Assessing Machine Learning Robustness to Sample Selection Bias

Evaluating the effectiveness of semi-supervised learning techniques

Viraj Biharie[1]

Supervisor: Responsible Professor[1]**: Joana de Pinho Gonçalves, Supervisor**[1]**: Yasin Tepeli**

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Viraj Biharie Final project course: CSE3000 Research Project
Thesis committee: Responsible Professor: Joana de Pinho Gonçalves, Supervisor: Yasin Tepeli, Examiner: Julian Urbano Merino

Abstract

This paper tackles the problem of sample selection bias in machine learning, where the assumption of train and test sets being drawn from the same distribution is often violated. Existing solutions in domain adaptation, such as semi-supervised learning techniques, aim to correct this bias, but their ability to generalize to unseen test sets remains unexplored. To address this issue, specific semi-supervised methods (self-training and co-training) are trained on biased training sets and tested with an unbiased test set drawn from the same distribution. The results of this paper demonstrate that the semi-supervised methods consistently outperformed or matched the baseline models, with self-training exhibiting greater improvement. Through this study, a promising approach is presented to mitigate sample selection bias in machine learning.

1 Introduction

Sample selection bias is a widely acknowledged problem where the dataset used to train a model may not accurately represent the population it is intended to generalize to. This can lead to unfairness and unintended biases that disadvantage less well-represented groups. [18] In the field of healthcare, biases are of significant concern. When the training data for medical algorithms lack representation from diverse populations, certain demographic groups can experience underdiagnosis or suboptimal care. [14] Another instance of selection bias emerges in finance in the decisions made by managers. This bias arises from unobservable differences within the samples, influencing managers' choices and subsequent outcomes. Consequently, it can result in distorted conclusions and flawed evaluations of the actual impact of managers' financial decisions. [20]

One common type of sample selection bias is where the selection of samples depends on both the feature space and the class labels, and the number of labeled samples is scarce. The issue lies in the limited understanding of the generalization capabilities of models trained on biased data, even though biased training data are often observed in practical applications. [1]

This study will focus on evaluating the effectiveness of specific semi-supervised learning [5] techniques that have already received considerable attention in previous research. The focus will be on two techniques: self-training and co-training.

Self-training has been selected due to its unique approach of utilizing labeled data along with valuable unlabeled data to assist in the creation of the decision boundary. This method stands out by using pseudo-labels, which are generated for the unlabeled data to approximate their true labels, thereby creating a pseudo-decision boundary that enhances the learning process. [17]

Co-training has been chosen for its distinct characteristics that go beyond self-training. By employing multiple models, co-training introduces a more sophisticated framework for the learning process. This method stands out by utilizing the diversity of the models and their respective interpretations of the unlabeled data, enabling a more robust decision boundary. [2]

While other techniques exist, studying and evaluating these representative approaches will provide valuable insights into the broader field of semi-supervised learning.

Previous research in domain adaptation has focused on learning models that correct the domain shift between the train set and a specific test set [22] [9], but the generalizability of these methods to any unseen test set has not yet been investigated. Some methods have been proposed to address sample selection bias, such as importance weighting, subspace mapping, deep domain adaptation, minimax estimation, and semi-supervised learning. [10]

Within the context of semi-supervised learning techniques, prior studies have examined the effectiveness of various machine learning models for generating the pseudo-decision boundary required for self-training. [19] [7] Drawing insights from these investigations, this research employs support vector classification (SVC) [11] and logistic regression [4] as the supervised machine learning models. These methods have been established as effective in prior research, justifying their selection for this study.

In regards to co-training. Previous research has focused on generalizing domain adaptation methods from a source domain to a new target domain. [6] However, these studies assume that the source domain is representative of the entire domain, neglecting the presence of sample selection bias.

This research aims to evaluate the effectiveness of semi-supervised learning techniques in reducing sample selection bias when labeled data is limited. The study will investigate the following research questions:

- How effectively do the synthetic datasets generated in this research manifest the intended bias?
- How does the performance of self-training and co-training vary with different quantities of labeled data?
- How does the performance of self-training and co-training adapt to varying levels of bias in the data?

To accomplish this, synthetic datasets with simulated biases are generated, and an evaluation framework is implemented to compare the performance of the different semi-supervised learning techniques. The goal is to assess the generalizability performances of supervised and semi-supervised methods trained on these biased toy datasets. By providing insights into the suitability of different methods for addressing sample selection bias, this research aims to contribute to the development of machine learning models that are more fair and reliable.

2 Methodology

2.1 Formal Problem Description

The formal problem addressed in this research is to assess the effectiveness of semi-supervised learning techniques, specifically self-training and co-training, in mitigating sample selection bias. The study aims to evaluate the performance of these techniques on synthetic datasets that have

intentionally biased samples. The objective is to determine whether these techniques can effectively learn from a limited labeled dataset and utilize the available unlabeled instances to improve classification performance while reducing the impact of the biased nature of the data. By quantitatively analyzing the results using appropriate evaluation metrics, the research seeks to provide information on the potential of these techniques to address sample selection bias in real-world scenarios.

2.2 Incorporating Biased and Unbiased Datasets for Domain Adaptation

In this research paper, an innovative approach is proposed for domain adaptation in machine learning. Traditionally, domain adaptation methods learn from a labeled training set and an unlabeled test set. [10] In this study, an approach is used that strays from the conventional method.

This approach involves dividing the synthesized dataset into two parts: a training set and a test set. This division is carried out in a manner that ensures both sets have a distribution similar to that of the original dataset.

After the split, the training set is further divided into two subsets. One subset intentionally introduces bias and includes labels, while the other larger subset remains unbiased and doesn't utilize the labels (representing the original distribution). Labels are assigned to all data points initially, but the unlabeled subset does not utilize these labels. For more details on the biasing process, please refer to Section 2.6.

Once the training phase is completed, the model performance is assessed using a separate test set, following the conventional approach. The test set uses labeled data, enabling a comprehensive evaluation of the model's performance.

This methodology aims to investigate the effectiveness of incorporating biased and unbiased datasets during domain adaptation. By utilizing the labeled biased training set and the larger, unbiased unlabeled set, this approach offers the potential to enhance the model's ability to adapt to different domains and improve generalization performance.

Evaluation of the model using the separate test set enables a rigorous comparison of its performance with established methods, providing information on the effectiveness of the proposed approach for domain adaptation.

In summary, this research paper introduces a new methodology for domain adaptation in machine learning, involving the utilization of a biased training set alongside an unbiased unlabeled set. The proposed method is evaluated using a separate test set, allowing for a comprehensive analysis of its performance in comparison to traditional domain adaptation techniques.

2.3 Self-Training with Dynamic Thresholding and Ensemble Models

The self-training algorithm used in this research paper implements a modified version of the traditional self-training algorithm, with the aim of improving its performance in semi-supervised learning scenarios. The motivation behind modifying the traditional algorithm stems from the subpar results observed when applying it to certain models. By making adjustments to the algorithm, this research aims to address the limitations and enhance its effectiveness in handling semi-supervised learning tasks. Self-training, as a technique, involves initially training a model on a labeled dataset and subsequently updating iteratively using confident predictions made on unlabeled data.

In this modified implementation, the self-training algorithm starts by training the model on the labeled data. Then, probabilities are predicted for the unlabeled data using the trained model. By setting a threshold, confident predictions are identified based on the maximum predicted probability.

One significant difference in this approach is the use of dynamic thresholding. The concept of dynamic thresholding draws inspiration from the 'Dash' framework, which combines semi-supervised learning with the utilization of dynamic thresholds. [21] The threshold value determines the minimum confidence level that a sample must meet to be included in the confident predictions. Instead of a fixed threshold, this method adjusts the threshold value gradually with each iteration. This flexibility allows for a more adaptable inclusion of confident predictions during the self-training process.

Moreover, unlike traditional self-learning that typically uses only one model, this approach employs an ensemble of models to train a single model. The concept of the ensemble of models was inspired by the methodology of multi-view training, which uses multiple models to collectively make a unified decision. [23] The reason for using this ensemble is to improve the model's ability to understand different patterns and to reduce the risk of fitting too closely to the training data. By combining the strengths of different models, each with their own biases and learning capabilities, the ensemble approach aims to create a more versatile and robust trained model.

By incorporating dynamic thresholding and ensemble models, this modified self-training implementation offers potential improvements over the typical self-training approach, as implemented in libraries like scikit-learn. These modifications enable a more refined and adaptive utilization of unlabeled data, leading to potentially enhanced classification performance in semi-supervised learning tasks. Please refer to Appendix B for the pseudo-code implementation corresponding to this modefied self-training process described in this section.

