International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

An Enhancement of Adaboost Algorithm Applied in Online Transaction Fraud Detection System

Christian S. Ortega[1], Lance Daniel P. Lim[2], Alrhia Ruby S. Bautista[3], Vivien A. Agustin[4]

[1,2,3]Author, Pamantasan ng Lungsod ng Maynila
[4]Co-Author, Pamantasan ng Lungsod ng Maynila

Abstract

This study is focused on the enhancement of the AdaBoost model for online transaction fraud detection to improve performance in detecting fraudulent activities. The study addresses the limitations of AdaBoost, including class imbalance, long training times, and overfitting. Three techniques were integrated to optimize the model. SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset, Quantile-Based Capping (QBC) to reduce training time, and Early Stopping to prevent overfitting. The dataset used for training consisted of online transaction records, with model performance evaluated using standard classification metrics. Results show that applying Heron-Centroid SMOTE led to an increase in accuracy, from 0.8150 to 0.8198, while significantly improving recall from 0.2962 to 0.3773, indicating better identification of minority class instances. The F-measure rose from 0.4135 to 0.4798, reflecting a better balance between precision and recall. The G-mean improved from 0.5337 to 0.5970, showing overall better classification performance. QBC reduced training time from 1.45 to 1.32 seconds, and Early Stopping increased accuracy from 0.73 to 0.816. These findings suggest that the proposed enhancements significantly improve AdaBoost's efficiency and reliability, making it more effective for online fraud detection.

CHAPTER ONE
INTRODUCTION

This section of the study provides a comprehensive context and background information about the chosen algorithm and its intended application. It includes an overview of the algorithm and problems found and the researchers objective to enhance the algorithm.

1.1 Background of the Study

Adaptive Boosting, or the AdaBoost algorithm, is a supervised learning model commonly used for data classification by combining multiple weak learners to produce a strong learner. This approach is particularly effective in detecting fraudulent activities in potential transaction fraud by evaluating and assessing differences between normal transactions and fraudulent ones through pattern learning. Machine learning algorithms, such as AdaBoost, have shown promising results in fraud detection, as evidenced by Wang et al. (2020). Their research demonstrated AdaBoost's effectiveness in identifying

fraudulent online transactions. They trained AdaBoost to recognize subtle patterns indicative of fraud by analyzing large datasets containing both genuine and fraudulent transactions. Through its iterative process of combining weak learners into a strong classifier, AdaBoost achieved high accuracy in distinguishing fraudulent from legitimate transactions. These findings emphasize the importance of utilizing advanced machine learning techniques like AdaBoost to mitigate the growing threat of fraud in digital transactions.

While AdaBoost is considered one of the best algorithms for classification, it still has several areas to be enhanced. AdaBoost algorithm may produce an imbalanced dataset during the training process, leading to inaccurate classification results. It may result in longer training times, as the algorithm assigns more weight to misclassified data initially caused by outliers, thereby affecting the overall efficiency of the model. Lastly,

AdaBoost faces challenges in generalizability due to overfitting on biased weights that tend to focus on training data noise, which impacts the overall effectiveness of the model when given unseen data.

Given that AdaBoost has its limitations when it comes to classification capabilities, the researchers found a method to help AdaBoost and enhance its overall performance by using these methods. SMOTE (Synthetic Minority Over-sampling Technique) is a method used to address class imbalance in datasets by generating synthetic examples for the minority class. It creates new instances by interpolating between existing minority class samples. According to Taghikhah, Kumar et al. (2024), the quantile-based method helps in normalizing the flow of pre-trained features and detecting outliers based on their log-likelihood values. This method outperforms existing unsupervised outlier detection techniques and reduces the need for extensive negative training data, resulting in better classification performance. Although accuracy can be improved with SMOTE, there is a risk wherein it could introduce noisy instances and overfitting problems as it randomly duplicates minority class samples (Meng, D. & Li, Y., 2022). With the problem of overfitting in mind, the generalizability of the algorithm would decrease, requiring another method in order to mitigate its effects. The model may improve and provide good results. However, at a certain point during its training process, the results given by the training subset will begin showing improvement, while the loss function of the validation set starts to increase, showing that the results are declining. This is known as overfitting. To avoid this, early stopping can be used to terminate the training process depending on the model updates (Cai, Y. et al, 2022). With early stopping, the training process of the model is stopped before it can learn further from noise and irrelevant patterns that could harm its generalizability, thus reducing its risk of overfitting.

Enhanced AdaBoost will be applied to an online fraud detection system to ensure data integrity with every online transaction. Fraud is one of the biggest problems in our world today. It occurs when people try to deceive others for their own gain, whether for

money, tangible items, or intangible things such as personal identity. Fraud creates many negative impacts on individuals, establishments, businesses, and finance. According to Hashedi & Mangalingam (2021), the rise of fraud cases is the reason why fraud detection systems are created. Fraud detection is a way to detect and predict possible fraudulent activities. It is utilized not only by banks and financial institutions, but also in e-commerce. As technology evolves, fraud detection systems enhance their ability to detect people and their fraudulent activities, hence reducing the risk of financial, reputational, and identity fraud.

1.2 Statement of the Problem

Adaptive Boosting is one of the well-known algorithms used for classification tasks because it has been proven in some studies to provide quality and accurate classification results. However, the algorithm still has room for improvement in its overall performance and has limitations that limit the quality of its general effectiveness. These limitations may lead to harm in some applications, such as detecting and classifying fraud from online transactions.

This study aims to address the following:

1. **Imbalanced datasets can significantly affect machine learning algorithms, including AdaBoost, by biasing the model towards the majority class.**

This often results in less accurate predictions for the minority class, which can hinder performance in critical applications (Randhawa et al., 2018). In AdaBoost, class imbalance complicates the weight adjustment process, as the algorithm focuses more on the majority class, leading to biased predictions and reduced overall model accuracy (Zhou et al., 2019).

2. **Longer training time due to more weight on misclassified data initially caused by outliers produced by previous weak learners during the iteration when training the data.**

According to Desarda (2023), AdaBoost produces a longer training time during the iteration process because after training a classifier, adaboost will assign weight to each of the training items but assign more weight to the misclassified data.

3. **Difficulty in Generalization due to overfitting based on biased weight normalization after multiple iteration updates which tends to focus or memorize specific patterns in only the training data noise.**

According to Modarres et al (2020), AdaBoost focuses on the weight of each sample from examples in its previous iterations during training. In ensemble learning algorithms like AdaBoost, in each iteration, it produces noise in the training process which leads to overfitting during its learning.

1.3 Objectives of the Study

The primary aim of this study is to improve the AdaBoost algorithm to significantly boost its classification performance. By addressing the algorithm's limitations, the study seeks to enhance its accuracy and reliability. This research desires to make AdaBoost a more effective tool for high-performance classification tasks in various practical applications.

1.3.1 Specific Objectives

1. To utilize Synthetic Minority Oversampling Technique (SMOTE) technique to balance class distribution by generating synthetic samples for the minority class.
2. To use Quantile Based Capping to limit the values of outliers to specific thresholds with using the formula:(lower bound = (Q1–1.5*IQR), upper bound = (Q3+1.5*IQR))
3. Utilize the Early Stopping regularization technique to prevent overfitting by calculating a validation error for each of the iterations, comparing the accuracy between the results in training and testing, and if the validation error begins to increase or the accuracy does not improve after a certain number of iterations, the training process halts.

1.4 Significance of the Study

The significance of this study lies in its potential to benefit individuals through the application of an enhanced version of the AdaBoost algorithm in an Online Transaction Fraud Detection System

The following are the beneficiaries of this study:

Clients - This study significantly benefits clients of the online transaction platforms by addressing fraud detection challenges in digital transactions, thereby strengthening security practices. By leveraging the AdaBoost algorithm, this research optimizes fraud detection accuracy, offering consumers enhanced protection against financial fraud, particularly identity theft. Moreover, the study contributes to advancing technology, providing valuable insights and methodologies to bolster fraud detection performance, ultimately ensuring safer digital transactions for the clients of online transactions platforms.

Business Owners - This study will offer significant advantages to business owners by helping them to reduce costs and also ensures the security and safety of clients which will foster a greater trust and confidence among clients. It will also enhance the business reputation and relationships between clients.

Future Researchers - This paper provides valuable insights into the advancements achieved in transaction fraud detection using the AdaBoost algorithm, making it an essential resource and reference point. It establishes a foundation for further research and innovation in this field, encouraging the continuous improvement of fraud detection methods and procedures. By leveraging the AdaBoost algorithm's capabilities, this study enhances the accuracy and efficiency of identifying fraudulent activities, ultimately contributing to more robust security measures and protecting financial and personal data.

1.5 Scope and Limitations

This study focuses on the enhancement of the AdaBoost algorithm for fraud detection in online transactions. It aims to address the limitations of the model, which includes imbalanced datasets, long training time, and generalization. The researchers proposed SMOTE, Quantile-Based Capping, and Early Stopping as solutions to these limitations. The algorithm is capable of classification, which would allow it to discern fraudulent activity, possibly mitigating the threat of such activities. AdaBoost will evaluate and assess differences by identifying patterns and relationships between datasets that contain both fraudulent and non-fraudulent transaction activities.

Despite these promising capabilities, AdaBoost comes with its own set of limitations. First, AdaBoost may not provide accurate results when it is provided with imbalanced datasets. Another challenge encountered by the algorithm is the tendency for its training time to increase due to the presence of outliers which are produced from the weak learners that it creates in each iteration as it is being trained. Additionally, while AdaBoost may provide accurate results during its training, the results may no longer be accurate when it is being tested as it may overfit with how it was trained due to biases in the weights, effectively reducing its generalizability.

1.6 Definition of Terms

This section of the study outlines the specific terminology employed to facilitate the execution and comprehension of the research processes. The terms used in this study may utilize a meaning that is specific within the confines of this paper. As such, the following are the operational definitions for the terms utilized within this study by the researchers:

AdaBoost - A machine learning algorithm that combines multiple weak classifiers to create a strong classifier.

Algorithm - A set of rules or instructions for solving problems.

Classification - Classifying data based on attributes.
Decision tree - Graphical decision-making tool for machine learning classification and regression.
Epoch – refers to when the algorithm has completed one pass in the training dataset.
Fraud - The act of deceiving people for their own gain.
Fraud Detection - Detecting possible fraudulent activities during transactions between sellers and buyers.
Generalizability – The ability of a model to generalize its results towards different settings, groups, and situations.
Generalization – Capability of trained models to make accurate predictions from new, unseen data.
Machine Learning - Allowing computers to learn and predict from data without the need of explicit programming.
Noise – data that includes meaningless information. Random or unpredictable fluctuations in data that disrupt the ability to identify the targeted patterns or relationships.
Outliers - A data point that falls significantly outside the typical range of values in a set of data.
Regression - Predicting continuous outcomes by modeling the relationship between variables.
Smote - (Synthetic Minority Over-sampling Technique) is a method that generates synthetic instances of the minority class in imbalanced datasets to enhance model performance.
Unseen Data – data that was not part of the training data introduced to the model.

