks and choose the correct model based on the data structure. The choice of query strategy is dependent on the underlying ML algorithm. If the underlying ML algorithm reduces expected error or variance, use the query strategy that reduces the expected error or variance. If there is a cluster structure in the data, choose a density-weighted AL or GBSSL algorithm. For probabilistic models use uncertainty sampling. If the labels are to be used for different tasks, use multi-task active learning. To account for labeling costs, use cost-sensitive active learning. If labeling costs vary over time, cost-sensitive AL allows modeling of the labeling cost as a deterministic or stochastic function. Suppose that data is generated from an experiment due to actual data be-

ing too expensive to acquire. Predicting "actual" data based on generated data will therefore predict noisy labels. In many cases some instances will be difficult to label manually. Some people have a limited attention span and therefore the quality of the labels will decrease over time. Crowdsourcing can be a solution as it allows several people to label the instances, making it easier to detect errors. Another solution is to use repeated labeling to reduce uncertainty in the oracle and model. The label distribution challenge (C3) involves problems concerning uncertainty in the label distribution such as skewness. If the label distribution is skewed then AL might not outperform PL. A solution for this is to search for class representative instances using guided learning which can create a more balanced dataset.

RQ2: What existing algorithms exist that may be utilized to solve the labelling challenges?

This research question was answered by studying previous research on machine learning algorithms that will reduce or eliminate manual data labeling. The most popular machine learning algorithms were identified by conducting a systematic mapping study. In addition, 87 datasets that are used to evaluate algorithms were identified. The datasets are distributed across four datatypes, image, text, sound, and numerical. The majority of these datasets are images and numerical. A taxonomy of algorithms consisting of AL and SSL were provided and the applications where these algorithms are used were outlined. This was done to conclude what algorithms are used for a specific application. The classes of SSL algorithms are *GBSSL, GMM, MVL* and *S3VM*, and the classes of AL algorithms are *uncertainty sampling, density-weighted methods, expected variance reduction* and *QBC*. SSL algorithms all rely on different assumptions on the dataset. Theoretically, if the dataset only satisfies a few assumptions, then not every algorithm class will work on the dataset. The difference in assumptions may the reason why algorithms of different classes are not evaluated together. The optimal AL algorithm varies but in almost every case AL outperforms random sampling. All algorithms have been primarily evaluated on image data and secondly on text data. Fewer studies have been found involving numerical data. Only two studies involved sound data, one utilizing GBSSL and the other S3VM. No AL algorithms were applied to sound data. Uncertainty sampling was applied to images and text datasets but not to numerical data. QBC algorithms were mostly used on text and a

few studies involved numeric data.

RQ3: What Machine Learning algorithms for Data Labeling are optimal for achieving high accuracy while maintaining low labeling costs?

Based on the systematic mapping study, GBSSL algorithms are the most popular algorithms for automatic labeling. 13 GBSSL algorithms from the open-source Python package GraphLearning were evaluated on 12 of the 87 popular datasets. The algorithms in USB were evaluated due to the recent popularity of the DSSL algorithms. Each simulation was performed many times to account for the stochastic nature of the algorithms. The datasets used in the evaluation were of different data types to demonstrate to practitioners how algorithms may perform differently on different data types. The number of labeled instances varied to illustrate how the performance of the algorithms is affected when the number of labeled instances increases. In addition, noise is added to see if the performance increases in the presence of noise. All algorithms are ranked utilizing the Bradley-Terry model and a Binomial model is applied to calculate the probability of algorithms achieving a certain threshold. For GBSSL algorithms, the probability of achieving an accuracy above 90% is considered, and for DSSL algorithms, the probability of achieving an error-rate less than 10%. The results demonstrate that Poisson2 is the optimal labeling algorithm that outranks all other algorithms. Poison 2 has high probability of achieving an accuracy above 90% for all datatypes except for text where noise is required. For DSSL, many algorithms perform similarly and therefore there is uncertainty in their ranks. Based on results the following algorithms are recommended: FreeMatch for all datatypes and for large allocation of labels, SimMatch for image datatypes and small allocation of labels. Finally, SoftMatch is recommended for text datatypes and small allocation of labels. None of the algorithms achieves an error rate below 0.1 with and without labels. The performance of both GBSSL and DSSL algorithms increases the more labels are available and the performance is increased in the presence of noise, indicating that the algorithms perform well on real-world datasets.

RQ4: Do benchmark datasets contribute to a fair evaluation of Graph-based Semi-Supervised algorithms?

Constructing benchmark tests for evaluating algorithms is essential in developing generic algorithms. However, constructing benchmark tests may be time-consuming for practitioners because the appropriate datasets must be selected to represent the purpose for which the algorithm is designed. Based on the results from evaluating GBSSL, many algorithms work better with certain datatypes than others. This observation suggests that many datasets are too easy for evaluating algorithms. To illustrate that datasets may be inappropriate for evaluating GBSSL and DSSL algorithms, the Jöreskog item response theory model is utilized. The Jöreskog model calculates three parameters: the ability of the algorithms to learn a dataset, the datasets' discrimination, and the datasets' difficulty. Optimal datasets have high discrimination and high difficulty. For GBSSL, only Reuters, Ohsumed, and Musk have higher discrimination and high difficulty, the other datasets have low discrimination and low difficulty. Therefore only 25% of the datasets are suitable for evaluating GBSSL algorithms. For DSSL algorithms, 2 out of 3 algorithms have high discrimination and difficulty parameters and are therefore suitable for evaluating DSSL. The datasets that are unsuitable for evaluating DSSL algorithms are all image datasets, it is therefore concluded that many benchmark tests need to be updated to include more datasets that have high difficulty and high discrimination, particularly image datasets.

RQ5: What are the pros and cons of utilizing SSL for Drone classification in a real-world Doppler-Radar dataset

Many studies on SSL utilize benchmark datasets, and a few utilize real-world datasets from the industry. One reason for this is that these algorithms are patented and are not shared for legal reasons. Another reason is that practitioners want to develop generic algorithms and therefore utilize benchmark datasets. To demonstrate the pros and cons of utilizing DSSL on a real-world dataset, an algorithm based on FixMatch is applied to a dataset containing Pulse-Doppler data. A supervised CNN is trained, and pseudo-labeling and consistency regularization are utilized to incorporate unlabeled data. Bayesian linear regression is utilized to investigate the relative improvement of DSSL compared to DSL. The number of available labels varies to determine if the

performance is correlated with the number of labels. The label distribution is skewed, so labels are upsampled to balance the distribution. The number of labels is upsampled to 1000, 2000, 3000, 4000, and 5000. The results demonstrate that unlabeled data will improve performance when there are at least 2000 instances of each label. The increase in performance becomes less significant as the number of available labels increases. On the downside, increasing the number of labels and incorporating unlabeled data will increase computation time. On the positive side, utilizing unlabeled data will help reduce overfitting and underfitting

12.2 Summary of Contributions

This thesis examines the challenges and mitigation strategies associated with the problem of missing labels in supervised learning.

Paper A reports AL and SSL algorithms that may be used to solve labeling-related issues. AL selects instances to be labeled according to a query strategy, thereby reducing the need for labeling. SSL labels data automatically or strives to improve the performance of supervised learning by incorporating unlabeled data, thereby eliminating labeling completely.