2.4 Co-Training: Incorporating Disagreement-based Augmentation and Independent Modeling

The initial approach employed in this research paper was a modified implementation of traditional co-training, a semi-supervised learning technique where two models are trained simultaneously using labeled and unlabeled data. However, this custom implementation yielded somewhat disappointing results. To address this, the CTClassifier method from the mvlearn library [3] was adopted as an alternative. In particular, the mvlearn method is specifically designed for two-class scenarios, whereas the previous implementation supported multiple classes.

The CTClassifier method from the mvlearn library is a co-training classifier used for semi-supervised learning. It takes advantage of either two views of input data or a single matrix with distinct estimators. The co-training approach involves training two classifiers on different views of the data and iteratively updating them by sharing confident predictions on unlabeled instances. This method aims to improve the model generalization by leveraging diverse perspectives and combining the strengths of multiple classifiers.

In comparison, the initial co-training implementation was developed from scratch. This allowed for customization and control over the co-training process to align with the specific requirements of the study. During each iteration of the co-training process, both models were trained on the labeled data. Subsequently, the models predicted labels for the unlabeled data, and instances with significant disagreements between the models, based on a specified threshold, were identified. These disagreement instances and their predictions were incorporated into the labeled data, augmenting its size and enabling the models to learn from a larger set of instances in subsequent iterations. This process of training, predicting, identifying disagreement instances, and incorporating them into the labeled data was repeated for a specified number of iterations.

The motivation behind incorporating the disagreement instances into the labeled data was to embrace diverse perspectives offered by the two models. By capturing different aspects of the data and reducing the risk of overfitting, this co-training approach aimed to enhance performance. This strategy enabled the models to learn from multiple sources of information, including the unlabeled data, resulting in improved performance compared to relying solely on labeled data.

In summary, while the initial bespoke co-training implementation offered customization and control, its performance fell short of expectations. However, it is important to note that both the initial implementation and the adopted CTClassifier method from the mvlearn library are evaluated and described in section 3. It is worth emphasizing that the primary focus of this research is not to identify the best-performing methods, but rather to comprehensively evaluate and compare different approaches. By assessing the performance of both methods, this study aims to provide valuable insights into their effectiveness and suitability for the given problem.

2.5 Dataset generation

The datasets are generated by randomly creating a specified number of clusters, where each cluster represents similar data points and shares the same class label. However, a single class can have multiple clusters representing it. The number of clusters cannot be lower than the number of classes due to technical limitations. The specific number of clusters does not hold any intrinsic meaning; rather, it determines how the data points are distributed among the clusters. The primary objective is to distribute the data across these clusters to enable the application of machine learning models such as logistic regression and support vector classification, facilitating the creation of decision boundaries between the classes. This clustering strategy also makes it easier to visualize the data, allowing for a clear depiction of how the decision boundary evolves when biases are introduced.

The data used for the semi-supervised learning techniques is generated with customizable parameters. These parameters include:

- n_samples: The total number of samples
- n_features: The number of features or dimensions in each sample
- n_classes: The number of distinct classes in the data
- n_clusters: The number of clusters present in the data.
- density: A parameter that controls the density between clusters. Increasing this value results in more density and higher density around the cluster centers.
- min_dist: The minimal distance between cluster centers. The default value is set at 0.4, ensuring that the clusters are separated by a certain distance.

By adjusting these parameters, the generated data can be customized to suit different experimental settings and requirements for the semi-supervised learning techniques. The parameter ranges for the experimental setup are defined as follows: n_samples, n_features, n_classes, and n_clusters are all integer values. The parameter density is a floating-point value ranging from 0 to 1, with a recommended range of 0 to 0.1 to ensure consistent clusters. Additionally, the parameter min_dist is a float between 0 and 1.

Using the given parameters, the data generation process follows these steps:

1. Calculate the number of samples per cluster by distributing the total number of samples evenly among all clusters
2. Randomly generate the specified number of cluster centers
3. Check if every pair of cluster centers is at least the specified minimal distance away from each other. If not, regenerate a new set of random cluster centers and repeat the distance check
4. Calculate the number of clusters that should be assigned to each class, aiming for an even distribution by dividing the given number of clusters over the given number of classes using integer division
5. For each cluster center:
    - Assign a class label to the cluster. This assignment ensures an even distribution of clusters across the classes. If the number of clusters per class is not equal, the remaining clusters are randomly assigned to the classes with less clusters
    - Create a covariance matrix where each element on the diagonal is equal to the specified density parameter
    - Use the cluster center and the covariance matrix to generate the specified number of random samples around this cluster center. The density of samples around the center is controlled by the density parameter. Lower values of this parameter lead to

increased density around the cluster center, consequently reducing the level of density observed between different clusters.

6. Return the generated data, labels, and cluster labels

Please refer to Appendix A for the pseudo-code implementation corresponding to the dataset generation process described in this section.

To initiate the evaluation process, the semi-supervised learning techniques were initially assessed on a single instance of a generated dataset. However, relying solely on results from a single dataset instance is insufficient to draw meaningful conclusions about the techniques' generalization capabilities.

To address this limitation and enhance the robustness of the evaluation, multiple datasets were created using different random seeds. This approach ensures reproducibility while also introducing diverse data distributions for testing purposes. Evaluating the techniques' performance across multiple datasets provides a more comprehensive understanding of its efficiency and generalization capabilities.

For all dataset instances, the same parameters were used to generate the datasets, unless explicitly stated otherwise. Please refer to Appendix C for the exact values throughout the experiments.

These parameters were chosen because, through manual inspection, it has been observed that the training data consistently contains a bias due to the changes in the decision boundary. Additionally, while it is important to note that having large datasets can be beneficial for training robust machine learning models, the sheer size of these datasets also poses a bottleneck. The amount of time it will take to process and execute a single experiment increases significantly when dealing with large datasets, impacting the overall efficiency of the algorithm's performance evaluation over several iterations. However, in this specific case, the datasets used for evaluation are not excessively large and do not currently pose a bottleneck in evaluating the algorithm's performance.

2.6 Bias Induction

The generated synthetic datasets are split into a labeled training, test, and unlabeled set. The purpose is to bias the training set such that this set is no longer representative of the original distribution. This is done by using the clusters that are present within the training set.

The determination of the number of clusters per class relies on two factors: the specified total number of clusters and the specified total number of classes. The calculation of the number of clusters is determined by dividing the total number of clusters by the total number of classes using integer division. This allocation ensures an even distribution of clusters among the classes. Any remaining clusters that cannot be evenly distributed are arbitrarily assigned to the classes with the least number of clusters.

Each cluster per class is initially assigned a weight of 1. This ensures an equal representation of clusters within each class.

To introduce bias, a random cluster is selected and assigned a higher weight, currently set to 20. This weight can be adjusted according to the desired level of bias in the dataset.

Subsequently, based on the assigned weights, samples are selected from the training set using weighted probabilities. The indices corresponding to these samples determine which ones will be included in the biased dataset. Clusters with higher weights have a greater probability of being selected.

By modifying the weights assigned to the clusters, the biasing process affects the composition of the training set, potentially favoring specific clusters and introducing bias into the learning process.

This biasing procedure aims to maintain the class balance, whether the classes were initially balanced or unbalanced.

2.7 Performance Evaluation

Next, the self-training and co-training techniques are implemented and applied to the generated biased datasets. These techniques, both utilizing a linear and non-linear classifier, are chosen due to their potential in utilizing unlabeled instances to improve classification performance while addressing sample selection bias. The linear classifier used is Linear Regression, while the non-linear classifier employed is Support Vector Classification (SVC). Linear regression is a statistical technique that aims to model the relationship between a dependent variable and one or more independent variables by fitting a straight line that minimizes the sum of the squared differences between the observed and predicted values. [8] SVC is a machine learning algorithm that finds the optimal hyperplane in a high-dimensional feature space to separate different classes of data points by maximizing the margin between them while minimizing the classification errors. [16] The paper's implementation of the SVC algorithm utilizes the 'sklearn.svm.SVC' implementation from scikit-learn (sklearn). [15] By default, it uses the non-linear "Radial basis function" (RBF) kernel, which is often referred to as the Gaussian kernel.

The models are trained in an iterative manner. Initially, a small labeled subset of the biased dataset is used to train the models. The models then utilize the available unlabeled instances to refine their performance.

To assess the effectiveness of the semi-supervised learning techniques, the following performance metrics are employed: accuracy, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics were selected based on their frequent utilization as evaluation parameters in existing literature. The F1-score is a metric that combines precision and recall to measure the accuracy of a binary classification model by calculating the harmonic mean of the two. [12] AUC-ROC is a performance metric that quantifies the ability of a binary classification model to distinguish between classes by calculating the area under the curve generated by plotting the true positive rate against the false positive rate at different classification thresholds. [13] By comparing the results obtained from the self-training and co-training approaches, valuable insights are gained regarding the extent to which these techniques can address the sample selection bias, improve classification accuracy in the presence of biased data, and generalize effectively to unseen data. The focus lies on assessing the models' ability to extend their learning from the

biased dataset to new, unseen instances, thereby determining their robustness and applicability in real-world scenarios.

