IMPROVED ADAPTIVE BOOSTING IN HETEROGENEOUS ENSEMBLES FOR OUTLIER DETECTION: PRIORITIZING MINIMIZATION OF BIAS, VARIANCE AND ORDER OF BASE LEARNERS

JOASH KIPROTICH BII

DOCTOR OF PHILOSOPHY

(Computer Science)

JOMO KENYATTA UNIVERSITY

OF

AGRICULTURE AND TECHNOLOGY

2023

Improved Adaptive Boosting in Heterogeneous Ensembles for Outlier Detection: Prioritizing Minimization of Bias, Variance and Order of Base Learners

Joash Kiprotich Bii

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science of the Jomo Kenyatta University of Agriculture and Technology

2023

DECLARATION

This thesis is my original work and has not been presented for a degree in any other university.

Signature ……. Date ……..

Joash Kiprotich Bii

This thesis has been submitted for examination with our approval as university supervisors

Signature ……. Date ……..

Dr. Richard Rimiru, PhD

JKUAT, Kenya

Signature ……. Date ……..

Prof. Waweru Ronald Mwangi, PhD

JKUAT, Kenya

DEDICATION

To my dear parents, Rev. Dr. Stephen K. Ngenoh and Mrs. Alice Ngenoh, my father and mother in law Dr. Waititu and Mrs. Waititu, my dear wife Phyllis Bii and my children Abigael Chepkemoi and Mark Kipkorir. You are my source of strength and joy in life. God bless you ALL.

ACKNOWLEDGEMENT

My appreciation goes to the management of the Jomo Kenyatta University of Agriculture and Technology (JKUAT) for according me an opportunity to study my PhD in their esteemed institution of higher learning. Special thanks to my supervisors Dr. Richard Rimiru and Prof. Waweru Mwangi who guided me throughout the programme. Their high degree of professionalism and commitment enabled me complete the study within the acceptable period. My other gratitude goes to Dr. Lawrence Nderu, Chairman of Computing Department, and Dr. Rimiru who inspired and mentored me in the field of Data Mining, Prof. Waweru in his guidance and support throughout the journals and publications. My sincere gratitude goes to the director of the School of Computing and Information Technology (SCIT), Dr. Michael Kimwele, and including Prof. Stephen Kimani, Dr. Kaburu, Dr. Petronilla Muriithi, Dr. Ann Kibe, Dr. Musau, Prof. George Okeyo, Dr. Agnes Mindila, Prof. Wilson Cheruiyot, Prof. Wafula, Dr. Ogada Kennedy, Dr. Tobias Mwalili and other staff of JKUAT who actively participated in providing feedback, comments and guidance during the coursework, project thesis and seminar presentations. May God Bless You All.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF APPENDICES

LIST OF ABBREVIATIONS

Acc	Accuracy Rate
ANN	Artificial Neural Network
AUC	Area Under the Curve
AUROC	Area Under Receiver Operating Characteristic
CNN	Convolutional Neural Networks
CRISP-DM	Cross-Industry Process for Data Mining Methodology
DT	Decision Tree
Err	Error Rate
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
K-NN	K -Nearest Neighbours
LOF	Local Outlier Factor
LOoP	Local Outlier Probability
LR	Linear Regression
ML	Machine Learning
NB	Naive Bayes Classifier
ODDS	Outlier Detection Datasets

P	Precision
PCA	Principle Component Analysis
R	Recall
RNN	Recurrent Neural Networks
ROC	Receiver Operating Curve
Spe	Specificity
SVM	Support vector machine
TN	True Negative
TP	True Positive

ABSTRACT

Real-world data suffer from corruption caused by human errors, for instance, rounding errors, wrong measurements, biases, faults, or rare events, including malicious activities like credit card fraud or cyber activities that cause unusual patterns or outliers in data. The detection of outliers is a difficult task that requires complex ensemble models. The ideal outlier detection ensemble should assess the strengths and optimize the results of its base detectors while carefully combining their outputs to create a robust overall model and achieve unbiased accuracy with minimal variance. Existing outlier detection ensembles fuse numerous detectors (weak learners) in either parallel or sequential order to increase detection accuracy by obtaining a combined result through a majority vote. However, trusting the results of all weak learners may deteriorate overall ensemble performance as some learners may produce erroneous results depending on the types of data and their underlying rules. The general objective was to develop an outlier detection model by integrating multiple yet different (heterogeneous) base detectors into one model (ensemble), by first selecting highly accurate base detectors through training and evaluating every model by their error rates, and then implementing the adaptive boosting technique, where misclassified samples got to be feedback for the next detector (to minimize bias), then strategically combining all their decisions (to minimize variance), in order to obtain a strong detector by a combination function. The research's specific objectives were: identifying weak learners by analyzing their initial biases and variances, analyzing fusion strategies, developing and evaluating an outlier detection model with a focus on minimizing bias, variance, and order of base learners. The CRISP-DM methodology was employed. Outlier datasets were drawn from ODDS library. The model was validated against four other baselines, and test results were compared using performance measures such as Recall, Precision, ROC and AUC values. The experiments showed improvement in results in at least 8 out of ten datasets in terms of average AUCROC even when the least of outliers (single cases up to 10%) were used.

Keywords: Outliers, Weak learners, Ensembles, Bias, Variance

CHAPTER ONE

INTRODUCTION

1.1 Motivation

Outlier detection, also known as anomaly detection, is a critical task in various domains such as finance, cyber security, and fraud detection (Varun & Bhatia, 2020). The goal is to identify data points that deviate significantly from the normal behavior of the majority of the data, which could indicate potential anomalies or abnormalities that require further investigation. Traditional approaches for outlier detection often rely on statistical methods or rule-based techniques, but these methods may have limitations in handling complex and high-dimensional data.

In recent years, deep learning techniques have gained significant attention in outlier detection due to their ability to automatically learn complex patterns and representations from raw data (Li et al., 2021). Deep learning techniques, such as autoencoders, recurrent neural networks (RNNs), and convolutional neural networks (CNNs), have shown promising results in outlier detection tasks. These methods have been able to capture intricate patterns and representations from the data, enabling them to detect outliers with high accuracy (Buda et al., 2018). However, despite the success of deep learning techniques, they still have some limitations. Deep learning models are known to be data-hungry and require large amounts of labeled data for training, which may not always be available in outlier detection scenarios, especially when dealing with rare or novel anomalies (Gupta et al., 2019). Moreover, they are typically black-box models, making it difficult to interpret and explain their decisions, which may be a concern in applications where interpretability is crucial, such as in finance or healthcare (Lipton, 2018). Furthermore, training deep learning models can be computationally expensive, and they are prone to overfitting, which may not be feasible in resource-constrained environments.

In contrast, traditional machine learning algorithms like k-nearest neighbors and decision trees are more suitable for resource-constrained environments since they require fewer computational resources and are less prone to overfitting.

Moreover, heterogeneous ensembles, which combine multiple diverse models or algorithms, have emerged as a promising approach for outlier detection (Chen et al., 2019). Heterogeneous ensembles leverage the strengths of different models or algorithms, allowing them to compensate for each other's weaknesses and improve overall performance. For example, an ensemble may combine a decision tree-based method with a deep learning-based method, or a clustering-based method with an instance-based method. By integrating diverse models or algorithms, heterogeneous ensembles can enhance outlier detection performance, even when limited labeled data or resources are available, and provide interpretability through model diversity (Huang et al., 2019).

Several studies have demonstrated the effectiveness of heterogeneous ensembles in outlier detection. For instance, Liu et al. (2018) proposed an ensemble approach combining multiple outlier detection algorithms, including clustering-based, density-based, and distance-based methods, to achieve better performance compared to individual methods. Chen et al. (2019) developed a heterogeneous ensemble approach that combined autoencoders with one-class SVMs to detect anomalies in network traffic data, achieving higher accuracy and interpretability compared to standalone models. Xu et al. (2020) proposed a hybrid ensemble approach that combined deep learning-based autoencoders with clustering-based KNNs for detecting fraud in credit card transactions, demonstrating superior performance in terms of accuracy and robustness compared to single models.

In light of these, while deep learning techniques have shown promising results in outlier detection tasks, they still have limitations in terms of data requirements, interpretability, and computational efficiency. Heterogeneous ensembles, on the other hand, offer a viable alternative by combining diverse models or algorithms to compensate for weaknesses and enhance performance, even in challenging scenarios with limited data or resources. By leveraging the strengths of different methods, heterogeneous ensembles can provide improved outlier detection accuracy, interpretability, and robustness, making them a compelling choice for outlier detection tasks.

1.2 Background of the Study

Machine learning (ML) refers to the process of training computer systems to learn and improve their performance on specific tasks by utilizing algorithms and statistical models (Murty et al., 2021). These algorithms are designed to learn from data and make predictions or decisions based on that data. ML algorithms are commonly used in classification, clustering, association rule mining, and other fields that require pattern recognition (Alpaydin, 2020). ML algorithms are also widely used in outlier detection, which involves the identification of observations that deviate from expected patterns or behaviors (Barnett & Lewis, 2021). As the field of ML continues to evolve, new algorithms and techniques are being developed to improve performance and enable new applications across various industries.

In data analysis tasks, one of the first steps towards obtaining a clear study is the detection of outlying observations, that is, observations that don't fit a clear definition of what constitutes typical behaviour or what constitutes outlier behaviour (Aggarwal, 2021). In a variety of application disciplines, these observations or behaviours are frequently referred to as outliers, contaminants, anomalies, discordants, faults, oddities, defects, aberrations, noise, errors, damages, unexpected, or quirks (Chen, Du, & B, 2020). Although outliers are often considered unusual, errors or noise, they may carry important information. Sometimes if left undetected, outliers may lead to model misspecification and incorrect results. Therefore, it is critical to recognize them before modeling and analysis (Prasada et al., 2020).

Several definitions for outliers have been put forward. According to Grubbs, outlier detection is the process of identifying patterns in data that differ from what would be considered normal behaviour (Grubbs, 1969). According to Hawkins, an outlier is an observation that deviates so much from other observations that it raises the possibility that it was generated by a different mechanism (Hawkins, 1980). According to Johnson, it is an observation in a data collection that seems discordant with the rest of the data (Johnson, 2022). Similarly, Barnett and Lewis (Barnett & Lewis, 2021) indicate that an observation that appears to differ significantly from other members of the sample in which it occurs is referred to as an outlier.

The process of detecting outliers in datasets dates back to the 18th century. In 1850, the first statistical method was created (Beckman & Cook, 1983) to address the problem of outliers in the data. There have been arguments on whether outliers should be kept as part of data as they provide very useful information about the data. For instance, Barnett (Barnett & Lewis., 2021) says that one should not delete extreme observations just due to their gap from the remaining data. Prior, other researchers favored cleaning the data from outliers as they distorted the estimates. Cousineau and Chartier (2010) claims that outliers are always the result of some spurious activity and should be deleted. Whether to delete or keep the outliers in the data may depend on the domain of use of that data.

Numerous applications, such as the identification of credit card fraud, have called for the use of outlier detection techniques (Allam, 2019), clinical trials, voting irregularity analysis (Aggarwal, 2021), data cleansing, network intrusion (Aggarwal, 2020), severe weather prediction (Kalinichenko et al., 2014), geographic information systems, athlete performance analysis (Pawar & Mahindrakar, 2015), and other data-mining tasks to anomaly pattern detection for disease outbreaks (Skelsey et al., 2021).

Foorthuis (2021) point out that anomalies are of interest because they may represent both novel, interpretable results and artifacts of the data, such as measurement flaws, sampling errors, standardization failures, and false distributional assumptions. Because they can be the basis for comparisons and help identify underlying reasons, outliers can occasionally be more beneficial to research than specific data points. For example, a scenario of an entire town being infected by a disease: if there is any case of a disease-negative person, studying this outlier would be more medically useful than studying the rest of the population. The case of the discovery of one HIV-1 resistant woman in Nairobi, Kenya (Fowke et al., 1996) led to the discovery of natural immunity and more insight into combatting the virus.

Identifying fraudulent activity is another practical usage of outlier detection (Roy & Garg, 2022). Suspicious activity, such as money laundering, should have different signatures in the data than normal usage (Gomez et al., 2021). In computer science,

malware can be identified by creating baseline usage models of safe programs and then identifying attacks based on the deviation (Tang et al., 2014). Manufacturing and industrial processes require similar identification of defects to prevent costly recalls (Niemann et al., 2022). Genomic abnormalities such as differential gene expression in malignant cell lines are also anomalies compared to benign cells (Tasaki et al., 2020).

Outlier detection acts as an essential intermediate step in data analysis. Outliers can make modeling difficult due to the discordance they introduce into the data. As a measure, the isolation of outliers can improve the performance of predictive modeling by offering better data quality and reducing outlier's influence on the model fitting. In other applications, however, identifying outliers is the primary purpose of analysis, for instance, in the case of fraud detection. Several studies have been conducted to handle issues of detecting outliers. Defining outliers by their distance to neighboring examples is a popular approach to finding unusual examples in a dataset, among other techniques. However, because seemingly normal behaviour is grouped in the dataset, outlier detection algorithms may not be able to detect all forms of fraud. It is also possible that the outlier detection approach misses the genuine outlier. Assume, for instance, that the intruder completes a routine transaction amount to a bank account that is not on a denylist within the expected timeframe. In that situation, the transaction won’t be classified as unusual and won’t ever be marked as such.

Outliers can significantly affect the performance of machine learning algorithms, leading to biased models and unreliable predictions. Outlier detection ensembles are a popular approach to improve the robustness and accuracy of outlier detection algorithms. These ensembles combine the outputs of multiple outlier detection algorithms to identify and remove outliers that are consistent across different algorithms while retaining non-outlying data points. The use of outlier detection ensembles has been shown to improve the accuracy and robustness of outlier detection algorithms in a variety of applications, including finance, healthcare, and cyber security (Barnett & Lewis, 2021).

However, the design and optimization of outlier detection ensembles can be challenging due to the trade-off between bias and variance. Ensembles that use multiple weak learners can reduce the bias but increase the variance, while ensembles that use multiple strong learners can reduce the variance but increase the bias. Therefore, careful selection and combination of outlier detection algorithms are critical to achieving optimal performance in outlier detection ensembles.

Imbalance datasets are another common challenge in many applications of machine learning, where the number of data points in one class is significantly smaller than that in the other classes. Imbalance datasets can lead to biased models and poor performance, especially in applications where the minority class is of particular interest, such as fraud detection, disease diagnosis, or anomaly detection (Cervantes et al., 2020). Outlier detection ensembles can also be used to address the challenges of imbalance datasets. Specifically, outlier detection ensembles can identify and remove outliers that are present in the majority class but not in the minority class, which can help to reduce the bias and improve the performance of machine learning models. However, the design and optimization of outlier detection ensembles for imbalance datasets can be more complex than that for balanced datasets due to the trade-off between detection accuracy and minority class retention. Careful selection of outlier detection algorithms and appropriate combination techniques are required to achieve optimal performance in outlier detection ensembles for imbalance datasets (Xu et al., 2019; Wang & Sun, 2018).

In general, outliers may indicate fraudulent cases or just entry errors or may exist in datasets due to one reason or another – but either way, detection of outliers is vital for database consistency and integrity (Carter, 2019), and when detected at an early stage, can reduce financial losses in many organizations or save a life in case of medical or health-related instances. It is against such encounters that outliers are examined in this research. This research investigated outliers, outlier detection, the various detection techniques, and how heterogeneous detection methods could be combined to improve the accuracy of detecting anomalies in high-dimensional data sets.

1.3 Problem Statement

An ideal outlier detection ensemble should take into account the strengths of individual base learners while carefully combining their outputs in order to create a strong learner so as to achieve non-biased overall detection accuracy with minimal variance. Existing outlier detection ensembles utilize either parallel or sequential combination structures to fuse multiple detectors (weak base learners) in order to, hopefully, improve the overall detection performance by taking a joint overall result (majority vote) from the detectors. The parallel combination structure is designed with the intention to reduce variance while the serial combination is designed with the intention to reduce bias (Zhao et al., 2019).

Unfortunately, trusting the results from all the weak learners may have a negative effect on the ensemble's performance as a whole because some learners may produce inaccurate results depending on the type of data and the learner's underlying rules, particularly in the context of outliers - which lack ground truth (Chen et al., 2019). Outlier detection ensembles that mark instances as anomalous when they are not, or anomalous instances being marked as safe, can make the outlier detection ensembles unsafe, untrustworthy, or redundant. In certain applications, such as medical diagnosis, misclassifications could have catastrophic or irreparable consequences (Khullar, Jha, & Jena, 2015). In the financial industry, misclassifying fraudulent transactions as legitimate ones can have severe economic consequences (Kieu & Nguyen, 2020). In cybersecurity, misclassifying a malicious file as benign can result in the failure of intrusion detection systems and can have significant implications for an organization's security (Bhattacharyya & Kalita, 2019).

The need arose to study which detectors to select as base learners and in what way, on what kind of data, and in what order. This research engaged in a cross-industry process for data mining (CRISP-DM) involving various tests and experiments to measure the effect of combining heterogeneous base detectors (weak learners) on high dimensional data with the aim of improving overall detection accuracy while prioritizing the minimization of bias, variance, and the order of base learners.

1.4 Objectives

1.4.1 General Objective

This study's main objective was to create and evaluate a model for outlier detection utilizing a heterogeneous hybrid ensemble to provide improved performance and accuracy while prioritizing the minimization of bias and variance and order of base learners.

1.4.2 Specific Objectives

(i) To determine which classifiers constitute weak learners for constructing the base (detectors) for outlier detection.
(ii) To analyse different combination methods or fusion strategies (order) using the selected base learners for the outlier detection ensemble.
(iii) To create a model for outlier detection that combines several selected weak learners into a hybrid ensemble to improve performance while prioritizing minimizing bias, variance, and order of base learners.
(iv) To evaluate the developed ensemble model for outlier detection accuracy.

1.5 Study Questions

The following research questions guided the study:

(i) What classifiers make good base detectors (weak learners) for building an outlier detection ensemble?
(ii) Do combinations of heterogeneous detectors for outlier detection improve accuracy?
(iii) What combination sequences or fusion strategies are compatible for weak learners for better outlier detection?
(iv) How does the combination or fusion of weak learners affect variance and bias?
(v) How can the developed ensemble model be tested for outlier detection accuracy?

1.6 Assumptions of the Study

The following assumptions guided the study:

(i) Combining opinions of multiple weak learners into one strong learner achieves better performance and accuracy.
(ii) Combination sequence or order of weak learners affects performance and accuracy.
(iii) The choice of structure, whether parallel (independent) or serial, affects the variance and bias of an ensemble outcome.

1.7 Justification of the Study

The following grounds justified the work in this research:

Firstly, the rapid development of classification ensembles enabled efficient approaches for other machine learning problems, including outlier detection (Rayana, Zhong, & Akoglu, 2017). While ensemble techniques for classification and clustering have undergone many studies and are already being successfully used, little research has been done on ensemble learning for outlier detection because it is challenging to get ground truth. It was necessary to conduct research, capitalize on this gap, and develop a new or enhanced ensemble approach for outlier detection.

Secondly, in many data analysis tasks, a large number of variables are recorded or sampled, and one of the initial stages to getting a clear analysis is the detection of outlying observations. Although often considered errors or noise, they may carry important information. Detected outliers are candidates for peculiar patterns that may otherwise lead to wrong modeling, bias, and incorrect results. Therefore, it's critical to recognize them before modeling and analysis (Prasada et al., 2020).

Thirdly, it is often the case that groups of people can often make better decisions than individuals, especially when group members each come in with their own biases. The same concept could be applied to machine learning. Since classifiers are

learning models, it was necessary to investigate whether models could achieve overall performance and accuracy by combining the opinions in an ensemble. In contemporary decisions making, it is always also the case to leave out alternatives that do not yield good results and use the alternatives that yield better results. This concept could further be applied to classifiers by finding out whether combining the strengths of accurate learners while alleviating the weaknesses of the less accurate ones could build better ensembles for outlier mining.

1.8 Scope of the Study

This study limited itself to machine learning. It limited itself to the study of outliers and methods for outlier detection - and how those methods could be tuned in such a way as to provide improved performance and accuracy in the detection of outliers in high dimensional data. The researcher sought to find out what classifiers composed weak learners for building an outlier detection ensemble based on different classification methods and then identified a combination strategy that could fuse the weak learners to achieve a strong learner. Moreover, the researcher tested the combination sequences and fusion structures used to provide improved performance and accuracy while minimizing variance and bias in outlier detection ensembles. This research used publicly available and online databases as data sources. The data sources included the ODDS machine learning repository datasets; ten datasets were utilized in training and testing the model. Four existing state-of-the-art ensembles were compared to the proposed model to verify improvements.

1.9 Organization of Thesis

The following is how the rest of the thesis is structured: Chapter 2 reviews existing literature based on the research objectives. Issues addressed include discussion on various outlier detection algorithms, aspects of outlier detection, ensemble formation methods, concepts of bias and variance, outlier detection methods, evaluation metrics and performance measures, and an analysis of existing related works. This chapter completes by emphasizing the research gap.

Chapter 3 provides a description of the research methods used in this study. Issues discussed include data collection methods and preprocessing approaches. The chapter concludes by providing a detailed description of the proposed method for the study.

Chapter 4 provides the results of the experiments. The experiments examine the performance of the proposed technique, the comparative performance of outlier detection ensembles, and comparisons of the effectiveness of the proposed approach with other existing approaches. The chapter concludes by providing a summary of the experimental findings.

Chapter 5 offers a summary and discussion of the study. The summary includes a review of the objectives, methodology, and experimental findings. Discussion includes issues of outlier detection and bias-variance reduction, results for outlier detection using the proposed method, and performance of the proposed method in relation to the literature provided in chapter 2.

Chapter 6 provides conclusions and future work of the research study. The chapter highlights the significant contributions made based on the study findings. It explains the achievements of the research and ends by offering recommendations for further work.

1.10 Chapter Summary

A brief background of the study has been given in this chapter. The basic concepts of outlier detection, aspects of outlier detection, and ensemble formation have been introduced. The research objectives, justification, and scope have been provided. The next chapter reviews existing literature.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter reviews existing literature dealing with ensembles and model learning approaches in relation to outlier detection.

2.2 The Ensemble Concept

An Ensemble is a collection of trained classifier models whose predictions are combined to reach a final decision (Sabzevari et al., 2022). In ensemble learning, a machine learning paradigm, multiple learners are trained to tackle the same issue. Ensemble methods strive to generate many hypotheses and combine them, unlike traditional machine learning approaches that try to learn one hypothesis from training data (Pintelas & Livieris, 2020). The base learners of an ensemble are a group of learners that make up the ensemble. The ability of an ensemble to generalize is typically substantially higher than that of base learners. Ensemble learning is superior to random guessing because it can elevate poor learners to strong learners who can make more accurate predictions. Weak learners are also known as base learners. A base learning method, such as a decision tree or a neural network, generates base learners from training data. Most ensemble methods produce homogeneous base learners using a single base learning algorithm. However, some approaches produce heterogeneous learners using multiple learning algorithms. Hansen and Salamon's pioneering research in the 1980s discovered that predictions made by a group of classifiers are typically more accurate than predictions made by the best single classifier (Hansen & Salamon, 1990). The second research was conducted in 1989, where Schapire proved that weak learners could be boosted to strong learners, and the proof resulted in boosting, one of the most influential ensemble methods (Schapire, 1990).

Kumar (2022) gives four main reasons why classifiers are combined: Firstly, for statistical reasons: By averaging numerous classifiers, the worst classifier can be

avoided. This reason supported the earlier works of (Fumera & Roli, 2005) as it was efficient; however, it did not ensure that the combination would outperform the best classifier.

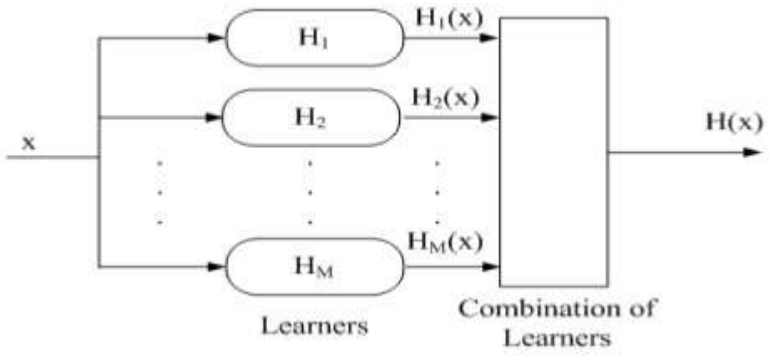

Figure 2.1: The concept of an ensemble.

In figure 2.1, the outputs of the base learners (weak learners) $H_m(x)$ with $m \in \{1,..., M\}$ are combined to produce the output of the ensemble given by $H_{(x)}$. x is the training sample.

Secondly, for representational reasons: in some cases, adding more classifiers can enhance the performance of the best classifiers used alone. Many experimental pieces of evidence show that this is possible if the classifiers in an ensemble generate varying predictions. This idea has a theoretical basis in specific situations, such as linear combinations (Kumar, 2022).