Paper B identifies labeling-related challenges faced by the industry and the mitigation strategies used to address these challenges. The results from paper A are used to improve the mitigation strategies.

Paper C-G report on empirical evaluations of algorithms and datasets.

Papers C and E are empirical evaluations of algorithms. Papers C evaluates GBSSL algorithms for the automatic labeling of data, while Paper E assesses DSSL algorithms that enhance the performance of SL by incorporating labeled data. Both chapters compare the performance of algorithms using multiple datasets of different datatypes and study how the performance changes when the number of labeled instances varies. Noise is added to the dataset to simulate the performance of the algorithms on datasets with high complexity. We also calculate the probability of an algorithm reaching a specific performance.

Papers D and G evaluate the datasets used to evaluate the GBSSL algorithms and DSSL algorithms. If a dataset is too easy for the algorithms to learn from, then it is unsuitable for evaluating algorithms.

Lastly, in Paper G, we develop a DSSL algorithm on a real-world dataset collected from Saab. The algorithm utilizes unlabeled data to outperform

supervised learning. The number of instances used to train the classifier is varied to investigate how much the performance increases.

12.3 Future Work

This thesis presents mitigation strategies for challenges faced by practices that lack labels for training supervised classification models. The most popular AL and SSL algorithms are presented and evaluated to help practitioners choose their optimal algorithm based on their use-case. In addition, the optimal datasets for evaluating GBSSL and DSSL algorithms are presented to help practitioners choose datasets to include in benchmark tests. Last, a case study demonstrates that SSL can improve the performance and of SL at the cost of computation time.

Based on the empirical evidence from Papers A-G, practitioners know which GBSSL and DSSL algorithms are optimal in different use-cases. However, the IRT studies suggest that image datasets are inappropriate for evaluating algorithms. This may come from the fact that many algorithms were initially developed to work on image datasets and are not originally evaluated on text, numerical, and audio datasets. In addition, it is demonstrated that DSSL may outperform DSL, but it is not understood why SSL outperforms SL. In particular, why do DSSL algorithms seem more effective than other SSL algorithms? In SSL entropy regularization and the maximum a posteriori framework are utilized to incorporate unlabeled data in the loss function [62]. This means that the prior distribution must be specified, and this prior is chosen according to the maximum entropy principle [62]. However, this is based on theory rather than empirical evidence. In addition, the theoretical studies demonstrating that unlabeled data is useful are limited because they rely only on a type of algorithm and loss [33], [140]. By not relying on empirical evaluations and limiting our thinking to pure theoretical constructs, it is possible that we may miss out on new ways to make use of unlabeled data.

Therefore, in future research it is relevant to explore new ways to improve the performance of SL through empirical evidence and simulations utilizing data rather than relying on theoretical principles that lack empirical evaluation.

References

[1] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016.

[2] A. Klenke, *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[3] J. O. Berger, *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[4] G. Casella and R. Berger, *Statistical inference*. CRC press, 2024.

[5] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[6] E. D. Sontag, *Mathematical control theory: deterministic finite dimensional systems*. Springer Science & Business Media, 2013, vol. 6.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[9] H. Cloud Factory, *Crowd vs. managed team: A studo on quality data processing at scale*, https://go.cloudfactory.com/hubfs/02-Contents/3-Reports/Crowd-vs-Managed-Team-Hivemind-Study.pdf, 2020.

[10] S. Shafaei, S. Kugele, M. H. Osman, and A. Knoll, "Uncertainty in machine learning: A safety perspective on autonomous driving," in *Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings 37*, Springer, 2018, pp. 458–464.

[11] J. Zheng, D. Lin, Z. Gao, S. Wang, M. He, and J. Fan, "Deep learning assisted efficient adaboost algorithm for breast cancer detection and early diagnosis," *IEEE Access*, vol. 8, pp. 96 946–96 954, 2020.

[12] Cognilytica Research, "Data Preparation & Labeling for AI 2020," Cognilytica Research, Tech. Rep., 2020, pp. 1–35.

[13] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data-ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[14] V. Dogra and S. Verma, "Challenges and opportunities in labeling for text classification," *Think India Journal*, vol. 22, no. 17, pp. 4390–4400, 2019.

[15] MedCase, *Medical data annotation: What you need to know*, "https://www.medcase.h
data-labeling-the-key-to-successful-machine-learning-applications acessed, 2023.

[16] T. Fredriksson, D. I. Mattos, J. Bosch, and H. H. Olsson, "Data labeling: An empirical investigation into industrial challenges and mitigation strategies," in *Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21*, Springer, 2020, pp. 202–216.

[17] H. S. Alenezi and M. H. Faisal, "Utilizing crowdsourcing and machine learning in education: Literature review," *Education and Information Technologies*, vol. 25, no. 4, pp. 2971–2986, 2020.

[18] J. C. Chang, S. Amershi, and E. Kamar, "Revolt: Collaborative crowdsourcing for labeling machine learning datasets," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017, pp. 2334–2346.

[19] J. Zhang, V. S. Sheng, T. Li, and X. Wu, "Improving crowdsourced label quality using noise correction," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1675–1688, 2017.

[20] K. Crowston, "Amazon mechanical turk: A research tool for organizations and information systems scholars," in *Shaping the Future of ICT Research. Methods and Approaches: IFIP WG 8.2, Working Conference, Tampa, FL, USA, December 13-14, 2012. Proceedings*, Springer, 2012, pp. 210–221.

[21] IBM, *Ibm, what is data labeling?* "https://www.software-center.se/", 2025.

[22] E. Mosqueira-Rey, E. Hernandez-Pereira, D. Alonso-Rios, J. Bobes-Bascaran, and A. Fernandez-Leal, "Human-in-the-loop machine learning: A state of the art," *Artificial Intelligence Review*, pp. 1–50, 2022.

[23] B. Settles, "Active learning, volume 6 of synthesis lectures on artificial intelligence and machine learning," *Morgan & Claypool*, 2012.

[24] B. Settles, "Active learning literature survey," 2009.

[25] T. Fredriksson., J. Bosch., and H. H. Olsson., "Machine learning models for automatic labeling: A systematic literature review," in *Proceedings of the 15th International Conference on Software Technologies - Volume 1: ICSOFT,*, INSTICC, SciTePress, 2020, pp. 552–561, ISBN: 978-989-758-443-5.

[26] J. Baldridge and M. Osborne, "Active learning and the total cost of annotation," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004, pp. 9–16.

[27] A. Culotta and A. McCallum, "Reducing labeling effort for structured prediction tasks," in *AAAI*, vol. 5, 2005, pp. 746–751.

[28] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, Atlanta, vol. 3, 2013, p. 896.

[29] A. Kapoor, E. Horvitz, and S. Basu, "Selective supervision: Guiding supervised learning with decision-theoretic active learning.," in *IJCAI*, vol. 7, 2007, pp. 877–882.

[30] B. Settles, "Curious machines: Active learning with structured instances," Ph.D. dissertation, Citeseer, 2008.

[31] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[32] T. J. O'neill, "Normal discrimination with unclassified observations," *Journal of the American Statistical Association*, vol. 73, no. 364, pp. 821–826, 1978.