In addition to the performance metrics (accuracy, F1-score, and AUC-ROC score), a baseline comparison was conducted to further evaluate the algorithm's effectiveness. The baseline evaluation was performed on the same dataset before the application of the self-training algorithm. This comparison allows for a direct measurement of the algorithm's improvement and effectiveness.

2.8 Experimental Setup

In order to thoroughly evaluate the performance of the self-training and co-training techniques, comprehensive testing has been conducted under various conditions. The techniques were subjected to the following scenarios: the manipulation of bias weight, the variation of the size of the unlabeled training set, and the adjustment of the number of features. In these scenarios the baseline scores are being compared to the scores obtained from the self-training and co-training methods, where the scores are measured using the accuracy, F1-score and AUC-ROC score.

To comprehensively assess the performance across all dataset instances, the scores of each individual instance were recorded. Subsequently, the average score and its variance were calculated for both the baseline and the self-training scenarios. The use of error bar plots provided a visual representation of the results, offering valuable insights into the distribution and variability of the scores.

In the error bar plots, the line represents the average score across all dataset instances for the metrics of accuracy, F1-score, and AUC-ROC. This line serves as an indicator of the overall performance trend. Additionally, error bars are included in the plot to depict the variance associated with each score. The length of the error bars reflects the extent of variability observed among the dataset instances.

By utilizing error bar plots, a more comprehensive understanding of the algorithm's performance consistency and reliability can be obtained. The average scores provide an overview of the algorithm's overall effectiveness, while the error bars convey the degree of variation within the dataset instances. This information is crucial for interpreting the robustness and generalizability of the self-training algorithm in the context of the evaluation.

To address readability issues caused by displaying the variances of all the different scores, the results are presented in two different ways. In the main plots, basic scores were plotted to provide an overview of the performance, while in the appendix, additional plots were included with error bars representing the variances. The error bar plots in the appendix offer a more detailed depiction of the distribution and variability of scores for accuracy, F1-score, and AUC-ROC across all dataset instances. This approach enables a comprehensive assessment of the algorithm's performance consistency and reliability, enhancing the understanding of its robustness and generalizability.

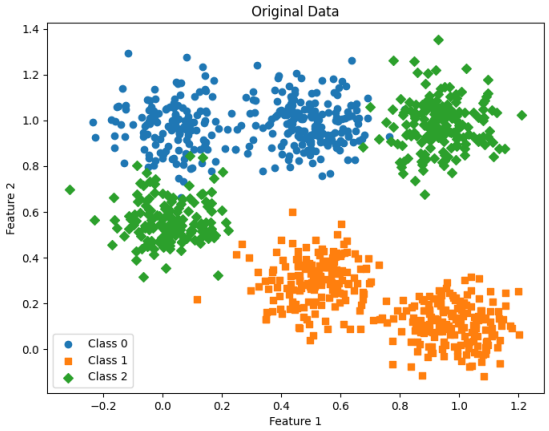


Figure 1: *The entire dataset created using the method explained in Section 2.5. Different symbols and colors are used to represent each class in the dataset. The dataset will be split into three parts: the training set, unlabeled set, and test set. Although all data points have labels initially, the labels in the unlabeled set are not used for further analysis.*

3 Results

3.1 Generated Dataset

In Figure 1, a generated dataset is depicted, which consists of 6 clusters and 3 classes. The dataset comprises a total of 1000 samples.

Observing the figure, it is evident that each cluster contains approximately an equal number of samples. Additionally, the density around the center of each cluster appears to be similar across all clusters (as specified by the overlap parameter).

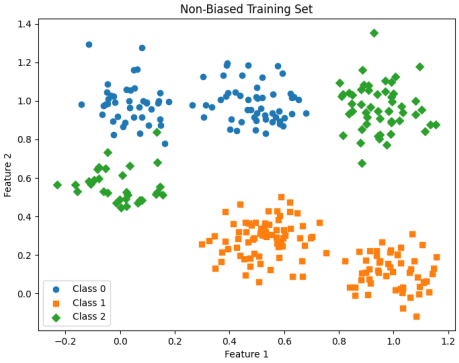
Furthermore, it is worth noting that the cluster centers are positioned at least a specific distance away from each other. In this instance, the default value of 0.4 was utilized as this minimum distance between cluster centers.

3.2 Biased Datasets

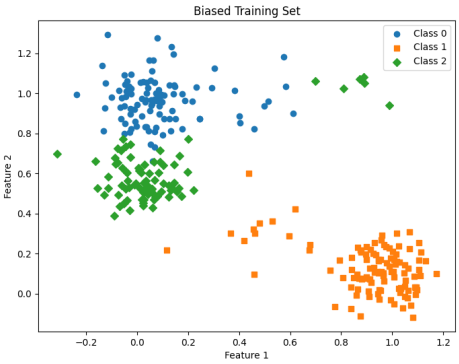
Figure 2 illustrates the training sets of the previously generated dataset. Figure 2a displays the dataset when the training data is sampled without bias, ensuring a relatively equal number of samples are selected from each cluster. In Figure 2b, the dataset is depicted after applying bias during the training data sampling process. Notably, in Figure 2b, it is evident that one cluster from each class has been sampled significantly more times compared to the other clusters within the same class, creating an imbalanced representation.

Finally, to demonstrate the bias induced by the sampling process in the training set, the datasets are presented in Figure 3 after applying the decision boundary and assigning new labels based on it. Logistic regression is utilized to calculate the decision boundary; however, due to the complexity of displaying the boundary itself, only the datasets after applying the decision boundary are shown.

Figure 3a showcases the non-biased training set, where the decision boundary has resulted in labeling the upper right cluster primarily as "Class 2". In contrast, 3b illustrates the

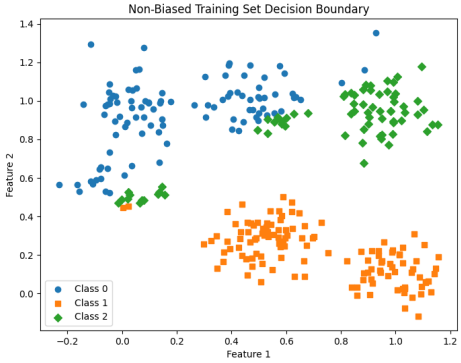


(a) *Non-Biased Training Set*

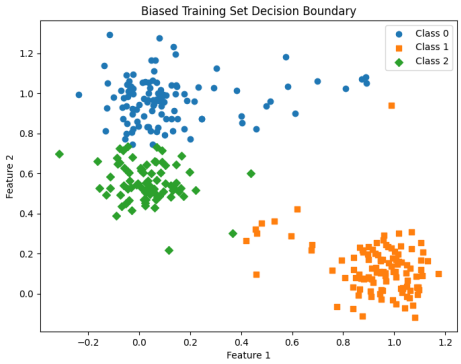


(b) *Biased Training Set*

Figure 2: *This figure illustrates the training set obtained from the dataset shown in Figure 1 after the partitioning into the training set, unlabeled set, and test set. The visualizations depict the same dataset instance both with and without the biasing, following the biasing process detailed in Section 2.6. In this particular representation, a bias weight of 20 is applied.*



(a) *Non-Biased Training Set*



(b) *Biased Training Set*

Figure 3: *Training Sets after applying Decision Boundary. This figure depicts the data after applying the decision boundary obtained through logistic regression. The decision boundary separates the data points into different classes based on their assigned labels. The labeled data points are displayed in the figure, showcasing their classification after the application of the decision boundary.*

biased training set, where the same cluster is predominantly labeled as ”Class 0” (although only a limited portion of the cluster remains after sample selection biasing).

Furthermore, a notable change in the decision boundary is observed for the bottom left cluster. In Figure 3a, this cluster is mostly classified as ”Class 0,” whereas in Figure 3b, it is nearly entirely classified as ”Class 2.”

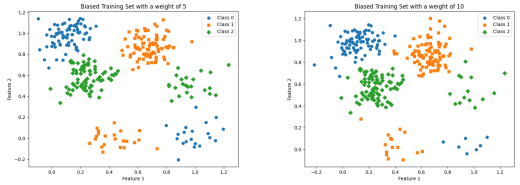
These differences in label predictions between the two sets emphasize that the sampling process has indeed altered the decision boundary. Although the decision boundary itself is not displayed, the shifts in labeling patterns demonstrate the impact of the sampling bias on the classification outcomes.

To demonstrate the impact of manipulating the bias weight, Figure 4 is presented, which showcases the changes in a specific dataset instance as the bias weight is altered. In Figure 4a, the bias weight is set to 5, revealing a slight bias where certain clusters are larger for each class. However, the figure still exhibits the presence of other clusters, indicating a relatively balanced distribution.