Thirdly, for computational reasons: some algorithms carry out an optimization task to discover local minima and suffer from them. To avoid finding locally optimal solutions, the back propagation technique for neural networks, for instance, uses a random initialization. Finding the best classifier is complex and is used to create multiple initializations to find an optimal classifier. The fusion of such classifiers stabilized and improved the best single classifier result (Breve et al., 2007).

Finally, some applications need to utilize more than just a single classifier; for instance, in sensor fusion, a set of learners are required. The availability of classifiers

that have varying abilities in various feature subspaces and the difficulty of building a pattern of classifiers by adjusting parameters are necessary (Chen et al., 2019).

2.3 Constructing Ensembles

An ensemble is constructed in two steps. To begin with, a large number of base learners are created in a parallel structure, as demonstrated in figure 2.2. All classifiers are run separately, and the results are combined using a combination rule, such as taking the average or using weighted voting.

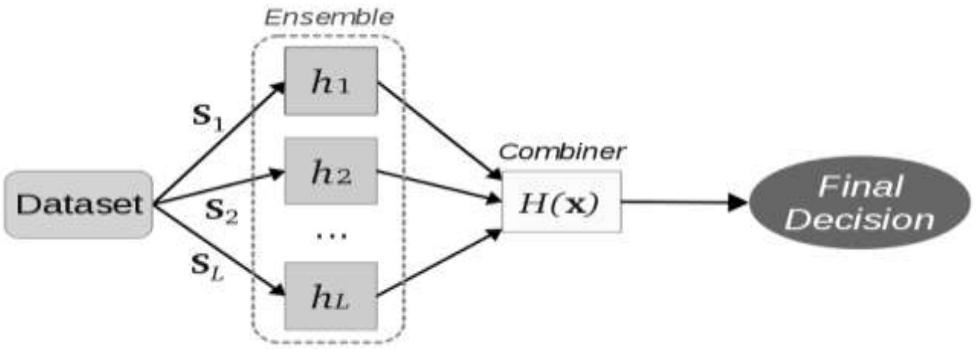

Figure 2.2: Independent (parallel) Combination of Weak Learners

In a sequential structure, a primary classifier is used. When the first classifier cannot classify a new pattern because it was rejected, a second classifier is introduced that is trained to be accurate on the errors of the first classifier. Using a third and fourth classifier, and so forth, is an option, as depicted in figure 2.3. This way, less iterations can be expected as errors reduce from level to level to the final classifier. After that, the base learners' results are combined, and a final optimal decision is made by either taking a majority vote in case of classification or a weighted average in case of regression (Pintelas & Livieris, 2020).

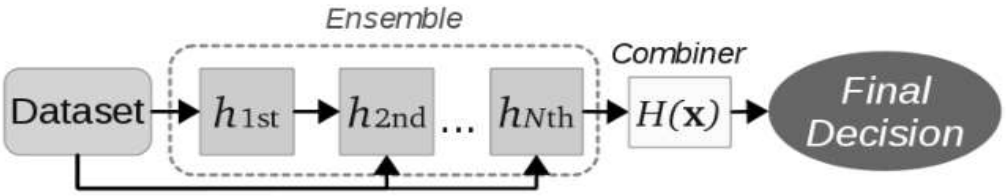

Figure 2.3: Sequential or Serial Combination

Bagging (Breiman, 1996) and boosting (Schapire, 1990; Freund & Schapire, 1996) are two of the various ways that ensembles of detectors can be trained and integrated.

2.3.1 Bootstrapping and Bagging

Bagging, also known as bootstrap aggregation (Breiman, 1996), is a method that may be used in both classification and regression. It uses bootstrap sampling to lower variance and increase the accuracy of specific predictors. Bagging uses a base-learning technique to train many base learners from a specific bootstrap sample. A bootstrap sample is created by subsampling the training data set with replacement and ensuring that the sample size is the same as the training data set. Some training examples may appear in a bootstrap sample, while others may not, with the probability of seeing an example at least once being around 63.2% (Bauer & Kohavi, 1999). After obtaining the base learners' results, bagging combines them by taking a majority vote, and the most-voted class is predicted. Bagging conventionally uses classifiers of the same type e.g., decision trees. Figure 2.4 shows the bootstrapping approach. Using bootstrap, produce several training samples $z_1 \ldots z_n$; then each is fed to a weak learner H_m. A majority vote produces the final decision $H_{(x)}$.

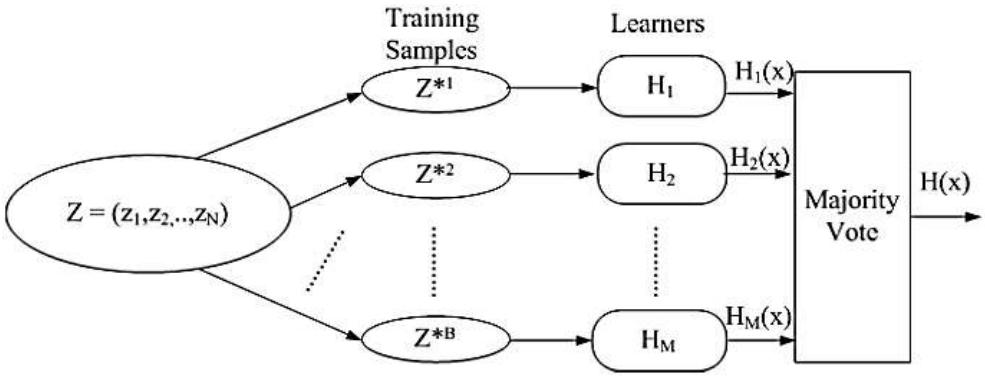

Figure 2.4: The Bagging Approach.

The bootstrap procedure is as shown in algorithm 1:

Input: Size-N sample $Z = \{z_1, z_2, \ldots, z_N\}$ of a (potentially infinite) population P.
B, number of bootstrap samples.
Output: Estimate $\widehat{T}(P)$ of the population statistic.

1: **for** $b = 1$ to B **do**
2: Draw, with replacement, N samples from Z, obtaining the b-th bootstrap sample Z_b^*.
3: Compute, for each sample Z_b^*, the estimate of the statistic $\widehat{T}(Z_b^*)$.
4: **end for**
5: Compute the bootstrap estimate $\widehat{T}(P)$ as the average of $\widehat{T}(Z_1^*)$,...,$\widehat{T}(Z_B^*)$.
6: Compute the accuracy of the estimate, using, *e.g.*, the variance of $\widehat{T}(Z_1^*)$,...,$\widehat{T}(Z_B^*)$.

Algorithm 2.1: Procedure for Bootstrapping

Bagging offers two key advantages over learning a classifier traditionally, that is, from the entire training set: first, it improves classifier stability and accuracy, and second, it lowers classifier variation. Algorithm 2 below depicts the bagging procedure. Bagging, however, omitted the bias term and only reduced variance. (Schapire & Freund, 1997) later developed an ensemble known as boosting to make up for this shortfall.

Algorithm 2 Bagging Procedure for Classification

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
B, number of bootstrap samples.
Output: $H : \mathscr{X} \to \{-1, +1\}$, the final classifier.

1: **for** $b = 1$ to B **do**
2: Draw, with replacement, N samples from Z, obtaining the b-th bootstrap sample Z_b^*.
3: From each bootstrap sample Z_b^*, learn classifier H_b.
4: **end for**
5: Produce the final classifier by a majority vote of $H_1, \ldots, H_B$, that is, $H(\mathbf{x}) = \text{sign}\Big(\sum_{b=1}^{B} H_b(\mathbf{x})\Big)$.

Algorithm 2.2: Procedure for Bagging

2.3.2 Boosting

Boosting uses a different method of resampling than bagging. The weak learners are created in stages, with examples that were incorrectly classified by previous classifiers being chosen more frequently than those that were correctly classified. The boosting procedure is listed in point in Algorithm 3 below.

Algorithm 3 Boosting Procedure for Classification

Input: Dataset $Z = \{z_1, z_2, \ldots, z_N\}$, with $z_i = (\mathbf{x_i}, y_i)$, where $\mathbf{x}_i \in \mathscr{X}$ and $y_i \in \{-1, +1\}$.
Output: A classifier $H : \mathscr{X} \to \{-1, +1\}$.

1: Randomly select, without replacement, $L_1 < N$ samples from Z to obtain Z_1^*.
2: Run the weak learner on Z_1^*, yielding classifier H_1.
3: Select $L_2 < N$ samples from Z, with half of the samples misclassified by H_1, to obtain Z_2^*.
4: Run the weak learner on Z_2^*, yielding classifier H_2.
5: Select all samples from Z on which H_1 and H_2 disagree, producing Z_3^*.
6: Run the weak learner on Z_3^*, yielding classifier H_3.
7: Produce the final classifier as a majority vote: $H(\mathbf{x}) = \text{sign}\Big(\sum_{b=1}^{3} H_b(\mathbf{x})\Big)$.

Algorithm 2.3: Technique of Boosting

The training set is randomly partitioned into three partitions without replacement, as shown in Algorithm 3, Z_1*, Z_2*, and Z_3*. For a given instance, if the first two classifiers (H1 and H2) agree on the class label, this is the final decision for that instance. The set of instances on which they disagree defines the partition $Z3*$, which is used to learn H3. Schapire has shown that this learning method is strong. Furthermore, using this approach repeatedly can reduce the overall error. In other

words, each learner can serve as their boosting mechanism. Figure 2.5 shows the boosting approach.

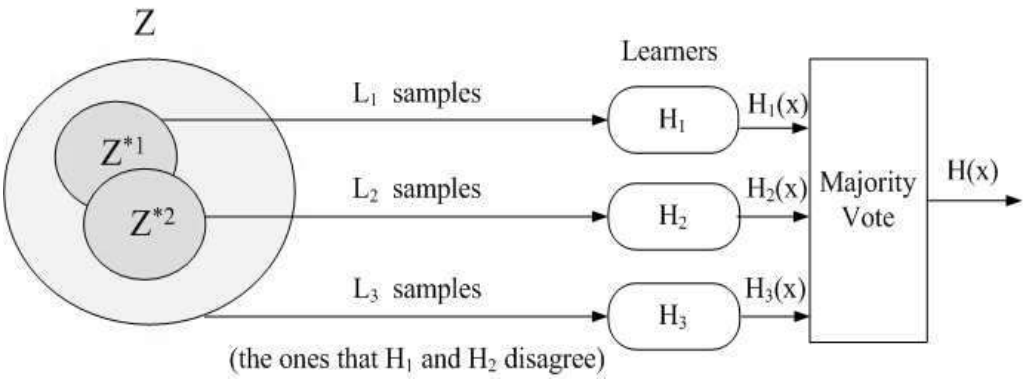

Figure 2.5: A Boosting Approach

2.3.3 Adaptive Boosting (Ada-Boost)

(Freund, 1996) proposed a boosting algorithm called adaptive boosting, based on, and improving, the ideas presented by (Schapire, 1990). By integrating multiple classifiers, each of which is acquired by applying a specific learning method to a separate collection of instances, he increases the accuracy of learning binary classifiers. Adaptive boosting uses weights of instances of similar training data in place of random subsamples. Due to the fact that the training set is utilized repeatedly, its size is not critical. To create the final classifier, AdaBoost uses a weak learner to learn a group of classifiers. Using reweighted copies of the training data, the weak classifiers are designed successively, with the weights based on the performance of the prior classifiers. In this manner, the weak learner concentrates on patterns that the preceding weak learner failed to classify at each iteration accurately. This is illustrated in algorithm 4 below:

The adaboost algorithm pseudocode (Freund, 1996)

1. Given examples $(x_1, y_1), \ldots, (x_n, y_n)$

   where: $y_i = 0, 1$ *for negative and positive examples respectively*.

2. Initialize weights $w_i = {}^1/_2M, {}^1/_2L$ for y_i = zero, one respectively

Where; M, L are *the no. of negatives & positives.*

3. For r = 1 to R
    a) Normalize w, to $\sum_{i=1:n} w_i = 1$
    b) Choose classifier h_j, with the lowest error ε_j : $\varepsilon_j = \sum_i w_i \mid h_j(x_i) - y_i \mid$.
    c) Update the weights for each example:

$$w_i = \begin{cases} w_i \beta_t & \text{example } x_i \text{ is classified correctly} \\ w_i & \text{otherwise} \end{cases}$$

4. The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1:T} \alpha_t h_t(x) \geq \lambda \sum_{t=1:T} \alpha_t, \text{ where } \lambda = \tfrac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Algorithm 2.4: Adaptive Boosting

Because of normalization, the number of iterations increases, and the error rates increase, which leads to smaller α values for weak learners selected later in the training process. The final classification function is the sum of the predictions of the selected weak learners multiplied by the corresponding α values.

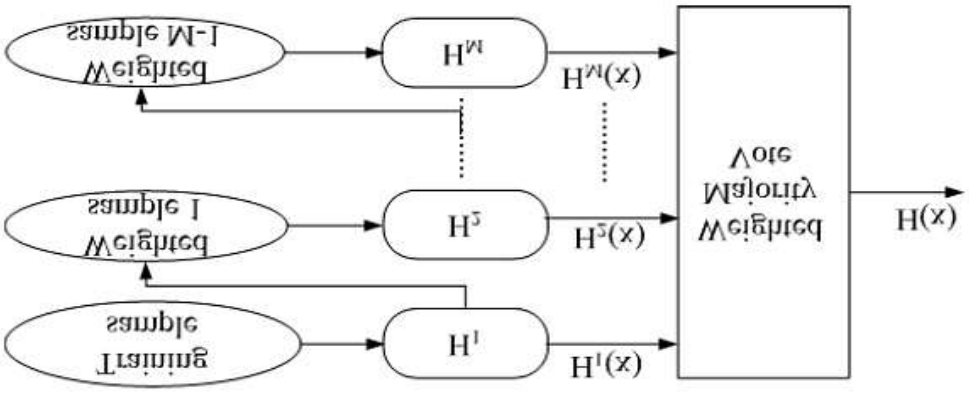

Figure 2.6: Adaptive Boosting Algorithm.

In figure 2.6, a distinct weighted copy of the training examples is used to train each weak learner. The final prediction is a product of those weights with a majority vote from the selected weak learners that were adaptively trained on the weighted examples.

2.2.4 Stacking

Stacking combines numerous detectors by learning a meta-level (or level-1) model based on the decisions of the base-level (or level-0) classifiers. The output of the detectors for a test instance, along with the true class of that instance, forms a meta-instance. After that, the meta-instances are used to train a meta-detector, and all the training data is used to train the base-level classifiers. When a new instance must be classified, the output of the base-level classifiers is computed first, followed by the output of the meta-level classifier, which yields the final result (Aggarwal, 2017). The algorithm 5 below summarizes stacking.

Algorithm Stacking

Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier H
Step 1: learn base-level classifiers
for $t = 1$ to T **do**
    learn h_t based on D
end for
Step 2: construct new data set of predictions
for $i = 1$ to m **do**
    $D_h = \{x_i', y_i\}$, where $x_i' = \{h_1(x_i), ..., h_T(x_i)\}$
end for
Step 3: learn a meta-classifier
learn H based on D_h
return H

Algorithm 2.5: Staking Algorithm

Two main challenges when handling an ensemble of heterogeneous base detectors are: selecting what detectors to use as base learners and combining the detectors' outcomes into a single final decision.

2.4 Selecting Detectors

A simple method is an evaluation and selection where each model is evaluated typically using the 10-fold cross-validation on the training set and selecting the best one on the test set (Tsoumakas et al., 2014). Other research proposes selecting a learning algorithm based on its performance in similar learning domains. Training

meta-instances are produced by recording the predictions of each algorithm, using the whole training data both for training and testing. Performance data is generated using m k-fold cross-validations and averaging the m evaluations for each training instance. Although every ensemble combines multiple detector outcomes into a single decision, their building paradigms usually differ in the diversity generation mechanism among the base learners and the strategy of combining them.

2.5 Combining Detectors

Since base learners may make mistakes in different instances, strategically combining them can reduce the total error (Khullar, 2015). Therefore, diversity among the base detectors is one of the key issues in ensemble formation. The homogeneous formation is the popular approach where different sets of the original training dataset are used to train other instances of one base classifier, for example, in bagging and boosting. These formations could be biased to some specific characteristics of the dataset because of their training using a single type of base learner. These formations could be biased to some specific characteristics of the dataset because of their training using a single type of base learner. Using different base learners to create an ensemble is one approach for introducing diversity, i.e., a heterogeneous ensemble, which is beneficial for learning other characteristics of the training dataset.

The other critical point is the combination of base learners' outcomes into a final decision. There are numerous combination approaches, such as majority voting, weighted majority voting, summation, product, maximum and minimum, fuzzy integral, Dempster-Shafer-based fusion, or decision templates (Zhao et al., 2019). Voting, unweighted or weighted, are two main methods for combining not only Heterogeneous but also Homogeneous models (Sabzevari et al., 2022). In voting, each model generates a class value (or ranking, or probability distribution), and the ensemble proposes the class with the most votes (or the highest average ranking or average probability). The winning class must receive at least 50% (the majority) of the votes. In contrast to majority voting, weighted voting assigns each model a coefficient (weight), which is often equivalent to its classification accuracy.

2.6 Outlier Detection

Outlier detection is the process of identifying patterns in data that differ from what would be considered normal behaviour (Aggarwal, 2021). Outliers are detected by analyzing the system's events, where each event is designated by a data instance. Features (i.e., attributes) are used to describe the data instance. The ability to distinguish normal from deviant is reliant on features.

Hawkins defines an outlier as “an observation that deviates so substantially from other observations as to provoke suspicion that a separate mechanism generated it” (Hawkins, 1980). In most literature, authors have described an outlier as an observation that “appears to be inconsistent” with the remainder of a data set; this is the main problem when dealing with outliers. Outlier detection methods try to solve this problem using different approaches, including statistical and probabilistic knowledge, distance and similarity-dissimilarity functions, metrics and kernels, accuracy when dealing with labeled data, association rules, properties of patterns, and other specific domain features.

2.7 Key Aspects of Outlier Detection Problem

Exploring the invisible data spaces is a significant obstacle in outlier detection. Finding the outlying points by computing a measure of normalcy or ordinariness with respect to their nearby points is an easy way to locate outliers. However, several aspects make this very challenging. (Bii, Rimiru & Mwangi, 2020) described seven factors: (i) it is quite challenging to define a normal zone that includes every conceivable normal activity. (ii) Normal behaviour frequently evolves, therefore a current understanding may not be accurate in the future.

(iii) The line separating a normal point from an outlier is frequently blurry, making it possible for the outlying point to be normal and vice versa. (iv) Every application domain imposes a unique set of criteria and limitations, hence the precise definition of an outlier varies for each. (v) Labelled data for training or testing is not always available. (vi) It might be difficult to define normal behaviour when outliers are caused by malicious behaviour because the malicious opponents adjust to make the

outlying observations seem normal. (vii) Noise is present in most data and simulates the actual outliers, making it difficult to separate and eliminate.

Due to these difficulties, the majority of outlier detection algorithms introduce numerous elements, such as the nature of the data, the characteristics of the outliers that need to be found, the significance of the normal, etc. The aspects are, in most scenarios, determined by the application domain in which the technique is applied. For these reasons, numerous approaches to the outlier detection problem have been investigated across multiple fields. e.g., data mining, statistics, information theory, and machine learning. Narrowing down Hawkins's concept, for example, two major outlier detection techniques were derived: distance-based techniques and density-based techniques. While density based strategies identify data points located in a lower density area than their closest neighbours, distance based techniques identify data points far from their nearest neighbours (Xu et al., 2022).

2.8 Ingredients of Outlier Detection

The first ingredient of any outlier detection technique is taking into account the data's characteristics, the outliers' characteristics, and the limits and rules that form the problem design. Secondly is the application area in which the technique is to be used. Some techniques target particular domains, while others are general and are developed in a more general way. Finally, the concepts and ideas from one or more knowledge disciplines are non-trivial (Merza, 2021).

2.8.1 Type of Outliers

Point outliers, contextual outliers, and collective outliers can be categorized based on the number of data instances included in the idea of outliers.

2.8.1.1 Point Outliers

A point outlier is a data instance that has an anomaly but is not part of a larger dataset. For example, a system event occurs when a user attempts to visit a restricted server. Point outliers do not fit the situations where outlying behavior is an aggregate

of data instances (Merza, 2021). Figure 2.7 illustrates point outliers. N1 and N2 are regions of normal behavior; Points o1 and o2 are outliers.

2.8.1.2 Contextual Outliers

These are data points that are considered anomalous in a particular context. Each data point is defined by contextual and behavioural attributes (Aggarwal, 2021). Examples are the time of day, season, and geographical location. Figure 2.8 illustrates contextual outliers. T1 is normal, but T2 is an outlier.

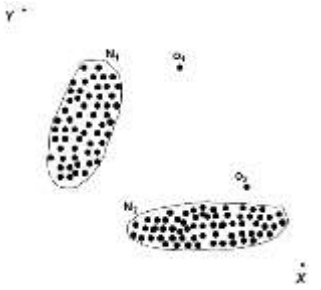

Figure 2.7: Point outliers

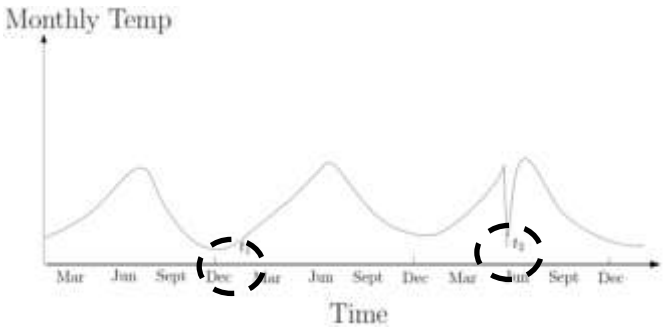

Figure 2.8: Contextual Outliers

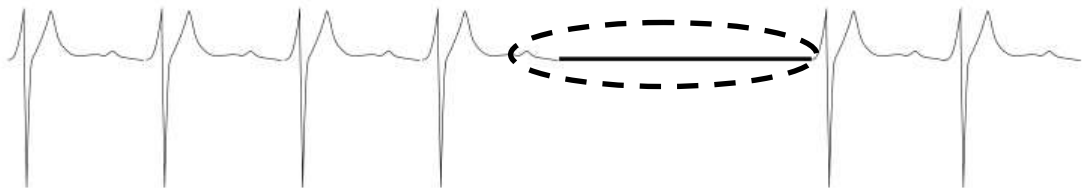

Figure 2.9: Collective Outlier - showing an anomalous subsequence

2.8.1.3 Collective Outliers

It defines a group of instances that exhibits an anomalous behavior compared to the other groups of instances (Merza et al., 2021). An individual instance is not necessarily anomalous on its own. Collective outliers are meaningful only when the data has spatial or sequential nature (Aggarwal, 2021). Sequence outliers are

presented sequentially (Figure 2.9). E.g., an unusual sequence of commands could signal malicious action that jeopardizes system security.

2.8.1.4 Vector Outliers and Trajectory Outliers

Vector outliers are detected in vector-like data representations, such as relational databases. The data is presented in tuples, each of which has its attributes. Based on numbers, the data collection can be split into low-dimensional or high-dimensional data (Zhang, 2008). Trajectory outliers are characterized by key movement features, such as the average, minimum, and maximum values and velocities. A weighted sum distance function is defined to compute the difference in trajectory based on the key features (Knorr et al., 2000).

2.8.2 Type of Input Data

The type of data for input to a detection technique is vital. Every data instance is described using a set of attributes called features, variables, fields, dimensions, or characteristics. They might have a single characteristic (univariate) or numerous attributes (multivariate), which can be binary, categorical, or continuous. Any outlier detection technique's goal is to identify the ideal combination of features that will enable the detection algorithm to produce more accurate findings while consuming fewer resources.

2.8.2.1 Point Input Data

If input data has no structure amongst its instances, they are referred to as point data. Algorithms taking point data sets are found in the medical records outlier detection domain (Barai & Dey, 2017).

2.8.2.2 Sequential Input Data

If data instances are ordered, i.e., defined sequentially in a data set, they’re sequential data. E.g. time-series data (Javier, 2017).

2.8.2.3 Spatial Input Data

When data instances have a well-specified spatial structure, i.e., the location of a data instance with respect to other data is significant and well specified, they are spatial data. For example, ecological data (Larson & Moore, 2022).

2.8.2.4 Spatio-Temporal Input Data

Data instances can be structured to have temporal or sequential component, they give rise to Spatio-temporal data, for instance, the climate data (Wu et al., 2008).

2.8.3 Type of Detection

In order to build a predictive model, a training data set is required. The labels associated with a data instance signify if that instance is normal or an outlier. And based on the extent to which these labels are utilized, outlier detection techniques are categorized on whether labeled instances of outliers can be obtained or whether the objects can be assumed as normal or outliers. Within each category, there are methods for detecting and evaluating outliers. The techniques under user-labeled instances are supervised, semi-supervised, and unsupervised techniques. Algorithms for supervised learning use labeled data to find outliers, where records are classified as “normal” or “outlier”. Since unsupervised learning approaches use unlabeled data, outliers (and normals) are unknown (Carter, 2019). Concerning detecting outliers on the assumption of normal data versus outliers, the techniques used include clustering, statistical, and distance-based or proximity techniques. Figure 2.10 summarizes the techniques under this section.