[33] V. Castelli and T. Cover, "The relative value of labeled and unlabeled samples in pattern recognition," in *Proceedings. IEEE International Symposium on Information Theory*, IEEE, 1993, pp. 355–355.

[34] T. Bartz-Beielstein, C. Doerr, D. v. d. Berg, *et al.*, "Benchmarking in optimization: Best practice and open issues," *arXiv preprint arXiv:2007.03488*, 2020.

[35] P. Sanders, "Algorithm engineering–an attempt at a definition," in *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, Springer, 2009, pp. 321–340.

[36] W. A. S. Program, *Wallenberg autonomous systems program*, "https://wasp-sweden.org/", 2025.

[37] P. R. Halmos, *Measure theory*. Springer, 2013, vol. 18.

[38] V. Vapnik, "Principles of risk minimization for learning theory," *Advances in neural information processing systems*, vol. 4, 1991.

[39] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. MIT press Cambridge, MA, USA, 2017, vol. 1.

[40] A. K. Tyagi *et al.*, "Machine learning with big data," in *Machine Learning with Big Data (March 20, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*, 2019.

[41] L. Atlas, D. Cohn, and R. Ladner, "Training connectionist networks with queries and selective sampling," *Advances in neural information processing systems*, vol. 2, 1989.

[42] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, no. 2, pp. 201–221, 1994.

[43] I. Dagan and S. P. Engelson, "Committee-based sampling for training probabilistic classifiers," in *Machine Learning Proceedings 1995*, Elsevier, 1995, pp. 150–157.

[44] D. Angluin, "Queries and concept learning," *Machine learning*, vol. 2, no. 4, pp. 319–342, 1988.

[45] D. Angluin, "Queries revisited," in *International Conference on Algorithmic Learning Theory*, Springer, 2001, pp. 12–31.

[46] E. B. Baum and K. Lang, "Query learning can work poorly when a human oracle is used," in *International joint conference on neural networks*, Beijing China, vol. 8, 1992, p. 8.

[47] D. E. Graff, E. I. Shakhnovich, and C. W. Coley, "Accelerating high-throughput virtual screening through molecular pool-based active learning," *Chemical science*, vol. 12, no. 22, pp. 7866–7881, 2021.

[48] N. Khoshnevis and R. Taborda, "Application of pool-based active learning in physics-based earthquake ground-motion simulation," *Seismological Research Letters*, vol. 90, no. 2A, pp. 614–622, 2019.

[49] A. Lang, C. Mayer, and R. Timofte, "Best practices in pool-based active learning for image classification," 2021.

[50] T. Fredriksson, J. Bosch, H. H. Olsson, and D. I. Mattos, "Machine learning algorithms for labeling: Where and how they are used?" In *2022 IEEE International Systems Conference (SysCon)*, IEEE, 2022, pp. 1–8.

[51] B. Settles, M. Craven, and L. Friedland, "Active learning with real annotation costs," in *Proceedings of the NIPS workshop on cost-sensitive learning*, Vancouver, CA: 2008, pp. 1–10.

[52] T. M. Mitchell, "Generalization as search," *Artificial intelligence*, vol. 18, no. 2, pp. 203–226, 1982.

[53] P. Melville and R. J. Mooney, "Diverse ensembles for active learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 74.

[54] G. Ngai and D. Yarowsky, "Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking," *arXiv preprint cs/0105003*, 2001.

[55] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," *Advances in neural information processing systems*, vol. 13, 2000.

[56] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," *Advances in neural information processing systems*, vol. 17, 2004.

[57] H. Akaike, "A new look at the bayes procedure," *Biometrika*, vol. 65, no. 1, pp. 53–59, 1978.

[58] H. Akaike, "Information measures and model selection," *Int Stat Inst*, vol. 44, pp. 277–291, 1983.

[59] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[60] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[61] E. T. Jaynes, "Information theory and statistical mechanics. ii," *Physical review*, vol. 108, no. 2, p. 171, 1957.

[62] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[63] J. Bridle, A. Heading, and D. MacKay, "Unsupervised classifiers, mutual information and'phantom targets," *Advances in neural information processing systems*, vol. 4, 1991.

[64] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1168–1175.

[65] Y. Fan, A. Kukleva, D. Dai, and B. Schiele, "Revisiting consistency regularization for semi-supervised learning," *International Journal of Computer Vision*, vol. 131, no. 3, pp. 626–643, 2023.

[66] M. Zheng, S. You, L. Huang, F. Wang, C. Qian, and C. Xu, "Simmatch: Semi-supervised learning with similarity matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 14 471–14 481.

[67] J. Li, C. Xiong, and S. C. Hoi, "Comatch: Semi-supervised learning with contrastive graph regularization," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9475–9484.

[68] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," *Advances in neural information processing systems*, vol. 33, pp. 6256–6268, 2020.

[69] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.

[70] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[71] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," *arXiv preprint arXiv:1511.06709*, 2015.

[72] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," *arXiv preprint arXiv:1808.09381*, 2018.

[73] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," *Advances in neural information processing systems*, vol. 28, 2015.

[74] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[75] K. Sohn, D. Berthelot, N. Carlini, *et al.*, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in neural information processing systems*, vol. 33, pp. 596–608, 2020.

[76] Y. Xu, L. Shang, J. Ye, *et al.*, "Dash: Semi-supervised learning with dynamic thresholding," in *International conference on machine learning*, PMLR, 2021, pp. 11 525–11 536.

[77] D. Berthelot, R. Roelofs, K. Sohn, N. Carlini, and A. Kurakin, "Adamatch: A unified approach to semi-supervised learning and domain adaptation," *arXiv preprint arXiv:2106.04732*, 2021.

[78] B. Zhang, Y. Wang, W. Hou, *et al.*, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," *Advances in neural information processing systems*, vol. 34, pp. 18 408–18 419, 2021.

[79] H. Chen, R. Tao, Y. Fan, *et al.*, "Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning," *arXiv preprint arXiv:2301.10921*, 2023.

[80] Y. Wang, H. Chen, Q. Heng, *et al.*, "Freematch: Self-adaptive thresholding for semi-supervised learning," *arXiv preprint arXiv:2205.07246*, 2022.

[81] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[82] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[83] D. Berthelot, N. Carlini, E. D. Cubuk, *et al.*, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *arXiv preprint arXiv:1911.09785*, 2019.

[84] Y. Wang, H. Chen, Y. Fan, *et al.*, "Usb: A unified semi-supervised learning benchmark for classification," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3938–3961, 2022.

[85] P. Indyk, R. Levi, and R. Rubinfeld, "Approximating and testing k-histogram distributions in sub-linear time," in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, 2012, pp. 15–22.

[86] A. Subramanya and J. Bilmes, "Semi-supervised learning with measure propagation.," *Journal of Machine Learning Research*, vol. 12, no. 11, 2011.

[87] M. Karlen, J. Weston, A. Erkan, and R. Collobert, "Large scale manifold transduction," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 448–455.

[88] G. Călinescu, H. Karloff, and Y. Rabani, "An improved approximation algorithm for multiway cut," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 48–52.

[89] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 51, no. 2, pp. 271–279, 1989.

[90] J. Lafferty, A. McCallum, F. Pereira, *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Icml*, Williamstown, MA, vol. 1, 2001, p. 3.