CHAPTER TWO

REVIEW OF RELATED LITERATURE

2.1 Related Literature

The AdaBoost algorithm, an ensemble learning technique for classification and regression tasks (GeeksforGeeks, 2024), iteratively trains a sequence of weak classifiers. Each new classifier focuses on data points that previous ones misclassified, giving them higher weight in the training process. This approach progressively improves the overall model's accuracy.

According to Saini, A. (2024), AdaBoost was introduced by Freaund and Schapire in 1997 and since that, the algorithm is used for binary classification where in it outputs only 2 options. It enhances prediction accuracy by combining multiple weak learners to make a strong learner. Adaboost focuses more on much higher weight data because that kind of data needs more attention to be solved than the common ones.

AdaBoost's slower training time is primarily attributed to how it manages misclassified data. When the algorithm misclassifies data points, it increases their weights in subsequent iterations, causing the algorithm to spend more time correcting these errors. This reweighting process can significantly prolong the training duration, particularly when working with noisy or complex datasets. AdaBoost's focus on improving the accuracy of misclassified samples results in a slower convergence rate.

According to Gajendra(2022), Adaboost works by assigning sample weights to create a base learner that are called stumps. This creates the initial step in creating weak learners and calculates the errors initially as well as to calculate the performance of the base learner. Next thing to do is to update the weights for both classified and misclassified data and normalize the weight to create buckets which means creating a new dataset.

As noted in the Journal of Big Data, this emphasis on misclassified data, while enhancing accuracy,

leads to delayed convergence. Similarly, sources such as MDPI Applied Sciences highlight that the iterative reweighting of misclassified points is a key factor contributing to AdaBoost's extended training time (Ding, Zhu, Chen, & Li, 2022).

According to the study by Wang and Sun (2021) entitled "The Improved AdaBoost Algorithms for Imbalanced Data Classification," class imbalance poses a critical challenge in classification tasks, resulting in diminished performance in recognizing minority classes. The authors assert that traditional AdaBoost algorithms inadequately address this issue, often neglecting the minority class during the training process. Their proposed algorithm introduces weighted vote parameters aimed at improving classification accuracy specifically for imbalanced datasets.

Based on the study of Li et al. (2019), “Improved PSO_AdaBoost Ensemble Algorithm for Imbalanced Data” class imbalance presents a significant challenge in machine learning classification tasks, where traditional AdaBoost algorithms often fail to accurately represent minority classes. To address this, the authors propose AdaBoost-A, an enhanced version that incorporates the Area Under the Curve (AUC) to improve error calculations and classification performance for imbalanced data. Additionally, the study introduces PSOPD-AdaBoost-A, an ensemble method using Particle Swarm Optimization to optimize weak classifiers and improve computational efficiency, particularly for datasets with severe class imbalances.

In the study "An Imbalanced Multifault Diagnosis Method Based on Bias Weights AdaBoost (BW-AdaBoost)", Jiang et al. (2022) address the problem of imbalanced datasets in fault diagnosis, particularly in multifault systems, which are more complex than single faults. The method uses K-nearest neighbor undersampling to focus on boundary samples and bias weights in weak classifiers to emphasize minority fault samples. This strategy significantly improves both accuracy and F1 score, making the method more effective in detecting less frequent faults, while also reducing computational overhead through a hierarchical classification structure.

According to Ahmed, I. (2023), One common method used for oversampling is SMOTE, which works by creating new minority class examples by interpolating between existing samples. Oversampling can be an effective technique for improving the accuracy of the model, but it could lead to overfitting. Another technique that can be used is weighting, which assigns different weights to different classes while the model is being trained. This, however, also has a risk of overfitting despite being able to improve the accuracy of the model during training. The use of a regularization technique is important as it can prevent overfitting for oversampling and weighting techniques.

According to Misra S. & Li H. (2020), AdaBoost is an ensemble method that utilizes trees by training and deploying them in a series. By implementing boosting, AdaBoost uses a set of weak classifiers connected in a series so that each of the weak classifiers would improve the classification samples that were misclassified in the previous weak classifiers to create a strong classifier. These decision trees are called “stumps” in boosting methods because each of the decision trees tend to be shallow models that do not overfit but can be biased. Classification accuracy increases when more weak learners are added to the series of the model, but this may lead to severe overfitting and a drop in the capability of the model for generalization. While AdaBoost is useful for imbalanced datasets, it also underperforms when noise is present.

In the 2023 study of Hao, L. & Huang, G. titled “An improved AdaBoost algorithm for identification of lung cancer based on electronic nose”, they state that evaluating the effectiveness of an algorithm does not lie just within its performance but also its generalization ability and robustness. The study proposed

a version of AdaBoost which showed improved performance due to the multiple integrated heterogeneous classifiers, and generalization due to the implemented k-fold cross-validation.

According to a 2024 study by Ibragimov, B. & Gusev, G. titled "Learn Together Stop Apart: An Inclusive Approach to Ensemble Pruning", they utilized Early Stopping via cross-validation and state that the quality estimation of the training set used in the learning process is biased compared to unseen data. It would be conventional to utilize validation sets in order to control the generalization ability of the algorithm. Such quality estimates can often be highly dependent on particular train-validation splits, which therefore can be noisy. The common way to go about this is by using cross-validation.

According to Ning W. et al in their 2023 study titled "A Credit Card Fraud Model Prediction Method Based on Penalty Factor Optimization AWTadaboost", they state that AdaBoost is prone to overfitting when noisy samples are present. Their theoretical analysis showed that the traditional AdaBoost is overfitting in a noisy training set, leading to a degradation of classification accuracy. They utilized a penalty factor which is constructed from the number of consecutive misclassified samples, using it for the reconstruction of the sample weight assignments in order to prevent the model from overfitting on the noisy samples.

2.2 Related Studies

Randhawa et al (2018) AdaBoost is a machine learning system that detects credit card fraud, a major issue in financial services. This study analyzes real-world credit card data sets using standard models and hybrid approaches such as AdaBoost and majority voting. Assess the effectiveness of the model using both publicly available data sets and real-world data from financial institutions. They add noise to data samples to test the adaptability of the algorithms. The majority voting mechanism is effective in detecting credit card fraud.

Nithin et al (2020) Fraud detection is a crucial aspect of financial services, causing billions of dollars in losses annually. This proposed system uses two mechanisms: fraud prevention and fraud detection. During fake exchanges, the first mechanism prevents misrepresentation, while the second mechanism guesses the fraudster. Credit card fraud is a significant issue, and there is a lack of research on analyzing real-world data due to confidentiality issues. This project uses machine learning algorithms, including AdaBoost, to detect credit card fraud. The efficiency of the 12 model is evaluated using publicly available credit card data sets and real-world data sets from financial institutions. Experimental results show that boosting algorithms achieve high accuracy rates in detecting fraud cases.

Sailusha et al (2020) AdaBoost is a machine learning algorithm that focuses on accuracy, precision, recall, and F1-score for credit card fraud detection. The project aims to address the increasing issue of fraud in online transactions and e-commerce platforms. The AdaBoost algorithm is compared to the Random Forest algorithm, ensuring the best detection method. Based on the confusion matrix, we plot the ROC curve and consider the algorithm with the highest accuracy, precision, recall, and F1-score as the best.

Ileberi et al (2021) Mentioned in the research develops a machine learning framework for detecting credit card fraud, addressing the rise in fraud due to e-commerce and FinTech advancements. The study uses a real-life dataset of unequal data from European credit cardholders and applies the Synthetic Minority Over-sampling Technique (SMOTE) to balance the data. It then tests six machine learning methods: Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), Extreme Gradient Boosting (XGBoost), Decision Tree (DT), and Extra Tree (ET), along with Adaptive Boosting

(AdaBoost). The framework is evaluated using metrics such as accuracy, recall, precision, Matthews Correlation Coefficient (MCC), and Area Under the Curve (AUC), as well as testing it on a highly skewed synthetic dataset. Results show that AdaBoost significantly improves performance, with the boosted models outperforming existing methods.

Zhou et al (2019) The paper proposes a risk control algorithm for integrated learning to detect credit card fraud. It combines SMOTE and principal component analysis, a single-layer decision tree, and the improved Adaboost algorithm. The method achieves an accuracy rate of 96.50% and an F-measure value of 97.3% when tested on a commercial bank data sample. The method outperforms traditional risk control technology and effectively completes fraud detection work, highlighting the importance of robust data mining in credit card fraud prevention.

Lv et al (2019) This paper explores the use of the SMOTE method, the AdaBoost algorithm, and a cost-sensitive algorithm to process imbalanced data in finance, information security, and industrial systems. The imbalanced consumption data of credit cards in the UCI database is processed using these methods. The results show that the SMOTE-AdaBoost method outperforms traditional AdaBoost, and the cost-sensitive algorithm increases the weight of minority class samples.

Pan et al. (2020) conducted the study "Learning imbalanced datasets based on SMOTE and Gaussian distribution." This study highlighted the Border-SMOTEs technique, an improved SMOTE technique that oversamples the boundary to improve prediction precision. Nevertheless, it is imperative that we enhance the robustness of Border-SMOTEs, as they are susceptible to being influenced by imbalanced datasets in various fields. The expansion of minority data and the disregard of the distributional characteristics of minority samples are the primary constraints of Border-SMOTEs. To resolve the issue of unbalanced classification, we introduce an innovative oversampling method, Adaptive-SMOTE, which concentrates on the distribution density of minority samples. To enhance the distributional character of minority data, Adaptive-SMOTE adaptively partitions it into two subsets, Inner and Danger. These subsets are jointly employed to balance the original data by generating new minority data. This method of sampling accurately represents the distributional characteristics of minority data and enhances the classification accuracy of the majority data.

According to Hussein et al. (2019), the issue of imbalanced datasets is a prevalent challenge in machine learning, where standard algorithms frequently encounter difficulties in representing minority classes. The study concentrated on the Synthetic Minority Oversampling Technique (SMOTE), a widely used method that generates synthetic examples to address class imbalance. However, they identified limitations in SMOTE's approach, particularly its tendency to introduce noise and borderline examples, which worsen model performance. Noise arises when examples from one class intrude into another's safe zone, while borderline examples near the class boundary further
complicate accurate classification. Advanced SMOTE (A-SMOTE) was suggested as a solution to these obstacles. This method refines the positioning of synthetic samples by excluding those that are closer to the majority class. The performance of A-SMOTE was superior to that of other oversampling methods when tested on 44 datasets with varying imbalance ratios. This was achieved by more effectively resolving the structural issues of imbalanced data, thereby enhancing the accuracy of the classifier.

A proposed method by Liang et al, (2020) called LR-SMOTE, based on both K-Means and SVM.They introduced LR-SMOTE, an improved oversampling technique that produces new minority class samples that are closer to the center of the minority distribution, thereby reducing outliers and maintaining data integrity. They tested LR-SMOTE on four UCI public datasets and six datasets they

created themselves. The results showed that it did better than regular SMOTE in terms of G-means, F-measure, and AUC values. This means that it might be able to improve the accuracy of classification for data that isn't balanced.