2.8.3.1 Supervised Outlier Detection

Supervised detection approaches use examples and rules to identify characteristics distinguishing normal behaviour from an outlier. Each new observation is given a class, one of the two. The supervised method could be faced with the problem associated with an imbalance class where the outlier datasets may be observed to be the minority class.

This can be alleviated by re-sampling the dataset by either under sampling the majority normal classes or oversampling the minority outlier classes. Alternatively, artificial or synthetic data sets for outlier classes can be generated to boost the number of outlier samples. The performance assessment metric should be based on recall and receive operating characteristics rather than accuracy (Sun et al., 2019).

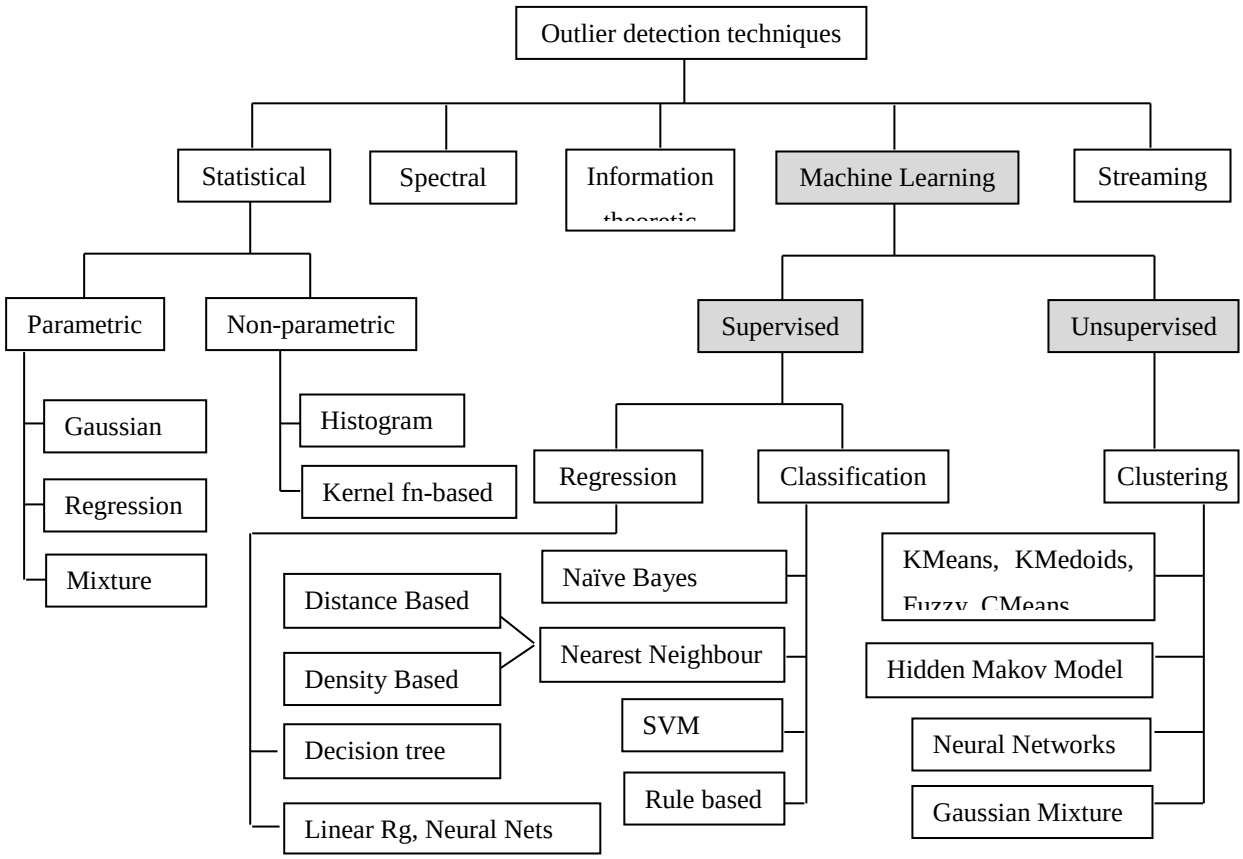

Figure 2.10: Taxonomy of Outlier Detection Techniques

2.8.3.2 Semi-supervised Outlier Detection

A semi-supervised method combines the advantages of both supervised and unsupervised methods. If there are some labeled normal items, the labeled examples and nearby unlabeled objects can be utilized to train a model for normal objects. Outliers are data objects that do not fit the model of normal items in this context (Ruff et al., 2019).

2.8.3.3 Unsupervised Outlier Detection

Unsupervised detection methods rely less on rules or examples and more on data distribution. Based on a version of this distribution, the data distribution and unsupervised algorithms find outliers in datasets. As a result, these algorithms do not require tagged instances, which makes processes like fraud detection more realistic since the information is not readily available. The two underlying presumptions for this detection are that outlying behaviour may be distinguished from normal behaviour and that the number of "normal" records is significantly higher than the number of "outlier" records. One of the main challenges and weaknesses of the unsupervised method of outlier detection is the failure of the method to detect collective outliers effectively. This is explained by the fact that while normal items may not exhibit strong patterns, a group of outliers may exhibit great similarities in a restricted space. For instance, routine activities vary in some intrusion or virus detection systems, and unsupervised methods may have a high false-positive rate but still miss many true outliers. The clustering method for detecting outliers can be affected by the presence of noise in the dataset since it is hard to distinguish between noise and outliers. Unsupervised algorithms for outlier detection have the advantage of spotting previously unknown outlier records (Zhao & Hryniewicki, 2018).

2.8.3.4 Statistical Outlier Detection

According to Amil, (2019) statistical outlier detection techniques depend on the statistical approaches that assume a distribution to fit the given dataset. Data is assumed to be generated following established distribution models like Gaussian mixture, Poisson distribution, etc. Statistical approaches presume that outliers are data that do not fit a statistical model, such as a stochastic model, and that normal data do. Whether the data support the statistical model's underlying assumption determines their effectiveness. They use discordance tests depending on data distribution parameters like mean, variance, etc. A drawback of statistical techniques is that the method assumes that the data modeled consists of a single feature or attribute and that data distribution follows a known distribution. It is vital to note that real-life data may consist of multivariate high-dimensional objects whose statistical

distribution model may not be prior known or difficult to establish. Statistical techniques can model parameterized or non-parameterized data (Amil et al., 2019).

2.8.3.5 Parametric Outlier Detection

The parameterized technique, like the Mahalanobis distance technique, is the distance between two points in multivariate space. It measures distance relative to the centroid - a point in multivariate space where all means from all variables intersect. The bigger the distance, the further the data point is from the centroid. Between two objects, the Mahalanobis distance is calculated as:

$$d\ (Mahalanobis) = [(X_B - X_A)^T * C^{-1} * (X_B - X_A)]^{0.5} \quad \text{Equation (2.1)}$$

Where X_A and X_B are two objects, and C is the sample covariance matrix.

The disadvantage here is the inverse correlation matrix needed for the calculations, which can't be calculated if variables are highly correlated (Hamid, 2019).

2.8.3.6 Non-parametric Detection

With non-parameterized techniques, the model of normal data is learned from input data without any prior structure. Histograms are used to detect outliers. A problem faced in the structure of histograms is the establishment of appropriate bin sizes. Where the bin size is too small, the normal object can be observed as in outlying bins or being empty, resulting in a false positive. Where the bin size is too large, the outliers can be observed in some frequent bins resulting in a false negative. By using kernel density estimation to determine the probability density distribution of the data, the histogram's bin size issue can be resolved (Jiawei et al., 2012). The object is most likely normal if the predicted density function is high; otherwise, it is an outlier.

2.8.3 Distance based Outlier Detection

Knorr and Ng (1998) introduced the concept of local neighborhood or k-nearest neighbors (kNN) of the data points. KNN uses the distance to the k^{th} nearest neighbors of every point, denoted as D_k, to rank points so that outliers can be

discovered and ranked. Formally, given k and n points, a point is an outlier if the distance to its k^{th} nearest neighbor is smaller than the corresponding value for no more than $n - 1$ other point. Importantly, the distance-based methods require a user to specify a distance threshold. Figure 2.11 shows the steps of determining the class of a new instance using KNN. There are three algorithms under this category, namely index-based, nested-loop, and cell-based algorithms (Xu et al., 2022).

In the nested loop, for any object o, calculate its distance from other objects and count the number of other objects in the r-neighborhood. If other objects are within r distance, the inner loop is terminated; else, o is an outlier. In cell-based, also called grid-based, an attempt is made to improve efficiency by reducing the computation cost of objects in the data set. Rather than evaluating each object, it evaluates groups of objects. The grid-based method partition the data space into a multi-dimensional grid. Each cell is a hypercube with a diagonal length. It applies the process of pruning using the level-1 & level 2 cell properties, i.e., for any possible point x in cell C and any possible point y in a level-1 cell, $dist(x,y) \leq r$; and for any possible point x in cell C and any point y such that $dist(x,y) \geq r$, y is in a level-2 cell.

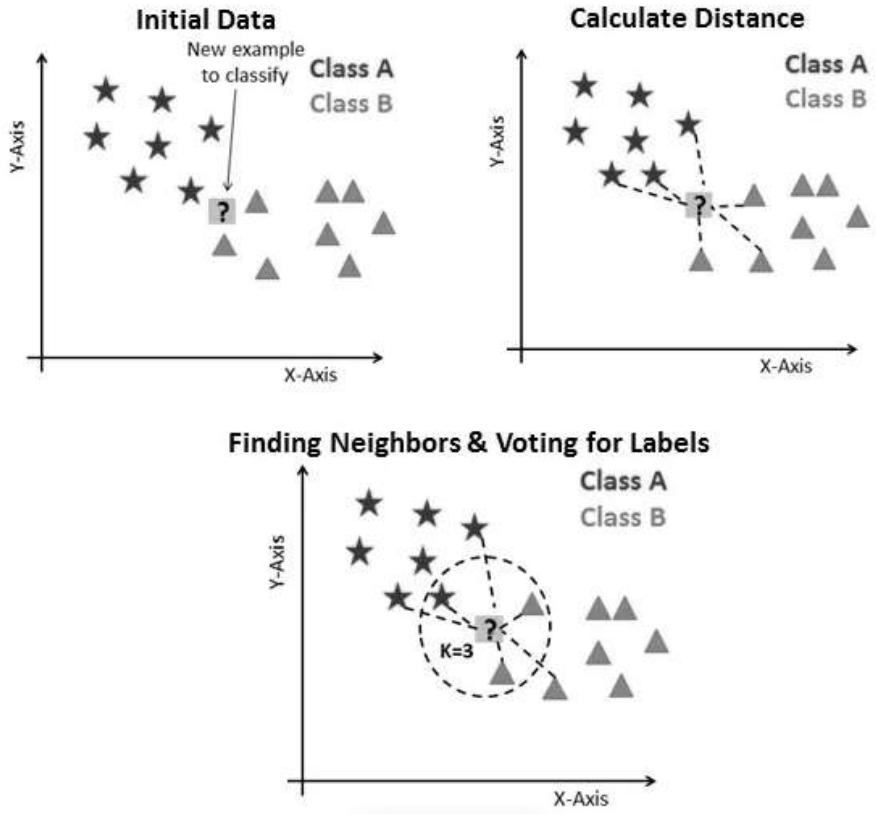

Figure 2.11: K-Nearest Neighbour with k=3

In Shaikh and Kitagawa's (2012) work, managing and mining outlier data was becoming vital with the increased use of devices responsible for generating outlier data, e.g., sensors. Their work extended the notion of distance-based outliers for outlier data of Gaussian distribution. Since the distance function for Gaussian distributed objects was computationally costly, they proposed a cell-based approach to accelerate the computation.

The work of (Angiulli et al., 2010) and (Chander et al., 2022) surveyed a distributed method for detecting distance-based outliers in large data sets. They talked about a modification to the fundamental technique that decreases the quantity of data needed to be exchanged to decrease communication costs and increase overall runtime.

2.7.3.8 Density-Based Outlier Detection

Density-based methods find outliers in spatial data, where outliers are objects having a low local density of an object's neighborhood of objects. The definition of the spatial neighborhood is based on Euclidean distance and graph connectivity (Aggarwal, 2017). Breuning (2000) proposed a more robust scheme than distance-based schemes, focusing on local outliers compared to their local neighborhoods instead of the global data distribution. The density around an outlaying object significantly differs from the density around its neighbors. It involves examining the local density of the point being studied and the local densities of its nearest neighbors. Figure 2.12 illustrates the density-based outliers, where O1, O2 represent local outliers in C1, while O3 is global, and O4 is an inlier.

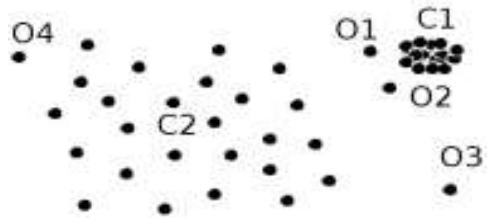

Figure 2.12: Density-based clusters and outliers

Liu and Cao (Liu et al. 2018 and Cao et al. 2014) examined the problem of density-based local outlier detection on outlier data sets described by discrete instances. They proposed a density-based local outlier concept based on uncertain data. The local outlier factor (LOF) is described in terms of an object's reachability distance and local reachability density. From figure 2.13, the reachability distance from *o'* to *o is*:

$$reachdist_k(o \leftarrow o') = max\{dist_k(o), dist(o, o')\} \qquad \text{Equation (2.2)}$$

where: *k* is a user-defined parameter and local reachability density of *o is*:

$$lrd_k(o) = \frac{\|N_k(o)\|}{\sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)} \qquad \text{Equation (2.3)}$$

hence, LOF of an object *o* is the average of the ratio of l_rd_k (*o*) and *o*'s k-nearest neighbors:

$$LOF_k(o) = \frac{\sum_{o' \in N_k(o)} \frac{lrd_k(o')}{lrd_k(o)}}{\|N_k(o)\|} = \frac{\sum_{o' \in N_k(o)} lrd_k(o') * \sum_{o' \in N_k(o)} reachdist_k(o' \leftarrow o)}{||N_k(o)||^2}$$

Equation (2.4)

The lower the $l_rd_k(o)$ and the higher the local reachability density of the kNN of *o*, the higher LOF. This way, a local outlier whose local density is relatively low compared to the local densities of its kNN is captured.

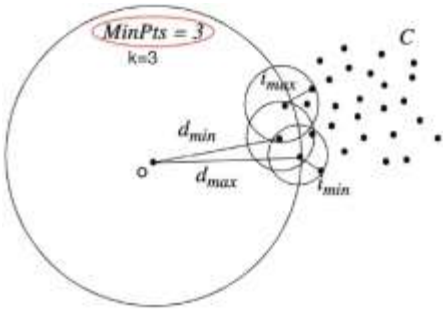

Figure 2.13: Determining Local Outlier Factors (LOF)

In a nutshell, distance-based approaches counter the limitations of statistical approaches. E.g., Manhattan or Euclidean metrics measure the distances between pairs of points. Their effectiveness depends on the proximity measure. The advantage is that, unlike distribution-based methods, distance-based methods are non-parametric and do not rely on any assumed distribution to fit the data (Zhao et al., 2019). The disadvantage is that they are ineffective in high-dimensional space due to the curse of dimensionality, noise, and abnormal deviations that may be rooted in lower-dimensional subspaces that cannot be observed in the full data space.

2.8.3.9 Clustering-based Outlier Detection

Clustering-based outlier detection approaches presumptively assign outliers to small or sparse clusters or none, while normal data are assumed to belong to large, dense clusters. Prior to the initial iteration, K-Means initializes k centroids. Every other

iteration involves assigning each data point to the closest centroid, which is followed by a calculation of a new cluster mean, which serves as the new centroid. The model is said to be stable when the previous iteration's centroid equals the present iteration's result. When the size of the data increases, it uses linear time complexity and converges to a local minimum. On the other hand, k-means clustering is highly sensitive to outliers and cannot be perceived as robust. Whereas k-means clustering selects the mean of a cluster as a centroid, k-median selects the median. Hence k-median clustering is a more robust technique for outlier detection (Angelin & Geetha, 2020). It minimizes every point's 1-norm distance (Manhattan) to its assigned cluster centroid. A local minimum of the Manhattans distance between the centroid and its assigned points is reached by computing the sum of differences between two vectors in a dimensional space.

Some advantages of using the clustering method include: data labels are not required, it works for many types of data, clusters can be viewed as summaries of the data, and once the clusters are obtained, you need only to compare any object against the clusters to determine whether it is an outlier. However, many researchers disagree and claim that clustering algorithms shouldn’t be viewed as outlier detection techniques because their sole goal is to organize the objects in a dataset so that clustering functions can be optimized. The purpose of removing outliers from a dataset using clustering is to lessen their negative impact on the clustering output, which contrasts with the various definitions of outliers in outlier detection, which are more objective and independent of how clusters in the input dataset are identified. The critical variation between clustering and density-based methods is that clustering methods segment the *points*, whereas density-based methods segment the *space*. In nearest neighbor methods, the distance of each data point to its nearest neighbor is determined, while in clustering, the first step is to use a clustering algorithm to determine the dense regions of the data set. Secondly, a measure of the fit of the data points to the different clusters is used to compute an outlier score for the data point.

2.8.3.10 Sliding Window Technique in Outlier Detection

The sliding window is utilized in this method for streaming data. The sliding window is correctly selected, and the window size is not based on the data point. This strategy is ineffective since certain outliers were viewed as inliers in another window (Kaur & Garg, 2016).

2.8.3.11 DSS and LDSS Techniques in Outlier Detection

Outlier detection in large data sets can be accomplished using distributed methodologies such as the Distributed Solving Set (DSS) and Lazy Distributed Solving Set (LDSS). Although the DSS method has a supervisor node, other nodes concurrently do its core calculation, and the partial result synchronises once the work is finished. In Lazy DSS, subsets of the nearest neighbours are computed for each node, starting with the smallest, and sent to the supervisor node (Angiulli et al., 2010; Chander et al., 2022). Table 2.1 compares various outlier detection algorithms.

Table 2.1: Comparisons of Outlier Detection Techniques (Thudumu et. al., 2020)

	Technique	Efficiency (L=Low; A=Average H=High)	Computational Cost (L=Low; Avg=Average; H=High)	Scalability (N=No; Y=Yes)	Application	High Dimensional Data (N=No; Y=Yes)
1	Statistical Based Outlier Detection	L	H	N	Statistical	N
2	Depth Based Outlier Detection	L	H	N	Statistical	N
3	Distance Based Outlier Detection	A	L	Y	According to the distance between each point	Y
4	Density Based Outlier Detection	H	H	Y	Local density neighborhood of the data points	Y
5	Clustering Based Outlier Detection	H	L	Y	Basing on the data clusters	Y
6	Classification Based Outlier Detection	H	L	Y	Normal training data	Y
7	Sliding Window technique for outlier detection	L	H	Y	Streaming Data	Y
8	DSS & LDSS detection techniques	H	L	Y	Large/high dimensional datasets	Y

2.8.4 Outlier Detection Methods in High Dimensional Data

In statistics, a dataset's dimensionality is the number of attributes, and when there are several attributes, the dataset is said to be high dimensional, and computations become very challenging (Kamalov, 2020). The number of features may be more than the number of observations in high-dimensional data. The curse of dimensionality refers to what happens when more variables are added to a multivariate model. The more dimensions added to a data set, the more difficult it

becomes to predict certain quantities. In these cases, the traditional outlier detection approaches such as PCA are ineffective. Distance and probability density metrics are created to calculate the differences between one object and the other, but they lose meaning across the entire space in high-dimensional datasets (Tang, 2015).

As dimensionality increases, the range between the nearest and farthest neighbors gets closer. Adopting techniques based on the nearest neighborhood, will result in outlier scores close to one another. Therefore, identifying outliers in different subspaces becomes an option for solving practical outlier detection problems. Subspace selection and outlyingness measurement are the two main issues examined in the majority of studies on subspace outlier detection. On the basis of the various assumptions, many subspace selection techniques have been presented. As standard outlier identification procedures, distinct metrics are created for different reasons. Using statistical approaches, Keller et al. (2012) selected subspaces with high contrast. Kriegel et al. (2009) explained the outlyingness of a data object in a subspace made up of its nearest neighbors. Muller et al. (2008) created a data object's outlier score using the dimensions of the relevant reference cluster. SOD, a technique to identify data objects that do not fit well into their axis-parallel subspaces, was first introduced by Kriegel et al. in 2009. A reference point that spans the axis-parallel subspace of a data object *o* shares at least n of its k-nearest neighbours with that of *o* is said to be the reference point. Here, n and k are two user-input parameters; *n* is the bare minimum of neighbours a reference point and *o* must share, and *k* is the minimal number of neighbours a data object must take into account. The variation of reference points in each dimension and the distance between the outlier and the mean values of the reference points in each dimension are used to determine the outlier score of *o*. Additionally, Rehman et al. (2020) provided a method to evaluate the contribution of a few chosen subspaces when an item deviates from the area indicated by the closest cluster. The chosen subspaces are designed to offer a clear distinction between the object and its immediate surroundings, which are modelled as a collection of objects within a given radius. The selected radius depends on the subspace. It calculates an overall outlier score for a selected object that only considers contributions from subspaces where the object has a noticeably low density (twice standard deviations from the mean). An

adaptable neighbourhood that expands in accordance with the amount of features in the subspace is used to overcome the difficulty of calibrating density metrics in various subspaces. The caliber of the chosen subspaces and the suitability of the adopted outlyingness measurements determine how efficient subspace identification approaches are (Rehman & Khan, 2020).

2.8.5 Output of Outlier Detection

A key requirement for any outlier detection technique is the manner in which the outliers are reported. Guo et al. (2018) address two reporting techniques:

2.8.5.1 Labelling Techniques

Each test case is given a label (normal or outlier). The methods operate as classification algorithms; given a set of instances for the test input, they output a set of outliers and a set of typical instances. Such methods have the advantage of giving a precise set of outliers. The disadvantage is that they do not rank various outliers or distinguish between them. A 0-1 judgment isn't useful as a weight is typically connected to an outlier in a trend.

2.8.5.2 Scoring Techniques

Depending on how much a pattern is thought to be an outlier, these techniques give it an outlier score. So, a ranked list of outliers is the result. The top few outliers may be examined, or outliers may be chosen using a cut-off criteria. The decision of the threshold to select a set of outliers is the drawback of a ranked list of outliers since it requires random selection and is not simple to execute.

2.8.6 Evaluation of Outlier Detection Technique - Performance

Evaluation metrics are defined from a matrix with the number of examples correctly and incorrectly classified for each class, called a confusion matrix. The binary classification problem's confusion matrix (which has only two classes - positive and negative), is shown in Table 2.2.

Table 2.2: Confusion Matrix

	Predicted Class	
Actual Class	***Positive***	***Negative***
Positive	TP	FN
Negative	FP	TN

False positives (FP) are examples detected as positive, whose true class is negative. False negatives (FN) are examples detected as negative, whose true class is positive. True positives (TP) are examples correctly predicted as positive and are actually from the positive class. True negatives (TN) are examples correctly predicted as belonging to the negative class.

2.8.6.1 Accuracy Measure (Acc.)

The accuracy rate (Acc.) evaluates the effectiveness of the classifier by its percentage of correct predictions. It determines how close the measurement comes to the quantity's true value. Equation (2.5) shows how *accuracy is* calculated:

$$Acc = \frac{|TN| + |TP|}{|FN| + |FP| + |TN| + |TP|} \qquad \text{Equation (2.5)}$$

Where; $|X|$ denotes the cardinality of set X.

2.8.6.2 Error Rate (Err.)

The error rate is the complement of *accuracy* defined as (*Err*) in Equation (2.6). It evaluates a classifier by its percentage of incorrect predictions. *Acc* and *Err* are general measures and can be directly adapted to ensemble outlier detection.

$$Err = \frac{|FN| + |FP|}{|FN| + |FP| + |TN| + |TP|} = 1 - Acc \qquad \text{Equation (2.6)}$$

2.8.6.3 Sensitivity / Recall (R) and Specificity (Spe.)

Sensitivity or recall (*R*), or true positive rate, is the proportion of examples belonging to the positive class which was correctly predicted as positive. The specificity (*Spe*) is the percentage of negative examples correctly predicted as negative. *R* and *Spe* are shown in Equation (2.7) and Equation (2.8) as:

$$R = \frac{|TP|}{|TP| + |FN|} \qquad \text{Equation (2.7)}$$

$$Spe = \frac{|TN|}{|FP| + |TN|} \qquad \text{Equation (2.8)}$$

2.8.6.4 Precision (P)

Precision (P) is a measure that estimates the probability that a positive prediction is correct. It is given by Equation (2.9) and may be combined with the recall originating from the F-measure. A constant β controls the trade-off between the precision and the recall, as in Equation (2.10). Usually set to 1.

$$P = \frac{|TP|}{|TP| + |FP|} \qquad \text{Equation (2.9)}$$

$$F - measure = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R} \qquad \text{Equation (2.10)}$$

2.8.6.5 Mean Squared Error (MSE)

MSE is used to evaluate individual detectors and ensemble as a whole. It measures the squared difference between predicted and actual values, with lower MSE values indicating better performance. Adaptive ensembles use MSE to monitor and adjust the contribution of individual detectors to improve accuracy by giving more weight to those performing well and reducing the influence of those not (Johnson, 2022).