[91] Y. Boykov, O. Veksler, and R. Zabih, "Markov random fields with efficient approximations," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, IEEE, 1998, pp. 648–655.

[92] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning*, Semi-Supervised Learning. MIT Press, Jan. 2006, pp. 193–216.

[93] *sklearn.semi_supervised.LabelSpreading — scikit-learn 0.23.2 documentation.*

[94] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[95] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[96] C. Garcia-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. G. Percus, "Multiclass data segmentation using diffuse interface methods on graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1600–1613, 2014.

[97] Z. Shi, S. Osher, and W. Zhu, "Weighted nonlocal laplacian on interpolation from sparse data," *Journal of Scientific Computing*, vol. 73, no. 2, pp. 1164–1177, 2017.

[98] A. Jung, A. O. Hero III, A. Mara, and S. Jahromi, "Semi-supervised learning via sparse label propagation," *arXiv preprint arXiv:1612.01414*, 2016.

[99] M. Jacobs, E. Merkurjev, and S. Esedoḡlu, "Auction dynamics: A volume constrained mbo scheme," *Journal of Computational Physics*, vol. 354, pp. 288–310, 2018.

[100] M. Flores, J. Calder, and G. Lerman, "Algorithms for lp-based semi-supervised learning on graphs," *arXiv preprint arXiv:1901.05031*, 2019.

[101] J. Calder, B. Cook, M. Thorpe, and D. Slepcev, "Poisson learning: Graph based semi-supervised learning at very low label rates," in *International Conference on Machine Learning*, PMLR, 2020, pp. 1306–1316.

[102] J. Calder, *Graph-based clustering and semi-supervised learning*, https://github.com/j 2023.

[103] U. o. G. Chalmers University of Technology, *Software center*, "https://www.software-center.se/", 2025.

[104] B. B. N. de França and G. Travassos, "Simulation based studies in software engineering: A matter of validity.," in *CIbSE*, 2014, pp. 308–321.

[105] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[106] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, 2008, pp. 1–10.

[107] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[108] B. B. N. de França and G. H. Travassos, "Reporting guidelines for simulation-based studies in software engineering," in *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, IET, 2012, pp. 156–160.

[109] T. Fredriksson, J. Bosch, and H. H. Olsson, "Machine learning models for automatic labeling: A systematic literature review.," *ICSOFT*, pp. 552–561, 2020.

[110] J. Cohen, "The earth is round (p<. 05).," *American psychologist*, vol. 49, no. 12, p. 997, 1994.

[111] J. P. Ioannidis, "Why most published research findings are false," *PLoS medicine*, vol. 2, no. 8, e124, 2005.

[112] C. A. Furia, R. Feldt, and R. Torkar, "Bayesian data analysis in empirical software engineering research," *IEEE Transactions on Software Engineering*, 2019.

[113] M.-A. Storey, N. A. Ernst, C. Williams, and E. Kalliamvakou, "The who, what, how of software engineering research: A socio-technical framework," *Empirical Software Engineering*, vol. 25, pp. 4097–4129, 2020.

[114] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[115] C. P. Robert *et al.*, *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer, 2007, vol. 2.

[116] D. I. Mattos, J. Bosch, and H. H. Olsson, "Statistical models for the analysis of optimization algorithms with benchmark functions," *IEEE Transactions on Evolutionary Computation*, 2021.

[117] A. Agresti, *Categorical data analysis*. John Wiley & Sons, 2003, vol. 482.

[118] D. Issa Mattos and É. Martins Silva Ramos, "Bayesian paired comparison with the bpcs package," *Behavior Research Methods*, pp. 1–21, 2022.

[119] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[120] M. Cattelan, "Models for paired comparison data: A review with emphasis on dependent data," *Statistical Science*, pp. 412–433, 2012.

[121] C. Spearman, "The proof and measurement of association between two things.," 1961.

[122] A. Binet and T. Simon, "Méthodes nouvelles pour le diagnostic du niveau intellectuel des anormaux," *L'année Psychologique*, vol. 11, no. 1, pp. 191–244, 1904.

[123] L. L. Thurstone, "A method of scaling psychological and educational tests.," *Journal of educational psychology*, vol. 16, no. 7, p. 433, 1925.

[124] F. Lord, "A theory of test scores.," *Psychometric monographs*, 1952.

[125] G. Rasch, *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[126] K. G. Jöreskog, "Statistical analysis of sets of congeneric tests," *Psychometrika*, vol. 36, no. 2, pp. 109–133, 1971.

[127] D. S. Triangulation, "The use of triangulation in qualitative research," in *Oncol Nurs Forum*, vol. 41, 2014, pp. 545–7.

[128] AzatiSoftware, *AzatiSoftware automated data labeling with machine learning*, https://azati.ai/automated-data-labeling-with-machine-learning, 2019.

[129] R. Board and L. Pitt, "Semi-supervised learning," *Machine Learning*, vol. 4, no. 1, pp. 41–65, 1989.

[130] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *arXiv preprint arXiv:1406.2080*, vol. 2, no. 3, p. 4, 2014.

[131] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[132] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[133] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[134] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and information systems*, vol. 35, no. 2, pp. 249–283, 2013.

[135] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *Language, Knowledge, and Representation*, Springer, 2004, pp. 103–111.

[136] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[137] S. Dasgupta, M. L. Littman, and D. McAllester, "Pac generalization bounds for co-training," *Advances in neural information processing systems*, vol. 1, pp. 375–382, 2002.

[138] V. R. de Sa, "Learning classification with unlabeled data," in *Advances in neural information processing systems*, Citeseer, 1994, pp. 112–119.

[139] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[140] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recognition Letters*, vol. 16, no. 1, pp. 105–111, 1995.

[141] J. Ratsaby and S. S. Venkatesh, "Learning from a mixture of labeled and unlabeled examples with parametric side information," in *Proceedings of the eighth annual conference on Computational learning theory*, 1995, pp. 412–417.

[142] F. G. Cozman, I. Cohen, M. C. Cirelo, *et al.*, "Semi-supervised learning of mixture models," in *ICML*, vol. 4, 2003, p. 24.

[143] V. Vapnik and V. Vapnik, *Statistical learning theory 156–160*, 1998.

[144] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 441–448.

[145] S. I. Daitch, J. A. Kelner, and D. A. Spielman, "Fitting a graph to vector data," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 201–208.

[146] P. S. Dhillon, P. P. Talukdar, and K. Crammer, "Inference-driven metric learning for graph construction," in *4th North East Student Colloquium on Artificial Intelligence*, 2010.

[147] X. Zhu, J. S. Kandola, J. Lafferty, and Z. Ghahramani, *Graph kernels by spectral transforms.* 2006.

[148] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy, "Semi-supervised learning using randomized mincuts," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 13.

[149] S. Baluja, R. Seth, D. Sivakumar, *et al.*, "Video suggestion and discovery for youtube: Taking random walks through the view graph," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 895–904.

[150] P. P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 442–457.

[151] M. Orbach and K. Crammer, "Graph-based transduction with confidence," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2012, pp. 323–338.