According to Laureano et al. (2019), SMOTE has shortcomings when generating synthetic data, particularly in noisy regions and with imbalanced distributions of minority class samples. The authors propose a new method that employs affinity propagation for clustering, allowing for the generation of clusters and their exemplars without needing to specify the number of clusters in advance. Their proposed AP-SMOTE outperforms both traditional SMOTE and K-means SMOTE on Liver Disease and Yeast datasets, achieving average F-measure and G-mean scores of 0.685 and 0.756, respectively.

The article entitled "Handling Unbalanced Data with Smote Adaboost (Qadrini , 2022). The study focuses on the classification of data using Adaboost and SMOTE Adaboost, a resampling technique that effectively manages unbalanced classes. SMOTE Adaboost was selected for its great accuracy and ability to reduce overfitting. The authors suggest a cost-sensitive algorithm that is based on dynamic sampling and bagging to develop a multi-fusion imbalanced ensemble learning algorithm. BPSO-Adaboost-KNN, an ensemble algorithm, incorporates feature selection and boosting into the ensemble to optimize classification performance by balancing the classes in the dataset. The accuracy, Confusion Matrix (CM), and AUC of the results will be compared to those of classification without resampling. Unbalanced data is a substantial issue in data mining. The objective of the study is to resolve class imbalances in wine quality data, with an emphasis on the minority and majority classes. Performance can be diminished by explicitly applying the classification algorithm to imbalanced datasets.

According to Ratsch, G. et al. (2022), Outliers can greatly affect the overall performance of the model since it confuses the algorithm on focusing more on the weight of the errors on misclassified data rather than the common data which affects the accuracy that the model can give. It has negative and positive effects. The positive effect of outliers is that some of it needs to be tolerated and is beneficial substantially increasing the margins on the remaining point while can greatly affect it with completely removing them.

According to Wang, Jiang, Wen, and Song(2019), AdaBoost is highly effective in classifying security levels in mobile intelligence, achieving 94% accuracy with efficient resource use. It has a lower runtime during classification compared to basic methods. However, the algorithm requires significant computational resources during training due to its focus on misclassified data, which increases the time needed to process the model. Despite this, AdaBoost excels in classification tasks after training, making it a robust choice for complex applications.

According to Bahad & Saxena (2020), the key factors which affect the difference between the actual and predicted values of a model are noise, variance, and bias. Their study builds a model and evaluates AdaBoost and other Gradient Boosting Ensemble Machine Learning Algorithms for the prediction of the disease known as diabetes based on human matrices that are related to human health. In order to reduce the bias associated with the random sampling that was conducted for the data, they utilized the Train-Test Split and K-Fold cross-validation methods to determine what the optimal number of iterations were, and to prepare the training and test data. The candidates for their

prediction classifiers in the study were AdaBoost with Naïve Bayes as the base learner, AdaBoost with Decision Tree as the base learner, and Gradient Boosting. Their experimental results show that Gradient Boosting had better accuracy results compared to the AdaBoost. The results for accuracy and precision of Gradient Boosting and AdaBoost with decision trees as a base learner were nearly identical in some

aspects such as recall and F1-score, whereas AdaBoost with AdaBoost with Naïve Bayes as the base learner fell behind in terms of those factors and in performance.

According to Modarres et al (2020), one of the problems encountered by ensemble learning is overfitting. It is a phenomenon in which the model yields very good results on the training data but a very high error on test data. Choosing the right degree of freedom via cross-validation and regularizations are the two ways to deal with this phenomenon. The likelihood of overfitting is that the criterion of fitting the model is different from the standards used to evaluate it. Unseen data samples should be utilized when testing in order to properly measure the effectiveness of the model. Overfitting happens when the model begins to memorize instead of learning during training. Also, ensemble learning algorithms such as AdaBoost focuses on misclassified samples in each of the iterations, it encounters problems with noisy datasets as it would memorize the noise from the dataset, resulting in noise being present in its training results, leading to overfitting in its learning step. Classifiers with low bias tend to have higher variances, and those with higher bias tend to have lower variances, and the goal of each classifier is to reduce the amount of composition of bias and variance.

According to Wang, R. et al. (2023), despite the promise that machine learning offers to many of the fields in medicine, concerns were raised regarding its potential biases and poor generalization when it came to different variables, such as genders, age, distributions, races and ethnicities, hospitals, and data acquisition equipment and protocols. As machine learning is being applied to different areas to solve the problems in clinical sciences, an increase in the amount of discussion around bias in particular and ethical issues in general have been circulating. A large amount of work has been done in order to identify biases and techniques are under development for their mitigation. Different machine learning literature have well-established safeguards and procedures against poor generalization, which, if disregarded, could lead to more biased predictions. Although balanced datasets are desirable for building unbiased models, trained models may still be biased due to unobserved evaluation factors even for such balanced datasets. In some situations, bias cannot be removed by preprocessing and model selection alone as the source of the bias may be located elsewhere in the algorithm.

According to Belitz & Stackelberg (2021), Ensemble tree machine learning regression models are helpful for understanding and evaluating environmental systems. However, the output from these models can be systematically biased, wherein they overestimate smaller values and larger values are underestimated. The study evaluated five methods for bias-correction, namely the Empirical Distribution Matching (EDM), Regression of observed on estimated (ROE) values, Linear Transfer Function (LTF), Z-score Transform (ZZ), and the use of another Machine Learning model for estimating the residuals (ML2-RES). A sixth method was added, the ROE-Duan, for the evaluation of two case studies that were developed by using log-transformed concentration. The bias-correction methods were calibrated using training data and their performance was assessed with the use of the training and holdout data. Their results show that the EDM method was the most effective, and the point-scale ROE method was the least effective for correcting systematic bias among the 4 case studies evaluated. However, when RMSE-F was computed in retransformed concentration, the best results for the Central Valley Case Study was given by the EDM method, and the ROE-Duan method for the Mississippi Embayment Case Study. The values of the other three metrics varied amongst the different methods. For correcting introduced bias (bias in the mean), the ROE-Duan method was the most effective.

According to Chen, Z. et al (2023), Machine Learning software has spread over into a wide range of critical decision-making applications, such as in the hiring process, criminal justice, credit risk

prediction, and admissions. Many concerns regarding the exhibition of unfair behavior related to protected attributes such as gender and race have been observed. Unfair software behavior may result in unacceptable and unethical consequences that adversely affect minorities and/or historically disadvantaged groups. The increase in the focus for increasing the reliability of machine learning in order to deal with complex decision making and making predictions brings with it the potential to provide unfair results due to software bias. With the emergence of various bias mitigation methods, many have pinned these methods against one another in order to test how these methods deal with different scenarios. Researchers often only use one or two metrics in order to measure the effects of bias and bias mitigation methods on ML performance, which overlooks the other metrics widely used in the industry and academia. Some choose to measure the performance based on accuracy, while others utilize the F1-score. This study evaluated 17 representative bias mitigation methods in 8 widely-adopted benchmark tasks. 11 Machine Learning performance metrics, 4 fairness metrics, and 20 fairness-performance tradeoff measures were also used. Researchers evaluate bias mitigation methods in diverse tasks in order to improve the generalizability of the results. Both traditional Machine Learning algorithms and DNNs for implementation were used, which reduced the influence of model selection on the generalization of the results. Among the bias mitigation methods that were studied, RW (Reweighting) performed the best in retaining ML performance. This is because RW only adjusts the weights of the examples in the training data. It does not modify their features or labels, which avoids the addition of additional noise for the algorithm. RW modifies the weights of different training samples so that it tackles training data problems, which is recognized as one of the root causes of bias in Machine Learning Software.

2.3 Synthesis

The AdaBoost algorithm is widely recognized for its effectiveness in classification tasks, particularly for converting weak learners into a strong ensemble classifier. Adjusting the weights of misclassified samples in each iteration, AdaBoost prioritizes challenging data points, thereby improving overall model accuracy. However, despite its strengths, the algorithm still faces key challenges, such as imbalanced datasets, susceptibility to outliers, and overfitting during training. These limitations significantly affect its performance, especially in critical applications like online transaction fraud detection.

Several studies have highlighted these issues in the AdaBoost algorithm. The imbalanced datasets lead to inaccurate classification results, as the algorithm tends to miscalculate the weight of the minority class (Lv et al., 2019). The outliers present during the training process result in longer training times, as AdaBoost assigns additional weight to misclassified data produced by weak learners (Zhou et al., 2019). Overfitting happens because of biased weight normalization during updates. This makes AdaBoost focus too much on the patterns in the training data, which makes it bad at applying to new data (Wang et al., 2020). These problems hinder AdaBoost's effectiveness in detecting fraudulent transactions and necessitate a more balanced, efficient approach.

To address these problems in the current algorithm, we proposed enhancements to the algorithm. For the imbalance dataset the Synthetic Minority Oversampling Technique (SMOTE) will be used to generate synthetic samples for the minority class, which improves the balance of class distributions and enhances classification accuracy. Quantile-Based Capping can reduce the impact of outliers by limiting extreme values, leading to faster and more efficient training. The use of early stopping mechanisms prevents

overfitting by halting the training process when validation errors increase, allowing the model to better generalize to new data. These improvements make AdaBoost a more reliable tool for fraud detection, enhancing its performance in real-world applications.

2.4 Comparative Analysis

Methods	Accuracy	Speed	Classification Performance
AdaBoost	99.43%	Moderate	99.48%
Naive Bayes	90.93%	Moderate	Not specified
Logistic Regression	95.35%	Fast	Not specified
ANN	94.81%	Slow	Not specified

Table 2.1 AdaBoost Accuracy and Classification Performance Rate

Models such as the AdaBoost algorithm, Naive Bayes algorithm, Logistic Regression algorithm, and Artificial Neural Networks (ANN) algorithm are just a few of the algorithms designed for classification purposes. They are widely used for predictive tasks, where they categorize data into predefined classes based on their features. This leads to valuable classified data for applications such as fraud detection, medical diagnoses, or even email classification.

With the data shown in Table 2.1, these models were used for credit card fraud detection in the study by Gedela & Karthikeyan entitled "Credit Card Fraud Detection using AdaBoost Algorithm in Comparison with Various Machine Learning Algorithms to Measure Accuracy, Sensitivity, Specificity, Precision, and F-score" published in February 2022. These models were evaluated to determine their accuracy, sensitivity, specificity, precision, and F-score to identify the most suitable algorithm for credit card fraud detection. The study involved a dataset of 284,807 transactions, with 492 identified as fraudulent. The dataset was split into a training set (80%) and a test set (20%).

As the test was conducted, it was found that the AdaBoost algorithm obtained the highest average performance with an accuracy of 99.43% and an F-score of 99.48%.

Other algorithms performed as follows: Naive Bayes (90.93%), Logistic Regression (95.35%), ANN (94.81%), and Decision Trees (94.81%).