2.8.6.6 ROC

Receiver Operating Characteristic which relates *R* and *spe*, give way for the comparison of models by showing False Positive Rates on the X-axis, the probability of target=Y when its true value is N, against True Positive Rate on Y axis, the probability of target=Y when its true value is Y.

2.9 Bias-Variance Trade-off for Outlier Detection

As quantification of bias-variance necessitates labeled data, the bias-variance trade-off is effectively explained in the context of supervised learning, for example, in classification. However, as there is no ground truth available, outlier detection problems must be tackled using unsupervised methods. By considering the dependent variable (actual labels) as an unobserved variable, the bias-variance trade-off can be measured. Most anomaly detection methods produce ratings for the data points' anomalousness, unlike classification. By converting the anomalousness scores to class labels, these anomaly detection techniques could be thought of as two-class classification problems with a majority class (normal points) and an uncommon class (anomalous points) (Rayana & Akoglu, 2016). The points which achieve scores above a threshold are considered anomalies and get label 1, and those below get label 0. However, deciding a threshold is difficult for heterogeneous detectors as they provide scores with different scaling and ranges. Fortunately, there exist unification methods that turn these anomalousness scores into probability estimates to make them comparable without altering the order of the data points (Gao & Tan, 2006; Kriegel et al., 2009).

The bias-variance trade-off for outlier detection could be explained using concepts from classification because the unsupervised outlier detection problem now resembles a classification problem with only real unobserved labels. According to Rayana and Akoglu (2016), the expected error of outlier detection can be separated into two main categories: irreducible error and reducible error, that is, noise). The accuracy of the detector can be improved by minimizing the reducible error. Additionally, the reducible error can be divided into two types: (i) error resulting from squared bias and (ii) variance-related error. While reducing both of these types

of errors, there is a trade-off. The difference between the prediction of the target model and the average models is the bias given as:

$$Bias(x) = b(x) - \overline{b(x)} \qquad \text{Equation (2.11)}$$

Such that, $b(x)$ returns the predicted value of x by the fitted weak/base learner or model b; and $\overline{b(x)}$ returns the average of all the predicted values of x predicted using all possible models fitted over all possible samples.

Variance is the difference between the mean of the obtained values and the predictions of each model obtained from various samples. This gives information of how much the models from the sample vary from the mean model.

$$Variance(x) = E_{sample}[\widehat{b(x)} - \overline{b(x)}] \qquad \text{Equation (2.12)}$$

Where, $\widehat{b(x)}$ returns the predicted value of x using the estimated (predicted) model on the sample. Noise is defined as the irreducible error that a model cannot predict.

In defining bias-Variance trade-off, two parts are necessary: bias-variance noise decomposition and bias-variance complexity tradeoff.

In considering a true function $y = f(x) + \epsilon$, where ϵ is normally distributed with zero mean and standard deviation σ, and given a set of training sets $D: \{(x_i, y_i)\}$, an unknown function $b(x) = w.x + \xi$ is fitted to the data by minimizing the squared

error $\sum[y_i - b(x_i)]^2$. So that, given a new data instance x^* with the observed value $y^* = f(x^*) + \epsilon$, the objective is to understand the expected error $\boldsymbol{E}[(y^* - b(x^*))^2]$. Thus, expected error is broken down as *bias*, *variance* and *noise* such that:

Expanding the error; $\boldsymbol{E}[(b(x^*) - y^*)^2]$

$$= \boldsymbol{E}\left[\left(b(x^*)\right)^2 - b(x^*)y^* - b(x^*)y^* + (y^*)^2\right]$$

$$= \boldsymbol{E}\left[\left(b(x^*)\right)^2 - 2b(x^*)y^* + (y^*)^2\right]$$

$$= \boldsymbol{E}\left[\left(b(x^*)\right)^2\right] - 2\boldsymbol{E}[b(x^*)]\boldsymbol{E}(y^*) + \boldsymbol{E}[(y^*)^2] \quad (\because \boldsymbol{E}(Z - \bar{Z})^2 = \boldsymbol{E}(Z^2) - \bar{Z}^2)$$

$$= \boldsymbol{E}\left[\left(b(x^*) - \bar{h}(x^*)\right)^2\right] + \left(\bar{b}(x^*)\right)^2 - 2\bar{b}(x^*)f(x^*) + \boldsymbol{E}\left[\left(y^* - f(x^*)\right)^2\right] + \left(f(x^*)\right)^2$$

$$= \boldsymbol{E}\left[\left(b(x^*) - \bar{b}(x^*)\right)^2\right] + \boldsymbol{E}\left[\left(y^* - f(x^*)\right)^2\right] + \left(\bar{b}(x^*) - f(x^*)\right)^2$$

$$= Var\left(b(x^*)\right) + \boldsymbol{E}(\epsilon^2) + Bias^2\left(b(x^*)\right) \quad = Bias^2\left(b(x^*)\right) + Var\left(b(x^*)\right) + \sigma^2$$

Equation (2.13)

The predicted error comprises three components, bias, variance, and noise, according to Equation (2.13). The learning algorithm is mostly to cause the reduction in bias, which results from the discrepancy between the average and the best prediction. The variance, which is the discrepancy between any prediction and the mean prediction, is frequently introduced by utilizing several training sets. Since the difference between the ideal prediction and the true function is so small, the noise is frequently insignificant. Typically, this noise is difficult to reduce as it is always unknown.

Hence the expected error becomes fairly equal to the sum of the squared bias and variance, as shown in Equation (2.14).

Expected error $\mathbf{E}[(h(x^*) - y^*)^2] = \text{Bias}^2(h(x^*)) + \text{Var}(h(x^*))$ Equation (2.14)

It is clear that when complexity (i.e., the number of weak learners) increases, the bias of an ensemble decreases and the variance of a model increases. This supports the claim in the work of (Rayana et al., 2017). Bias, variance, and complexity all have a close relationship as depicted in Figure 2.14. There is a trade-off between the three components and from this point; the ideal model complexity can be established. Furthermore, the bias as well as variance can be the least since the combination of both can reach the lowest in the error curve as shown in figure 2.14.

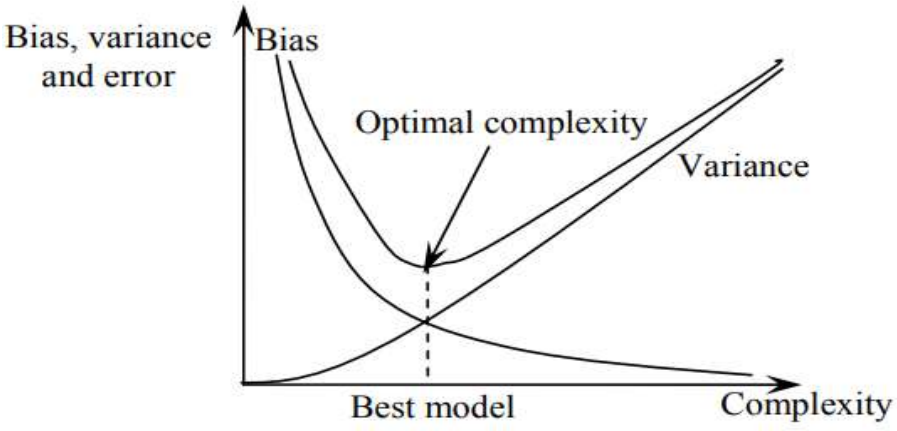

Figure 2.14: Bias, Variance, and Ensemble Complexity

The bias and variance trade-off could be conceptualized as follows:

(i) A detector with low bias is particularly flexible in fitting data well and will fit each training set differently, yielding high variation; and
(ii) The rigid detectors will have low variance and perhaps high bias.

The objective is to increase accuracy as far as possible by minimizing both bias and variance while enhancing the outlier ensemble approach by adding the concept of complexity in the ensemble formation (Bii, Rimiru & Mwangi, 2020).

2.10 Bias Variance Reduction Methods

Outlier ensemble learning tries to reduce the variance by combining different base detectors. An approach proposed by (Lazarevic & Kumar, 2005) created an ensemble to find outliers in high-dimensional, noisy datasets using randomly chosen feature subsets from the original features. (Prasada et al., 2020) used various training data subsamples to create an ensemble of trees to identify outliers based on the length of the path from the root to the leaves. (Aggarwal, 2017) discussed algorithmic trends, classification, and key components of outlier ensembles, such as base model diversity and model combination. (Zimek et al., 2013), without discussing the subsample selection, the subsampling technique was investigated analytically and empirically, and results were enhanced by layering an ensemble of subsamples on top of one another. (Aggarwal & Sathe, 2015) elaborated the concepts of classification borrowed into outlier detection, the bias-variance trade-off theory, and clarified some subsampling techniques. It further suggested some improved subsampling and feature bagging methods.

On base detector fusion, Rayana and Akoglu, (2015) presented unsupervised strategies to select a subset of trusted detectors while omitting inaccurate ones in an unsupervised way. It further proposed a method (CARE) that incorporates either sequential or parallel framework building blocks to reduce bias and variance (Rayana et al., 2016). These two phases correspondingly involve (i) consecutively eliminating outliers from the dataset to build a better data model on which outlierness is estimated (sequentially) and (ii) combining the results from individual base detectors and across iterations (independently). (Jiang et al., 2022) selects strong learners by AUC scores and introduces a Meta learner (stacking) to improve predictions and reduce bias.

2.11 Theories Underlying Outlier Detection Ensembles

Outlier detection ensembles are ML techniques used to identify anomalies or outliers in data. They typically combine multiple outlier detection algorithms to improve the accuracy and robustness of the results. The main theories applied in this study underlying outlier detection ensembles are:

2.11.1 The Diversity Theory

The Diversity Theory suggests that combining outlier detection methods that are diverse in terms of their underlying assumptions, feature selection techniques, and decision rules can lead to better detection accuracy and robustness. This is because the diversity in the methods helps to cover a wider range of possible outlier patterns and reduces the risk of false positives and false negatives (Liu et al., 2018).

2.11.2 The Decision Fusion Theory

The Decision Fusion Theory proposes that combining the decisions of multiple outlier detection algorithms can lead to better results than any individual algorithm. This is achieved by using a voting scheme, where each algorithm gives a vote for the presence or absence of an outlier, and the final decision is made based on the majority of votes (Kandhari et al., 2018).

2.11.3The Meta-learning Theory

The Meta-learning Theory suggests that a meta-classifier can be trained to combine the outputs of multiple outlier detection algorithms. The meta-classifier is trained on a labeled dataset of both normal and outlier instances and learns to predict whether a new instance is an outlier or not based on the outputs of the underlying algorithms (Rayana, 2016).

These theories can be applied in various ways to develop outlier detection ensembles. For example, Liu et al. (2014) proposed a method called Copula-Based Outlier Detection (COFOD), which combines multiple outlier detection methods using a clustering-based approach that maximizes diversity. Kandhari et al. (2018) developed

an ensemble method called Outlier Ensemble Detection (OED), which uses a voting scheme to combine the outputs of multiple outlier detection algorithms. Rayana (2016) proposed a meta-learning approach called Meta Outlier Detection (MetaOD), which trains a meta-classifier on the outputs of multiple outlier detection algorithms.

2.12 Related Works to the Proposed Problem

According to the research on classification ensembles, combining the outputs of many heterogeneous base algorithms will reduce the ensemble's overall variance (Aggarwal & Sathe, 2015). This is also true for outlier ensembles. Contrariwise, this combination does not provide enough ground for reducing bias in the outlier ensemble. For an ensemble to outperform its constituent classifiers in classification, two fundamental requirements must be met: (i) the base classifiers must be accurate (better than random), and (ii) they must be diverse (making uncorrelated errors) (Brownlee, 2019). Getting better-than-random accuracy in supervised learning is easy, and studies have shown that ensembles frequently perform better when the models are significantly diverse (Zhao et. al., 2019).

An ensemble based on diversity (Div-E) was proposed by (Schubert et al., 2012). For anomaly ensembles, it is unreasonable to assume that all detectors will be somewhat accurate (i.e., better than random), in contrast to classification ensembles, as some may not be able to recognize the (type of) anomalies in the given data. The diversity-based strategy Div-E would probably produce subpar results if inaccurate detectors existed since it is prone to choosing inaccurate detectors for the sake of diversity.

The Full-Ensemble (Full-E) chooses all of the detector findings and then all of the consensus outcomes to aggregate at both stages (Rayana & Akoglu, 2017). It is thus a naive method that, in the face of unreliable detectors, is likely to produce subpar results. Base detectors and consensus methods are arbitrarily chosen in selective ensemble procedures to produce the outcome. By doing this, the algorithms that are not chosen are eliminated and do not affect the outcome. Estimating detector/consensus result weights and using a weighted rank aggregation technique to combine the findings are alternatives to utilizing binary selection criteria.

In order to improve the relative influence of individual detectors on the final ranking by learning relative weights w_i for the individual rank lists, (Klementiev et al., 2007) previously suggested an unsupervised learning algorithm called ULARA for this type of rank aggregation (where P n_i=1 w_i= 1) (Rayana & Akoglu, 2015). Their strategy is informed by the idea that each individual rank list's relative contribution to the final ranking should be decided by how frequently it agrees with other rank lists in the pool. Large relative weights are assigned to rank lists that concur with the majority, whereas small relative weights are assigned to rank lists that do not. The total deviation from the average ranking of the various data points is used to calculate the agreement. Therefore, the objective is to distribute weights in a way that minimizes the overall weighted variance.

The bias-variance decomposition approach was employed in studies by Rayana, Zhong, and Akoglu (2017) that sought to better understand the theoretical underpinnings of ensemble performance. Initially, (Tumer & Gosh, 1996) provided a framework for analyzing the simple averaging combination rule based on manipulation used in bias-variance decomposition, stating that the ensemble error will be *M* times smaller than the error of the individuals if the classifiers are statistically independent and just equal to the average error of the individuals if the classifiers are correlated. When low-biased detectors with high variance are combined, the variance can be decreased, and when low variance detectors are combined, the bias can be decreased.

The SELECT technique by Rayana and Akoglu (2015) is based on the presumption that there are erroneous detectors out there that, when paired with accurate ones, can harm the ensemble as a whole. By using a pseudo-ground truth algorithm that averages the outputs for a vertical selection and uses majority voting for a horizontal selection among the many detectors, each of which may have biases in different directions, the method eliminates the inaccurate detectors. In order to lessen bias and variance and hence increase accuracy, it carefully chooses detectors and combines their outputs in another two steps.

As stated in the reduction of bias and variance, the other three techniques (Full-E, Div-E, and ULARA) can fall short of producing results that are more accurate than SELECT. Full-E damages the final ensemble because it mixes all of the base detectors' output, including the biased ones. ULARA determines relative weights depending on how well the detectors agree with one another, although it does not completely ignore detectors with significant bias. Just for the sake of diversity, Div-E chooses the more diverse detectors, which ultimately means choosing the ones with greater bias, lowering the accuracy as a whole. The SELECT strategy is heuristic since it cannot be relied upon to deliver the best result for various datasets. As a result, it may act unexpectedly for some data sets. Bias reduction in unsupervised learning, such as outlier detection, is a challenging task, and adopting the heuristic method to accomplish short-term objectives is quite reasonable to increase accuracy.

Prasada et al. (2020) combined both point and global outlier detection methods and proposed a method which handles data having imbalanced classes, which enhanced the performance of outlier detection but did not consider complete elimination of bias and variance. Other works, like MEOD (Jiang et al., 2022) select strong learners by their AUC scores and introduce a Meta learner by stacking method to improve predictions. This method was able to reduce ensemble bias and variance because of two levels of base learner combination. Of importance is that it agreed that different detection methods can be used together to detect outliers and reduce overall bias. Christy et al., (2015) proposed a technique where two algorithms are used for outlier detection. They suggested the use of distance-based outlier detection and cluster-based outlier detection algorithm as an ensemble. The later proved better than the former method.

According to Ma et al. (2021), large dimensional data does not make it difficult to identify outliers using distance-based techniques. It is widely believed that distance-based approaches identify all locations about equally as good outliers as the dimension of the data grows (Ma et al., 2021). Feldbauer and Flexer (2018) offered evidence to support the assertion that such a perspective is overly simplistic in their study on nearest neighbors in unsupervised distance-based outlier detection. They showed that in high-dimensional environments, distance-based techniques could

provide more disparate outlier scores. The ensemble bias and variance were not their main concerns (Feldbauer & Flexer, 2018).

In order to identify outliers, Yan et al. (2016) developed a novel hybrid approach called pruning-based K-nearest neighbor (PB-KNN), which combines the density-based, cluster-based, and K-nearest neighbor algorithm (KNN). The detection efficiency of KNN decreases with increasing data size. The PB-KNN method significantly reduces and prunes data dimensionality for outlier detection, outperforming the k-nearest neighbor (KNN) and local outlier factor (LOF) in terms of accuracy and efficiency, according to experimental results. When forming an ensemble, however, it does not take the learners' order or the bias-variance perspective into account (Yan et al., 2016).

All of the current attempts to create outlier ensembles either integrate the results from every base detector (Zhang et al., 2016) or enhance detector diversity to increase the likelihood that individual errors will be made (Shebuti, 2017; Kriegel et al., 2018). The concept of data locality in DCSO (Zhao & Hryniewicki, 2018) and dynamic learner fusion in (Wang & Mao, 2019), and learner combination methods in LSCP (Zhao et al., 2019) emphasized the importance of data relationships in local domains and showed that outliers could be global and local.

SVM is another effective algorithm for classification tasks. Its key advantage is its ability to generalize better by maximizing class margins. It tries to find the decision boundary that maximizes the distance between the nearest data points of different classes. By doing so, it improves the separation of data classes through contrast, making it easier to distinguish between them. It is also less susceptible to overfitting, as it focuses on finding the most significant features of the data. Overall, SVM has proven useful in various applications, from image recognition to text classification, and its principle can be borrowed into outlier detection tasks (Cervantes et al., 2020).

(Jaw, 2021) however, suggests that neither of these strategies would work well in the presence of inaccurate detectors as combining all outcomes, including inaccurate outcomes, deteriorates the overall ensemble performance.

Table 2.3: Critique of work done for outlier detection ensembles

Author	Year	Approach	Findings	Weaknesses
Feldbauer & Flexer	2018	Nearest neighbors in unsupervised distance-based outlier detection	Distance-based techniques may provide more disparate outlier scores in high-dimensional environments.	Does not completely ignore detectors with significant bias.
Shebuti	2017	Enhancing detector diversity	Enhancing detector diversity increases the likelihood that individual errors will be made.	Chooses inaccurate detectors for the sake of diversity.
Zhao & Hryniewicki	2018	Dynamic Classifier Selection	Data relationships in local data domains are important for outlier detection.	Did not consider bias-variance perspective
Aggarwal & Sathe	2015	Classification Ensembles	Combining outputs of many heterogeneous algorithms reduces overall variance.	Not adaptive in training its learners
Rayana & Akoglu	2017	Full-Ensemble	Base detectors and consensus methods are arbitrarily chosen	It is likely to produce poor results in the face of unreliable detectors.
Rayana, Zhong, & Akoglu	2017	Bias-Variance Decomposition by V-Select and H-Select	Variance can be decreased when low-biased detectors with high variance are combined, and the bias can be decreased when low variance detectors are combined.	Does not consider high dimensional data. Training is not adaptive, hence does not utilize errors of other learners.
Zhao et al	2019	Dynamic learner fusion	Outliers can be both global and local.	Not adaptive
Brownlee	2019	Classification Ensembles	For an ensemble to outperform its constituent classifiers in classification, the base classifiers must be accurate and diverse.	Does not utilize unsupervised approach to outlier detection which is key
Prasada et al.	2020	Combined Point and Global Outliers	Enhanced performance of outlier detection.	Does not consider elimination of bias and variance.
Ma et al.	2021	Distance-based techniques	Use of distance-based algorithms enhances outlier detection.	Does not consider ensemble bias
Jiang et al.	2022	Stacking of the outputs of base detectors	Able to reduce ensemble bias and variance using two levels of learner fusion.	It is not adaptive in its training approach.

2.13 Research Gap

Effective ensembles for outlier detection are still one of the challenging tasks in machine learning and data mining. Existing outlier detection ensembles utilize either parallel or sequential combination structures to fuse multiple detectors (weak base learners) in order to, hopefully, improve the overall detection performance by taking

a joint overall result (majority vote) from the detectors. The parallel or independent combination of base detectors is intended to reduce variance, while the serial or sequential combination is intended to reduce bias. The literature review has shown that there is no general approach for completely reducing the bias or variance or ensemble complexity problem. Detecting outliers is a difficult task because outliers are not only always very few but also mimic true labels in a dataset. Some learners may provide inaccurate results in an outlier detection ensemble, which may decrease the ensemble's overall performance, especially in the case of outlier datasets lacking ground truth. This depends on the type of data and the underlying rules of each learner. One of the directions of addressing the bias-variance dilemma is to study the underlying nature of the data. This way, both local and global outliers are considered. It is also vital that once outliers are discovered, a contrast between them is maximized to improve detection efficiency. More research is needed for the ever-challenging and emerging outlier problems in real-life applications. The review established that combining different base detectors in an ensemble induces diversity, enabling learning of different data characteristics and discovery of new information.

2.14 Chapter Summary

The literature has surveyed ensembles for outlier detection and methods for bias-variance decomposition. The review found that the goal of detection is to accurately predict the target class for each case in the data. Outlier detection is an unsupervised problem that requires unsupervised learning. The most commonly used learning algorithms have been discussed. The review found that different algorithms perform differently, and the choice of datasets may affect the performance of the base detectors. Some of the algorithms reviewed include distance-based metrics like KNN and density-based metrics like LOF. Different combinations of base detectors through bagging, boosting, and stacking methods (ensembles) were also discussed.

The review found that ensemble techniques produce models that perform better than individual algorithms used in constructing an ensemble. Outliers can be found in dense regions, that is, local to the other data points or neighbourhoods, and also, they can be global, that is, in far-off regions compared to other data. Identifying outliers in

local and global domains improve models' accuracy. Outliers can be detected and analyzed using different methods. The methods under user-labeled examples include supervised, semi-supervised, and unsupervised methods. When detecting outliers on assumptions about normal data or outliers, the methods used include statistical, proximity, and clustering-based methods. An ensemble of several algorithms that detect outliers in different domains improves outlier detection performance. The review also established that existing outlier detection ensembles combined all base model outcomes, including poor biased base models hurting the ensemble's overall accuracy. The bias-variance dilemma or tradeoff is difficult since high bias can reduce variance, and high variance can reduce bias. Most learning algorithms were developed for binary classes and are biased towards certain points. The literature found that the bias-variance dilemma can be handled by applying decomposition techniques such as bagging, boosting, and stacking.

The literature review found that detecting outliers is difficult because the size of outlier representation in most datasets is too small, and a large number of detection algorithms were designed to be biased towards the prediction of the majority class (inliers). Most outliers mimic true classes and are hard to identify. More research is needed for the ever-challenging outlier detection problem in real-life applications. Separating the margin between outliers and inliers can bring a contrast between the two and can make outlier detection efficient.

The review found that outlier detection models and ensembles can be tested and evaluated using different performance metrics. The common approaches for evaluating the performance of classifiers include cross-validation, k-fold cross-validation, random subsampling, confusion matrix, receiver operating curves characteristic (ROC), and area under the curve (AUC). The review also found that outlier detection has continued to be an active research field in data mining, and a unifying framework of outlier detection methods does not exist. The review concludes by providing the research gap identified in the study. The review study established a need for further research in outlier detection methods for high-dimensional datasets. The next chapter introduces the research methodology used in this study.

2.15 Conclusion

Ultimately, this study's goal is to enhance the performance and accuracy of outlier detection by a hybrid heterogeneous classifier ensemble while reducing bias and variance. The literature on ensembles demonstrates that merging results from different base methods reduces the ensemble's total variance, which is also the case for outlier ensembles. However, due to the absence of ground truth, controlled bias reduction is fairly challenging; hence this combination does not offer any evidence for doing so. Some heuristic approaches for reducing bias and variance, like SELECT, could build up the researcher's concept. This method is considered a hybrid ensemble. By translating the outlier-ness scores to class labels, the outlier identification problem can be viewed as a binary classification task with a majority class (inliers) and a minority class (outliers). Outliers with label 1 are the points with scores over a certain threshold (label 0 for inliers below the threshold). The unsupervised outlier detection problem is then transformed into a classification challenge using only unobserved labels. The researcher was able to explain the bias-variance trade-off for outlier detection using concepts from classification. Reducible error and irreducible error are the two primary parts of the predicted error in outlier detection (i.e., error due to noise). The reducible error can be reduced to the absolute minimum to increase the detector's accuracy. The reducible error can be divided into bias- and variance-related errors. While reducing both of these types of errors, there is a trade-off. The degree to which a detector's expected output deviates from the actual label (unobserved) throughout training data is known as the detector's bias. The predicted difference between a detector's output from one training set and its expected output from all the training sets is called a detector's variance.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, the study's methodology is explained. The first section provides an overview of the research methodology used. The second section describes the methods for data collection and pre-processing. The third section entails a detailed description of the proposed method. The chapter comes to a close with an introduction to the next chapter.