[152] M. S. T. Jaakkola and M. Szummer, "Partially labeled classification with markov random walks," *Advances in neural information processing systems (NIPS)*, vol. 14, pp. 945–952, 2002.

[153] A. A. D. Corduneanu, "The information regularization framework for semi-supervised learning," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

[154] A. Corduneanu and T. S. Jaakkola, "On information regularization," *arXiv preprint arXiv:1212.2466*, 2012.

[155] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization.," in *AISTATS*, vol. 1, 2005.

[156] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR'94*, Springer, 1994, pp. 3–12.

[157] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 148–156.

[158] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1070–1079.

[159] R. Hwa, "Sample selection for statistical parsing," *Computational linguistics*, vol. 30, no. 3, pp. 253–276, 2004.

[160] A. Fujii, K. Inui, T. Tokunaga, and H. Tanaka, "Selective sampling for example-based word sense disambiguation," *arXiv preprint cs/9910020*, 1999.

[161] M. Lindenbaum, S. Markovitch, and D. Rusakov, "Selective sampling for nearest neighbor classifiers," *Machine learning*, vol. 54, no. 2, pp. 125–152, 2004.

[162] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[163] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *Proceedings of the ninth ACM international conference on Multimedia*, 2001, pp. 107–118.

[164] A. K. McCallumzy and K. Nigamy, "Employing em and pool-based active learning for text classification," in *Proc. International Conference on Machine Learning (ICML)*, Citeseer, 1998, pp. 359–367.

[165] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 79.

[166] Z. Xu, R. Akella, and Y. Zhang, "Incorporating diversity and density in active learning for relevance feedback," in *European Conference on Information Retrieval*, Springer, 2007, pp. 246–257.

[167] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 287–294.

[168] N. Abe, "Query learning strategies using boosting and bagging," *Proc. of 15ˆ< th> Int. Cmf. on Machine Learning (ICML98)*, pp. 1–9, 1998.

[169] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Technical report, Ver. 2.3 EBSE Technical Report. EBSE, Tech. Rep., 2007.

[170] C. Zhang and W.-S. Zheng, "Semi-supervised multi-view discrete hashing for fast image search," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2604–2617, 2017.

[171] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, IEEE, 2007, pp. 469–475.

[172] F. Wu, X.-Y. Jing, J. Zhou, *et al.*, "Semi-supervised multi-view individual and sharable feature learning for webpage classification," in *The World Wide Web Conference*, 2019, pp. 3349–3355.

[173] Z. Yu, L. Su, L. Li, Q. Zhao, C. Mao, and J. Guo, "Question classification based on co-training style semi-supervised learning," *Pattern Recognition Letters*, vol. 31, no. 13, pp. 1975–1980, 2010.

[174] X. Cui, J. Huang, and J.-T. Chien, "Multi-view and multi-objective semi-supervised learning for hmm-based automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 7, pp. 1923–1935, 2012.

[175] Y. Xia, F. Liu, D. Yang, *et al.*, "3d semi-supervised learning with uncertainty-aware multi-view co-training," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 3646–3655.

[176] K. Lim, J. Y. Lee, J. Carbonell, and T. Poibeau, "Semi-supervised learning on meta structure: Multi-task tagging and parsing in low-resource scenarios," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 8344–8351.

[177] C. Chen, Y. Li, H. Qian, Z. Zheng, and Y. Hu, "Multi-view semi-supervised learning for classification on dynamic networks," *Knowledge-Based Systems*, vol. 195, p. 105 698, 2020.

[178] R. Yan and M. Naphade, "Semi-supervised cross feature learning for semantic concept detection in videos," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 657–663.

[179] R. G. Colares, P. Machado, M. de Faria, A. Detoni, V. Tavano, *et al.*, "Microalgae classification using semi-supervised and active learning based on gaussian mixture models," *Journal of the Brazilian Computer Society*, vol. 19, no. 4, pp. 411–422, 2013.

[180] L. Bull, K. Worden, and N. Dervilis, "Towards semi-supervised and probabilistic classification in structural health monitoring," *Mechanical Systems and Signal Processing*, vol. 140, p. 106 653, 2020.

[181] L. Shi, R. Mihalcea, and M. Tian, "Cross language text classification by model translation and semi-supervised learning," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2010, pp. 1057–1067.

[182] J.-T. Huang and M. Hasegawa-Johnson, "On semi-supervised learning of gaussian mixture models for phonetic classification," in *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, Association for Computational Linguistics, 2009, pp. 75–83.

[183] N. Kumar and S. P. Awate, "Semi-supervised robust mixture models in rkhs for abnormality detection in medical images," *IEEE Transactions on Image Processing*, vol. 29, pp. 4772–4787, 2020.

[184] R. Lang, R. Lu, C. Zhao, H. Qin, and G. Liu, "Graph-based semi-supervised one class support vector machine for detecting abnormal lung sounds," *Applied Mathematics and Computation*, vol. 364, p. 124 487, 2020.

[185] S. Wang, X. Guo, Y. Tie, I. Lee, L. Qi, and L. Guan, "Graph-based safe support vector machine for multiple classes," *IEEE Access*, vol. 6, pp. 28 097–28 107, 2018.

[186] T. Zhang and Z.-H. Zhou, "Semi-supervised optimal margin distribution machines.," in *IJCAI*, 2018, pp. 3104–3110.

[187] K. Zhang, C. Li, Y. Wang, X. Zhu, and H. Wang, "Collaborative support vector machine for malware detection," *Procedia Computer Science*, vol. 108, pp. 1682–1691, 2017.

[188] Z. Yang and Y. Xu, "A safe screening rule for laplacian support vector machine," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 309–316, 2018.

[189] M. P. Kumar and M. K. Rajagopal, "Detecting facial emotions using normalized minimal feature vectors and semi-supervised twin support vector machines classifier," *Applied Intelligence*, vol. 49, no. 12, pp. 4150–4174, 2019.

[190] R. Rastogi and S. Sharma, "Fast laplacian twin support vector machine with active learning for pattern classification," *Applied Soft Computing*, vol. 74, pp. 424–439, 2019.

[191] H. Pei, Q. Lin, L. Yang, and P. Zhong, "A novel semi-supervised support vector machine with asymmetric squared loss," *Advances in Data Analysis and Classification*, pp. 1–33, 2020.

[192] M. Zhao, T. W. Chow, Z. Zhang, and B. Li, "Automatic image annotation via compact graph based semi-supervised learning," *Knowledge-Based Systems*, vol. 76, pp. 148–165, 2015.

[193] J. Tang, H. Li, G.-J. Qi, and T.-S. Chua, "Image annotation by graph-based inference with integrated multiple/single instance representations," *IEEE Transactions on Multimedia*, vol. 12, no. 2, pp. 131–141, 2009.

[194] J. Tang, R. Hong, S. Yan, T.-S. Chua, G.-J. Qi, and R. Jain, "Image annotation by knn-sparse graph-based label propagation over noisily tagged web images," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 2, pp. 1–15, 2011.

[195] Y. Liu and K. Kirchhoff, "Graph-based semi-supervised learning for phone and segment classification.," in *INTERSPEECH*, 2013, pp. 1840–1843.

[196] X. Zeng, D. F. Wong, L. S. Chao, and I. Trancoso, "Graph-based semi-supervised model for joint chinese word segmentation and part-of-speech tagging," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 770–779.