Metrics	AdaBoost Regression	Linear Regression	Random Forest Regression
MRR (Material Removal Rate)	95.6%	55%	94.8%
Handling Outliers	Residuals between ±0.004, outliers	Not specified	Not specified
Sensitivity to Regressors	Negligible effect with changes	Not specified	Highest R^2 with 200 regressors

Table 2.2 AdaBoost in Handling outliers

Table 2.2 illustrates the comparison between three algorithms used for classification with the study entitled "A Comparative Study of Linear, Random Forest, and AdaBoost Regressions for Modeling Non-Traditional Machining"and showed that AdaBoost Regression is the most effective with achieving high accuracy rate at 95.6% compared to Random Forest Regression at 94.8%. AdaBoost also has better control with handling outliers and its also less sensitive to the number of regressors which makes it more flexible and easier to implement. (Shanmugasundar et al., 2021).

Metric	PSO AdaBoost	ACO XGBoost
Accuracy	79.22	69.0
Precision	73.46	53.38
Recall	65.45	65.45
F1 score	69.23	60.0
AUC score	76.16	68.08

Table 2.3 AdaBoost vs. XGBoost Metrics

Table 2.3 Illustrates about the comparison between two ensemble learning strategies which are Particle Swarm Optimization(PSO) integrated AdaBoost and Ant Colony Optimization (ACO) merged with XGBoost in being able to investigate the early identification of diabetes. This study resulted in having PSO AdaBoost outperform the ACO XGBoost with its overall performance based on the data shown above. (Konda et al. 2023)

CHAPTER THREE DESIGN AND METHODOLOGY

This chapter provides an overview of the tools, methods, and underlying concepts utilized in the study. It aims to guide the understanding of the research approach and evaluate the credibility of the results from the proposed enhancements.

3.1 Research Design

This section shows how the data gathering, data preparation, and data analysis was conducted by the researchers of this study.

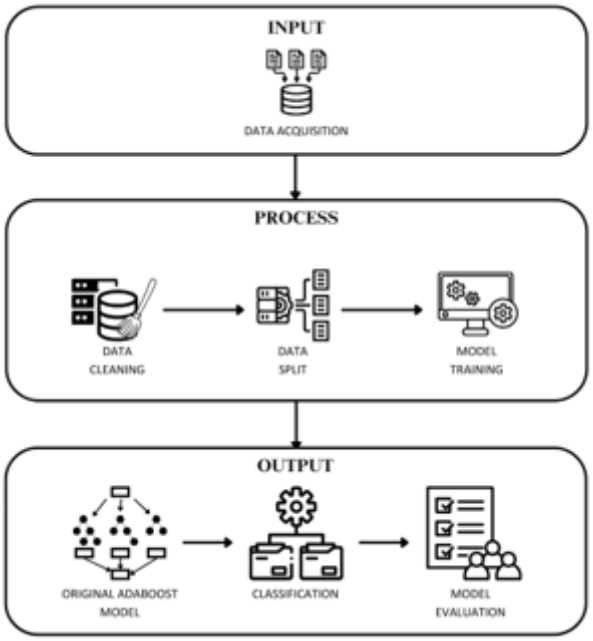

Figure 3.1 Conceptual Framework

Figure 3.1 shows the steps of how the researchers will develop the AdaBoost algorithm. First, the raw data that will be used for the model must be acquired. Next, the acquired data must be cleaned. After cleaning, the data will then be split into two
different sets, namely the training set and the test set. The model will then be trained by using the two datasets. The AdaBoost model will then create different weak classifiers, with each classifier learning

from the previous weak classifiers. After creating a sufficient amount of weak classifiers, the results from all of the created weak classifiers will be combined into a single strong classifier. The performance of the model will then be evaluated by using different metrics based on the results of its output.

3.1.1 Data Acquisition

The data collection strategy for this research incorporates the use of UCI credit card dataset which contains personal attributes like id, available credit balance, gender, level of education, age among other aspects. This data set creates a ground for current and advanced modeling of the technique. Therefore, using this dataset, the researchers utilized the standard method performance, and the performance of the modified algorithm and thus help to understand the efficiency and effectiveness of the modifications brought forth in this work.

3.1.2 Data Cleaning

The UCI credit evaluation dataset was thoroughly checked. Inconsistent data such as missing values, data duplicates, and outliers were removed. Irrelevant features were discarded or changed to make it more clean. Noise was not observed during the inspection. Additionally, categorical variables had to be properly encoded so that the data could be used for some of the latest ensemble models. For numerical log features, if a log transform had been applied then all preprocessing operations could have been executed automatically. Achieving this comprehensive data cleaning was a critical step to increase the efficiency of the improved algorithm and to ensure that more robust and relevant results came from it.

3.1.3 Data Analyzation

This stage of the process entails the assessment of the researchers and evaluation of the data collected in order to achieve accurate and unbiased results. The data was scrutinized in order to determine which features are the most advantageous for the purpose of testing the old and new algorithms. The aim of the researchers is to test the performance of the improvements that were introduced to the algorithm in this paper and to ensure that the findings are well justified by the evidence presented.

3.1.4 Evaluation Matrix

The Traditional and Enhanced Adaboost algorithms are trained using the X_train and Y_train, and the performance of both is assessed using the x_test and y_test. Assessment measures like accuracy, precision, recall, and F1-score are used to compare the two models' performance, where it would compare the predicted labels with the actual labels of the test dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

The model's accuracy is primarily determined by the percentage of precisely predicted instances. It is determined by dividing the total number of instances by the sum of true positive (TP) and true negative (TN) predictions. A high accuracy score suggests that the model is producing a significant number of accurate predictions, whereas a low accuracy score suggests that the model is frequent.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall refers to the percentage of true positive predictions, made from all of the actual positive cases in the considered dataset. Recall is also known as sensitivity or true positive rate and is defined as the ratio of number of True Positives divided by the total number of positive instances, that is, True Positives and

False Negatives combined. A high recall score means that a majority of positive instances are identified, i.e. a majority of true positives are captured. On the other hand, a low recall score implies that many of the positive instances are missed by the model which may lead to the occurrence of false negatives or type II errors, where positive cases are classified as negatives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

The positive predictive value, which is also called the precision, can be defined as the fraction of true positive predictions to the whole number of positive predictions that would have been made by that model. This can be determined by calculating the fraction of true positive results (TP) to total positive predicted cases which is TP plus false negative cases (TP + FP). When a precision score is high, it indicates the model is reliable in determining positive cases as the chances of false positives are slim. On the other hand, a low score is a warning sign indicating that the model produces many type I errors, or false positives, which means predicting a positive outcome, whereas the instance is in fact negative.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score computes the harmonic mean of precision and recall. It is a good performance metric to balance out precision and recall as it takes into consideration the extreme negatives of the two (i.e., a low F1-score if either precision or recall is low). While accuracy is a great measure for evaluating the model's performance, it can be misleading in the presence of a class imbalance. In that sense, other performance metrics such as precision, recall, and F1-score are essential to assessing its overall performance (Walker, 2024).

$$G\text{-}mean = \left(\frac{TP}{TP + FN} \times \frac{TN}{TN + FP} \right)^{\frac{1}{2}}$$

The G mean, short for geometric mean, is a measure of a model performance by considering both the recall and precision of the model. It is very useful when dealing with imbalanced datasets since accuracy may not reflect how good the model is across the classes. The computation of G-Mean is based on the square root of the product of sensitivity which is also called true positive rate and specificity which is also referred to a true negative rate. A high G mean score denotes the capability of the model to accurately predict both positive and negative instances without over predicting one class. Low G mean score on the other hand means that the model is overpredicting one of the classes since the minority class cannot be accurately classified. Due to this, there may be poor classification performance generalization across the different classes.

3.2 Proposed Algorithm

The following is the proposed pseudocode for the AdaBoost algorithm. The proposed pseudocode implements SMOTE, Quantile Based Capping, and Early Stopping to the pseudocode of the AdaBoost Algorithm in order to enhance it.

1. Data Acquisition
2. Data Input
3. Data Split (Training and Testing Data)
4. Model Training
    a. Weight Initialization
    b. Utilize SMOTE to balance the dataset (Objective 1)

        i. Divide data into Minority and Majority Categories
        ii. Compute for the degree of imbalance
        iii. while imbalance ratio =! 1:1
            1. Get a minority instance
            2. Get minority instance 2 nearest neighbors
            3. Create Synthetic data point

    c. Calculate for Interquartile range (Objective 2)
        i. Calculate for the 25th percentile as Q1
        ii. Calculate for the 75th percentile as Q3
        iii. Set data between the range of the 25th and 75th percent as the Interquartile range
        iv. Set lower bounds as data under 25th percentile
        v. Set upper bounds as data under 75th percentile
    d. Quantile based capping checks the values of each feature
        i. if feature < 25th percentile then replace value with lower bound
        ii. if feature > 75th percentile then replace value with upper bound
        iii. else the value remains unchanged

e. Loop for each weak classifier
   i. A weak classifier is trained on the current weighted dataset.
   ii. Computes the error rate based on the predictions of the weak classifier.
   iii. Computes the weight of the weak classifier based on its error rate.
   iv. Updates the weights of the samples.
   v. Calculates a normalization factor and normalizes the weights for the next iteration.
   vi. Monitor performance for Early Stopping: (Objective 3)
      1. Get the parameters from the current loop
      2. Get the performance of the current loop
      3. if performance of current loop > best loop
         a. Save the parameters from the current loop as the best loop
         b. Reset Performance Improvement Counter
      4. else Add to Performance Improvement Counter
   vii. Early Stopping:
      1. if Performance Improvement Counter reaches the maximum set value then Early Stopping terminates the training loop

5. Model Evaluation
   a. Calculate Accuracy with the formula: $\frac{TP+TN}{TP+FP+TN+FN}$
   b. Calculate Recall: $\frac{True\ Positives}{True\ Positives + False\ Negatives}$
   c. Calculate Precision: $\frac{True\ Positives}{True\ Positives + False\ Positives}$
   d. Calculate F1-score: $F1 = 2\ \frac{Precision\ x\ Recall}{Precision + Recall}$
   e. Calculate G-mean: $\left(\frac{TP}{TP+FN} x \frac{TN}{TN+FP}\right)^{\frac{1}{2}}$

3.3 System Requirements

The researchers of this study utilized Python and Visual Studio Code to simulate the algorithm. A platform for coding, testing, and refinement of both AdaBoost and the proposed enhanced version of the AdaBoost algorithm were established due to the flexibility of Python and the integrated development environment provided by Visual Studio Code. Visual Studio Code is an open-source code editor created by Microsoft that supports many programming languages, debugging, and extensions.

The researchers used an 11th-gen Intel i5 processor and 8 GB RAM as it adequately meets the computational capacity for iterative runs of the AdaBoost algorithm. Such specifications can competently handle medium-sized datasets, such as the UCI credit card dataset, without needing high-performance servers or GPUs. A local setup thus provides for effective development, testing, and debugging through Visual Studio Code, offering repertoire oversight over the computational environment. At the same time, such hardware configuration strikes an optimal cost-benefit ratio regarding computational resources consumed in training and testing AdaBoost models.