3.2 Methodology

The Cross-Industry Process for Data Mining was employed in this research. This methodology outlines a process for planning a data mining project in a structured manner. It is a reliable and consistent approach comprising six stages that are all linked together. Business understanding comes first, followed by data understanding, data preparation, modeling, evaluation, and deployment. The first through the fifth stages follow a loop pattern because of the complexity involved in data mining. The looping processes ensure that the outcomes are not only consistent but also reliable. Stage one is about understanding the business goals as it emphasizes stating the objectives or aims of a project in general. In relation to this study, the objectives below were put forward:

(i) To find out what classifiers constitute weak learners for constructing the base (detectors) for the outlier detection ensemble.

(ii) To identify different combination sequences or fusion strategies (order) from the selected base learners for the outlier detection ensemble.

(iii) To create a model for outlier detection that utilizes multiple weak learners in a hybrid ensemble structure to provide improved performance and accuracy while prioritizing the minimization of bias, variance, and classifier fusion order.

(iv) To evaluate the developed ensemble model for outlier detection accuracy.

The second stage is data understanding. This stage starts with data collection, followed by tasks that enable the identification of data quality issues, data insights, and detection of interesting patterns, which form hypotheses for hidden information. In this study, we adopted secondary datasets from the Outlier Detection Datasets (ODDS) library of Stony Brook University (Rayana, 2016), New York. Section 3.3 provides a detailed description of the datasets.

The third stage is data preparation, also known as the data preprocessing stage. This stage encompasses all activities that transform the initial raw data into a final cleaned dataset ready for analysis. It has four major steps: step one is data consolidation, which involves data collection, data selection, and data integration. The second step is data cleaning, which involves imputing all missing data values and eliminating the noise and inconsistency in the data. Step three is data transformation. Details of this step are discussed in section 3.4. The fourth step is data reduction, which entails feature reduction and data resampling. In this study, feature reduction was based on the importance of features as determined by their weight.

In phase four, modeling techniques are chosen and applied to a ready dataset to solve specific business needs. As part of the process of developing a new model, existing models are assessed and compared. In this study, an outlier detection model was proposed, as discussed in section 3.5.

The fifth phase is model testing and evaluation, where the models' accuracy and generalizability are assessed to ensure that they generalize well against unknown data. Data tables, charts, and other visualization techniques are frequently used to interpret knowledge patterns. Several experiments were carried out in this study to evaluate the models' performance, as discussed in Chapter 4.

The last phase is model deployment, which involves code writing and execution. It entails configuring the model code for scoring, classifying, or categorizing previously unseen data, as well as creating a mechanism for incorporating that data into the solution of the original business problem. The coded model should follow all

the steps discussed above to generate an executable model that transforms new unprocessed data in the same way as was during its development. Figure 3.1 below illustrates the cross-industry standard process for data mining.

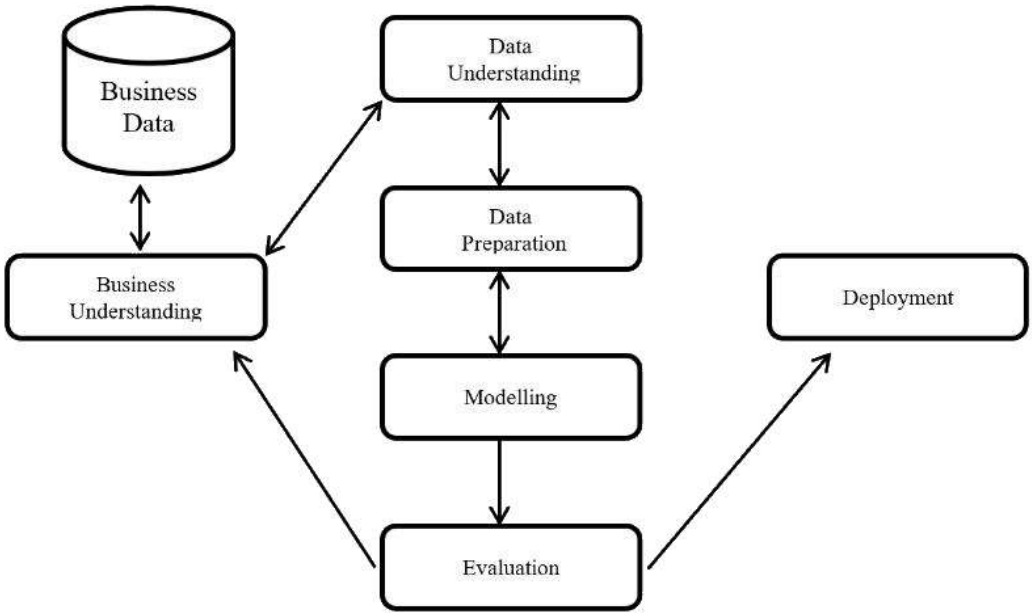

Figure 3.1: Cross-industry Standard Process for Data Mining

3.3 Data Collection Approaches

3.3.1 Dataset Description

This study focused on outlier detection; hence, outlier detection datasets from ODDS were adopted. The study used 10 multidimensional datasets, i.e. 3 digits datasets, 1 alphabet dataset, 4 medical datasets, 1 time-series dataset, and 1 biological dataset. These datasets had both high and low dimensionality. Each of them is explained below:

(i) Mnist Dataset

The MNIST dataset originally included 60,000 handwritten digits spanning from 0 to 9 for training and 10,000 for testing. In a gray-scale image with a size of 28px (height) x 28px (width), each digit is normalized and centered. The digit-0 class is sampled as inliers, and 700 images from the digit-6 class are sampled as outliers,

transforming it for outlier detection. In addition, 100 features are randomly selected from a pool of 784.

The classes have samples represented as:

Class 0 (digit 0)	6903 (90.793%)
Class 1 (digit 6)	700 (9.207%)

A total of 7603 images were sampled, of which digit 6 with 700 (9.207%) images was considered a rare/outlier class.

(ii) Letter Dataset

The original letter dataset included 26 capital letters in the English alphabet represented in 16 dimensions. To transform it for outlier detection, three data letters are randomly selected to represent the normal class, and then their pairs are joined randomly, doubling their dimensionality. A few letters that don't fit into the normal class are randomly selected for the outlier class. 1500 normal data points and 100 outliers (6.25%) are sampled across 32 dimensions. Class samples are represented as:

Class 0	1500 (93.75%)
Class 1	100 (6.25%)

A total of 1600 data points were sampled, of which class 1 with 100 (6.25%) samples was considered as rare/outlier class.

(iii) Cardio Dataset

Expert obstetricians divided fetal heart rate and uterine contraction scans into 3 groups in the Cardio dataset: pathogenic, normal, and suspect. The pathologic class of 176 (9.61%) was converted into outliers for the purpose of outlier detection, the normal class was considered as inliers, and the suspect class was discarded. The dataset has 21 attributes, 2 classes, and 1831 instances.

The classes have sample representations as follows:

Class0 (normal)	1655 (90.387%)

Class1 (pathologic)	176 (9.612%)
Class2 (suspect)	295 (discarded)

Pathologic class was considered a rare/outlier class and was down sampled to 176 points, while the suspect class was discarded.

(iv) Breastw Dataset

Breast Cancer Wisconsin (breastw) contains records of scans for breast cancer cases. It has two classes: benign and malignant. It is a 9-dimensional dataset containing 683 instances, of which 239 represent malignant tumors and were taken as the outlier class. The classes have samples represented as:

Class0 (benign)	444 (65.01%)
Class1 (malignant)	239 (34.99%)

(v) Ann Thyroid Dataset

The original thyroid disease (ann-thyroid) dataset has 3772 training and 3428 testing instances. The issue is determining whether or not a patient referred to the clinic is hypothyroid. Hence, three classes are created: normal, hyperfunction, and subnormal functioning. For outlier detection, both training and testing instances are used. The normal class is used as inliers, while the hyperfunction and subnormal classes are considered outliers. The dataset consists of 7200 instances, 3 classes, and 6 real attributes. The classes have samples represented as:

Class0 (normal)	6666 (92.58%)
Class1 (hyper function) + Class 2 (subnormal)	534 (7.42%)

Class 1 (hyper function) and class 2 (subnormal) were both considered as outliers.

(vi) Pima Dataset

The original Pima dataset for diabetes is a dataset for binary classification. The selection of examples from the wider database was subject to a number of restrictions including; all patients being female, at least 21 years of age, and of Pima heritage. It

has 8 attributes, 2 classes, and 768 instances, of which 268 (34.9%) are considered outliers.

The classes have sample representations as follows:

Class0 (normal)	500 (65.105%)
Class1 (diabetic)	268 (34.895%) - outliers

(vii) Vowels Dataset

The original Vowels dataset is a multivariate time series data with 9 male speakers' utterances of two Japanese vowels, 'a' and 'e.' A single utterance gives a time series range of lengths 7-29, with every point consisting of 12 features. For the purpose of outlier detection, each frame in the training set is treated as a separate data point. 50 (3.43%) outliers (Class 1-Speaker) are included in the sample. Classes 6, 7, and 8 are regarded as inliers, while other classes are disregarded. The ODDS dataset has 12 attributes, 2 classes, and 1456 instances. Classes have sample representations as:

Class0 (Class6 + Class7 + Class8)	1406 (96.566%)
Class1 (speaker)	50 (3.434%)

Class1 (speaker) was considered as the outlier.

(viii) Thyroid Dataset

The original thyroid disease (ann-thyroid) dataset has 3772 training and 3428 testing instances. The issue is determining whether or not a patient referred to the clinic is hypothyroid. Hence, three classes are created: normal, hyperfunction, and subnormal functioning. For outlier detection, only the training instances are used. The normal and subnormal classes are used as inliers, while the hyperfunction class is considered outliers. The dataset has 6 real attributes, 3 classes, and 3772 instances. The classes have samples represented as:

Class0 (normal) + Class 3 (subnormal)	3679 (97.534%)
Class1 (hyper function)	93 (2.466%)

Class 1 (hyper function) was considered as outliers.

(ix) Pendigits Dataset

The original Pen-Based Recognition of Handwritten Digits (pendigits) dataset has 16 features and classes 0 to 9. A collection of 250 samples from 44 writers, of which samples from 30 writers are taken as training and cross-validation data while the rest are for testing. For outlier detection, the original collection of handwritten samples is reduced to 6,870 points, of which 156 are outliers. The quantity of objects in one class, digit-0, is decreased by 10% because all classes have similar frequencies. The classes have sample representations as follows:

Class0 (digits1-9)	6714 (97.729%)
Class1 (digit-0)	156 (2.271%) – outliers

Digit 0 was taken as the outliers.

(x) Optdigits Dataset

The original Optical Recognition of Handwritten Digits (Optdigits) is a multi-class classification dataset that includes inliers made up of the instances of digits 1 through 9, and outliers made up of the instances of digit 0, which are down-sampled to 150. The dataset has 5,216 instances with 64 attributes and two classes with sample representation as follows:

Class0 (digits 1-9)	5066 (97.14%)
Class1 (digit-0)	150 (2.86%) - outliers

Table 3.1 shows a summary description of the datasets. The details of the datasets and scatter plots are presented in appendix1.

Table 3.1: Summary Description of Datasets Used in the Study

Dataset	Dim *(d)*	Inst. *(n)*	Normal	Outliers	Frequency *(%)*	Attribute
Mnist	100	7603	6903	700	0.0921 (9.21%)	All numeric
Letter	32	1600	1500	100	0.0625 (6.25%)	All numeric
Cardio	21	1831	1655	176	0.0961 (9.61%)	All numeric
Annthyroid	6	7200	6666	534	0.0742 (7.42%)	All numeric
Pima	8	768	500	268	0.3490 (34.9%)	All numeric
Vowels	12	1456	1406	50	0.0343 (3.43%)	All numeric
Thyroid	6	3772	3679	93	0.0247 (2.47%)	All numeric
Pendigits	16	6870	6714	156	0.0227 (2.27%)	All numeric
Breastw	9	683	444	239	0.3599 (34.9%)	All numeric
Optdigits	64	5216	5066	150	0.0286 (2.86%)	All numeric

3.4 Data Preprocessing

The datasets used in the study were first converted into comma-separated values (CSV) files from their original mat format using the python command. The CSV files were then loaded into Python programming language. Python comes with several useful libraries and frameworks for data analysis. NumPy, SciPy, and Scikit libraries, in particular, are used for scientific calculations, advanced computations, and learning data mining and data analysis. In all ten datasets, there were no missing values. Numerical and categorical features were sorted out, and identical columns where all data points had the same values were removed. Further preprocessing entailed data normalization since the data were not a distribution and the features also had negative values occurring frequently. The data was standardized (rescaled) to between 0-1. The weak learners were all based on distance metric, that is, K-Nearest-Neighbours, and therefore feature scaling was important. Otherwise, features with a large range would have greatly influenced computing the distance. To retain outliers in the datasets, Feature scaling using python standardizer was used to maintain the outlier relationships. This was done by subtracting the mean value from

all values and dividing it by the standard deviation. The resulting variable had a mean of zero and a standard deviation of 1, and most importantly, not skewed by outliers, and the outliers were still present with the same relative relationships to other values. Afterwards, the entire dataset was randomly shuffled to reduce the risk of learning models with a single class. For all models, k nearest neighbours were set to a minimum of 10 and a maximum of the square root of *n*, where *n* is the number of instances in a dataset.

3.5 Proposed Method

This work proposed the development of a hybrid model for outlier detection utilizing heterogeneous weak learners to provide improved performance. It proposed an optimized adaptive boosting technique for outlier detection. This method combined a set of heterogeneous weak learners to obtain an optimized composite model that provided more accurate and reliable predictions than a single model. It used weighted versions of a single training dataset combined with random subsampling. The training dataset needed not to be as large as that required by many other methods. Successive weak classifiers (detectors) were trained using reweighted copies of the training instances based on the earlier classifiers' accuracy. Error rates were computed in the context of an instance's local neighbours rather than the global training set. Testing focused on local domains or regions within a training dataset. At every iteration, the training instances were weighted according to the misclassifications (errors) of previous classifiers and parameters optimized. This allowed weak learners to focus on poorly classified patterns by previous classifiers. The final result was a fusion of carefully selected results from individual learners.

The main steps in the proposed method are presented in Algorithm 1. Step 1: Different detection algorithms (base detectors) were selected with unique measures to score individual vectors. Step 2: Using a selection approach, a subset of the detector results were chosen. Step 3: The chosen results were fused using different fusion methods to create intermediate aggregate outcomes. Step 4: Subsets of the outcomes from Step 3 were selected. Step 5: The selected subset of outcomes was fused. In a nutshell, the method entailed the creation of heterogeneous weak learners

that would become the base detectors for the ensemble, assessing the capability or competency of each weak learner/base detector before fusion, selection of optimal detectors, the fusion of selected base detectors, the building of a hybrid heterogeneous model, and finally testing the model. To ascertain the presence of outliers in our datasets, scatter plots were created, as depicted in Appendix AI.

3.5.1 Phase 1: Weak learners

The proposed outlier detection model uses different types of weak learners (heterogeneous) to create a set of diverse base models. Heterogeneous weak learners were selected because ensemble methods are very effective when base classifiers of dissimilar types are used (Rayana, Zhong, & Akoglu, 2017). Based on the differences between classifiers, the unique properties in data can be discovered or learned. When base learners of the same type (homogeneous) are used, the advantages of learner fusion are lost unless different data subsamples, parameters, or features are used for training each classifier (Zhang et al., 2020). The proposed model uses different base learners (heterogeneous) to construct a group of models to improve efficiency. Distance-based and density-based methods were selected as unsupervised methods for outlier detection, as discussed in section 2.7. This study concentrated on unsupervised outlier detection methods, which score each individual data point and allow ranking according to the likelihood that the point is an outlier. Motivated by the critical importance of data locality and dynamic learner fusion in DCSO and LSCP and the concept of heterogeneous detector formations in SELECT (as discussed in section 2.11), we adopted a distance-based algorithm to detect global outliers and a density-based algorithm to detect local outliers. The k-NN and LOF algorithms were chosen as base algorithms because they are well established for identifying outliers. The former is based on distance, while the latter is based on density. Distance-based methods attempt to identify global outliers that lay far (distant) from the rest of the data. Contrarily, density-based approaches attempt to find local outliers that lay in less dense regions compared to their k-nearest neighbours. Other algorithms, such as the distribution-based algorithms, were not considered as they assume a distribution (Jiang & An, 2008). Furthermore, distance

and density-based algorithms have since outperformed distribution and cluster-based algorithms in high-dimensional datasets (Papadimitriou et al., 2003).

The weak learners have two characteristics: high bias, that is, low degree of freedom, and high variance, that is, too much degree of freedom. Through the combination of several weak learners, ensemble methods aim to create a strong learner that performs better by attempting to reduce bias and variance (section 2.9). Finding the best or most appropriate weak learner from a vast pool of weak learners and enhancing the performance of the final model based on the data at hand are the two key challenges in weak learner selection (i.e., optimization or improvement). To overcome these difficulties, a bias reduction-variance reduction trade-off was employed (section 2.9). We point out that increasing the ensemble's complexity could lead to better model selection and that an ensemble of the top models could enhance overall performance.

As stated in Chapter 2, the choice of an ensemble model that is efficient or high-performing is based on the bias variance trade-off. By prioritizing the minimization of these components, we obtained a criterion for determining a well-performing ensemble. The goal of creating an ensemble using this trade-off is to improve the overall performance on subsequent test data, that is, to better generalize results. As previously noted, if the ensemble model is too sophisticated for the training data, it will learn (memorize) certain data elements, such as noise and issues with the underlying structure, leading to high variance and low bias. The detectors may show low performance in the testing data or exhibit poor generalizations. However, where the ensemble model is not sufficiently complex, in which case, it may be unable to represent the underlying data structure regardless of the amount of data provided, which results in excessive bias. Hence well-performing ensemble is a formation of the minimization of bias, variance and the number of base detectors such that:

Detector selection = *minimization_of* (bias, variance, complexity)

In as much as the well-performing base learners can be selected using the bias-variance trade-off criterion above, the base learners still do not necessarily generalize well because of the challenges related to complex ensemble formations. For an

ensemble to detect outliers well, improving and optimizing the selected base learners from the previous phase is necessary. This discussion is given in section 3.5.4.

The ensemble formation made it possible to increase generalization performance by joining numerous dissimilar base learners and training them to do similar tasks. We pointed out that the generalization error could be lowered if the base learners (detectors) on which the averaging is done disagree or make different mistakes. For our outlier detection ensemble model $H(\boldsymbol{x})$ comprising of M base learners (detectors), $h_1(\boldsymbol{x}), \ldots h_M(\boldsymbol{x})$; its representation is such that;

$$H(\boldsymbol{x}) = \sum_{i=1}^{M} w_i h_i(\boldsymbol{x}) \qquad \text{Equation (3.1)}$$

where, the detector coefficients w are represented as α, and $h(x)$ as $b(x)$ of the r[th] base detector in section 3.5.2. For a test dataset $X_{test} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, the ensemble mean squared error (MSE) is defined as:

$$MSE = \left(\frac{1}{NM}\right) \sum_{i=1}^{N} \sum_{i=1}^{M} \left(y_i - b_j(\boldsymbol{x_i})\right)^2 \qquad \text{Equation (3.2)}$$

By introducing the average detector $\bar{b}(\boldsymbol{x_i}) = \frac{1}{M} \sum_{i=1}^{M} b_j(\boldsymbol{x_i})$ the MSE was broken into bias and variance in terms (Eq. 3.3, 3.4) such that:

$$bias^2 = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left(y_i - \bar{b}(\boldsymbol{x_i})\right)^2 \qquad \text{Equation (3.3)}$$

$$variance = \left(\frac{1}{NM}\right)\sum_{i=1}^{N}\sum_{i=1}^{M}\left(\bar{b}(\boldsymbol{x_i}) - b_j(\boldsymbol{x_i})\right)^2 \qquad \text{Equation (3.4)}$$

Equations (3.3) and (3.4) enabled us to find out the effects of bias and variance in the ensemble. It was evident that the bias term depended on the label y, while the variance was not the case. The variance term of the ensemble was broken down as:

$$var\big(B(x)\big) = E\left[\left(\bar{B}(\boldsymbol{x}) - B(\boldsymbol{x})\right)^2\right] = E\left[\left(\bar{B}(\boldsymbol{x}) - E\big(B(\boldsymbol{x})\big)\right)^2\right]$$

$$= E\left[\left(\sum_{i=1}^{M} w_i b_i(\boldsymbol{x})\right)^2\right] - \left(E\left[\sum_{i=1}^{M} w_i b_i(x)\right]\right)^2$$

$$= \sum_{i=1}^{M} w_i^2\left(E\left[b_i^2(\boldsymbol{x})\right] - E^2[b_i(\boldsymbol{x})]\right) + 2\sum_{(i<j)} w_i w_j\left(E\left[b_i(x).b_j(x)\right] - E[b_i(x)]E\left[b_j(x)\right]\right)$$

Equation (3.5)

where the expectation E was taken w.r.t. dataset D. The first summation of Equation (3.5) indicates the lowest boundary of the ensemble variance, which essentially is the variance's weighted average of base learners (detectors). The subsequent summation encompasses multiplied terms of the base learners, which disappear when the base learners become uncorrelated. Hence, the segment of interest in Eq. (3.5) was in the second part so that the variance of the ensemble was lowered or reduced.

Equations (3.3) and (3.5) presented many possible ways that could be used to lower the expected error of the outlier detection model anchoring on the bias-variance-complexity paradigm. The following ways were implemented: (i) increased the number of dissimilar-type base detectors with the train data to the greatest extent to

reduce bias, (ii) combined selected base detectors using different combination methods to lower variance; and (iii) maintained a suitable computational complexity of the overall model. This way, the overall model improved accuracy by lowering the expected error by the rational trade-off dispensation based on this bias-variance reduction theory.

In defining a weak learner, a model is a weak learner if it shows a misclassification rate lower than 0.5, i.e., ($\varepsilon_t < 0.5$) or predicts the class labels more accurately than random guessing. Formally, given a dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ of d size features that yields the hypothesis $b(x_i)$, for each $(i = 1, \dots, N)$; and in addition, with some small probability, the learner's training error is slightly below the one of a random guesser; and since the expected training error of a random guessing learner is 0.5, it implies, that if error ε_t

$$\varepsilon_t = \frac{\sum_{i=1}^{N} I(b(x_i) \neq y_j)}{N} \leq 0.5 - \gamma \qquad \text{Equation (3.6)}$$

is true for every real γ such that $0 < \gamma \leq 0.5$, then $b(x_i)$ signifies a weak hypothesis, and the classifier is referred to as weak learner.

In the proposed method, the dataset $\mathcal{D}$ of *d* size features is first divided into training and testing sets as: $X_{train} \in D^{nxd}$ to signify training data with n data points, and $X_{test} \in D^{mxd}$ to signify test data with m points in the ratio 70%:30% respectively.

Next, a pool of heterogeneous base models $B = \{b_1, \ldots, b_R\}$ is generated by initializing the two weak learning algorithms (k-NN and LOF) using a variety of parameter settings for each. Both of these algorithms use *k* number of neighbours and a distance metric, which could be Manhattan or Euclidean. All base detectors are then trained and used to classify outliers in X_{train}. In this phase, an adaptive boosting technique is adopted to create a strong learner from the weak learners that were initialized. This technique samples a training set X_{train} from the initial dataset $\mathcal{D}$ according to a uniform distribution, meaning the initial weight distributions $\{\text{W}\}$ are given a value of $1/N$, where N is the number of training data. These weight distributions are updated adaptively in each iteration based on prediction results. Correctly predicted samples from weak learners receive low weights because they are considered to be easy samples. Difficult samples receive higher weights. In this manner, in the next iteration, the learners are able to focus on the difficult samples and attempt to provide better predictions. The weighted error $\varepsilon_t(i)$ of each learner is then calculated, as shown in Algorithm 6, Step 1, and (Eq. 3.12). The weak learner with $\varepsilon_t > 0.5$ is discarded, as shown in Algorithm 1, Step 2. The learner with the lowest error is selected and its outputs are used for future fusion (Algorithm 6, step 3). In Step 4 in Algorithm 6, every learner's error rate ε_n is estimated using (Eq. 3.12) and a combination weight α_t is calculated using (Eq. 3.14) for every learner.

The weighted coefficients of base learners are used to calculate the overall outputs. The greater a weight is, the more influence the corresponding learner has on the

overall results. Therefore, over T iterations, the ensemble considers l weak learners with different combination weights or weighted coefficients α_t, as shown in

Algorithm 6, Step 5. The results of the selected base learners form an outlier score matrix $O(X_{train})$ as follows:

$$O(X_{train}) = [\boldsymbol{b}_1(X_{train}), \ldots, \boldsymbol{b}_r(X_{train})] \in D^{nxR}, \quad \text{Equation (3.7)}$$

where $\boldsymbol{b}_i(.)$ is the score vector from the r^{th} base detector. Each base detector score $b_r(X_{train})$ is normalized using the Z-norm function (Aggarwal, 2020) as $z = (b_r - \mu)/\sigma$, where μ is the mean and σ is the standard deviation. This process is summarized in figure 3.2 part A.