[197] C. Chen, Y. Gong, and Y. Tian, "Semi-supervised learning methods for network intrusion detection," in *2008 IEEE international conference on systems, man and cybernetics*, IEEE, 2008, pp. 2603–2608.

[198] W. Lin, Z. Gao, and B. Li, "Shoestring: Graph-based semi-supervised classification with severely limited labeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4174–4182.

[199] B. Yoon, Y. Jeong, and S. Kim, "Detecting a risk signal in stock investment through opinion mining and graph-based semi-supervised learning," *IEEE Access*, vol. 8, pp. 161 943–161 957, 2020.

[200] W. D. G. de Oliveira, O. A. Penatti, and L. Berton, "A comparison of graph-based semi-supervised learning for data augmentation," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, IEEE, 2020, pp. 264–271.

[201] S. M. K. Zaman, X. Liang, and H. Zhang, "Graph-based semi-supervised learning for induction motors single-and multi-fault diagnosis using stator current signal," in *2020 IEEE/IAS 56th Industrial and Commercial Power Systems Technical Conference (I&CPS)*, IEEE, 2020, pp. 1–10.

[202] J. Tang, H. Li, G.-J. Qi, and T.-S. Chua, "Integrated graph-based semi-supervised multiple/single instance learning framework for image annotation," in *Proceedings of the 16th ACM international conference on Multimedia*, 2008, pp. 631–634.

[203] J. Zhu, H. Wang, T. Yao, and B. K. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008, pp. 1137–1144.

[204] R. Liere and P. Tadepalli, "Active learning with committees for text categorization," in *AAAI/IAAI*, 1997, pp. 591–596.

[205] E. Ringger, P. McClanahan, R. Haertel, *et al.*, "Active learning for part-of-speech tagging: Accelerating corpus annotation," in *Proceedings of the Linguistic Annotation Workshop*, 2007, pp. 101–108.

[206] F. K. Nakano, R. Cerri, and C. Vens, "Active learning for hierarchical multi-label classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1496–1530, 2020.

[207] Y. Sheng, Y. Wu, J. Yang, W. Lu, P. Villars, and W. Zhang, "Active learning for the power factor prediction in diamond-like thermoelectric materials," *npj Computational Materials*, vol. 6, no. 1, pp. 1–7, 2020.

[208] S. Li, Y. Xue, Z. Wang, and G. Zhou, "Active learning for cross-domain sentiment classification," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[209] S. Bommaraveni, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Active content popularity learning and caching optimization with hit ratio guarantees," *IEEE Access*, vol. 8, pp. 151 350–151 359, 2020.

[210] R. C. Prati, G. E. Batista, and M. C. Monard, "Data mining with imbalanced class distributions: Concepts and methods.," in *IICAI*, 2009, pp. 359–376.

[211] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[212] R. C. Gonzalez and R. E. Woods, *Digital image processing, hoboken*, 2018.

[213] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: A survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 543–576, 2016.

[214] hackernoon.com, *Crowdsourcing data labeling for machine learning projects*, https://hackernoon.com/crowdsourcing-data-labeling-for-machine-learning-projects-a-how-to-guide-cp6h32nd, 2020.

[215] P. G. Ipeirotis, F. Provost, V. S. Sheng, and J. Wang, "Repeated labeling using multiple noisy labelers," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 402–441, 2014.

[216] A. Sheshadri and M. Lease, "Square: A benchmark for research on computing crowd consensus," in *First AAAI conference on human computation and crowdsourcing*, 2013.

[217] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *arXiv preprint arXiv:1911.00068*, 2019.

[218] P. Reason and H. Bradbury, *Handbook of action research: Participative inquiry and practice*. Sage, 2001.

[219] M. Staron, *Action Research in Software Engineering: Theory and Applications*. Springer Nature, 2019.

[220] T. DataScience, *What to do when your classification data is imbalanced?* https://towardsdatascience.com/what-to-do-when-your- classification-dataset-is-imbalanced-6af031b12a36, 2019.

[221] E. Bair, "Semi-supervised clustering methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 5, pp. 349–361, 2013.

[222] C. Körner and S. Wrobel, "Multi-class ensemble-based active learning," in *European conference on machine learning*, Springer, 2006, pp. 687–694.

[223] A. I. Schein and L. H. Ungar, "Active learning for logistic regression: An evaluation," *Machine Learning*, vol. 68, no. 3, pp. 235–265, 2007.

[224] A. Harpale, "Multi-task active learning," Ph.D. dissertation, Carnegie Mellon University, 2012.

[225] S. Arora, E. Nyberg, and C. Rose, "Estimating annotation cost for active learning in a multi-annotator environment," in *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, 2009, pp. 18–26.

[226] E. K. Ringger, M. Carmen, R. Haertel, *et al.*, "Assessing the costs of machine-assisted corpus annotation through a user study.," in *LREC*, vol. 8, 2008, pp. 3318–3324.

[227] S. Vijayanarasimhan and K. Grauman, "What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 2262–2269.

[228] B. C. Wallace, K. Small, C. E. Brodley, J. Lau, and T. A. Trikalinos, "Modeling annotation time to reduce workload in comparative effectiveness reviews," in *Proceedings of the 1st ACM International Health Informatics Symposium*, 2010, pp. 28–35.

[229] R. A. Haertel, K. D. Seppi, E. K. Ringger, and J. L. Carroll, "Return on investment for active learning," in *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, vol. 72, 2008.

[230] T. Fredriksson, D. I. Mattos, J. Bosch, and H. H. Olsson, "Assessing the suitability of semi-supervised learning datasets using item response theory," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, 2021, pp. 326–333.

[231] A. Subramanya and P. P. Talukdar, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, 2014.

[232] T. Fredriksson, "Opportunities, challenges and solutions for automatic labeling of data using machine learning," Ph.D. dissertation, Chalmers Tekniska Höskola (Sweden), 2023.

[233] T. Fredriksson, J. Bosch, and H. H. Olsson, "Classification of complex-valued radar data using semi-supervised learning: A case study," in *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, 2023, pp. 102–107.

[234] L. F. Cardoso, V. C. Santos, R. S. K. Francês, R. B. Prudêncio, and R. C. Alves, "Decoding machine learning benchmarks," in *Brazilian Conference on Intelligent Systems*, Springer, 2020, pp. 412–425.

[235] R. McElreath, *Statistical rethinking: A Bayesian course with examples in R and Stan*. CRC press, 2020.

[236] X. Mai and R. Couillet, "Random matrix-inspired improved semi-supervised learning on graphs," in *International Conference on Machine Learning*, 2018.

[237] H. B. Braiek and F. Khomh, "Machine learning robustness: A primer," *arXiv preprint arXiv:2404.00897*, 2024.

[238] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov, "Good semi-supervised learning that requires a bad gan," *Advances in neural information processing systems*, vol. 30, 2017.

[239] A. Laugros, A. Caplier, and M. Ospici, "Are adversarial robustness and common perturbation robustness independant attributes?" In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[240] W. Dai and D. Berleant, "Benchmarking robustness of deep learning classifiers using two-factor perturbation," in *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, 2021, pp. 5085–5094.