Python is essential in this study, supporting simulation, data visualization, and system development for online transaction fraud detection. Python libraries used include pandas for data handling, numpy for numerical operations, and scikit-learn modules such as AdaBoostClassifier and DecisionTreeClassifier for classification, along with train_test_split for data partitioning. PCA from sklearn.decomposition is applied for dimensionality reduction, while accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, and log_loss from sklearn.metrics help evaluate model performance. Additionally, Counter from collections aids in counting elements, where from numpy is used for conditional indexing, and metrics from sklearn provides additional evaluation functions. matplotlib.pyplot is used for visualizing the results, and time and operator are employed for performance tracking and optimization.

3.4 Methods and Tools

This section presents the methods and tools that were utilized to enhance the performance of the AdaBoost algorithm by exploring potential improvements and leveraging the capabilities of the model.

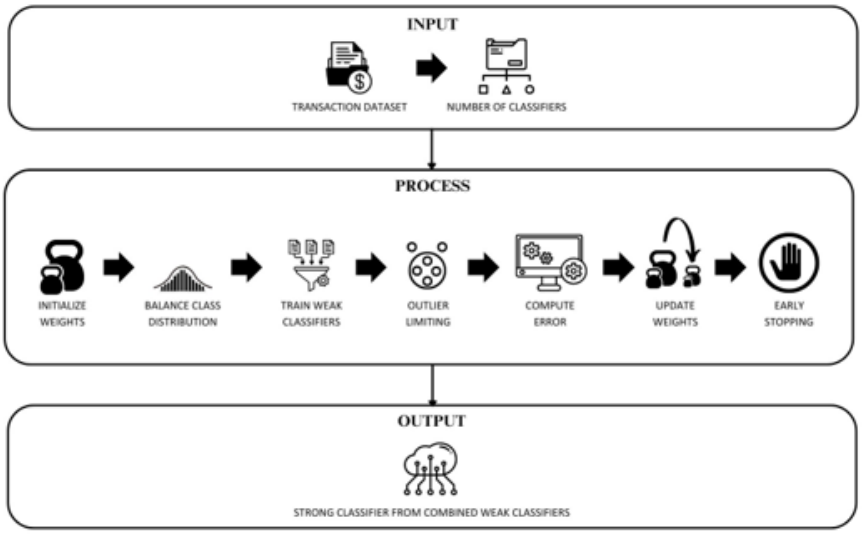

Figure 3.2 IPO Model of the Enhanced AdaBoost Algorithm

Figure 3.2 illustrates the flow of the Enhanced AdaBoost Algorithm, which is designed to further improve the performance of the AdaBoost algorithm.

It begins with the input phase, wherein the dataset is prepared for training and the number of classifiers

is identified. After that, the process phase occurs, initializing the weights of the dataset and balancing the class distribution using SMOTE as it trains the weak classifiers.

During the iteration process of training the data, it includes limiting the occurrence of outliers to reduce longer training times by using quantile-based capping, which adjusts the thresholds while calculating the weights for error and updating the weights of the data.

After normalizing the weights of the data, early stopping is implemented to avoid overfitting. The enhanced process of the model results in a stronger classifier from combined weak classifiers, leading to higher quality and more accurate classification results.

3.4.1 SMOTE (Synthetic Minority Oversampling Technique)

SMOTE (Synthetic Minority Oversampling Technique) addresses class imbalance by generating synthetic samples exclusively for the minority class, enhancing its representation in the dataset without duplicating existing samples. SMOTE works by identifying each minority instance's k-nearest neighbors within the minority class and creating new synthetic points along the lines connecting these neighbors. This process increases the number of minority class samples, providing a more balanced dataset and helping the model learn a more diverse and comprehensive representation of the minority class.

By using synthetic data, SMOTE prevents issues that might occur with simple duplication and enables the model to detect patterns in the minority class more effectively. This balanced approach improves the model's predictive accuracy and fairness, especially in cases where identifying the minority class accurately is essential.

3.4.2 Quantile Based Capping

The Quantile-Based Capping method addresses the longer time in AdaBoost by capping or setting limitations on extreme values within the dataset. In this way, it becomes easier to detect and limit the occurrence of outliers, which cause longer training times during the iterative process of training the data. Quantile-Based Capping works by using the formula lower_bound = Q1 - 1.5 * IQR and upper_bound = Q3 + 1.5 * IQR. Q1, or the first quartile, are the values in the data where 25% of the data falls below, and Q3, or the third quartile, are the values where 75% of the data falls below. The IQR, or interquartile range, is the range between the first and third quartiles. The lower bound value is determined by having data points lower than the lower bound, and the upper bound value is determined by having data points higher than the upper bound.

Eliminating the occurrence of outliers in the dataset will improve the performance of classification during the iterations, resulting in faster training of the data.

3.4.3 Early Stopping

Early stopping is a regularization technique that could be utilized in order to reduce or prevent overfitting. A training set and validation set is monitored during the training process and is halted when certain conditions are met. When the training set begins to improve and the results of the validation set begin to decline, this is a sign that the model is beginning to overfit with the training set during the training process, possibly learning from noise or irrelevant patterns. As such, while the model is in the training process, its performance is evaluated and the parameters from the iteration when it performed best in the training and validation set are saved so that the model can essentially return to that point. If the results of the training set begin to improve and the validation set does not, early stopping will terminate the training process and revert back to the point when the model performed best. At this point of the model, it is possible that it has been stopped from learning any of the noise or irrelevant patterns that may be present in the training set that would otherwise not be present in the validation set, giving

the model more generalizability.

3.4.4 Training and Testing Dataset

The dataset is divided into 80% for training and 20% for testing to effectively build and evaluate the AdaBoost model. The training set, represented by X_train (features) and Y_train (target), is used to train AdaBoost through iterative boosting, where the model learns by focusing more heavily on samples it initially misclassified. This adaptive weighting process allows AdaBoost to become increasingly accurate by combining multiple weak learners to form a robust ensemble. The testing set, represented by X_test (features) and y_test (target), contains unseen data that provides a true assessment of the model's ability to generalize. By comparing AdaBoost's predictions on X_test with the actual target values in y_test, performance metrics such as accuracy,
precision, and recall are calculated. These metrics indicate how well AdaBoost can classify new data, highlighting its effectiveness in managing challenging cases and enhancing predictive performance.

CHAPTER FOUR
RESULTS AND DISCUSSION

This chapter emphasizes the findings of the study and elaborates an in-depth analysis of the data gathered. The presentation of the results will follow a detailed discussion that interprets the implications, significant patterns, and any discrepancies found within the study. The overall performance of the proposed enhancement will be analyzed and compared to the traditional AdaBoost algorithm, delivering a comprehensive understanding of the achieved goals.

4.1 Class Distribution

The dataset is partitioned into training and testing sets, with 80% being used for training and 20% for testing, as displayed in Figure 4.1. It provides reproducibility when random_state=42. The training data is used to train an AdaBoost classifier with 50 estimators, which is eventually utilized to forecast the results of the test dataset.

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Evaluate AdaBoost without SMOTE
adaboost_clf_original = AdaBoostClassifier(n_estimators=50, random_state=42)
adaboost_clf_original.fit(X_train, y_train)
y_pred_original = adaboost_clf_original.predict(X_test)
```

Figure 4.1 Original Adaboost

The SMOTE Heron-Centroid aims to overcome the challenge of class imbalance by employing a unique geometric technique to generate synthetic data for the underrepresented class. To begin with, the algorithm separates the data into two categories. Minority and Majority classes, and also computes the degree of imbalance. The code defines the centroid of such a triangle as a synthetic data point, created using Heron's formula that finds the area of triangles, each formed by one minority instance and its two nearest neighbors. The process of inserting this synthetic instance into the minority class goes on until the desired imbalance ratio of nearly 1:1 is achieved, which signifies a balanced dataset. A balanced dataset is also returned, which seeks to provide an equal representation of the two classes and is likely to improve the model's ability to predict the minority class accurately.

```python
# Heron-Centroid SMOTE Function
def euclidean_distance(array1, array2):
    return np.sqrt(np.sum((array1 - array2) ** 2))

def hercenSMOTE(X, y, minority_class=1, majority_class=0):

    X_minority = X[y == minority_class]
    X_majority = X[y == majority_class]

    imbalanceRatio = len(X_majority) / len(X_minority)

    areas = []
    calculateMinority = X_minority.tolist().copy()

    # Perform Heron-Centroid SMOTE
    while imbalanceRatio > 1.0:
        # Update imbalance ratio
        imbalanceRatio = len(X_majority) / len(X_minority)

        for instance in calculateMinority[::-1]:
            distances = np.linalg.norm(X_minority - instance, axis=1)  # Vectorized distance calculation
            distances[distances == 0] = np.inf  # Ignore zero distance (itself)
            nearest_indices = np.argsort(distances)[:2]  # Get indices of 2 nearest neighbors