3.5.2 Phase 2: Establishing Weak Learners Local Domains

The proposed model assesses the capability or competency of each base detector prior to fusion; but most outlier data have no actual labels or ground truth information. Therefore, $O(X_{train})$ in Eq. (3.7) was utilized to generate a simulated ground truth for X_{train} (called target) using two methods: by mean denoted as (ADAHO_Avg), averaging all scores, (Eq. 3.8) and by maximization denoted as (ADAHO_Max), obtaining a maximum score across all detectors, (Eq. 3.9).

$$ADAHO\_Avg = \frac{\sum_{i=1}^{n} b_i(X_{train})}{r} \in \mathcal{D}^{nx1} \quad \text{Equation (3.8)}$$

$$ADAHO\_Max = Max\{b_1(X_{train}), \dots, b_r(X_{train})\} \in \mathcal{D}^{nx1}$$ Equation (3.9)

Both ADAHO_Max and ADAHO_Avg generate scores for training data, unlike the conventional (generic) methods that take the global average, denoted here as (G_AVG) and global maximization, denoted here as (G_MAX), and which only generate scores for test data. An aggregation $\emptyset$ representing ADAHO_Avg or ADAHO_Max (ADAHO_Avg $\cup$ ADAHO_Max) was then performed across all base detectors to yield the $target$, which is used for initial detector selection. Thus,

$$target = \emptyset\big(O(X_{train})\big) \in \mathcal{D}^{nx1}.$$ Equation (3.10)

In terms of precision, avgkNN yields better results than kNN, so it is used here for selecting the local domain (Burnaev, Erofeev, & Smolyakov, 2015). For a test instance $X_{test}^{(j)}$, the local domain ℓ_j is derived as a set of its k-nearest training objects based on Euclidean distance (Zhao & Hryniewicki, 2018) as:

$$\ell_j = \left\{x_i \mid x_i \in X_{train},\ x_i \in avgkNN^{(j)}\right\},$$ Equation (3.11)

where $avgkNN^{(j)}$ is the average of a set of a $X_{test}^{(j)}$'s nearest neighbours bound by the ensemble. In an attempt to tame the curse of dimensionality, this technique was adopted by borrowing the concept of feature bagging (Lazarevic & Kumar, 2005). This process is captured in figure 3.2 part B.

Input: Dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ where: $x \in X$ and $y \in Y = \{+1, -1\}$; a set of

heterogeneous weak learning algorithms {L}; number of iterations T.

Initialize: a set of weights {W} by setting $\{W_1(i) = 1/N\}$ for $(i = 1, \dots, N)$

For $t = 1, \dots, T$ rounds;

1. Fit/train X_{train} using a weak learning algorithm $\{L\}$ to get a classifier (base detector)

$b_t(x)$: $X \rightarrow \{-1, +1\}$ and calculate weighted error of b_t for the i^{th} training instance as:

$$\varepsilon_t(i) = \sum_{j \in N(i)} W_t' \,.\, I(b_t(x_j) \neq y_j), \tag{3.12}$$

where;

$$W_t' = \frac{W_t(j)}{\sum_{l \in N(i)} W_l(l)},$$

where $N(i)$ indicates the neighborhood of the i^{th} training instance and $I(.)$ is an

indicator function such that:

$$I_n^t = \begin{cases} (b_t(x_i) \neq y_i) = 1 \\ (b_t(x_i) = y_i) = 0 \end{cases}.$$

2. If $\varepsilon_t(i) > 0.5$, set $W_{t+1}(i) = 1/N \ (i = 1, \dots, N)$ and go to 1.

3. ***1st Selection:*** Goal: select b_t with the lowest $\varepsilon_t(i)$

4. Estimate the weighted error rate of this base detector for X as

$$\varepsilon_n = \frac{\sum_{j \in N(i)} W_t' . I(b_t(x_j) \neq y_j)}{\sum_{i=1}^{N} W_n^t}. \tag{3.13}$$

5. Calculate the weighted coefficient of c for detector b_t as (3.14)

$$\alpha_t(i) = \frac{1}{2} ln \left\{ \frac{1-\varepsilon_n(i)}{\varepsilon_n(i)} \right\}.$$

6. Re-weight and update examples (i.e., those incorrectly classified receive more weight and those correctly classified receive less weight) as follows:

$$W_{t+1}(i) = \frac{w_t(i)}{z_t} \times \begin{cases} exp(-\alpha_t) \; if \; b_t(x_i) = y_i \\ exp(\alpha_t) \; if \; b_t(x_i) \neq y_i \end{cases},$$

where Z_t is a normalization factor. (3.15)

7. Iterate: for each test instance, output the outlier score of X_{test}

while (test instance $X_{test}^{(j)}$ in X_{test})

{ - Set local domain ℓ_j using kNN

- From $target$, pick elements in ℓ_j to generate local simulated $target^{\ell_j}$

while (detector b_r in B)

{ - Obtain outlier scores related to X_{train} in the local domain $b_r(\ell_j)$

- Using (1), assess the local competency of b_r (i.e.,between $target^{\ell_j}$ and $b_r(\ell_j)$) }

- **Do**: score margin maximization as described in **section 3.5.4**

2nd selection: **if** (Maximization or Averaging)

{ Pick $B_r^*(X_{test}^{(j)}$, where B_r^* has the greatest Pearson correlation with $target^{\ell_j}$ }

else { Pick subgroup $\mathfrak{g}$ among similar base learners and

add to B_r^*

fusion: if (AvgM or MaxA)

{ Output: $avg\left(B_r^*(X_{test}^{(j)})\right)$} **else**

Output: $max\left(B_r^*(X_{test}^{(j)})\right)$}}

} end while

Output: Final ensemble (strong) detectorB_t.

$$H_t(x) = sign\left(\sum_{t=1}^{T}\left(\sum_{i=1}^{N}\frac{\alpha_t(i)}{d^2(x,x_i)}\right)b_t(x)\right),$$

where $d^2(x,x_i)$ is the Pearson correlation between the new test instance and i^{th} training instance.

Algorithm 3.1: ADAHO_OAAE

3.5.3 Phase 3: Optimal Weak Learner / Base Detector Selection

In addition to the first selection that was based on the error rate of each weak learner, a second selection was performed to filter out noisy outcomes by obtaining the local simulated label $target^{\ell_j}$ for every test instance, where the values of $target$ with respect to the local domain ℓ_j were used as follows:

$$target^{\ell_j} = \left\{target_{x_i} \mid x_i \in \ell_j\right\} \in \mathcal{D}^{|\ell_j|x1}, \qquad \text{Equation (3.16)}$$

where $|\ell_j|$ is the size of ℓ_j. The local training outlier scores $O(\ell_j)$ (figure 3.2, part C) were obtained from the previously generated training score matrix $O(X_{train})$ as follows:

$$O(\ell_j) = [b_1(\ell_j), \ldots, b_r(\ell_j)] \in \mathcal{D}^{|\ell_j| x R}. \quad \text{Equation (3.17)}$$

To determine the competence of each base detector in a local domain, we calculated the Pearson correlation similarity measure between the base detector score $b_r\left(X_{train}^{\ell_j}\right)$ and simulated label $target^{\ell_j}$. This method was considered to be more reliable for outlier detection because it used a similarity measure for evaluating detectors instead of absolute accuracy (Schubert, 2013), which was helpful because most outlier datasets were unpredictable and imbalanced. We then picked the base detector b_r^* with the greatest similarity measure relative to the optimal base detector in a test sample $X_{test}^{(j)}$ and its outlier score $b_r^*\left(X_{test}^{(j)}\right)$ was retained as an intermediate result for later use.

3.5.4 Improving Detectors by Score Margin Maximization

The scores separate the anomalies from the rest of the data, resulting in high scores for the former and low scores for the latter. A contrast between the two scores helps distinguish the anomalies from the other data in a dataset. These scores, however, do not usually represent a clear contrast between anomalies and the rest of the data, thus the need for score optimization. This method creates a clear contrast by maximizing the anomalies' scores and minimising the other data's scores. This clear contrast between scores simplifies the problem of anomaly detection, making it more effective. The adaptive score threshold (Clark, Liu, & Japkowicz, 2018) would be

ideal for separating anomalies from a dataset, but the anomaly scores do not always match. Our method is inspired by the work in (Cervantes et al., 2020) as discussed in section 2.11, which classifies data by determining the maximum margin separating the hyperplane. This algorithm is effective and can better generalize by maximizing class margins, which improves the separation of data classes through contrast. In this context, we refer to this contrast as the score margin and use it in Equation (3.18).

3.5.4.1 Optimization Maximization

In order to bring a clear distinction between the anomalies and the rest of the data, the anomaly score distributions of the two must have a positive margin (Reunanen, Raty, & Lintonen, 2020). However, the analysed dataset, here initialized as a_j^{norm},

contains unknown anomalies that prevent accurate margin calculations. Therefore, a robust measurement is necessary to establish the magnitude of the score margin. Let $a_j^{norm}(\mathcal{D})$ signify the normalized scores of the analyzed data of the jth base model

and let $a_j^{norm}(\mathcal{D}_o)$ signify the normalized scores of the known anomaly samples by

the jth base model. We define the score margin as the

difference $a_j^{norm}(\mathcal{D}_o) - a_j^{norm}(\mathcal{D})$. The unknown or unseen anomalies present in

the dataset should not affect this difference, and so we introduce the median (MED) value of the score distributions because it takes the 50th percentile of the distribution. Since anomalies are rare and few by definition, using the median method provides a robust measurement of the anomaly scores as it is not greatly affected by the unknown or unseen anomalies in the dataset a_j^{norm}. For the jth base model, its

optimization maximization is based on the values of the parameters that maximize the distance $MED\left(a_j^{norm}(\mathcal{D}_o)\right) - MED\left(a_j^{norm}(\mathcal{D})\right)$ between the medians of

a_j^{norm} scores. By maximizing this score margin, base models are more likely to distinguish between anomalies and the rest of the data, which improves the efficiency of the overall model. We achieve this maximization through minimization of the negative score margin using an objective function to be minimized for the jth base model as:

$$-op_j^{max} = MED\left(a_j^{norm}(\mathcal{D})\right) - MED\left(a_j^{norm}(\mathcal{D}_o)\right) \quad \text{Equation (3.18)}$$

In every loop, the base models are optimized, one at a time, and provided with updated parameters that maximize the distances between the anomalies and the rest of the data distribution scores. Most anomaly detection ensembles only select well-performing base models for fusion (Xu et al., 2019); however, in our work, instead of only selecting the well-performing base models, we first optimize their parameters and then adaptively train them to detect anomalies before fusion. Our approach is the first to explicitly optimize the parameters of the base models within an adaptive framework for anomaly detection.

3.5.4.2 Diversity between Outcomes

The base model optimization by score margin maximization enlarges the contrast between the scores of the anomalies and other data. It tries to improve the detection accuracy of every base model. However, on top of this, the results of the base models must also be diverse. Their errors should differ (Kriegel et al., 2009) in that they can be fused into one model to address the shortcomings of the individual base models. The contrast maximization equation (3.18) gives no assurance of diversity of outcomes from the base models. To introduce diversity between the base models' outcomes, we use equation (3.18) to account for the correlation and then adjust the optimization to reduce it, similar to (Reunanen, Raty, & Lintonen, 2020). Applying Pearson correlation ρ shows the dependency between the score vectors $a_j^{norm}(\mathcal{D})$

and a_l^{norm} of the jth and lth base models. For any two vectors (s, t), Pearson correlation (Murphy, 2012) is defined as:

$$\rho(s,t) = \frac{cov(s,t)}{std(s)std(t)} \qquad \text{Equation (3.19)}$$

where cov represents the covariance and std the standard deviation. Once we calculate the correlation ρ between the base models' scores, we take the average and use it to measure the margin at that point. Thus, the final expression of the function of the jth base model takes the form:

$$f_j = -op_j^{max} \times \left(1 - \frac{1}{P}\sum_{l=0}^{j-1} \left|\rho\left(a_j^{norm}(\mathcal{D}), a_l^{norm}(\mathcal{D})\right)\right|\right) \qquad \text{Equation (3.20)}$$

where $|.|$ represents the absolute value, ρ represents the correlation measure, P represents the total number of base models, op_j^{max} represents the optimization maximization in equation (3.18), a_j^{norm} and a_l^{norm} represents the normalized scores of the jth and lth base models respectively, $\mathcal{D}$ represents the dataset with unknown anomalies and $\mathcal{D}_o$ represents the anomaly examples. Our function f_j uses $\mathcal{D}_o$ primarily for two reasons: first, to create a contrast between the anomalies and other data, and second, to obtain diverse outcomes from the models prior to final fusion.

3.5.5 Phase 4: Fusion of Base Detector's Outcomes

Because our base detectors were heterogeneous, their scores varied in terms of range and interpretation. Therefore, fusing scores directly would have been inappropriate; hence an agreement was needed within the ensemble. Based on the literature (chapter 2), agreement methods could be grouped into two main categories: rank-based and score-based methods. In rank-based methods, detector scores are ordered into ranked lists, which make all detector scores equivalent and allow for easy fusion. Aggregation is then performed to merge all of the scores into a single ranked list. Similarly, score-based methods convert outlier scores into probabilities using either exponential or Gaussian scaling based on posterior probabilities, regularization, or normalization. This makes the outlier scores across different detectors comparable, meaning a final score could be calculated via averaging or maximization. Because rank-based aggregation yielded a relatively crude ordering of data instances (Rayana & Akoglu, 2016), we adopted the score-based method, which converted outlier scores into probabilities and provided binary classes for instances with probabilities greater than 50% receiving a value of one (i.e., outliers) and those with probabilities less than 50% receiving a value of zero (i.e., inliers). We then applied ADAHO_MaxA to the top-*h* performing detectors with respect to their targets or applied ADAHO_AvgM, where the average of *h* chosen detectors with respect to their targets was taken as a subgroup score. The final score was obtained by taking the maximum among all subgroup scores. To reduce bias, ADAHO_MaxA and ADAHO_AvgM were used to reduce the risk of picking only the best-performing base detector. The bias in the ensemble was significantly reduced by the fact that only the top-*h* performing base detectors with respect to their targets were selected and that the *h* detectors did not increase overall variance. The final result is a fusion of carefully selected outcomes $h_1(x), h_2(x), .. h_J(x)$ from different optimized

learners. Finally, after all *E* boosting ensembles were trained, the totals $H^{(j)}(x)$ were combined and the $sign[H(x)]$ taken as the decision label of point x. The modeling process is depicted in Figure 3.2 (showing information flow from leaner creation to fusion. Colors represent different stages. WkL are heterogeneous weak learners) and the algorithm is summarized in figure 3.3.

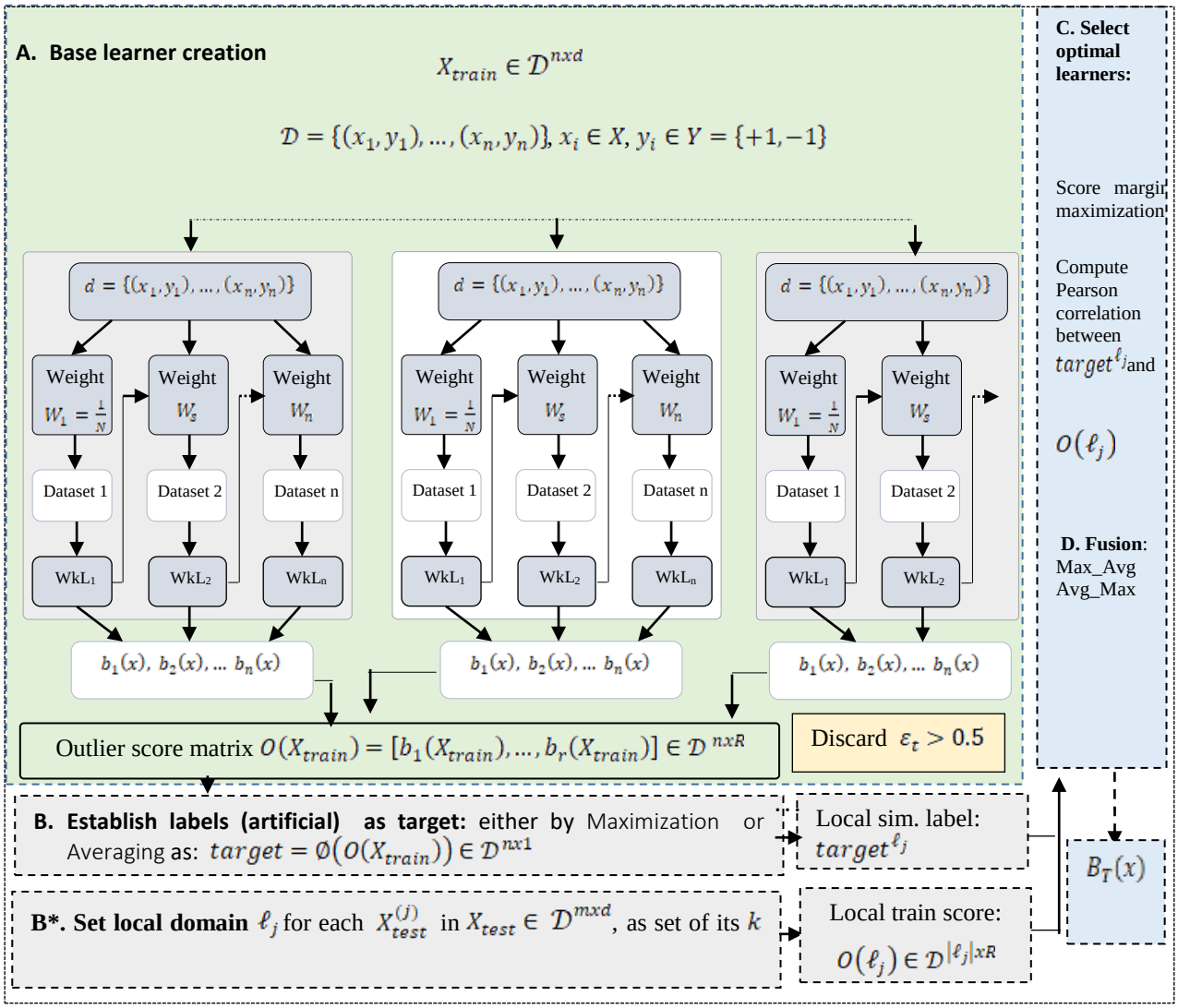

Figure 3.2: Proposed ADAHO_OAAE Model Ensemble.

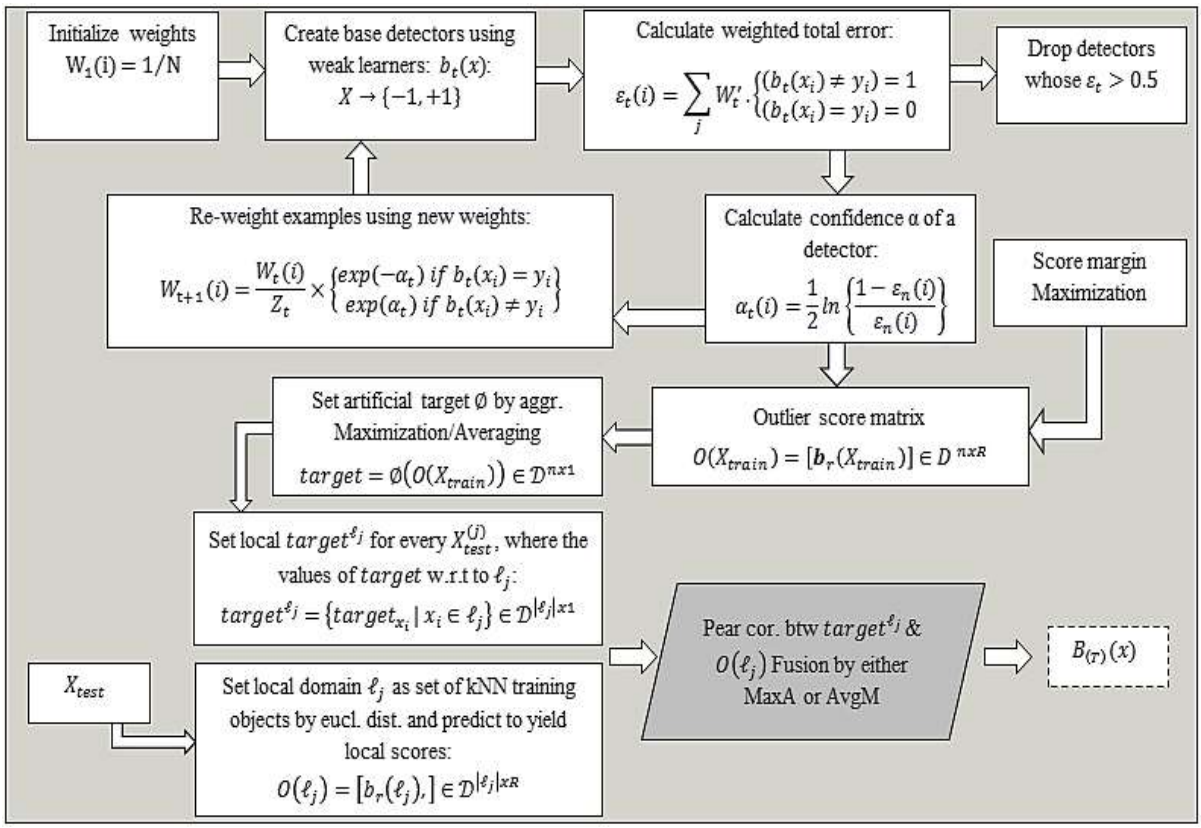

Figure 3.3: Summary of the Key Steps in Algorithm 6 (ADAHO_OAAE)

3.5.6 Phase 5: Testing and Validating the Proposed Method

Phase 5 involved testing the performance of the proposed method. Several weak learner combination methods were tested for outlier detection accuracy under AUROC. The different error rates were monitored and compared with the weights assigned to the weak learners in different training rounds. These error rates represented either a reduction or an increase in either the ensemble's bias or variance. The bias and variance were monitored as different combinations were applied. Stratified 10-fold cross-validation was used to validate the performance of the model. The initial dataset was first partitioned into 5 subsets with an approximately equal number of records in each subset. Each subset was used as the test partition, while the remaining subsets were combined to perform the role of the training partition. A paired T-test was employed for testing the difference in performances of the proposed model and the selected baseline algorithms. The paired t-tests were used to determine

whether the difference between two algorithms was significant or not. The paired T-test used a p-value set at 95% confidence interval. Several other tests were performed, including testing the merit for including each model in the method and the comparison of the method with the other well-known classification algorithms. The final test involved comparing the performance of the proposed model with other techniques for outlier detection. Values of ROC (Receiver Operating Characteristic) were utilized to gauge the effectiveness of the classifiers. The model's performance was assessed using other metrics, including True Positive, Precision, Recall, and AUROC scores.

3.6 Experimental Set Up

A heterogeneous outlier detection model was created using the proposed method discussed in section 3.5. The identification and qualification of appropriate but different types of weak learners for the heterogeneous ensemble for outlier detection were made. Both distance and density-based methods were selected. Experiments were done to prove the advantage of combining multiple learners as compared to just one. Also, further experiments were done by combining different kinds of weak learners and comparing their results to homogeneous combinations, as described in 3.5.1. To assess the capability or competency of each weak learner prior to fusion, tests at level 1 (weak learners), as described in 3.5.2, were done. The competencies of each base detector in their local domains were evaluated, as discussed in section 3.5.3. Optimization of base models was done to enlarge the contrast between inliers and outliers, as illustrated in 3.5.4. Because our model used weak learners of different types, i.e. heterogeneous, their outcomes were not combined directly. Several fusion techniques were compared in order to combine the weak learners. This is described in 3.5.5. A heterogeneous ensemble model was created and validated as described in section 3.5.6.

Experiments carried out included the following:

(i) Establishing the initial bias and variance of different kinds of weak learners (heterogeneous) combinations over the selected datasets.

(ii) Finding out the different error rates and various weight coefficients assigned to the weak learners over the different rounds of training so as to weed out those with high error rates.

(iii) Selecting and combining best performing or optimal learners and their overall effect in outlier detection performance.

(iv) Testing outlier detection performance by AUROC of the proposed method.

(v) T-Statistical test of proposed model verses other existing models

(vi) Comparisons of performance of proposed method and other existing algorithms in terms of kernel density estimates.

We contrasted the effectiveness of the suggested combination techniques with other existing combination techniques to reaffirm the performance of outlier detection. We also assessed the effectiveness of the suggested method with other existing ensemble formations such as ALOI (Schubert et al., 2013), BASE (Micenkova, McWilliams, and Assent, 2014), SELECT (Rayana & Akoglu, 2016) and ADAHO (Bii, Rimiru, & Mwangi, 2020) in order to demonstrate the superiority of the proposed method. The first two formations are symbolically referred to here as ALOI and BASE because of their relative design with respect to our proposed method. The results of the experiments were presented in the form of tables and graphs. Details of the experiment results are shown and discussed in Chapter 4.

3.7 Chapter Summary

This chapter presented the methodology known as Cross-Industry Process for Data Mining which provides a structured method for planning and executing a data mining project. The description of the datasets and the preprocessing process have been discussed. A detailed explanation of the proposed heterogeneous ensemble method for outlier detection has been provided. In a nutshell, for a given set of data in $\mathcal{D}$, an

outlier detection ensemble was built using $M = EB$, that is, M weak learners were

grouped into E boosting ensembles, each of which comprised B weak learners.