As the bias weight increases, as observed in Figures 4b and 4c, these specific clusters grow in size while the remaining clusters diminish. Notably, there is a threshold beyond which further increases in the weight have minimal effect on the dataset’s distribution. This is evident when comparing Figure 4c (bias weight of 50) and Figure 4d (bias weight of 100), where both datasets exhibit nearly identical distributions. This phenomenon was consistently observed across multiple dataset instances, resulting in testing with a maximum bias weight of 50 throughout the experiments.

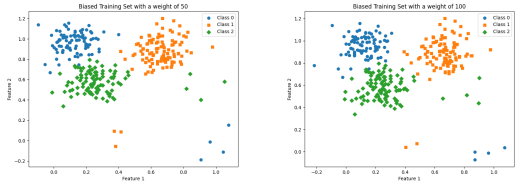
3.3 Manipulating Bias Weight

This section discusses the results of manipulating the bias weight parameter to control the level of biasing in the training data during sample selection. Both self-training and co-training techniques were experimented with under similar circumstances, using similar data instances. The only noticeable difference between them lies in how they used the training data to train the model.



(a) *Biased data with a weight of 5*

(b) *Biased data with a weight of 10*



(c) *Biased data with a weight of 50*

(d) *Biased data with a weight of 100*

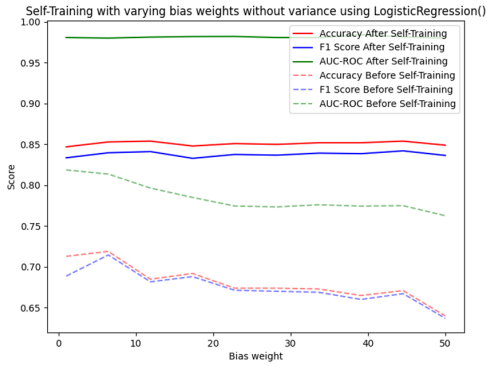
Figure 4: *The figures presented depict the training datasets after the entire dataset has been split into training, unlabeled, and test datasets. These figures showcase the impact of altering the bias weight parameter on the same dataset instance, allowing for a visualization of varying amounts of bias induced. The bias weight parameter influences the degree of bias present in the datasets, and these figures provide insights into how different levels of bias affect the data distribution.*

3.3.1 Self-Training Manipulating Bias Weight

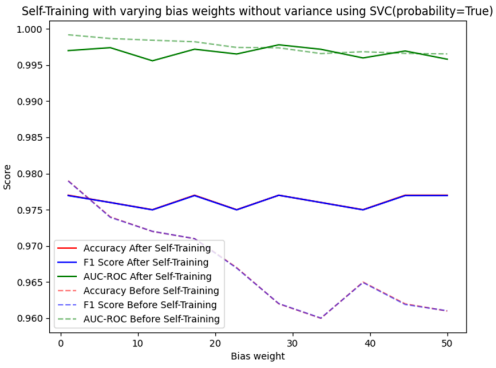
Figure 5 depicts the results of ten iterations of the self-training algorithm. It is important to note that the dataset instances used for both sub-figures were identical. Figure 5a represents the application of logistic regression, while 5b showcases the utilization of SVC. The dashed lines in the figure represent the baseline scores of the model obtained before applying self-training. These scores are based on the indicated supervised learning technique used solely on the training data. Please refer to Figure 11 in Appendix D for the plots that show the variance for these figures using error bars. In Figure 11 the scores are accompanied by error bars, where the length indicates the level of variance around the mean. However, due to the presence of other accuracy scores, the visibility of these error bars may be compromised.

Upon examining these plots, it can be deduced that the bias weight does not appear to pose a significant influence on the performance of the self-training model. Despite the increase in weight, the scores do not display a consistent downward trend. Furthermore, both figures demonstrate that the self-training algorithm surpasses the baseline approach. Additionally, when comparing Figure 5a (logistic regression) with Figure 5b (SVC), it is evident that logistic regression exhibits inferior overall performance relative to SVC. Moreover, Figure 11a displays larger error bars, indicating greater variability among the tested dataset instances for logistic regression. Another thing to notice is that especially in Figure 5b, the supervised models' performances decreases as the bias weight increases.

In Appendix E in Figure 14 plots are displayed showing the



(a) *Self-Training using Logistic Regression with varying weights*



(b) *Self-Training using SVC with varying weights*

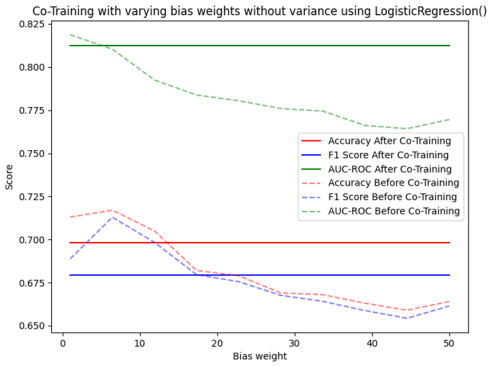
Figure 5: *The figure showcases the average scores of ten dataset instances trained using ten iterations of the modified self-training algorithm. The plots represent the scores without showing their individual variances. In Figure 5b, the accuracy and F1 score values coincide precisely, resulting in a purple-colored line, which is a blend of the red and blue colors representing the accuracy and F1 scores, respectively.*

result of executing this same experiment with the self training algorithm of sklearn, this implements the traditional way of self learning. Comparing Figure 14 with Figure 5 it can be seen that the for SVC both methods have a similar performance. However, for logistic regression it can be seen that in Figure 14a the self-training method has about the same performance as the baseline, whereas in Figure 5a the self-training method consistently outperforms the baseline.

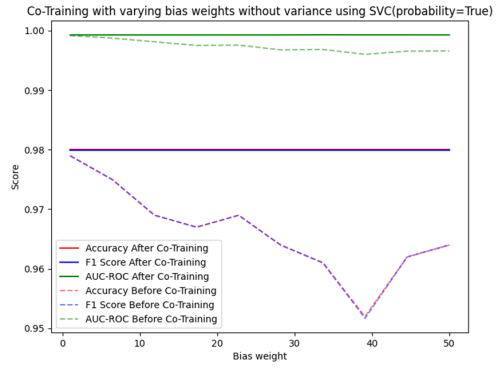
3.3.2 Co-Training Manipulating Bias Weight

From the analysis of Figure 6, several conclusions can be drawn regarding the performance of the co-training algorithm. Firstly, manipulating the bias weights in co-training does not appear to have any discernible effect on the algorithm's performance, as the performance remains consistently unchanged.

Moreover, in both figures, it can be observed that co-training slightly outperforms the baseline, although the performance gap is consistently small (observing the difference

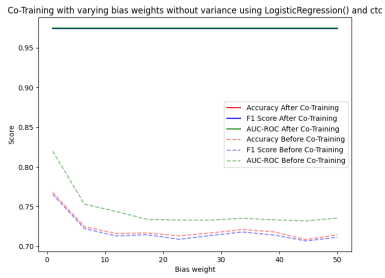


(a) *Co-Training using Logistic Regression with varying weights*

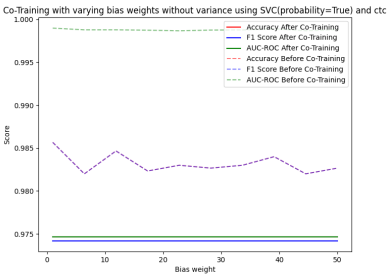


(b) *Co-Training using SVC with varying weights*

Figure 6: *The figure showcases the average scores of thirty dataset instances trained using ten iterations of the implemented co-training algorithm. The plots represent the scores without showing their individual variances. In Figure 6b, the accuracy and F1 score values coincide precisely, resulting in a purple-colored line, which is a blend of the red and blue colors representing the accuracy and F1 scores, respectively.*



(a) *Co-Training using Logistic Regression with varying weights and using the ctc classifier*



(b) *Co-Training using SVC with varying weights and using the ctc classifier*

Figure 7: *The figure displays the identical dataset instances as the previous experiments. However, in this figure, the ctc classifier from the mvlearn library is employed instead of the implemented co-training approach. The plots depict the scores without indicating their individual variances. Please refer to Appendix F Figure 16 for these figures with their corresponding variances. In Figure 7a, the accuracy, F1 score, and AUC-ROC values align perfectly, resulting in the use of a single line to represent these the lines obtained after applying co-training, whereas in Figure 7b the accuracy and F1 score align perfectly, thus only a single line has been displayed for this.*

in the scores gives a range of around 0.025 to 0.050 improvement for logistic regression and for SVC this range is 0.001 to 0.030). This suggests that the co-training algorithm is effective in improving the performance of the model, albeit marginally.

Comparing Figure 6a, which utilizes logistic regression, with Figure 6b, which employs SVC, it becomes evident that SVC consistently outperforms logistic regression in terms of both performance and variance. The performance of SVC is notably higher, while exhibiting lower variability compared to logistic regression. Please refer to Figure 12b in Appendix D to see this variance.