            neighbors = X_minority[nearest_indices]
            twoNeighborsDistance = euclidean_distance(neighbors[0], neighbors[1])
            s = (twoNeighborsDistance + distances[nearest_indices[0]] + distances[nearest_indices[1]]) / 2
            area = np.sqrt(s * (s - twoNeighborsDistance) * (s - distances[nearest_indices[0]]) * (s - distances
            areas.append((area, instance, neighbors[0], neighbors[1]))
            calculateMinority.pop()
```

Figure 4.2 Heron-Centroid SMOTE

Figure 4.3 Adaboost and Heron-Centroid SMOTE

Figure 4.3 shows the distribution of a dataset before and after applying Heron-Centroid SMOTE, visualized using AdaBoost classification results. On the left
side, we see the original imbalanced data, where the majority class is densely clustered, while the minority class has far fewer instances. This imbalance can hinder the AdaBoost classifier's ability to accurately learn patterns associated with the minority class. On the right side, after applying Heron-Centroid SMOTE, synthetic samples are added to the minority class, creating a more balanced distribution between the two classes. This balanced dataset enhances the classifier's ability to learn effectively from both classes, potentially improving prediction accuracy for the minority class.

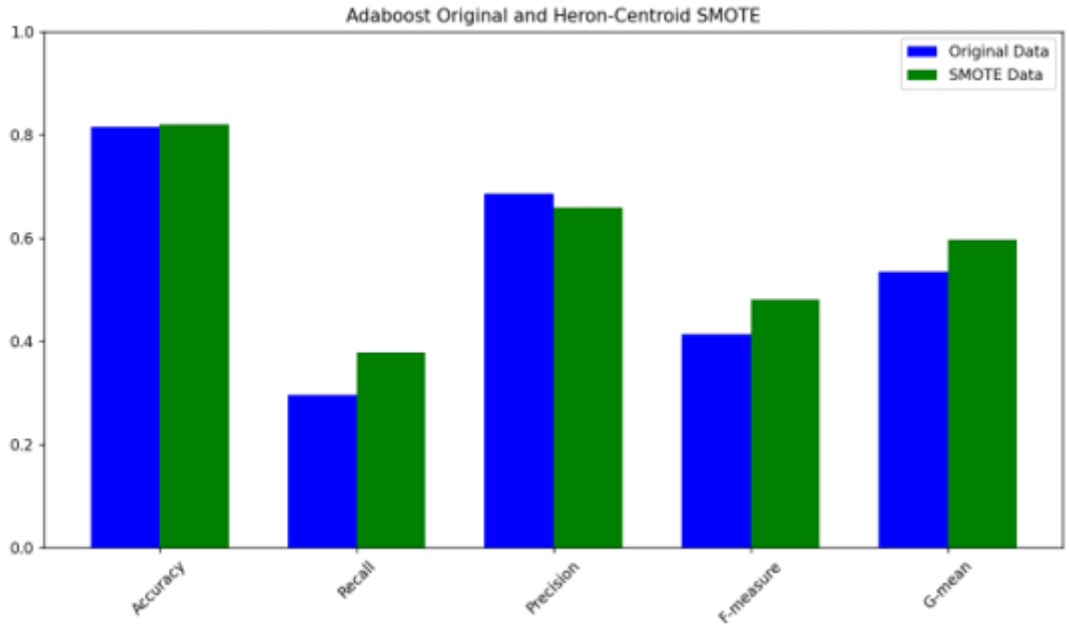

Figure 4.4 Results of Original AdaBoost and AdaBoost with Heron-Centroid SMOTE

The performance metrics of the AdaBoost classifier on the original imbalanced dataset and the dataset balanced with Heron-Centroid SMOTE are compared in Figure 4.4. The results on the original data are shown by the blue bars, while the results after applying SMOTE are represented by the green bars. The values of both datasets are comparable in terms of accuracy, with a slight rise from 0.8150 on the original data to 0.8198 on the SMOTE data. However, recall significantly improved after SMOTE, increasing from 0.2962 to 0.3773, showing a more
effective capacity to identify minority class instances. The use of synthetic samples results in a slight drop in precision, from 0.6848 to 0.6587. Furthermore, the F-measure increases from 0 .4135 to 0.4798, indicating a more advantageous balance between precision and recall. Also, the G-mean, which assesses the balance between recall and specificity, rises from 0.5337 to 0.5970, indicating that the balanced dataset shows improved classification performance.

4.2 Reduced Training Time

The process of enhancing the training time of AdaBoost starts with inputting and reading up the cleaned dataset and defining its features and labels which will help in training the data by learning from its patterns and relationships as well as to identify its target variable. Next step is to split the data into a training set and testing set where in 30% of the data will be used for testing and the remaining 70% will be used for training. This is done in able to monitor the performance of the model.

```
data = pd.read_csv(r"C:\Users\Kiel\Documents\Thesis\ucicleaned.csv")
X = data.iloc[:, :-1]
y = data.iloc[:, -1].values


X_train_raw, X_test_raw, y_train_raw, y_test_raw = train_test_split(X, y, test_size=0.3, random_state=42)
Adatest_raw = AdaBoostClassifier(n_estimators=50, learning_rate=1)
```

Figure 4.5 Data Preparation

```
start_time_raw = time.time()
model_raw = Adatest_raw.fit(X_train_raw, y_train_raw)
end_time_raw = time.time()
training_time_raw = end_time_raw - start_time_raw

y_pred_raw = model_raw.predict(X_test_raw)


Q1 = X.quantile(0.25)
Q3 = X.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR


X_capped = X.apply(lambda x: np.where(x < lower_bound[x.name], lower_bound[x.name],
                                      np.where(x > upper_bound[x.name], upper_bound[x.name], x)))


X_train_capped, X_test_capped, y_train_capped, y_test_capped = train_test_split(X_capped, y, test_size=0.3, r
Adatest_capped = AdaBoostClassifier(n_estimators=50, learning_rate=1)


start_time_capped = time.time()
model_capped = Adatest_capped.fit(X_train_capped, y_train_capped)
end_time_capped = time.time()
training_time_capped = end_time_capped - start_time_capped


y_pred_capped = model_capped.predict(X_test_capped)
```

Figure 4.6 Application of Quantile Based Capping

Quantile Based capping technique aims to help AdaBoost to make training time much faster by having to ease the outliers which initiates the model to focus more on weighted data and cause a longer training time. The process begins in calculating for Q1 and Q3 where Q1 is the 25th percentile and q3 is the 75th percentile. IQR or the Interquartile range serves as the range between Q1 and Q3. Lower and upper bounds are the data that falls under 25 percent and upper 75 percent of the data. Next step is to apply the quantile based capping to each of the features of x. In each value of x it checks if values are below the lower bound or above the upper bound and if they do the values are replaced with the lower and upper bound but if it is not the values remain unchanged. After splitting the data the initialization of the AdaBoost Classifier will begin wherein the n estimators specifies the number of weak learners and learning rate determines the contribution of each of the weak learner.

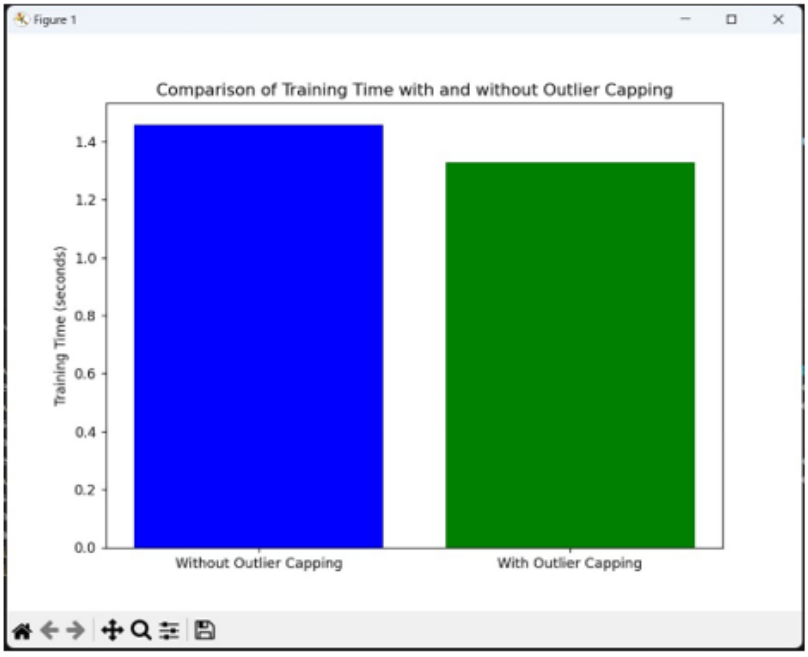

Figure 4.7 Training time without QBC vs Training time with QBC

Figure 4.7 presents a comparison of the training times for the AdaBoost algorithm with and without the implementation of Quantile-Based Capping. The left side of the figure indicates that the training time for AdaBoost without outlier capping is 1.45 seconds while the right side shows a reduced training time which is at rate of 1.32 seconds with the application of Quantile-Based Capping.
These results demonstrate that with the application of Quantile-Based Capping it significantly reduces the training time during the iteration process of AdaBoost algorithm. Reduced training time also enhanced the overall performance of the model which will lead to much accurate and effective classification.

4.3 Improved Test Set Performance

For this portion, the dataset is split into a training set and a test set. 80% of the data is used for the training set and 20% of the data for the test set. The number of estimators for the AdaBoost classifier is set to 50. For early stopping, the algorithm is set to wait for 10 epochs. If there is no improvement while monitoring the loss of the training and test dataset, early stopping will occur.

```
# Get Dataset CSV
data = pd.read_csv("C:\\AdaTesting\\AdaData\\ucicleaned.csv")

X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# define the number estimators
estima = 50

# Early Stopping Variables
improvementCounter = 10 # number of epochs to wait for improvement
```

Figure 4.8 Dataset Preparation and Parameters for AdaBoost and Early Stopping

After preparing the data, we first run AdaBoost as is with the set parameters. In each epoch during the training process, the log loss is monitored, and the accuracy of the training and test dataset are displayed. The test set acts as the validation set for the algorithm during training. AdaBoost will continue learning normally for this part and its results will be saved for comparison later.

```
# Number of Epochs
values = [i for i in range(1, estima+1)] # 50

# evaluate a decision tree for each depth
for i in values:
    dtc = DecisionTreeClassifier(max_depth=i)
    model = AdaBoostClassifier(n_estimators=estima, estimator=dtc, random_state=42)

    # fit model on the training dataset
    model.fit(X_train, y_train)
    # evaluate on the train dataset
    train_yhat = model.predict(X_train)
    train_acc = accuracy_score(y_train, train_yhat)

    train_loss = log_loss(y_train, train_yhat)
    exAdaTrainLoss.append(train_loss*0.1)

    # evaluate on the test dataset
    test_yhat = model.predict(X_test)
    test_acc = accuracy_score(y_test, test_yhat)

    test_loss = log_loss(y_test, test_yhat)
    exAdaTestLoss.append(test_loss*0.1)

    # summarize progress
    print('Epoch %d, train: %.4f, test: %.4f' % (i, train_acc, test_acc))
    exAdascore[0] = model.score(X_test, y_test) # Save Final Results
```

Figure 4.9 Monitoring Log Loss for AdaBoost

As observed in Figure 4.10, the log loss of the training set goes down very quickly, indicating that the performance of AdaBoost with the training set yields very high results. Meanwhile, the log loss of the test set fluctuates while moving upwards, indicating that the performance of AdaBoost with the test set gradually yields worse results as it continues. With the log loss of the training set gradually becoming lower, which indicates high performance, and the log loss of the test set fluctuating towards higher values, which indicates low performance, it can be seen that this model has overfitted.

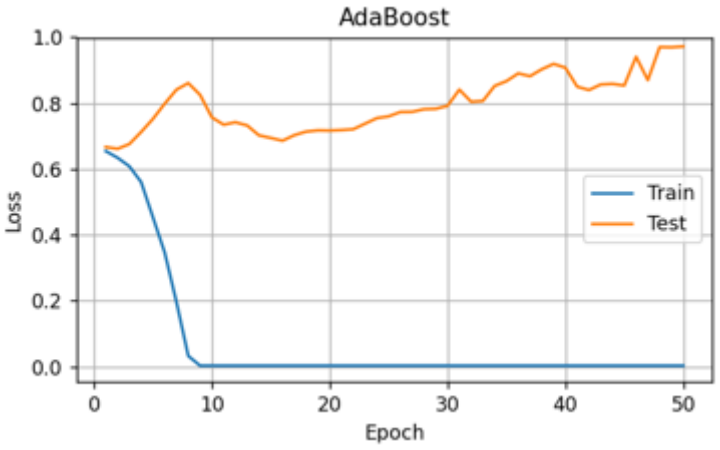

Figure 4.10 Log Loss of AdaBoost

After obtaining the results from AdaBoost, another model is created and trained under the same split dataset and parameters as the previous model, however, early stopping has been implemented for the second model. As set earlier, the algorithm will wait for 10 epochs, observing if there is any improvement or signs of overfitting before it performs early stopping. It will save the conditions and parameters of the epoch which gave the best results before the training process is terminated.