Basically, the ensemble had two levels: the first, or low level, contained all weak learners, and the second, or high level, contained all boosting ensembles. Because of this bi-level configuration, the model benefited from both bagging and boosting throughout her training. The bagging method generated multiple training subsamples from $\mathcal{D}$ on stratified sampling without replacement. Each subsample was then used

for training the boosting ensemble. In every boosting ensemble, the weak learners were of different kinds (heterogeneous). One ensemble was built with distance-based weak learners, while another with density-based weak learners. The ensembles executed adaptive boosting where weighted versions of the training dataset subsets d

were used. The training data was reweighted, with the weights based on the accuracy of the previous weak learners, and used to train new weak learners. Error rates were computed in the context of an instance's local neighbors, rather than a global training set. Testing instances focused on local domains or regions within a training dataset. The reweighting of data samples in every iteration allowed detectors to emphasize on patterns that were poorly predicted by the previous weak learners. Base detector scores were improved by a margin maximization process that brought a clear contrast between the classes. A final fusion of non-biased, low variance detectors was presented for outlier detection.

The next chapter provides the results of the experiments conducted in the study.

CHAPTER FOUR

RESEARCH RESULTS FINDINGS

4.1 Introduction

The findings of the experiments carried out for this research study are presented in this chapter. The experiments focused on examining different candidate base learners for the outlier detection ensemble, establishing their misclassification rates, local domains or regions of expertise, choosing optimal learners, testing the proposed method's detection performance, and comparing the proposed method with other existing methods. The chapter concludes by providing a summary of the experimental findings.

4.2 Examining the Base Detectors / Weak Learners

The framework creation process started with examining a set of candidate individual base classifiers for which KNN, DT, LOF, and LR were investigated. These four were considered simple yet met the base requirements of the description of weak learners from sections 3.5.1. KNN and LOF were selected because they are distance and density-based classifiers. The other two, DT and LR, widely used in homogeneous ensembles in most literature, were used to compare their performance to the distance and density-based methods in heterogeneous ensembles. This phase established a baseline for subsequent evaluations. Here, the classifiers were examined for three conditions necessary for a classifier to be admitted as a weak learner in an ensemble (sections 2, 3). That is, (i) they must exhibit a certain degree of bias and (ii) variance, (iii) and have an error rate less than 0.5, i.e., better than random guessing, and be simple and fast in terms of execution because several of them would be enjoined to form an ensemble. The results of the qualifying (individual) classifiers tested against the selected datasets are summarized in Table 4.1. KNN and LOF variants were selected for this study, as described in sections 3.5.1. Studies suggest that the base classifiers selected for a successful ensemble should be diverse (sections 2.4; 3.5.4.2) and have good individual performance. For the subsequent experiments and ensemble generation, these classifiers were

employed. Details of the experiment are first shown using the MNIST dataset, and a summary for the other nine datasets is provided thereafter.

4.2.1 Experiment 1: Establishing Weak Learners' Initial Bias over Different Number of Samples – Prior to Ensemble Formation (Criteria 1)

The experiment was conducted as per the description provided in section 3.5.1 Eq. 3.6. Table 4.1 represents the results of bias tests from 4 algorithms. We set the number of models for each classifier at 20 for consistency. We observed that with the increase in the number of training samples, the bias of the base learners decreased. This was seen as samples varied from the least at 100 samples to the highest at 5000 samples. The bias of the selected classifiers (KNN and LOF) decreased from 0.0028 to 0.0000007 and 0.0148 to 0.00003, respectively. The other classifiers did not show much change or reduction ability in bias and hence were not considered in other experiments as they did not satisfy criteria 1. Discussions are given in section 5.

Table 4.1: Initial Biases of Base Learners on MNIST Dataset over Different Samples

No. of samples	LR (20 models)	DT(20 models)	KNN(20models)	LOF(20 models)
100	2.13E-05	0.0000	0.0028	0.0148
300	1.63E-05	0.0000	0.0016	0.0129
500	1.61E-05	0.0000	0.0012	0.0101
700	1.56E-05	0.0000	0.0009	0.0085
1000	1.44E-05	0.0000	0.0007	0.0062
3000	1.43E-05	0.0000	0.0002	0.0015
5000	1.50E-05	0.0000	7.69E-07	0.00003

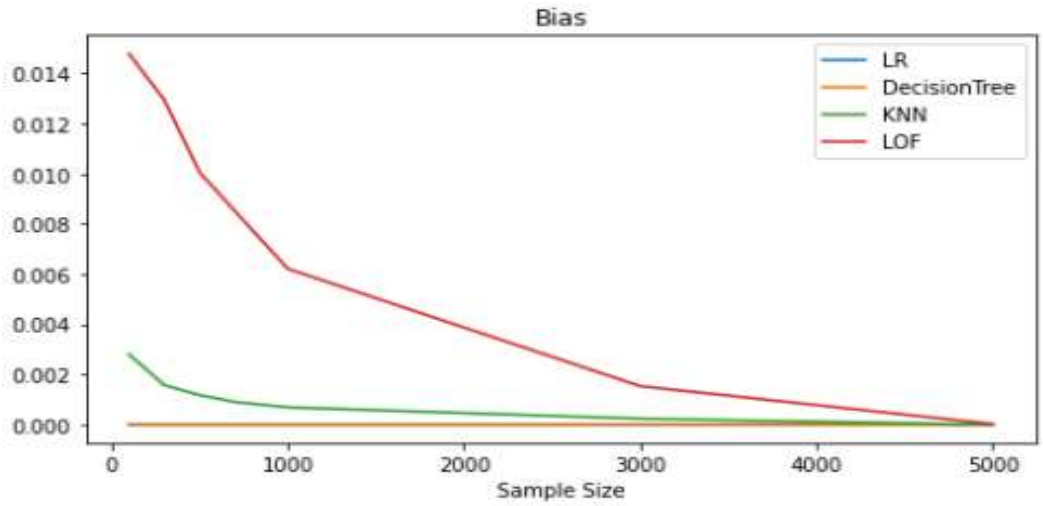

Figure 4.1: The Effect of Increasing the Training Samples on Model Bias

4.2.2 Experiment 2: Establishing Weak Learners' Initial Variances over Different Number of Samples (Criteria 2)

In experiment 2, the initial variances of the base classifiers were recorded, as shown in Table 4.2. Variations in the size of the training set on variance were not so huge. For the two selected classifiers, KNN and LOF, their variances tend to drop from 0.02 to 0.001 and 0.003, respectively. It implied that the more samples given in training, the more the classifiers' performance stabilizes. A high variance indicates that any little change in the dataset affects the behaviour of the classifier, and as a result, it generates different predictions in different training rounds.

Table 4.2: Initial Variances on MNIST over Different Number of Samples

No. of samples	LR(20 models)	DT(20 models)	KNN(20 models)	LOF(20 models)
100	0.0010	0.0000	0.0200	0.0190
300	0.0000	0.0000	0.0150	0.0200
500	0.0020	0.0000	0.0140	0.0180
700	0.0010	0.0000	0.0110	0.0190
1000	0.0010	0.0000	0.0110	0.0180
3000	0.0010	0.0000	0.0080	0.0140
5000	0.0010	0.0000	0.0010	0.0030

Figure 4.2 shows the different variances over different data sample sizes. The variance of each classifier seemed to drop with an increase in the size of the samples. It implies that the variance of a weak learner can be lowered by subsampling or by increasing the k-value in both KNN and LOF so that a high value of k lowers variance and a small value would cause model overfitting.

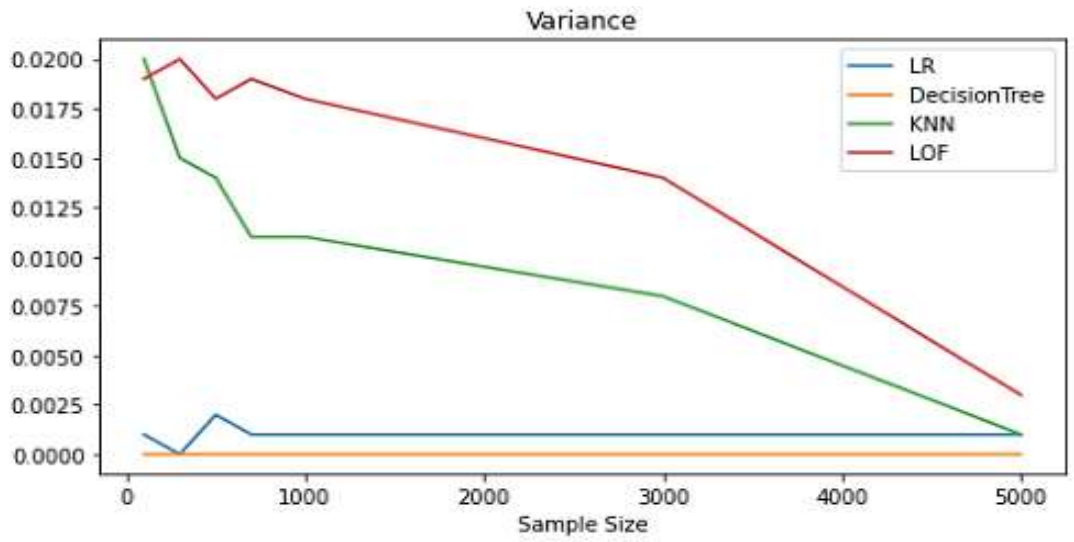

Figure 4.2: The Effect of Increasing Training Samples on Model Variance

For the MNIST dataset, figure 4.3 compares the classifiers bias with variance. The high variance indicated that the model was too flexible, and might have memorized the training data hence it does not generalize well because it is a sign of overfitting. This however qualified two classifiers, KNN and LOF, as good candidates for weak learners for the ensemble.

Table 4.3: Summary of Initial Biases and Variances on Selected Datasets

Dataset	KNN			LOF		
	BIAS	VARIANCE	MSE	BIAS	VARIANCE	MSE
Mnist	1.1741	0.0170	1.1912	1.3202	0.1044	1.4246
Letter	1.2024	0.0364	1.2388	1.2530	0.1203	1.3733
Cardio	1.2226	0.0280	1.2506	1.2842	0.1165	1.4006
Annthyroid	1.2420	0.0178	1.2597	1.3682	0.1167	1.4849
Pima	0.9994	0.0205	1.0199	1.0856	0.1037	1.1893
Vowels	1.2018	0.0387	1.2405	1.3477	0.1211	1.4687
Thyroid	1.2515	0.0189	1.2705	1.3396	0.1019	1.4415
Pendigits	1.3837	0.0327	1.4164	1.4004	0.1112	1.5116
Breastw	0.8499	0.0245	0.8745	1.2890	0.0708	1.3598
Optdigits	1.2950	0.0249	1.3199	1.3493	0.1022	1.4515

From Table 4.3 and figure 4.3, the initial biases and variances of the two detectors, KNN and LOF, were high. These high bias/variance base learners met the requirement of a 'weak' classifier as described in sections 3.5.1. These biases and variances would then be reduced through an ensemble whose goal is to learn the different characteristics of data – which is beneficial for outlier detection.

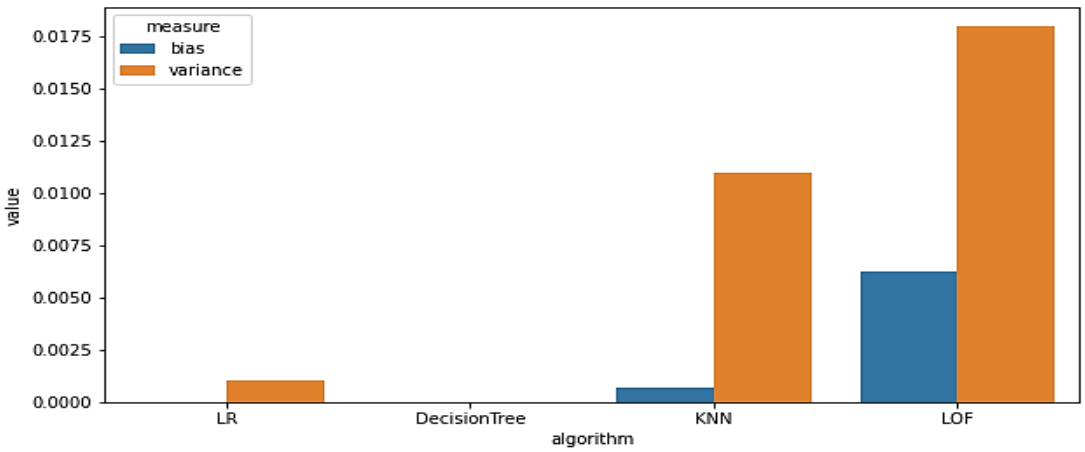

Figure 4.3: Comparison between Bias and Variance on MNIST Dataset

It was clear from Table 4.3 that as bias increased, variance decreased. This was true for all datasets applied in both KNN and LOF. For example, the bias of KNN over pendigits dataset was 1.3837, while its variance was 0.0327. This bias and variance combined, gave a total Mean Squared Error of 1.4164 (Table 4.3).

In this study, the two algorithms were selected for the reasons discussed in section 3: KNN and LOF. The former detects outliers far from the rest of the data instances, and the latter detects outliers in dense regions. KNN is distanced-based, while LOF is a density-based outlier detection algorithm. We generated g subsets of data for each ensemble formation. Each subset's samples were given uniform initial weights, which were then adjusted adaptively in every round of boosting depending on the error rates of the previous learners.

Initial Weak learner Selection by Trade-off: Bias-Variance

Based on the discussions of base detector selection by bias variance complexity trade-off in section 2.8, the outcomes of individual base algorithms were reported in Table 4.4. It was evident that the best performing base model was the Md_KNN in terms of the bias-variance complexity criterion over the two test datasets that were used. As regards detector complexity (i.e., the chosen model parameters like the choice of *k* or the distance metrics), LOF was able to give the most reduced selection values at model complexity of 30 and 37 base learners with selection values 0.110219 and 0.300299 respectively. Table 4.4 indicates the Selection value: = ($Bias^2$ + Variance) × influence of complexity (No. of models).

Table 4.4: Results of the Base Model's Bias and Variance.

Dataset	Base Model	Bias^2 $(E(\hat{y}) - y)^2$	Variance $E\left[(\hat{y} - E(\hat{y}))^2\right]$	Complexity	Selection value
Mnist	Lg_KNN	0.065413	0.053815	10	0.129164
	Avg_KNN	0.058445	0.081252	14	0.156271
	Md_KNN	0.075854	0.078639	12	0.171241
	LOF	0.032587	0.054014	30	**0.110219**
Letter	Lg_KNN	0.184577	0.084758	14	0.301290
	Avg_KNN	0.223143	0.105472	13	0.364666
	Md_KNN	0.254876	0.085347	8	0.351490
	LOF	0.128114	0.094756	37	**0.300299**

It was also observed that the bias of the LOF ensemble was the least of the 4 base models. To generate a committee of experts, we selected the models with the lowest error rates as they exhibited lesser bias and variance overall. The experiment also noted that the variance of the LOF model was comparatively large among the four base detectors, suggesting that different data characteristics were learned by its different base learners – which were important for outlier detection.

4.2.3 Experiment 3: Establishing the Error Rates of Learners (MNIST Dataset)

The third criterion was selecting weak learners for the ensemble by their error rates. Our method selected learners whose error rates $\varepsilon_t < 0.5$ as per the description in section 3.5.1. Tables 4.5 – 4.12 show the error rates ε_t, and the performance weights α_t assigned to every learner at every iteration during training. Lg_KNN utilized the largest distance between neighbours, and Ag_KNN utilized the mean distance, while Md_KNN utilized the median distance of the neighbourhood as the method for scoring outlierness of a data

point. To qualify learners as *'weak'*, those whose predictions were slightly better than random guessing ($\varepsilon_t < 0.5$) were chosen as intermediate learners for later use. The bold red font ε_t signified discarded learners based on this criterion.

Table 4.5: Weak Learners' Error Rates per Iteration using MNIST Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.1266	0.9657	0.1336	0.9347	0.1396	0.9093	0.2028	0.6844	0.1282	0.9585
1	0.2038	0.6814	0.2036	0.682	0.2146	0.6487	0.2688	0.5004	0.2081	0.6682
2	0.3348	0.3433	0.3306	0.3527	0.3216	0.3732	0.3481	0.3137	0.3129	0.3933
3	0.4084	0.1853	0.4104	0.1812	0.4229	0.1554	0.4212	0.1589	0.4139	0.1739
4	0.4725	0.0551	0.4625	0.0751	0.4472	0.1060	0.4588	0.0826	0.4652	0.0697
5	0.4795	0.0410	0.4881	0.0238	0.478	0.0440	0.4637	0.0727	0.4937	0.0126
6	0.4836	0.0328	0.4767	0.0466	0.4982	0.0036	0.4936	0.0128	0.3647	0.2775
7	**0.5203**	**-0.0406**	0.4904	0.0192	0.4996	0.0008	0.4993	0.0014	0.4206	0.1602
8	0.4844	0.0312	**0.5113**	**-0.0226**	**0.5038**	**-0.0076**	0.4957	0.0086	0.4713	0.0575
9	0.4865	0.0270	0.4909	0.0182	**0.5062**	**-0.0124**	**0.5015**	**-0.003**	0.4851	0.0298

Table 4.5 shows error rates per iteration using the MNIST dataset. The red bolded values indicate the misclassification rates where $\varepsilon_t > 0.5$ that were dropped. Out of 10 Lg_KNNs, 9 were set aside for future use. The same applied for Ag_KNN, and LOF. 8 Md_KNN weak learners were also put aside. For the combined heterogeneous weak learners, all 10 had misclassification rates less than 0.5 and were set aside for future use. A total of 5/50 learners were dropped, representing 10% of the total learners. Fig. 4.4 depicts the relationship between

error rates and weights. It was observed that as ε_t increased, α_t decreased.

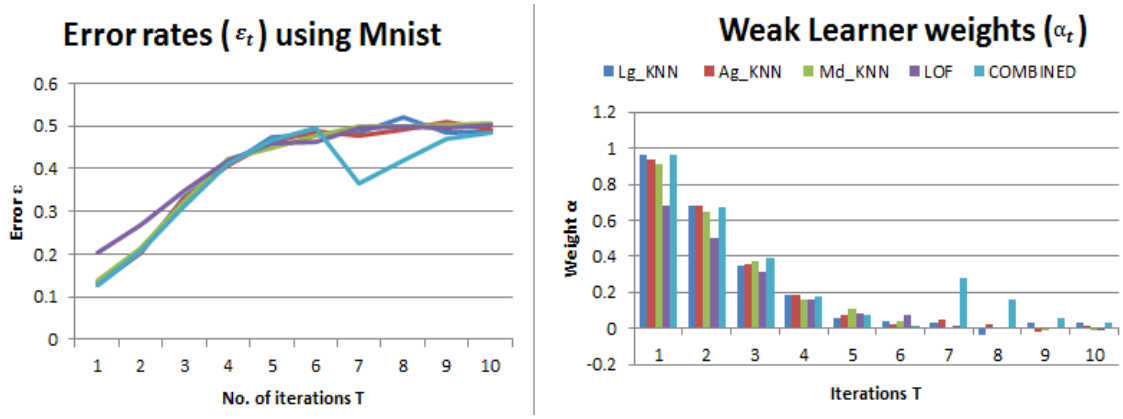

Figure 4.4: Comparison between Error Rates and Weights on MNIST Dataset

4.2.3 Experiment 4: Establishing Weak Learners' α_t using LETTER Dataset

Table 4.6 shows error rates per iteration using LETTER dataset. The red bolded values indicate where $\varepsilon_t > 0.5$. 9 Ag_KNN weak learners out of 10 were put aside for future use. The same applied for Lg_KNN and LOF. All 10 Md_KNN and 10 heterogeneous weak learners were put aside, since their $\varepsilon_t < 0.5$. A total of 3 learners were dropped during training representing 6% of the total learners.

Table 4.6: Weak Learners' Error Rates per Iteration using LETTER Dataset

	Lg_KNN (Learners = 10)		Ag_KNN (Learners = 10)		Md_KNN (Learners = 10)		LOF (Learners = 10)		COMBINED (Learners = 10)	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.1256	0.9702	0.1208	0.9924	0.1208	0.9924	0.1112	1.0393	0.1232	0.9812
1	0.2379	0.5821	0.2156	0.6457	0.2197	0.6337	0.2343	0.5921	0.2366	0.5857
2	0.3453	0.3199	0.2975	0.4296	0.3585	0.2909	0.3386	0.3348	0.3070	0.4071
3	0.4187	0.1641	0.4271	0.1468	0.4053	0.1917	0.4000	0.2027	0.3690	0.2683
4	0.4531	0.0941	0.4411	0.1183	0.4704	0.0593	0.4506	0.0991	0.4349	0.1309
5	0.4831	0.0338	0.4768	0.0464	0.4839	0.0322	0.4667	0.0667	0.4754	0.0492
6	0.4786	0.0428	0.4873	0.0254	0.4870	0.0260	**0.5057**	**-0.0114**	0.3999	0.2029
7	**0.5124**	**-0.0248**	**0.5026**	**-0.0052**	0.4830	0.0340	0.4996	0.0008	0.4534	0.0935
8	0.4870	0.0260	0.4983	0.0034	0.4977	0.0046	0.4889	0.0222	0.4638	0.0725
9	0.4917	0.0166	0.4981	0.0038	0.4927	0.0146	0.4875	0.0250	0.4703	0.0595

Figure 4.5: Comparison between Error Rates and Weights on LETTER Dataset

Figure 4.5 shows the relationship between ε_t and α_t. As error rates increased learner weights decreased per iteration. The negative $\alpha_t$were eliminated in rounds.

4.2.4 Experiment 5: Establishing Weak Learners' Error Rates (Cardio Dataset)

Table 4.7 shows error rates per iteration using the CARDIO dataset. The red bolded values indicate the misclassification rates where $\varepsilon_t > 0.5$. Out of 10 Lg_KNNs, 10 Ag_KNNs and 10 Md_KNNs, 2 weak learners from each were dropped because their $\varepsilon_t > 0.5$, i.e. 0.503, 0.5023, 0.5048, 0.5088, 0.5062 and 0.5008 respectively. Both LOF and the heterogeneous combined learners were all successful. A total of 6 learners were dropped representing 12%. This was attributed to the dense outlier clusters in the CARDIO dataset.

Table 4.7: Weak Learners' Error Rates per Iteration using Cardio Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.1169	1.0111	0.1176	1.0077	0.1218	0.9877	0.1232	0.9812	0.1218	0.9877
1	0.2413	0.5728	0.2473	0.5565	0.2407	0.5744	0.2434	0.5671	0.239	0.5791
2	0.3621	0.2831	0.3547	0.2992	0.3699	0.2663	0.3584	0.2912	0.3328	0.3478
3	0.4325	0.1358	0.4188	0.1639	0.4305	0.1399	0.443	0.1145	0.4234	0.1544
4	0.4665	0.0671	0.4543	0.0917	0.4612	0.0778	0.4598	0.0806	0.4539	0.0925
5	0.4887	0.0226	0.4797	0.0406	0.4825	0.035	0.4725	0.0551	0.4809	0.0382
6	0.4825	0.035	0.4854	0.0292	0.4781	0.0438	0.4934	0.0132	0.4574	0.0854
7	**0.5030**	-0.0060	0.4983	0.0034	**0.5062**	-0.0124	0.4964	0.0072	0.4919	0.0162
8	0.4964	0.0072	**0.5048**	-0.0096	**0.5008**	-0.0016	0.4992	0.0016	0.4541	0.0921
9	**0.5023**	-0.0046	**0.5088**	-0.0176	0.4737	0.0526	0.4917	0.0166	0.4998	0.0004

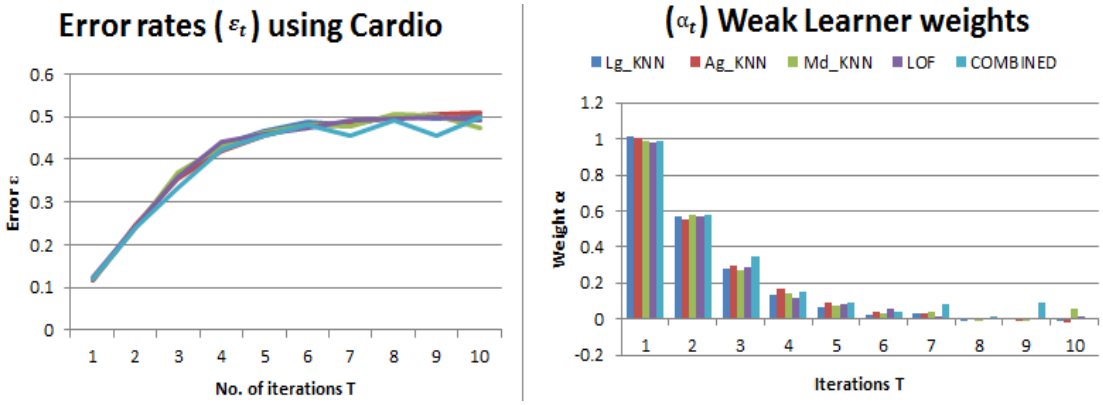

Figure 4.6: Comparison between Error Rates and Weights on Cardio Dataset

Figure 4.6 shows the relationship between error rates against weights when using the Cardio dataset. Again, as error rates increased, the weights α_t assigned to learners decreased since examples became harder in successive iterations.