These findings provide insights into the behavior and performance characteristics of the co-training algorithm in the context of bias weight manipulation.

Figure 7 presents the results obtained by training the model using the ctc classifier from the mvlearn library (as described in section 2.4). When comparing Figure 7a with the previous approach shown in Figure 6a, it becomes evident that the ctc approach consistently outperforms the previous approach and the baseline by a significant margin of twenty to thirty percent, when using logistic regression. Contrarily, Figure 7b illustrates that the ctc classifier underperforms compared to the baseline and the previous approach shown in Figure 6b when using SVC, indicating that the ctc classifier does not always outperform the other approach.

3.4 Manipulating Unlabeled Set Size

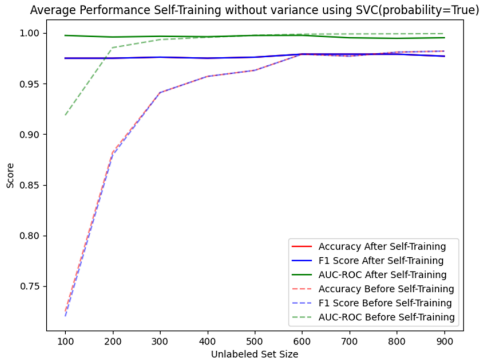
Previously, in each data generation process, the generated dataset was divided into a test set comprising 10% of the data, and a training set comprising 90% of the data. Subsequently, the training set was further split into 40% labeled data, which would be subjected to bias, and the remaining 50% unlabeled data. It is worth noting that although the labels for the unlabeled data were available during initialization, they were not utilized in the experiment.

In this particular experiment, the aforementioned split of labeled and unlabeled data will be manipulated. Specifically,

the split will be modified to range from 10% unlabeled data up to 90% unlabeled data. Considering that the entire dataset consists of 1000 samples this means that the range of unlabeled data will increase from 100 to 900. By doing so, the resulting plots will demonstrate the impact of varying proportions of unlabeled data on the observed outcomes.

3.4.1 Self-Training Manipulating Unlabeled Set Size

(a) *Self-Training using Logistic Regression with varying unlabeled set sizes*



(b) *Self-Training using SVC with varying unlabeled set sizes*

Figure 8: *This figure presents the mean scores obtained by splitting the entire dataset into different proportions of labeled and unlabeled data. The unlabeled data is utilized in conjunction with the labeled biased training set for self-training. The size of the test set remains constant at 100 samples throughout the experiment. In Figure 8b, the accuracy and F1 score have identical values, making it not possible to display both lines simultaneously.*

In Figure 8, the performance of the self-training algorithm is illustrated after ten iterations on ten different instances of datasets. The same datasets were used for both figures. Several important conclusions can be drawn from these figures.

Firstly, it is evident that in both figures, the self-training classifiers consistently outperform or match the baseline performance, demonstrating their effectiveness. Importantly, the self-training classifiers never exhibit lower performance compared to their respective baseline.

Furthermore, a notable observation is the variance in performance between logistic regression and SVC. In Appendix D in Figure 13a, where logistic regression is employed, the variance is considerably larger compared to Figure 13b, which utilizes SVC. Moreover, as the size of the unlabeled set increases, the variance decreases in the case of SVC, indicating improved stability in performance.

Additionally, Figure 8b demonstrates that when SVC is used, the self-training classifiers consistently exhibit favorable performance compared to the baseline, regardless of the size of the unlabeled set. This suggests that the self-training algorithm is robust and capable of achieving good results across different unlabeled set sizes.

However, in Figure 8a, it can be observed that as the size of the unlabeled set increases, the performance of the self-training classifiers decreases. Despite this decrease, the self-training classifiers consistently outperform the baseline, highlighting their superiority.

Overall, these findings emphasize the effectiveness of the self-training algorithm, showcasing its ability to improve performance and outperform the baseline in various scenarios.

In Appendix E in Figure 15 plots are displayed showing the result of executing this same experiment with the self training algorithm of sklearn, this implements the traditional way of self learning. Comparing Figure 15 with Figure 8 it can be seen that the for SVC both methods have a similar performance. However, for logistic regression it can be seen that in Figure 15a the self-training method has about the same performance as the baseline and even under performs under the baseline when size of the unlabeled set increases, whereas in Figure 8a the self-training method consistently outperforms the baseline.

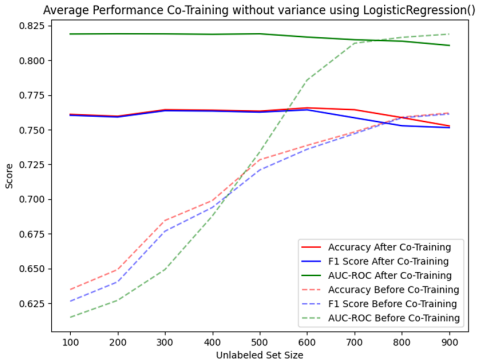
3.4.2 Co-Training Manipulating Unlabeled Set Size

The generated plots presented below are based on 30 different dataset instances and 10 iterations of the co-training algorithm. Due to its faster execution time compared to the self-training algorithm, co-training was applied to a larger number of datasets for comprehensive testing and evaluation.

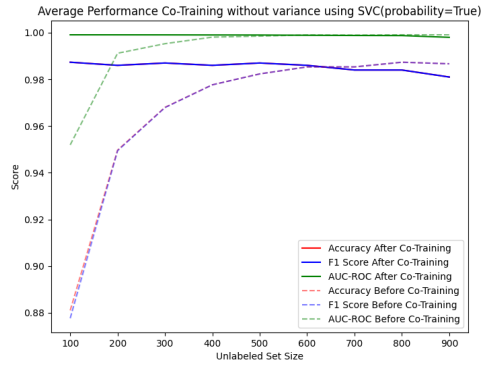
By analyzing Figure 9, several observations can be made regarding the impact of varying the size of the unlabeled set on the performance of the co-training classifiers. Firstly, it can be noted that changing the size of the unlabeled set has minimal effect on the co-training classifiers' performance.

When comparing the co-training classifiers to the baseline, it is evident in both figures that when the size of the unlabeled set is small, the co-training classifiers consistently outperform the baseline. However, as the size of the unlabeled set increases, the co-training classifiers slightly underperform in comparison to the baseline. This suggests that the co-training algorithm may not be as effective when faced with larger amounts of unlabeled data.

Additionally, a notable observation is that Figure 9b, which utilizes SVC in co-training, closely resembles Figure 8b, where SVC was used in self-training. This indicates that the performance of the co-training classifiers with SVC is comparable to that of the self-training algorithm with SVC. On the other hand, when comparing Figure 9a to Figure 9b, it becomes apparent that SVC consistently outperforms logistic

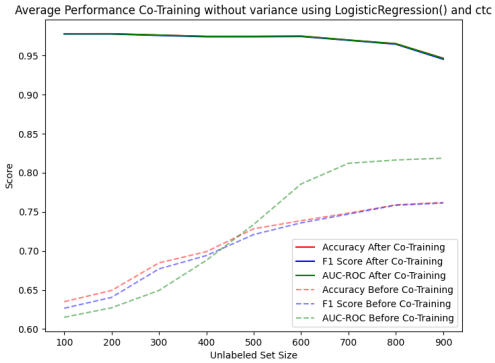


(a) *Co-Training using Logistic Regression with varying unlabeled set sizes*

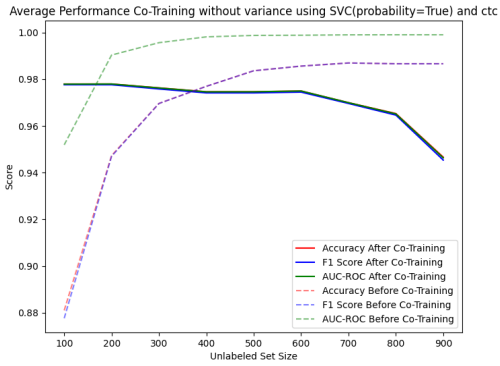


(b) *Co-Training using SVC with varying unlabeled set sizes*

Figure 9: *This figure presents the mean scores obtained by splitting the entire dataset into different proportions of labeled and unlabeled data. The unlabeled data is utilized in conjunction with the labeled biased training set for co-training. The size of the test set remains constant at 100 samples throughout the experiment. In Figure 9b, the accuracy and F1 score have identical values, making it not possible to display both lines simultaneously.*



(a) *Co-Training using Logistic Regression with varying unlabeled set sizes and using the ctc classifier*



(b) *Co-Training using SVC with varying unlabeled set sizes and using the ctc classifier*

Figure 10: *The figure displays the identical dataset instances as the previous experiments. However, in this figure, the ctc classifier from the mvlearn library is employed instead of the implemented co-training approach. The plots depict the scores while varying the size of the unlabeled set. Please refer to Appendix F Figure 17 for these figures with their corresponding variances.*

regression, and the variance associated with SVC is relatively smaller (Appendix D Figure 13).