```
# Check if we have a new best score
if val_score > highestValScore:
    highestValScore = val_score
    bestEpo = i
    stopCounter = 0  # Reset counter
    enAdascore[0] = model.score(X_test, y_test) # results of best epoch
    bestRes[0] = enAdaTrainLoss[i-1] # train loss of best epoch
    bestRes[1] = enAdaTestLoss[i-1] # test loss of best epoch
else:
    stopCounter += 1

# Check for Early Stopping
if stopCounter >= improvementCounter:
    print("\n")
    print(f"Early stopping triggered at Epoch {i}.")
    break
```

Figure 4.11 Application of Early Stopping

In Figure 4.12, it shows that the AdaBoost algorithm has triggered the conditions set for early stopping. The epoch that performed the best during this training process is epoch 2. The algorithm waited 10 epochs for any improvements in the performance of the model. After 10 epochs, there was no improvement in its performance, and early stopping is then triggered on epoch 12, terminating the training process.

```
Early stopping triggered at Epoch 12.
Best Validation Score: 0.8165
Can be found before epoch 12.
Best Epoch: 2
```

Figure 4.12 Early Stopping Report

With that, the results of AdaBoost with early stopping can be seen on Figure 4.13, wherein after epoch 2, the log loss of the training and test set begin to diverge. The log loss of the training set begins to decrease, which indicates better performance, and the log loss of the test set begins to increase, indicating a decrease in performance. Beyond epoch 2, the model no longer shows signs of improvement in its performance for the validation set, indicating that overfitting has begun to occur.

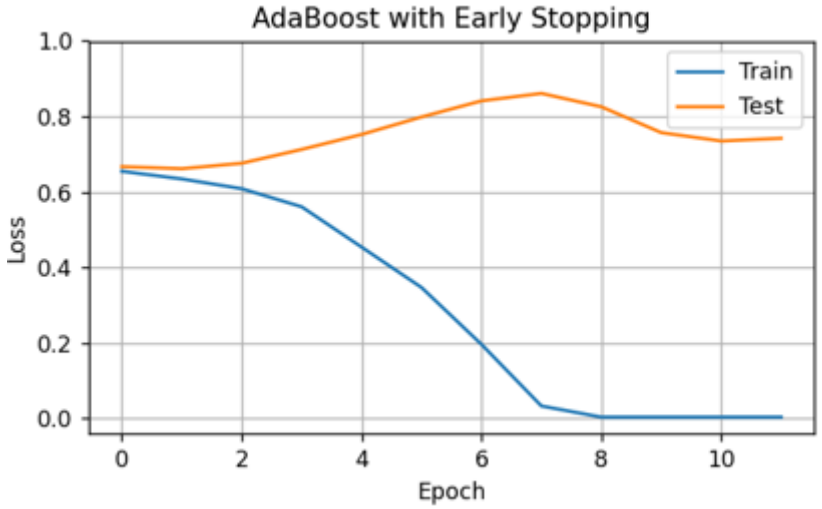

Figure 4.13 Log Loss of AdaBoost with Early Stopping

In Figure 4.14, we compare the performance between AdaBoost and AdaBoost with early stopping by

their end results. With AdaBoost, its output after its training was concluded yielded a lower performance compared to AdaBoost with the implementation of early stopping. This aligns with the presented log loss in the training and test data of AdaBoost in Figure 4.10 as it did overfit with the training set, leading to an accuracy of
0.73 after its training was concluded. Meanwhile, as presented in Figure 4.12, the algorithm triggered early stopping at epoch 2, stopping it before the model could overfit further. This reduced the chance for the model to overfit, allowing it to gain an accuracy of 0.816 after its training was terminated early.

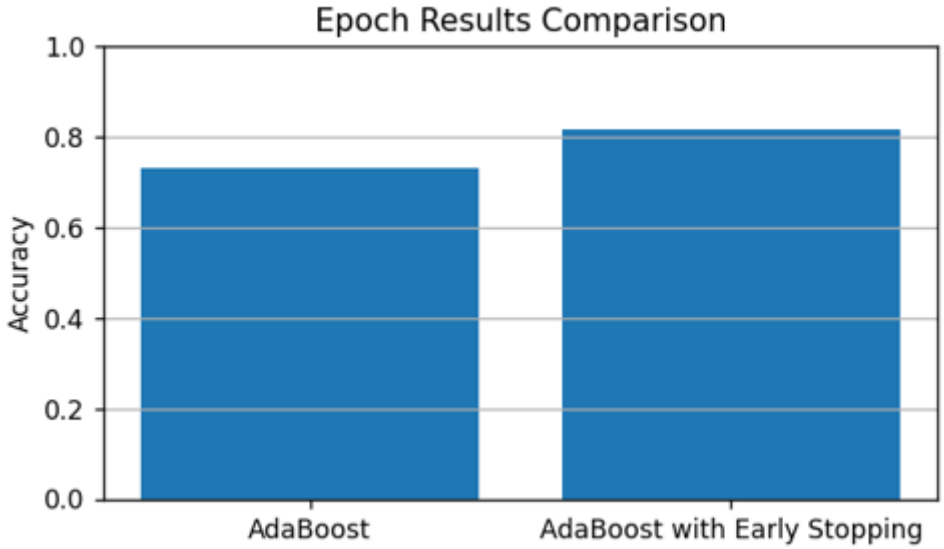

Figure 4.14 Comparison between an AdaBoost Model and an AdaBoost Model with Early Stopping

With these results in mind, it is evident that overfitting has a significant effect on the model should it be left unattended. AdaBoost on its own is still prone to overfitting. After early stopping was implemented onto AdaBoost, it was able to stop its training process early before it could overfit, increasing its performance by 11.81% or by preventing it from performing 10.566% less.

CHAPTER FIVE CONCLUSIONS & RECOMMENDATIONS

This section discusses the conclusions gained from the overall performance of the chosen algorithm as well as the recommendations for further implementation.

5.1 Conclusion

The use of SMOTE to address imbalanced dataset has proven to significantly enhance the overall accuracy of AdaBoost algorithm by balancing the dataset before training. This approach effectively reduces biases and leads to a better handling of relationships and patterns within the data. SMOTE ensures that there will be no bias when it comes to the majority class which leads to a much reliable result. A balanced dataset will result in the model's ability to detect and classify minority class instances which will enhance the models performance.

Using the method Quantile Based Caping, it was effectively applied to address longer training times during the iteration of training the data in the AdaBoost model. This statistical technique mitigates the influence of outliers by capping extreme values at a defined quantile or the values that are out of the range of quantile q and 3. By reducing the impact of outliers, it helps to streamline the dataset into minimizing the error result and excessive processing time.

The use of early stopping in the AdaBoost model has proven to be effective in addressing overfitting by stopping the training process before the model becomes too focused on the training data. Overfitting happens when AdaBoost continues to focus on reducing errors in the training dataset, often leading to difficulties in generalizing to new, unseen data. Early stopping mitigates this issue by monitoring performance metrics, such as validation error, and training is terminated when its performance no longer shows
signs of improvement. This ensures that the model can maintain its ability to accurately classify new data while preventing it from becoming too focused on the training dataset.

Hence, the three methods introduced to enhance the AdaBoost algorithm addressing imbalanced datasets, reducing long training times, and mitigating overfitting have proven effective in overcoming these specific limitations. By tackling imbalanced datasets, the algorithm ensures better representation and classification of minority classes. Strategies to reduce training time optimize computational efficiency without compromising model quality, while methods like early stopping solves for overfitting, preserving the model's generalization ability. The approach concludes that these enhancements have significantly improved the overall accuracy, efficiency, and effectiveness of the AdaBoost model's performance.

5.1 Recommendations

Further research should be done to analyze how each implemented modification (SMOTE, Quantile Based Capping, Early Stopping) has affected the AdaBoost algorithm and in what specific cases would each of the modifications be utilized for such to significantly improve the algorithm. Other modern techniques or methods may also be added alongside the modifications in order to compare what combination of enhancements would significantly increase the performance of the AdaBoost algorithm.

Further research should also be made to examine how vulnerable the AdaBoost algorithm truly is towards overfitting. Should there be findings of vulnerabilities towards overfitting, what would the different factors be that cause the AdaBoost algorithm to have such vulnerabilities. Different methods that prevent or reduce the risk of overfitting, such as other regularization techniques, should also be tested further in order to determine the effectiveness of each towards the possible vulnerability of the AdaBoost algorithm to overfitting.

LIST OF REFERENCES

1. Ahmed, I. (2023, May 22). How to fix imbalanced data (with python code for SMOTE). Retrieved from
2. https://ilyasbinsalih.medium.com/how-to-fix-imbalanced-data-60b905afc56c Alma Better. (2023, October 10). AdaBoost algorithm in machine learning [Bytes].
3. https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm?fbclid=IwAR27K9Cj0F5I7p2Kkd2eq1_F2Oj3fa17LLs4-mqXgDiVOAvgtCErY9eaUk chttps://medium.com/@nerminhafeez2002/mastering-adaboost-algorithm-a97e7d 70e151.
4. B, A. (2023c, April 12). *Adaptive Boosting (AdaBoost) Technique in Decision trees*. https://www.linkedin.com/pulse/adaptive-boosting-adaboost-technique-decision-tr ees-avinash-b/.
5. Bahad, P., Saxena, P. (2020). Study of AdaBoost and Gradient Boosting Algorithms for Predictive Analytics. *International Conference on Intelligent Computing and Smart Communication 2019* Retrieved from https://link.springer.com/chapter/10.1007/978-981-15-0633-8_22.

6. Belitz, K., & Stackelberg, P. E. (2021, May). Evaluation of six methods for correcting bias in estimates from ensemble tree machine learning regression models. *Environmental Modelling & Software, 139, 105006*. Retrieved from https://www.sciencedirect.com/science/article/pii/S1364815221000499.
7. Borketey, B. (2024). Real-Time fraud detection using machine learning. *Journal of Data Analysis and Information Processing, 12*(02), 189–209. https://doi.org/10.4236/jdaip.2024.122011.
8. Cai, Y., Wang, Z., Yao, L., Lin, T., & Zhang, J. (2022, September 21). Ensemble Dilated Convolutional Neural Network and Its Application in Rotating Machinery Fault Diagnosis. Computational Intelligence and Neuroscience, Vol. 2022, 6316140. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1155/2022/6316140.
9. Chen, Z., Zhang, J. M., Sarro, F., & Harman, M. (2023, May 27). A Comprehensive Empirical Study of Bias Mitigation Methods for Machine Learning Classifiers. *ACM Transactions on Software Engineering and Methodology, Vol. 32, No. 4, Article 106*. https://dl.acm.org/doi/full/10.1145/3583561.
10. Data Science Wizards (2023, July 8). Understanding the AdaBoost Algorithm. *Medium*. https://medium.com/@datasciencewizards/understanding-the-adaboost-algorithm-
11. 2e9344d83d9b.
12. Desarda, A. (2023, February 28). Understanding the AdaBoost algorithm. *Built In*. https://builtin.com/machine-learning/adaboost.
13. Dhote, P. (2020, August 5). AdaBoost, gradient boosting algorithm, XGBoost ensemble model. *Medium*.