[241] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[242] S. Agarwal, S. Godbole, D. Punjani, and S. Roy, "How much noise is too much: A study in automatic text classification," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, IEEE, 2007, pp. 3–12.

[243] W. J. Van der Linden and R. Hambleton, "Handbook of item response theory," *Taylor & Francis Group. Citado na pág*, vol. 1, no. 7, p. 8, 1997.

[244] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[245] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," 2001.

[246] W. J. van der Linden, *Handbook of Item Response Theory: Volume 1: Models*. CRC Press, 2016.

[247] K. Hori, H. Fukuhara, and T. Yamada, "Item response theory and its applications in educational measurement part i: Item response theory and its implementation in r," *Wiley Interdisciplinary Reviews: Computational Statistics*, e1531, 2020.

[248] F. Martinez-Plumed, R. B. Prudencio, A. Martinez-Uso, and J. Hernandez-Orallo, "Item response theory in ai: Analysing machine learning classifiers at the instance level," *Artificial intelligence*, vol. 271, pp. 18–42, 2019.

[249] A. L. Birnbaum, "Some latent trait models and their use in inferring an examinee's ability," *Statistical theories of mental test scores*, 1968.

[250] W. J. van der Linden, "Unidimensional logistic response models," in *Handbook of Item Response Theory, Volume One: Models*, Chapman and Hall/CRC, 2016, pp. 11–30.

[251] G. J. Mellenbergh, "Models for continuous responses," in *Handbook of Item Response Theory, Volume One: Models*, Chapman and Hall/CRC, 2016, pp. 153–163.

[252] D. I. Mattos, J. Bosch, and H. H. Olsson, "Statistical models for the analysis of optimization algorithms with benchmark functions," *arXiv preprint arXiv:2010.03783*, 2020.

[253] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[254] J. K. Kruschke and T. M. Liddell, "Bayesian data analysis for newcomers," *Psychonomic bulletin & review*, vol. 25, no. 1, pp. 155–177, 2018.

[255] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.

[256] B. Carpenter, A. Gelman, M. D. Hoffman, *et al.*, "Stan: A probabilistic programming language," *Journal of statistical software*, vol. 76, no. 1, 2017.

[257] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[258] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[259] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[260] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, Springer, 2016, pp. 630–645.

[261] F. Alimoglu, E. Alpaydin, and Y. Denizhan, "Combining multiple classifiers for pen-based handwritten digit recognition," 1996.

[262] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition," in *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*, Citeseer, 1996.

[263] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition using state-of-the-art techniques," in *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, IEEE, 2002, pp. 320–325.

[264] K. Huang, I. King, and M. R. Lyu, "Constructing a large node chow-liu tree based on frequent itemsets," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, IEEE, vol. 1, 2002, pp. 498–502.

[265] T. Liu, J. Bao, J. Wang, and Y. Zhang, "A hybrid cnn–lstm algorithm for online defect recognition of co2 welding," *Sensors*, vol. 18, no. 12, p. 4369, 2018.

[266] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, "Ms-capsnet: A novel multi-scale capsule network," *IEEE Signal Processing Letters*, vol. 25, no. 12, pp. 1850–1854, 2018.

[267] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, Springer, 2017, pp. 127–138.

[268] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *Security and Privacy*, vol. 1, no. 1, e9, 2018.

[269] C. Wang, Y. Song, A. El-Kishky, D. Roth, M. Zhang, and J. Han, "Incorporating world knowledge to document clustering via heterogeneous information networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1215–1224.

[270] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," in *2018 IEEE Data Science Workshop (DSW)*, IEEE, 2018, pp. 1–5.

[271] I. Gallo, S. Nawaz, and A. Calefati, "Semantic text encoding for text classification using convolutional neural networks," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, vol. 5, 2017, pp. 16–21.

[272] P. Bafna, D. Pramod, and A. Vaidya, "Precision based recommender system using ontology," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, IEEE, 2017, pp. 3153–3160.

[273] E. Frank, C. Chui, and I. H. Witten, "Text categorization using compression models," 2000.

[274] K. H. Lee, J. Kay, and B. H. Kang, "Active learning: Applying rinscut thresholding strategy to uncertainty sampling," in *Australasian Joint Conference on Artificial Intelligence*, Springer, 2003, pp. 922–932.

[275] Z. Xu, X. Xu, K. Yu, and V. Tresp, "A hybrid relevance-feedback approach to text retrieval," in *European Conference on Information Retrieval*, Springer, 2003, pp. 281–293.

[276] K. Toutanova, F. Chen, K. Popat, and T. Hofmann, "Text classification in a hierarchical mixture model for small training sets," in *Proceedings of the tenth international conference on Information and knowledge management*, 2001, pp. 105–113.

[277] Ö. Yılmaz, L. E. Achenie, and R. Srivastava, "Systematic tuning of parameters in support vector clustering," *Mathematical biosciences*, vol. 205, no. 2, pp. 252–270, 2007.

[278] L. Singh, S. Singh, and P. K. Dubey, "Applications of clustering algorithms and self organizing maps as data mining and business intelligence tools on real world data sets," in *2010 International Conference on Methods and Models in Computer Science (ICM2CS-2010)*, IEEE, 2010, pp. 27–33.

[279] H.-L. Li and Y.-H. Huang, "A diamond method of inducing classification rules for biological data," *Computers in biology and medicine*, vol. 41, no. 8, pp. 587–599, 2011.

[280] S. Dilmaç and M. Korürek, "Using abc algorithm for classification and analysis on effects of control parameters," in *2014 18th National Biomedical Engineering Meeting*, IEEE, 2014, pp. 1–4.

[281] J.-F. Liu, D.-R. Yu, Q.-H. Hu, and X.-D. Li, "Granular entropy based hybrid knowledge reduction using uniform rough approximations," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, IEEE, vol. 3, 2004, pp. 1878–1883.

[282] X. Xu and E. Frank, "Logistic regression and boosting for labeled bags of instances," in *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2004, pp. 272–281.

[283] D. M. Tax and R. P. Duin, "Learning curves for the analysis of multiple instance classifiers," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, Springer, 2008, pp. 724–733.

[284] X. Yuan, X.-S. Hua, M. Wang, G.-J. Qi, and X.-Q. Wu, "A novel multiple instance learning approach for image retrieval based on adaboost feature selection," in *2007 IEEE International Conference on Multimedia and Expo*, IEEE, 2007, pp. 1491–1494.

[285] J. Li, X. Li, and X. Yao, "Cost-sensitive classification with genetic programming," in *2005 IEEE congress on evolutionary computation*, IEEE, vol. 3, 2005, pp. 2114–2121.

[286] D. Pedreschi, S. Ruggieri, and F. Turini, "Measuring discrimination in socially-sensitive decision records," in *Proceedings of the 2009 SIAM international conference on data mining*, SIAM, 2009, pp. 581–592.

[287] L.-l. Zhang, X.-f. Hui, and L. Wang, "Application of adaptive support vector machines method in credit scoring," in *2009 International Conference on Management Science and Engineering*, IEEE, 2009, pp. 1410–1415.

[288] B. M. Shahshahani and D. A. Landgrebe, "The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon," *IEEE Transactions on Geoscience and remote sensing*, vol. 32, no. 5, pp. 1087–1095, 1994.