In summary, the experiments conducted on co-training demonstrate that the size of the unlabeled set has limited influence on the classifiers' performance. Moreover, SVC consistently outperforms logistic regression in co-training, showcasing its superiority in this context, while exhibiting a smaller variance.

Figure 10 presents the results obtained by training the model using the ctc classifier from the mvlearn library (as described in section 2.4). When comparing Figure 17a with the previous approach shown in Figure 9a, the ctc approach consistently outperforms the previous approach and the baseline by a margin of around ten percent, when using logistic regression. Contrarily, Figure 17b illustrates that the ctc classifier underperforms compared to the baseline and the previous approach shown in Figure 9b when using SVC, showing the same phenomena as in the previous experiment in that ctc outperforms when using logistic regression and underperforms when using SVC.

4 Responsible Research

It is important to note that since the generated data is synthetic, it does not hold any intrinsic meaning or represent real-world individuals or entities. The primary purpose of using synthetic datasets is to create controlled environments where biases can be intentionally introduced and systematically evaluated. This allows for a focused analysis of the effectiveness of the semi-supervised learning techniques in mitigating sample selection bias, without the potential ethical concerns and privacy implications associated with real-world data. Furthermore, by intentionally creating the bias in the synthetic datasets, we can ensure that the training data indeed contains the desired bias. This provides a distinct advantage compared to using pre-existing datasets, where the presence and extent of bias may be unknown or unintentional. The de-

liberate introduction of bias in the synthetic datasets allows for a controlled evaluation of the semi-supervised learning techniques' effectiveness in addressing and mitigating bias. This enhances the reliability and interpretability of the results, enabling a more accurate assessment of the techniques' performance in handling biased data.

5 Discussion

The results of this study provide valuable insights into the performance and behavior of self-training and co-training algorithms in semi-supervised learning settings. Through a comprehensive evaluation of various conditions and scenarios, several key findings have emerged.

First, the manipulation of bias weight was examined, and it was observed that increasing the bias weight had a significant impact on the clusters within the dataset. The clusters associated with certain classes increased in size while others diminished. However, there was a maximum bias weight beyond which further increases had minimal effect. This finding suggests that there is a threshold beyond which bias weight manipulation does not significantly improve algorithm performance. It was observed that the performance of the self-training and co-training algorithms was not significantly affected by changes in the bias weight. Regardless of the weight, the algorithms consistently outperformed or matched the baseline models. This indicates that these algorithms are robust.

Next, the effect of the size of the unlabeled training set was investigated. It was observed that in general self-training and co-training outperform the baseline when there is a small amount of unlabeled data, however they match or occasionally perform worse than the baseline when this amount of unlabeled data increases.

Furthermore, it was found that the accuracy and F1-score had consistently similar performance, indicating that the algorithms were effective in correctly classifying both the positive and negative instances. This balanced performance across the metrics demonstrates the reliability and consistency of the self-training and co-training algorithms.

In addition, the study compared the performance of the modified version of self-training with the traditional implementation using sklearn. Remarkably, the modifier version consistently outperformed or matched the traditional implementation. This result highlights the effectiveness of the modification in enhancing the algorithm's performance. The modified self-training approach shows promise as a reliable and improved method for semi-supervised learning tasks.

Similarly, for the most part, the co-training implementation using the ctc classifier from the mvlearn library outperformed or matched the modified implementation of co-training. The ctc classifier demonstrated superior performance, indicating its suitability for co-training algorithms. However, it is worth noting that when using SVC and having a large unlabeled set size, the co-training approach with the ctc classifier performed worse. This same phenomena was observed when comparing the ctc classifier when using SVC and changing the bias weight. These exceptions suggest that the performance of co-training can be affected by the choice of classifier and the size of the unlabeled set and the weight of the bias. Further investigation is needed to understand and mitigate these limitations.

Overall, the results of this study demonstrate the effectiveness of self-training and co-training algorithms in improving classification performance. The algorithms consistently outperformed or matched the baseline models across different conditions and scenarios, exhibiting robustness and reliability. The study also revealed insights into the impact of bias weight manipulation, unlabeled set size, and classifier selection on algorithm performance. These findings contribute to the understanding and practical application of self-training and co-training algorithms in semi-supervised learning tasks. Future research can build upon these findings to refine and optimize these algorithms further.

6 Conclusions and Future Work

This study provides valuable insights into the performance and behavior of self-training and co-training algorithms in semi-supervised learning. It highlights the effectiveness of synthetic datasets in manifesting intended biases, the adaptability of these algorithms and the impact of bias weight manipulation.

How effectively do the synthetic datasets generated in this research manifest the intended bias?

Based on visualization of the datasets, it was observed that the decision boundary changed when the bias was introduced among all the tested generated datasets. This indicates that a bias was present and that the synthetic datasets effectively manifested the intended bias. The presence of the bias allowed for the examination of the impact of bias weight manipulation and its influence on the algorithm's behavior and performance.

How does the performance of self-training and co-training vary with different quantities of labeled data?

The performance of self-training and co-training algorithms varies with different quantities of labeled data. When there is a small amount of unlabeled data, both self-training and co-training algorithms tend to outperform the baseline models. This suggests that utilizing a small number of unlabeled instances can improve classification performance. However, as the amount of unlabeled data increases, the performance of self-training and co-training algorithms may match or occasionally perform worse than the baseline.

In conclusion, the performance of self-training and co-training algorithms is influenced by the quantity of labeled and unlabeled data available. These algorithms demonstrate their effectiveness and outperform the baseline models when a small amount of unlabeled data is utilized. However, as the quantity of unlabeled data increases, the performance may not always surpass the baseline. This highlights the importance of carefully considering the balance between labeled and unlabeled data in semi-supervised learning tasks to achieve optimal results.

How does the performance of self-training and co-training adapt to varying levels of bias in the data?

The manipulation of bias weight in the dataset had a significant impact on the performance of self-training and co-

training algorithms. Increasing the bias weight affected the clustering of classes within the dataset, with certain clusters increasing in size while others diminished. However, there was a threshold beyond which further increases in bias weight had minimal effect on algorithm performance. This indicates that the algorithms are sensitive to bias in the data and can adapt to a certain extent, but there are limits to the benefits gained from bias manipulation. The performance of self-training and co-training algorithms was found to be robust and consistent across different weights of biases. Regardless of the bias weight, these algorithms consistently outperformed or matched the baseline models. This suggests that self-training and co-training can adapt to varying levels of biases within the training data.

In conclusion, the performance of self-training and co-training algorithms was consistent and robust across different quantities of labeled data. They demonstrated the ability to adapt and leverage both labeled and unlabeled data effectively. The algorithms were also sensitive to varying levels of bias in the data, with bias weight manipulation influencing the algorithm's performance. The synthetic datasets generated in this research successfully manifested the intended bias, allowing for the examination of bias effects on algorithm behavior. These findings contribute to our understanding of the behavior and performance of self-training and co-training algorithms in semi-supervised learning settings and provide valuable insights for their practical application.

By addressing these recommendations, future research can contribute to advancing the field of semi-supervised learning and refining the application of self-training and co-training algorithms in practical settings:

- Further Investigation of Bias Effects: While this study successfully examined the impact of bias weight manipulation, future research should delve deeper into the effects of bias in semi-supervised learning. Investigating the influence of different types and levels of bias on algorithm behavior and performance would provide a more comprehensive understanding of how biases affect these algorithms.
- Optimal Balance of Labeled and Unlabeled Data: Given the varying performance of self-training and co-training algorithms with different quantities of labeled data, future studies should explore the optimal balance between labeled and unlabeled data. Identifying the threshold at which the performance starts to decline and understanding the trade-offs between labeled and unlabeled data would assist in guiding the selection of data quantities for these algorithms.
- Enhanced Co-Training Approaches: Considering the exception where co-training using the ctc classifier performed worse with a large unlabeled set size and SVC, further research should focus on developing enhanced co-training approaches. Exploring alternative classifiers or modifying the co-training framework to mitigate these limitations would contribute to improving the performance of co-training algorithms in diverse scenarios.
- Evaluation of Other Performance Metrics: While this study focused on accuracy and F1-score as evaluation metrics, future research should consider exploring additional performance metrics to gain a more comprehensive understanding of algorithm performance. Metrics such as precision, recall, or area under the receiver operating characteristic curve (AUC-ROC) could provide further insights into the strengths and weaknesses of self-training and co-training algorithms.

References

[1] Lorraine K. Alexander, Brettania Lopes, Kristen Ricchetti-Masterson, and Karin B. Yeatts. *Eric Notebook*. UNC CH Department of Epidemiology, 2nd edition.