14. https://pradeep-dhote9.medium.com/adaboost-gradient-boosting-algorithm-xgboo st-ensemble-model-361ede2d442c.
15. Ding, Y., Zhu, H., Chen, R., & Li, R. (2022, June 9). An efficient AdaBoost algorithm with the multiple thresholds classification. *Applied Sciences, 12*(12), 5872. https://doi.org/10.3390/app12125872
16. Ekanayake, I., Meddage, D., & Rathnayake, U. (2022c). A novel approach to explain the black-box nature of machine learning in compressive strength predictions of concrete using Shapley additive explanations (SHAP). *Case Studies in Construction Materials, 16*, e01059. https://doi.org/10.1016/j.cscm.2022.e01059.
17. Gajendra, K. S. (2022, July 28). AdaBoost classifier. *Medium*. https://medium.com/@gajendra.k.s/adaboost-classifier-e43bc88ecc07.
18. GeeksforGeeks. (2024, June 03). *Implementing the AdaBoost algorithm from scratch*. https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratc h/.
19. Hao, L., & Huang, G. (2023, March). An improved AdaBoost algorithm for identification of lung cancer based on electronic nose. Retrieved from https://www.cell.com/heliyon/fulltext/S2405-8440(23)00840-X?uuid=uuid%3A58 67a349-3a02-44f7-8e97-e1067a381073.
20. Hussein, A. S., Li, T., Yohannese, C. W., & Bashir, K. (2019, November 9). A-SMOTE: A new preprocessing approach for highly imbalanced datasets by improving SMOTE. *International Journal of Computational Intelligence Systems, 12*(2), 1412–1422. https://link.springer.com/article/10.2991/ijcis.d.191114.002#preview.
21. Ibm. (2022). Calculating thresholds. *InfoSphere Master Data Management 11.6.0*.

https://www.ibm.com/docs/en/imdm/11.6?topic=thresholds-calculating&fbclid=I
22. wAR1FzzAR-6vDULgvZ91fm52M6IGsiTWkWqZSJX_OfVyPl_uPfYkUL30L56 I.
23. Ibragimov, B., & Gusev, G. (2024, August 24). Learn Together Stop Apart: An Inclusive Approach to Ensemble Pruning. Association for Computing Machinery. Retrieved from https://dl.acm.org/doi/10.1145/3637528.3672018.
24. Ileberi, E., Sun, Y., & Wang, Z. (2021, December 22). Performance evaluation of machine learning methods for credit card fraud detection using SMOTE and AdaBoost. *IEEE Journals & Magazine | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9651991.
25. Jiang, X., Xu, Y., Ke, W., Zhang, Y., Zhu, Q. X., & He, Y. L. (2022). An imbalanced multifault diagnosis method based on bias weights AdaBoost. *IEEE Journals & Magazine | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9712167.
26. Khan Academy.(n.d.). Interquartile range (IQR). Retrieved from: https://www.khanacademy.org/math/cc-sixth-grade-math/cc-6th-data-statistics/cc-6th/v/calculating-interquartile-range-iqr?fbclid=IwAR2NjPscMAUfhRVMSjolf7G7y87aQEmEQc6bXbOcgOO3bXbJniRbpzz_TeU#:~:text=The%20 IQR%20describes%20the%20middle,difference%20between%20Q3%20and%20Q1.
27. Konda, S., Goswami, C., Somasekar, J., Ramana, K., Yajjala, R., & Tirumanadham, N. S.
28. K. M. K. (2024, January 1). Optimizing diabetes prediction: A comparative analysis of ensemble machine learning models with PSO-AdaBoost and ACO-XGBoost. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/document/10370452.
29. Liang, X. W., Jiang, A. P., Li, T., Xue, Y. Y., & Wang, G. T. (2020, May 21). LR-SMOTE — An improved unbalanced data set oversampling based on K-means and SVM.
30. *Knowledge-Based Systems*, *196*, 105845. https://doi.org/10.1016/j.knosys.2020.105845.
31. Lv, M., Ren, Y., & Chen, Y. (2019). Research on imbalanced data : based on SMOTE-AdaBoost algorithm. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9094859.
32. Maydanchi, M., Ziaei, A., Basiri, M., Norouzi Azad, A., Pouya, S., & Ziaei, M. (2023, May 8). Comparative study of decision tree, AdaBoost, random forest, naïve Bayes, KNN, and perceptron for heart disease prediction. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/document/10115189/metrics.
33. Meng, D., & Li, Y. (2022, May). An imbalanced learning method by combining SMOTE with Center Offset Factor. *Applied Soft Computing, 120, 108618*. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S1568494622001156.
34. Misra, S., & Li, H. (2020). Noninvasive fracture characterization based on the classification of sonic wave travel times. *Machine Learning for Subsurface Characterization, 243–287*.Retrieved from https://www.sciencedirect.com/science/article/abs/pii/B9780128177365000090.
35. Modarres, Z. G., Shabankhah, M., & Kamandi, A. (2020, February 21). Making AdaBoost Less Prone to Overfitting on Noisy Datasets. *2020 6th International Conference on Web Research (ICWR)*. Retrieved from https://ieeexplore.ieee.org/abstract/document/9122292.
36. Ning, W., Chen, S., Qiang, F., Tang, H., & Jie, S. (2023). A Credit Card Fraud Model Prediction

Method Based on Penalty Factor Optimization AWTadaboost. Retrieved from
37. https://cdn.techscience.cn/files/cmc/2023/TSP_CMC-74-3/TSP_CMC_35558/TSP_CMC_35558.pdf.
38. Nithin, B., Rohit, R., & Sulthana, S. G. (2020, April). Credit Card Fraud Detection Using Adaboost. *International Journal of Scientific Research & Engineering Trends Volume 6, Issue2*.
39. https://ijsret.com/wp-content/uploads/2020/03/IJSRET_V6_issue2_198.pdf.
40. Obaido, G., Ogbuokiri, B., Swart, T. G., Ayawei, N., Kasongo, S. M., Aruleba, K., Mienye, I. D., Aruleba, I., Chukwu, W., Osaye, F., Egbelowo, O. F., Simphiwe, S., & Esenogho, E. (2022). An Interpretable Machine learning approach for Hepatitis B diagnosis. *Applied Sciences, 12(21), 11127.*
41. https://doi.org/10.3390/app122111127.
42. Pan, T., Zhao, J., Wu, W., & Yang, J. (2020b). Learning imbalanced datasets based on SMOTE and Gaussian distribution. *Information Sciences, 512*, 1214–1233. https://doi.org/10.1016/j.ins.2019.10.048.
43. Quadrini, L. (2022). Handling Unbalanced Data With Smote Adaboost. *Jurnal Mantik, 6(2) (2022) 2332-2336.*
44. https://iocscience.org/ejournal/index.php/mantik/article/view/2597/2219?fbclid=IwY2xjawFHdGBleHRuA2FlbQIxMAABHZYoWNq25s4ZZCG0i2-mApT7FH_4
45. FtL4FKA8Yv29nG7D7SAKdMFGMRwSIQ_aem_mNUjvv4ESBfF2Oc70oiRnQ
46. .Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018, March 28). Credit card fraud detection using AdaBoost and majority voting. *IEEE Journals & Magazine | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/8292883
47. Ratscht, G., Schölkopf, B., Smola, A., Müller, K.-R., Onoda, T., & Mikat, S. (n.d.). ν-Arc: Ensemble learning in the presence of outliers. *Proceedings of the Neural Information Processing Systems (NIPS)*. https://arxiv.org/pdf/2206.05379.
48. Sailusha, R., Gnaneswar, V., Ramesh, R., & Rao, G. R. (2020, May 1). Credit card fraud detection using machine learning. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9121114.
49. Santos, L. M., & Sison, A. M. (2019). Handling imbalanced data through affinity propagation and SMOTE. In *Proceedings of the 2019 International Conference on Data Science and Advanced Analytics (DSAA) (pp. 205-214).* ACM. https://doi.org/10.1145/3366650.3366665.
50. Shani, A. (2021, September 9). AdaBoost algorithm: Understand, implement and master AdaBoost.*Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/.
51. Shanmugasundar, G., Vanitha, M., Čep, R., Kumar, V., Kalita, K., & Ramachandran, M. (2021, November 11). A comparative study of linear, random forest and AdaBoost regressions for modeling non-traditional machining. *MDPI*. Retrieved from https://www.mdpi.com/2227-9717/9/11/2015.
52. Sun, J., Wang, G., He, G., Pu, D., Jiang, W., Li, T., & Niu, X. (2020, February 7). Study on the water body extraction using gf-1 data based on AdaBoost integrated learning algorithm. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.Retrieved from https://isprs-archives.copernicus.org/articles/XLII-3-W10/641/2020/isprs-archives

53. -XLII-3-W10-641-2020.pdf
54. Thanathamathee, P., & Lursinsap, C. (2013, September 1). Handling imbalanced data sets with synthetic boundary data generation using bootstrap re-sampling and AdaBoost techniques. *Pattern Recognition Letters*, *34*(12), 1339–1347. https://doi.org/10.1016/j.patrec.2013.04.019.
55. Walker, S. M. II (2024). F-Score: What are Accuracy, Precision, Recall, and F1 Score?
56. *KLU*. https://klu.ai/glossary/accuracy-precision-recall-f1.
57. Wang, F., Jiang, D., Wen, H., & Song, H. (2019, July 20). Adaboost-based security level classification of mobile intelligent terminals. *IEEE Access*. https://doi.org/10.1109/ACCESS.2019.2928938**.**
58. Wang, R., Chaudhari, P., & Davatzikos, C. (2023, January 30). Bias in machine learning models can be significantly mitigated by careful training: Evidence from neuroimaging studies. Retrieved from https://www.pnas.org/doi/10.1073/pnas.2211613120.
59. Wang, W., & Sun, D. (2021, July). The improved AdaBoost algorithms for imbalanced data classification. *Information Sciences*, *563*, 358–374. https://doi.org/10.1016/j.ins.2021.03.042.
60. Xu, Z., Shen, D., Nie, T., Kou, Y., Yin, N., & Han, X. (2021, September). A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data. *Information Sciences*, *572*, 574–589. https://doi.org/10.1016/j.ins.2021.02.056.
61. Zhou, H., Wei, L., Chen, G., Lin, P., & Lin, Y. (2019, December). Credit card fraud
62. identification based on principal component analysis and improved AdaBoost algorithm. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9051207.