[289] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, pp. 103–134, 2000.

[290] S. Baluja, "Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data," *Advances in Neural Information Processing Systems*, vol. 11, 1998.

[291] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, pp. 131–163, 1997.

[292] R. Bruce, "Semi-supervised learning using prior probabilities and em," in *International Joint Conference on Artificial Intelligence, Workshop on Text Learning: Beyond Supervision*, 2001.

[293] D. Mahajan, R. Girshick, V. Ramanathan, *et al.*, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 181–196.

[294] J. Hestness, S. Narang, N. Ardalani, *et al.*, "Deep learning scaling is predictable," *Empirically. arXiv*, vol. 1712, p. 2, 2017.

[295] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[296] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[297] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 687–10 698.

[298] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[299] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[300] CloudFactory.com, *The ultimate guide to data labeling for machine learning*, https://www.cloudfactory.com/data-labeling-guide, 2019.

[301] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[302] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[303] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[304] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[305] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training. arxiv 2019," *arXiv preprint arXiv:1904.12848*, 1904.

[306] T. Fredriksson, D. I. Mattos, J. Bosch, and H. H. Olsson, "An empirical evaluation of algorithms for data labeling," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, 2021, pp. 201–209.

[307] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[308] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[309] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[310] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.

[311] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," in *IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium*, IEEE, 2018, pp. 204–207.

[312] J. Yang, R. Shi, and B. Ni, "Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis," in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, IEEE, 2021, pp. 191–195.

[313] J. Yang, R. Shi, D. Wei, *et al.*, "Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[314] J.-C. Su and S. Maji, "The semi-supervised inaturalist-aves challenge at fgvc7 workshop," *arXiv preprint arXiv:2103.06937*, 2021.

[315] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[316] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[317] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 165–172.

[318] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar, "Importance of semantic representation: Dataless classification.," in *Aaai*, vol. 2, 2008, pp. 830–835.

[319] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.

[320] N. Asghar, "Yelp dataset challenge: Review rating prediction," *arXiv preprint arXiv:1605.05362*, 2016.

[321] E. Fonseca, M. Plakal, D. P. Ellis, F. Font, X. Favory, and X. Serra, "Learning sound event classifiers from web audio with noisy labels," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 21–25.

[322] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[323] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[324] D. Elworthy, "Does baum-welch re-estimation help taggers?" *arXiv preprint cmp-lg/9410012*, 1994.

[325] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," *Advances in neural information processing systems*, vol. 29, 2016.

[326] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[327] D. I. Mattos, L. Ruud, J. Bosch, and H. H. Olsson, "On the assessment of benchmark suites for algorithm comparison," *arXiv preprint arXiv:2104.07381*, 2021.

[328] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on psychological science*, vol. 6, no. 1, pp. 3–5, 2011.

[329] H. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Transactions on Information Theory*, vol. 11, no. 3, pp. 363–371, 1965.

[330] S. Fralick, "Learning to recognize patterns without a teacher," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 57–64, 1967.

[331] A. Agrawala, "Learning with a probabilistic teacher," *IEEE Transactions on Information Theory*, vol. 16, no. 4, pp. 373–379, 1970.

[332] D. W. Hosmer Jr, "A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample," *Biometrics*, pp. 761–770, 1973.

[333] G. J. McLachlan, "Estimating the linear discriminant function from initial samples containing a small number of unclassified observations," *Journal of the American statistical association*, vol. 72, no. 358, pp. 403–406, 1977.

[334] G. J. McLachlan and S. Ganesalingam, "Updating a discriminant function on the basis of unclassified data," *Communications in Statistics-Simulation and Computation*, vol. 11, no. 6, pp. 753–767, 1982.

[335] K. N. t. A. M. St, "Learning to classify text from labeled and unlabeled documents," 1998.

[336] M. Collins and Y. Singer, "Unsupervised models for named entity classification," in *1999 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, 1999.

[337] T. Joachims *et al.*, "Transductive inference for text classification using support vector machines," in *Icml*, vol. 99, 1999, pp. 200–209.

[338] B. R. Ernest, "Dynamic programming," *Mineola, NY, USA: Courier Dover Publ*, 2003.

[339] V. Castelli and T. M. Cover, "The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter," *IEEE Transactions on information theory*, vol. 42, no. 6, pp. 2102–2117, 1996.

[340] H. Schmutz, O. Humbert, and P.-A. Mattei, "Don't fear the unlabelled: Safe semi-supervised learning via simple debiasing," *arXiv preprint arXiv:2203.07512*, 2022.

[341] R. F. DeVellis, "Classical test theory," *Medical care*, vol. 44, no. 11, S50–S59, 2006.

[342] H. Baghi and S. F. Ferrara, "A comparison of irt, delta plot, and mantel-haenszel techniques for detecting differential item functioning across subpopulations in the maryland test of citizenship skills.," 1989.

[343] J.-P. Fox, *Bayesian item response modeling: Theory and applications*. Springer, 2010.

[344] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[345] B. Bischl, G. Casalicchio, M. Feurer, *et al.*, "Openml benchmarking suites," *arXiv preprint arXiv:1708.03731*, 2017.

[346] M. Awad and R. Khanna, *Efficient learning machines: theories, concepts, and applications for engineers and system designers*. Springer nature, 2015.

[347] *Machine learning: Why popular?* `https://data-flair.training/blogs/why-machine-learning-is-popular/`, Accessed: 2022-10-16.

[348] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[349] *Why deep learning over traditional machine learning*, `https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063`, Accessed: 2022-10-16.

[350] *Data labeling pros and cons*, `https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063`, Accessed: 2022-10-16.

[351] H. Bi, J. Sun, and Z. Xu, "A graph-based semisupervised deep learning model for polsar image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 2116–2132, 2018.

[352] M. I. Skolnik, "Introduction to radar," *Radar handbook*, vol. 2, p. 21, 1962.

[353] L. Zhu, X. Ma, P. Wu, and J. Xu, "Multiple classifiers based semi-supervised polarimetric sar image classification method," *Sensors*, vol. 21, no. 9, p. 3006, 2021.

[354] S. Ren and F. Zhou, "Semi-supervised classification for polsar data with multi-scale evolving weighted graph convolutional network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 2911–2927, 2021.

[355] N. Guberman, "On complex valued convolutional neural networks," *arXiv preprint arXiv:1602.09046*, 2016.

[356] W. Xie, G. Ma, F. Zhao, H. Liu, and L. Zhang, "Polsar image classification via a novel semi-supervised recurrent complex-valued convolution neural network," *Neurocomputing*, vol. 388, pp. 255–268, 2020.

[357] Q. Sun, X. Li, L. Li, X. Liu, F. Liu, and L. Jiao, "Semi-supervised complex-valued gan for polarimetric sar image classification," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, 2019, pp. 3245–3248.

[358] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, p. 100 057, 2021.

[359] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson, "Failing to learn: Autonomously identifying perception failures for self-driving cars," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3860–3867, 2018.

[360] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," *Advances in neural information processing systems*, vol. 27, 2014.

[361] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[362] A. Kurakin, C. Raffel, D. Berthelot, *et al.*, "Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring," 2020.

[363] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.