[2] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT '98, pages 92–100, New York, NY, USA, 1998. Association for Computing Machinery.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, ..., and Jake VanderPlas. Mvlearn: multiview machine learning in python. *Journal of Machine Learning Research*, 17(137):1–5, 2013.

[4] Haoran Chai, Yiqing Liang, Siyu Wang, and et al. A novel logistic regression model combining semi-supervised learning and active learning for disease classification. *Scientific Reports*, 8:13009, August 2018.

[5] Olivier Chapelle, Bernhard Scholkopf, and Eds. A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20:542–542, 2009.

[6] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22243–22255. Curran Associates, Inc., 2020.

[8] Thomas M.H. Hope. Chapter 4 - linear regression. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 67–81. Academic Press, 2020.

[9] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. *School of Computer Science, Univ. of Waterloo*.

[10] Wouter M. Kouw and Marco Loog. A review of domain adaptation without target labels. *arXiv preprint arXiv:1901.05335*.

[11] Yuanqing Li, Huiqi Li, Cuntai Guan, and Zhengyang Chin. A self-training semi-supervised support vector

machine algorithm and its applications in brain computer interface. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 1, pages I–385–I–388, 2007.

[12] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. Thresholding classifiers to maximize f1 score, 2014.

[13] Sarang Narkhede. Understanding auc-roc curve. *Towards Data Science*, 26(1):220–227, 2018.

[14] J. Odgaard-Jensen, G.E. Vist, A. Timmer, R. Kunz, E.A. Akl, H. Schünemann, M. Briel, A.J. Nordmann, S. Pregno, and A.D. Oxman. Randomisation to protect against selection bias in healthcare trials. *Cochrane Database of Systematic Reviews*, 4, 2011.

[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Derek A. Pisner and David M. Schnyer. Chapter 6 - support vector machine. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 101–121. Academic Press, 2020.

[17] Y. C. A. P. Reddy, P. Viswanath, and B. E. Reddy. Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, 7(1.8):81, 2018.

[18] Louisa H. Smith. Selection mechanisms and their consequences: Understanding and addressing selection bias. *Current Epidemiology Reports*, 7(4):179–189, 2020.

[19] Jafar Tanha, Maarten van Someren, and Hamideh Afsarmanesh. Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(2):355–370, February 2017.

[20] Jennifer Wu Tucker. Selection bias and econometric remedies in accounting and finance research. *Journal of Accounting Literature*, 29:31–57, 2010.

[21] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11525–11536. PMLR, 18–24 Jul 2021.

[22] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. *IBM T.J. Watson Research Center*.

[23] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

Appendix A

The pseudocode provided in Algorithm 1 offers a more detailed description of the data generation process:

Algorithm 1 generate_data (n_samples, n_features, n_classes, n_clusters, density, min_dist)

$n\_samples \leftarrow$ number of samples
$n\_features \leftarrow$ number of features
$n\_classes \leftarrow$ number of classes
$n\_clusters \leftarrow$ number of clusters
$density \leftarrow$ density parameter
$min\_dist \leftarrow$ minimal distance between cluster centers
$samples\_per\_cluster \leftarrow$ distribute $n\_samples$ evenly among $n\_clusters$
$cluster\_centers \leftarrow$ randomly generate $n\_clusters$ cluster centers
while not all cluster centers are at least min_dist apart **do**
    $cluster\_centers \leftarrow$ randomly generate $n\_clusters$ cluster centers
end while
$clusters\_per\_class \leftarrow n\_clusters$ divided by $n\_classes$
$data \leftarrow$ empty list
$labels \leftarrow$ empty list
$cluster\_labels \leftarrow$ empty list
for each $cluster\_id$ and $center$ **do**
    if $cluster\_id$ is less than $n\_classes \times clusters\_per\_class$ **then**
        $class\_id \leftarrow cluster\_id$ divided by $clusters\_per\_class$
    else
        $class\_id \leftarrow$ randomly choose a class from $n\_classes$
    end if
    $cluster\_samples \leftarrow samples\_per\_cluster[cluster\_id]$
    $cov\_matrix \leftarrow$ create covariance matrix with $density$ value
    $samples \leftarrow$ generate random samples around $center$ using cov_matrix and $cluster\_samples$
    add $samples$ to $data$
    add $class\_id$ to $labels$ for each $cluster\_sample$
    add $cluster\_id$ to $cluster\_labels$ for each $cluster\_sample$
end for
return $data$, $labels$, $cluster\_labels$

Appendix B

The pseudo-code provided in Algorithm 2 offers a more detailed description of the modified self training algorithm:

Algorithm 2 Self-Training Algorithm

procedure SELF_TRAINING (data_labeled, labels_labeled, data_unlabeled, num_iterations, threshold, model)
    $data\_labeled \leftarrow$ labeled data samples used for training
    $labels\_labeled \leftarrow$ labels corresponding to training data samples
    $data\_unlabeled \leftarrow$ unlabeled data samples used for training
    $num\_iterations \leftarrow$ maximum number of iterations
    $threshold \leftarrow$ minimum confidence level
    $model \leftarrow$ machine learning model used for training
    for $iteration$ in $range(num\_iterations)$ **do**
        Train the model using the labeled data: $model.fit(data\_labeled, labels\_labeled)$
        Compute the predicted probabilities for the unlabeled data: $unlabeled\_probabilities \leftarrow model.predict\_proba(data\_unlabeled)$
        Identify indices of unlabeled samples with confidence above the threshold: $confident\_indices \leftarrow$ max$(unlabeled_probabilities, axis = 1) \geq threshold$
        Select confident unlabeled samples based on the indices: $confident\_data \leftarrow data\_unlabeled[confident\_indices]$
        Select corresponding confident probabilities: $confident\_probabilities \leftarrow unlabeled\_probabilities[confident\_indices]$
        if len$(confident_data) == 0$ **then**
            break
        end if
        Assign labels to confident samples based on maximum probability: $confident\_labels \leftarrow$ argmax$(confident_probabilities, axis = 1)$
        Add confident samples and labels to the labeled data: $data\_labeled \leftarrow$ concat$((data_labeled, confident_data))$
        Add confident labels to the labeled labels: $labels\_labeled \leftarrow$ concat$((labels_labeled, confident_labels))$
        Decrease the confidence threshold: $threshold \leftarrow threshold - 0.05$
        Create a list of models: $models \leftarrow [model, RandomForestClassifier(n\_estimators = 100)]$
        for $model$ in $models$ **do**
            Train the model using the updated labeled data: $model.fit(data\_labeled, labels\_labeled)$
        end for
    end for
    return $model, data\_labeled, labels\_labeled$
end procedure

Appendix C

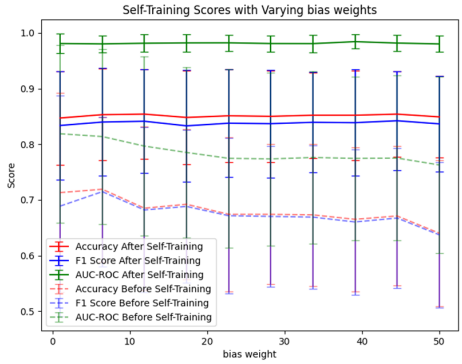
Table 1 provides the exact values that have been used for the parameters throughout the experiments.

Table 1: Parameters for data generation

Parameter	Value
$n\_samples$	1000
$n\_features$	2
$n\_classes$	3
$n\_clusters$	6
$density$	0.01

Appendix D

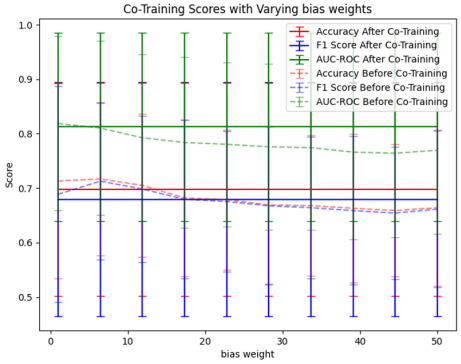
The figures (figure 11 - 13) presented in this appendix depict plots of the mean scores along with error bars, which provide a visual representation of the variance surrounding the scores.



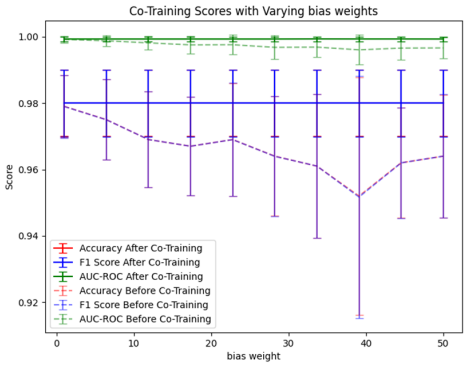
(a) Self-Training using Logistic Regression with varying weights

(b) Self-Training using SVC with varying weights

Figure 11: *The mean scores of the Self-Training approach are displayed, presenting the same scores as shown in Figure 5, but now accompanied by error bars in the plot to represent the variance.*

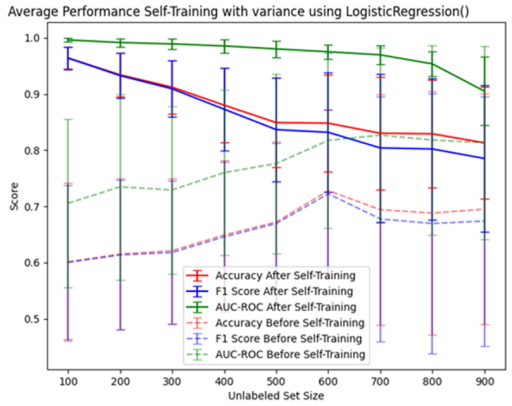


(a) Co-Training using Logistic Regression with varying weights

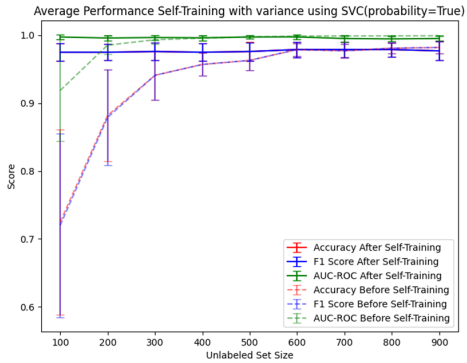


(b) Co-Training using SVC with varying weights

Figure 12: *The mean scores of the co-Training approach are displayed, presenting the same scores as shown in Figure 6b, but now accompanied by error bars in the plot to represent the variance.*



(a) Self-Training using Logistic Regression with varying unlabeled set sizes

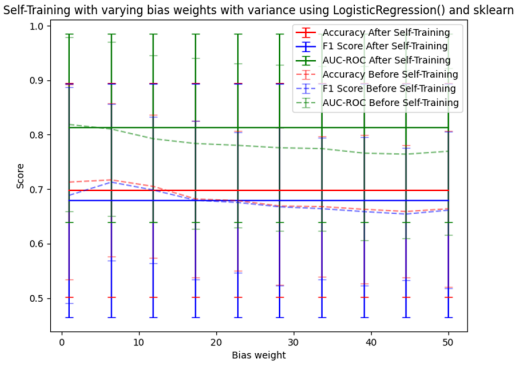


(b) Self-Training using SVC with varying unlabeled set sizes

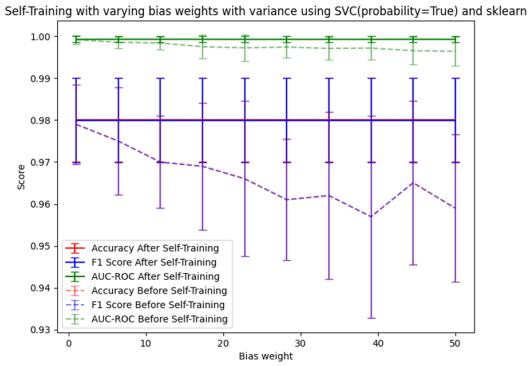
Figure 13: *The mean scores of the Self-Training approach are displayed, presenting the same scores as shown in Figure 8b, but now accompanied by error bars in the plot to represent the variance.*

Appendix E

The figures (figures 14 and 15) in this appendix display the plots of the mean scores using the self-training algorithm implemented by sklearn.

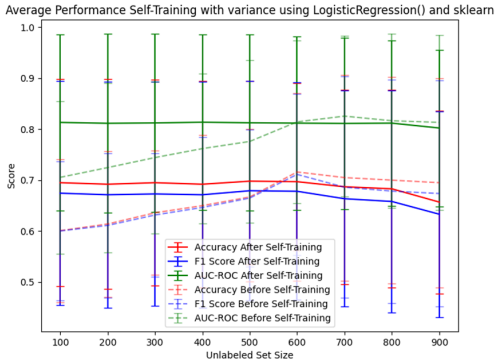


(a) Self-Training using Logistic Regression with varying weights

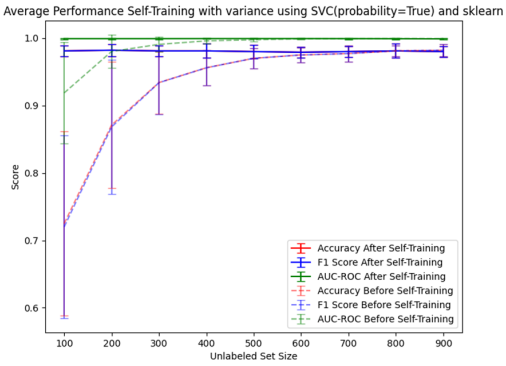


(b) Self-Training using SVC with varying weights

Figure 14: *The plot illustrates the mean scores achieved with different bias weights using the traditional self-training algorithm implemented with the sklearn library, instead of the modified algorithm utilized previously.*



(a) Self-Training using Logistic Regression with varying unlabeled set sizes

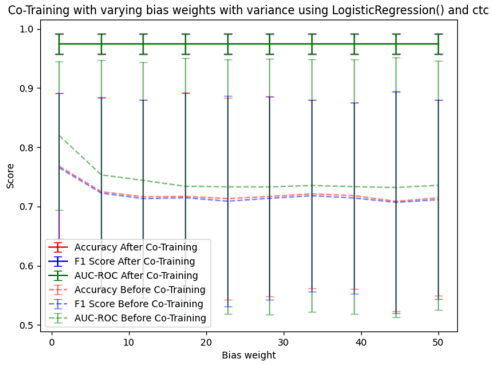


(b) Self-Training using SVC with varying unlabeled set sizes

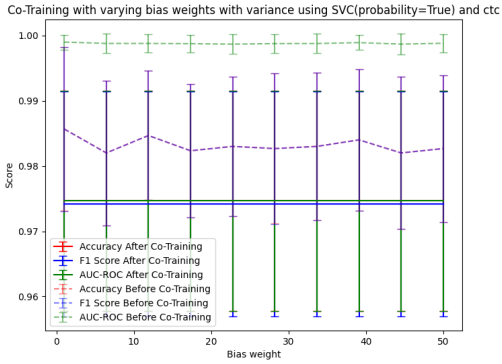
Figure 15: *The plot illustrates the mean scores achieved with different unlabeled set sizes using the traditional self-training algorithm implemented with the sklearn library, instead of the modified algorithm utilized previously.*

Appendix F

The figures (Figures 16 and 17) in this appendix display the plots of the mean scores using the ctc-classifier from the mvlearn library including the individual variances.

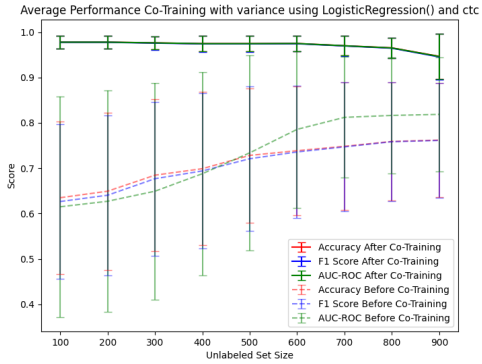


(a) *Co-Training using Logistic Regression with varying weights and using the ctc classifier*

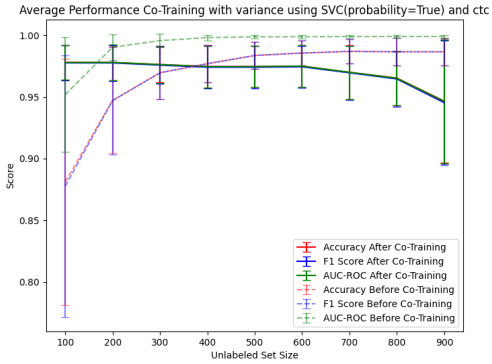


(b) *Co-Training using SVC with varying weights and using the ctc classifier*

Figure 16: *In this figure, the ctc classifier from the mvlearn library is employed instead of the implemented co-training approach. The plots depict the scores while also indicating their individual variances. In Figure 16a, the accuracy, F1 score, and AUC-ROC values align perfectly, resulting in the use of a single line to represent these the lines obtained after applying co-training, whereas in Figure 16b the accuracy and F1 score align perfectly, thus only a single line has been displayed for this.*



(a) *Co-Training using Logistic Regression with varying unlabeled set sizes and using the ctc classifier*



(b) *Co-Training using SVC with varying unlabeled set sizes and using the ctc classifier*

Figure 17: *In this figure, the ctc classifier from the mvlearn library is employed instead of the implemented co-training approach. The plots depict the scores while also indicating their individual variances, while varying the size of the unlabeled set.*