4.2.5 Experiment 6: Establishing Weak Learners' Error Rates using Annthyroid

Table 4.8 represents misclassification rates per iteration using the ANNTHYROID dataset. The red bolded values indicate the leaner's error rates where $\varepsilon_t > 0.5$ that were dropped. Out of 10 Ag_KNN learners, 8 were set aside for future use as 2 learners were dropped. Lg_KNN and Md_KNN lost one learner each. The rest 18 were set aside for future use. All 10 LOFs and the 10 combined heterogeneous learners had their misclassification rates < 0.5 and were all set aside for future use. A total of 4 learners were dropped representing 8%.

Table 4.8: Weak Learners' Error Rates per Iteration (ANNTHYROID Dataset)

	Lg_KNN		Ag_KNN		Md_KNN		LOF		COMBINED	
	(Learners = 10)		(Learners = 10)		(Learners = 10)		(Learners = 10)		(Learners = 10)	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.0962	1.1201	0.0914	1.1483	0.0980	1.1098	0.1156	1.0174	0.0974	1.1132
1	0.1847	0.7424	0.1658	0.8078	0.1840	0.7447	0.1921	0.7182	0.1717	0.7868
2	0.3237	0.3684	0.3031	0.4163	0.3214	0.3737	0.3208	0.3750	0.2946	0.4366
3	0.4079	0.1863	0.4084	0.1853	0.3894	0.2249	0.4161	0.1694	0.3928	0.2178
4	0.4687	0.0627	0.449	0.1024	0.4523	0.0957	0.4462	0.1080	0.4462	0.1080
5	0.467	0.0661	0.4744	0.0512	0.4887	0.0226	0.4787	0.0426	0.4765	0.0470
6	0.4704	0.0593	0.4911	0.0178	0.4864	0.0272	0.4877	0.0246	0.4254	0.1503
7	0.4977	0.0046	0.5006	-0.0012	0.4891	0.0218	0.4980	0.0040	0.4652	0.0697
8	0.4992	0.0016	0.4961	0.0078	0.5009	-0.0018	0.4985	0.0030	0.4728	0.0545
9	0.5040	-0.008	0.5103	-0.0206	0.4999	0.0002	0.4941	0.0118	0.4793	0.0414

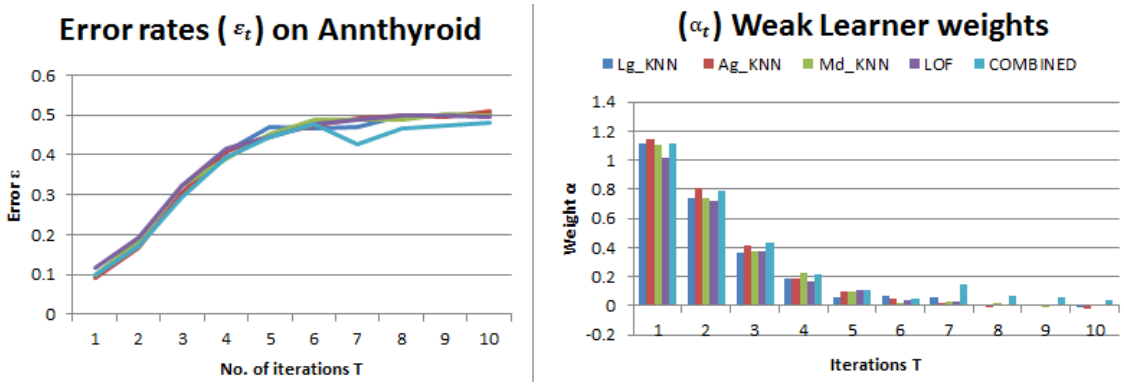

Figure 4.7: Comparison between ε_t and Weights α_t on ANNTHYROID Dataset

Figure 4.7 shows the relationship between training error rates against weights assignment on weak learners when using the Annthyroid dataset. With an increase in misclassification errors, decreasing learner weights were assigned.

4.2.6 Experiment 7: Establishing Weak Learners' ε_t using PIMA Dataset

Table 4.9 shows error rates per iteration using PIMA dataset. The red bolded values are $\varepsilon_t > 0.5$ that were dropped. Out of 10 Lg_KNN and 10 LOF learners, 8 of each were set aside for future use as 2 learners, from each, were dropped because their $\varepsilon_t > 0.5$. Ag_KNN and Md_KNN dropped one learner each. The 10 combined learners had their $\varepsilon_t < 0.5$ and none was dropped. 6 (12%) learners were dropped in total. The relationships with α_t assigned over different iterations are depicted in fig. 4.8. Learners with negative α_t were eliminated as well.

Table 4.9: Weak Learners' Error Rates per Iteration using PIMA Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.336	0.3406	0.3328	0.3478	0.3232	0.3696	0.336	0.3406	0.3376	0.337
1	0.4172	0.1671	0.4268	0.1475	0.4131	0.1756	0.4239	0.1534	0.4282	0.1446
2	0.4589	0.0824	0.4623	0.0755	0.4718	0.0565	0.4578	0.0846	0.4581	0.0840
3	0.4801	0.0398	0.4727	0.0547	0.4635	0.0731	0.4757	0.0486	0.4822	0.0356
4	0.4785	0.043	0.4885	0.023	0.4905	0.019	0.49	0.02	0.4744	0.0512
5	0.4928	0.0144	0.4935	0.013	0.5039	-0.0078	0.4947	0.0106	0.4979	0.0042
6	0.4848	0.0304	0.4925	0.015	0.4882	0.0236	0.5065	-0.013	0.4933	0.0134
7	0.5017	-0.0034	0.5081	-0.0162	0.4917	0.0166	0.4983	0.0034	0.4974	0.0052
8	0.5025	-0.0050	0.4992	0.0016	0.4989	0.0022	0.5001	-0.0002	0.4964	0.0072
9	0.4914	0.0172	0.4976	0.0048	0.4981	0.0038	0.5064	-0.0128	0.4946	0.0108

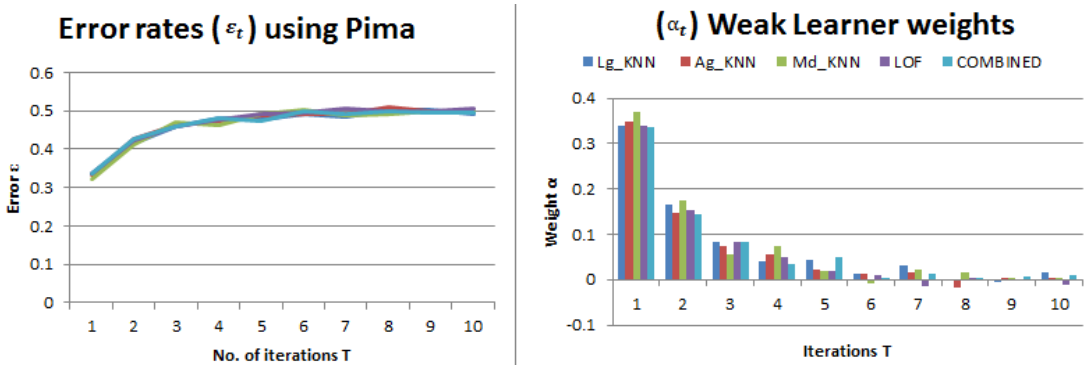

Figure 4.8: Comparison between Error Rates and Weights using PIMA Dataset

Figure 4.8: Comparison between Error Rates and Weights using PIMA Dataset

4.2.7 Experiment 8: Establishing Learners' Error Rates using Vowels Dataset

Table 4.10 represents misclassification rates per iteration using the VOWELS dataset. The red bolded values indicate the misclassification rates where $\varepsilon_t > 0.5$.

Out of 10 Lg_KNNs, 10 Ag_KNNs, and 10 LOF learners, 9 of each were set aside for future use. 1 learner from each was dropped because their error rates were greater than 0.5 (red font, bolded). Md_KNN lost two learners whose error rates were 0.5006 and 0.5034 respectively. The 10 combined heterogeneous learners had all their $\varepsilon_t < 0.5$ and therefore none was dropped. A total of 5

learners were dropped representing 10% of the total learners. Weights assignment in relations to these error rates are as depicted in Fig. 4.9. The learners whose coefficients were negative (black font, bold) were also dropped.

Table 4.10: Weak Learners' Error Rates per Iteration using VOWELS Dataset

	Lg_KNN (Learners = 10)		Ag_KNN (Learners = 10)		Md_KNN (Learners = 10)		LOF (Learners = 10)		COMBINED (Learners = 10)	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.0972	1.1144	0.0684	1.3058	0.0855	1.1849	0.0855	1.1849	0.1098	1.0464
1	0.1776	0.7663	0.1601	0.8287	0.1307	0.9474	0.1266	0.9657	0.1585	0.8347
2	0.2968	0.4313	0.2633	0.5144	0.2486	0.553	0.2055	0.6761	0.2186	0.6369
3	0.3648	0.2773	0.3478	0.3144	0.3709	0.2642	0.3304	0.3532	0.3298	0.3545
4	0.4408	0.1190	0.4055	0.1913	0.398	0.2069	0.3996	0.2036	0.4073	0.1876
5	0.4530	0.0943	0.4808	0.0384	0.4675	0.0651	0.4568	0.0866	0.4755	0.049
6	0.4697	0.0607	0.4852	0.0296	0.4711	0.0579	0.4476	0.1052	0.446	0.1084
7	0.4959	0.0082	**0.5033**	**-0.0066**	0.4996	0.0008	0.4954	0.0092	0.4819	0.0362
8	0.4919	0.0162	0.4944	0.0112	**0.5006**	**-0.0012**	**0.5079**	**-0.0158**	0.4592	0.0818
9	**0.5263**	**-0.0526**	0.4947	0.0106	**0.5034**	**-0.0068**	0.4949	0.0102	0.4651	0.0699

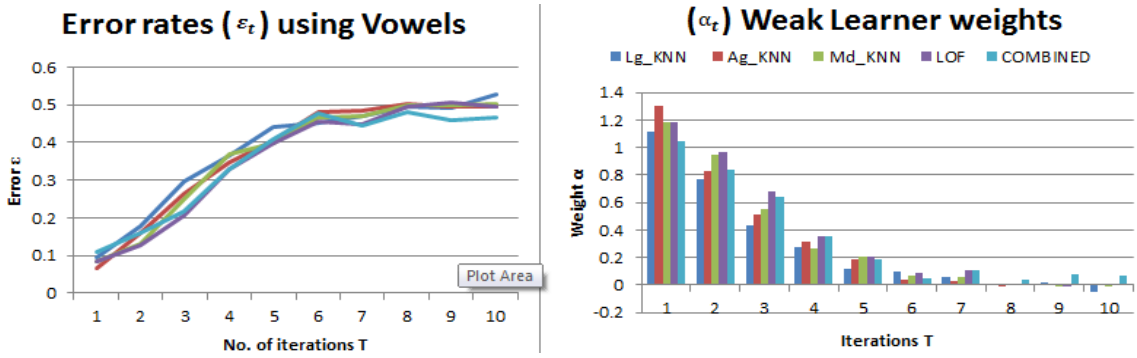

Figure 4.9: Comparison between ε_t and Weights α_t using Vowels Dataset

4.2.8 Experiment 9: Establishing Weak Learners Error Rates (Thyroid Dataset)

Table 4.11 shows training error rates per iteration using the THYROID dataset. The red font bold values indicate the misclassification rates where $\varepsilon_t > 0.5$. Out

of 10 Lg_KNNs, 10 Ag_KNNs, 9 of each were set aside for future use as one learner from each group was dropped because of their misclassification rates

being greater than 0.5 (red font, bold) i.e. 0.5171 and 0.5079 respectively. The other three, i.e. Md_KNNs, LOFs and the combined heterogeneous learners had their misclassification rates less than 0.5 and none was dropped. A total of 2 learners were dropped, while training using the THYROID dataset representing 4% of the total learners. Fig. 4.10 depicts the weights at every iteration.

Table 4.11: Weak Learners' Error Rates per Iteration using THYROID Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.0588	1.3865	0.0603	1.3731	0.0573	1.4002	0.0543	1.4287	0.0711	1.285
1	0.1395	0.9097	0.147	0.8792	0.1374	0.9185	0.1397	0.9089	0.1233	0.9808
2	0.2582	0.5277	0.2767	0.4804	0.2859	0.4577	0.2923	0.4421	0.2698	0.4978
3	0.3571	0.294	0.3932	0.2169	0.3648	0.2773	0.3548	0.299	0.3844	0.2355
4	0.447	0.1064	0.4325	0.1358	0.3973	0.2084	0.4527	0.0949	0.4346	0.1316
5	0.4603	0.0796	0.4643	0.0715	0.4711	0.0579	0.4541	0.0921	0.4644	0.0713
6	0.4623	0.0755	0.497	0.006	0.4978	0.0044	0.4901	0.0198	0.2966	0.4318
7	0.475	0.05	0.4791	0.0418	0.4699	0.0603	0.4914	0.0172	0.4233	0.1546
8	**0.5171**	**-0.0342**	0.4985	0.003	0.488	0.024	0.4927	0.0146	0.4105	0.1809
9	0.4948	0.0104	**0.5079**	**-0.0158**	0.4962	0.0076	0.4963	0.0074	0.4963	0.0074

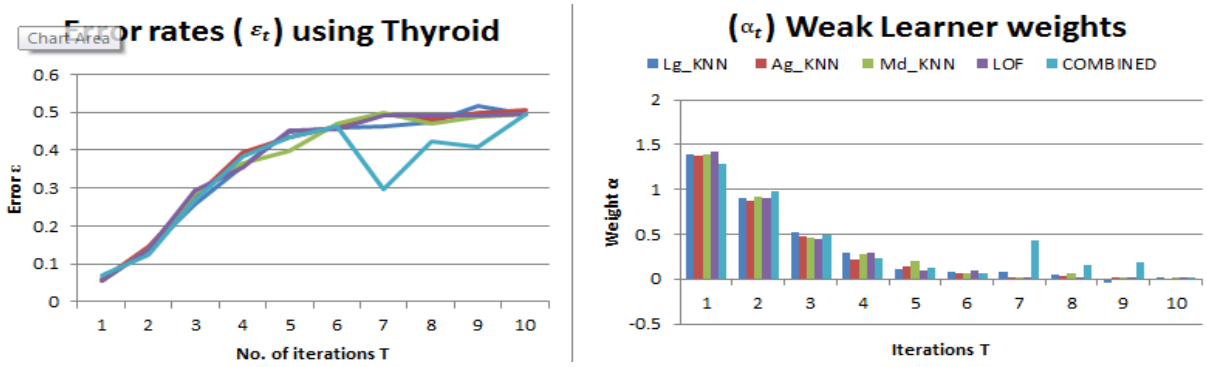

Figure 4.10: Comparison between ε_t and Weights α_t using THYROID Dataset

4.2.9 Experiment 10: Establishing Weak Learners' Error ε_t (Pendigits Dataset)

Table 4.12 represents training error rates per iteration using the PENDIGITS dataset. The red font bold values indicate the error rates $\varepsilon_t > 0.5$. Out of 10

Lg_KNNs, 10 Md_KNNs, and 10 LOFs, 9 of each were put aside for future use as one learner from each category was dropped due to misclassification rates greater than 0.5 (red font, bold) i.e. 0.5018, 0.5009 and 0.5092 respectively. For Ag_KNN, 2 weak learners were dropped as their error rates were higher at 0.5011 and 0.5212. All ten of the combined heterogeneous weak learners had their misclassification rates < 0.5 and all were set aside for future use. A total of 5 learners were dropped while training using the PENDIGITS dataset, which represents 10% of total learners. Negative learner coefficients were eliminated.

Table 4.12: Weak Learners Error Rates per Iteration over PENDIGITS Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.0776	1.2377	0.0738	1.2649	0.0744	1.2605	0.0886	1.1654	0.0824	1.2051
1	0.1277	0.9607	0.1249	0.9734	0.128	0.9594	0.1512	0.8626	0.1177	1.0072
2	0.248	0.5547	0.2675	0.5037	0.2571	0.5305	0.2713	0.494	0.2254	0.6172
3	0.3663	0.2741	0.3619	0.2836	0.3633	0.2805	0.3729	0.2599	0.3287	0.357
4	0.4258	0.1495	0.4404	0.1198	0.4308	0.1393	0.4163	0.169	0.4236	0.154
5	0.4635	0.0731	0.4606	0.079	0.4618	0.0765	0.4578	0.0846	0.4567	0.0868
6	0.47	0.0601	0.4739	0.0522	0.4757	0.0486	0.4939	0.0122	0.3409	0.3296
7	**0.5018**	**-0.0036**	0.4801	0.0398	0.4932	0.0136	0.4964	0.0072	0.433	0.1348
8	0.4919	0.0162	**0.5011**	**-0.0022**	0.4735	0.053	0.4899	0.0202	0.428	0.145
9	0.4852	0.0296	**0.5212**	**-0.0424**	**0.5009**	**-0.0018**	**0.5092**	**-0.0184**	0.4715	0.0571

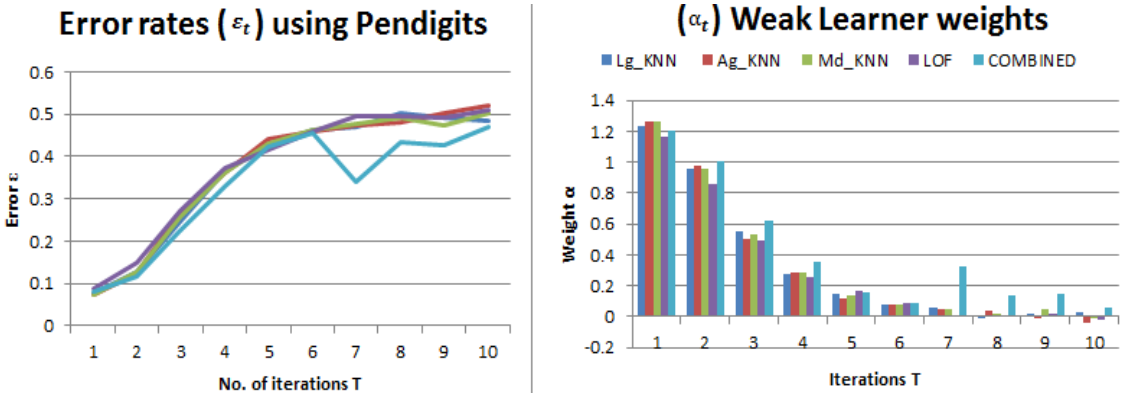

Figure 4.11: Comparison between ε_t and Weights α_t using Pendigits Dataset

4.2.10 Experiment 11: Establishing Learners' Error Rates (Breastw Dataset)

Table 4.13 shows training error rates per iteration using the BREASTW dataset. The red font bold values indicate the misclassification rates where $\varepsilon_t > 0.5$. Out of 10, 2-Lg_KNN learners were dropped with ε_t 0.5004 and 0.5097, respectively. 1 Ag_KNN was dropped having $\varepsilon_t = 0.5095$. Both Md_KNN and the combined heterogeneous learners dropped 3 weak learners each. All 10 LOF weak learners were retained as their misclassification rates were less than 0.5. Overall, a total of 9 learners were dropped while training using the BREASTW dataset, which represents 18% of the total learners. The high error rate learners were assigned lesser weights in every iteration as depicted in figure 4.12. The negative learner coefficients that were assigned were also dropped.

Table 4.13: Weak Learners' Error Rates per Iteration over BREASTW Dataset

	Lg_KNN (Learners = 10)		Ag_KNN (Learners = 10)		Md_KNN (Learners = 10)		LOF (Learners = 10)		COMBINED (Learners = 10)	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.2574	0.5298	0.2898	0.4482	0.2484	0.5536	0.2862	0.457	0.2628	0.5157
1	0.3882	0.2274	0.3632	0.2808	0.3803	0.2441	0.393	0.2174	0.3786	0.2477
2	0.4358	0.1291	0.47	0.0601	0.4418	0.1169	0.4514	0.0975	0.4513	0.0977
3	0.4456	0.1092	0.4864	0.0272	0.4764	0.0472	0.482	0.036	0.4734	0.0533
4	0.4643	0.0715	0.4574	0.0854	0.4787	0.0426	0.4955	0.009	0.4772	0.0456
5	0.4927	0.0146	0.485	0.03	0.4936	0.0128	0.4859	0.0282	0.4967	0.0066
6	**0.5004**	**-0.0008**	0.493	0.014	0.4678	0.0645	0.4849	0.0302	0.4935	0.013
7	**0.5097**	**-0.0194**	0.4979	0.0042	**0.5038**	**-0.0076**	0.4876	0.0248	**0.5013**	**-0.0026**
8	0.4849	0.0302	**0.5095**	**-0.019**	**0.5233**	**-0.0466**	0.4943	0.0114	**0.5101**	**-0.0202**
9	0.4989	0.0022	0.496	0.008	**0.5055**	**-0.011**	0.495	0.01	**0.5024**	**-0.0048**

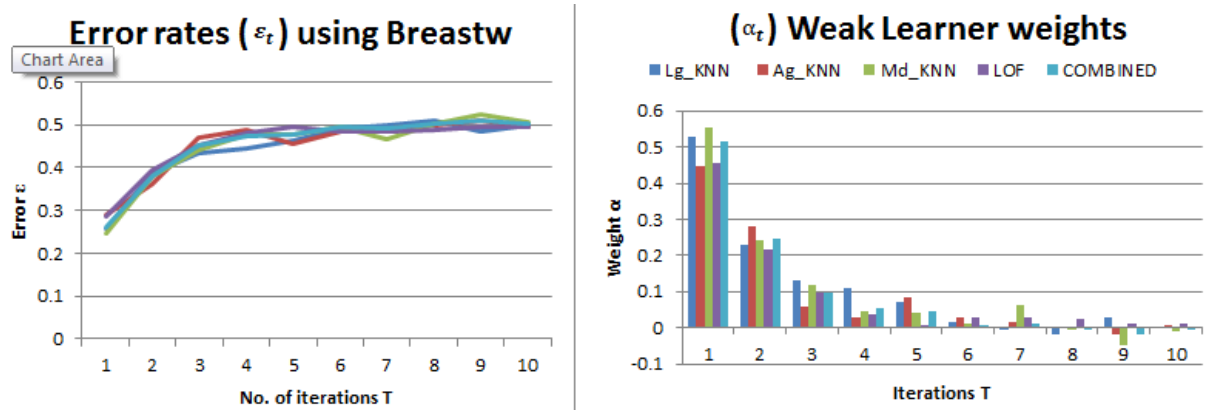

Figure 4.12: Comparison between ε_t and Weights α_t using Breastw Dataset

4.2.11 Experiment 12: Establishing Weak Learners' ε_t using Optdigits Dataset

Table 4.14 represent the training error rates of 10 iterations using the OPTDIGITS dataset. The red font bold values indicate the learners'

misclassification rates where $\varepsilon_t > 0.5$ and that were dropped. Out of 10 Lg_KNNs,

10 Ag_KNNs, 10 Md_KNNs and 10 LOFs, one weak learner was dropped in each group with error rates ε_t of 0.5111, 0.5021, 0.5095, and 0.5021 respectively.

All the 10 heterogeneous weak learners were retained as their misclassification rates were less than 0.5. Overall, a total of 4 learners were dropped while training using the OPTDIGITS dataset, which represents 8% of the total learners. Figure 4.13 depicts the assignment of decreasing weights to the different learners over the iterations depending on the error rates.

Table 4.14: Weak Learners' Error ε_t per Iteration using OPTDIGITS Dataset

	Lg_KNN		**Ag_KNN**		**Md_KNN**		**LOF**		**COMBINED**	
	(Learners = 10)		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**		**(Learners = 10)**	
#	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t	ε_t	α_t
0	0.0976	1.1121	0.0978	1.111	0.0974	1.1132	0.1032	1.0811	0.101	1.0931
1	0.2169	0.6419	0.2131	0.6532	0.208	0.6685	0.2106	0.6607	0.2166	0.6428
2	0.3155	0.3873	0.3248	0.3659	0.3199	0.3771	0.3231	0.3698	0.2958	0.4337
3	0.4090	0.1841	0.4329	0.135	0.4055	0.1913	0.3999	0.2029	0.4033	0.1959
4	0.4311	0.1387	0.4448	0.1109	0.45	0.1003	0.4288	0.1434	0.4329	0.1350
5	0.4772	0.0456	0.4529	0.0945	0.4517	0.0969	0.4753	0.0494	0.4782	0.0436
6	0.4719	0.0563	0.4957	0.0086	0.4914	0.0172	0.4819	0.0362	0.4027	0.1971
7	0.4971	0.0058	0.4846	0.0308	0.4942	0.0116	0.4971	0.0058	0.4465	0.1074
8	**0.5111**	**-0.0222**	0.4862	0.0276	0.4918	0.0164	0.4986	0.0028	0.4623	0.0755
9	0.4879	0.0242	**0.5021**	**-0.0042**	**0.5095**	**-0.019**	**0.5021**	**-0.0042**	0.4836	0.0328

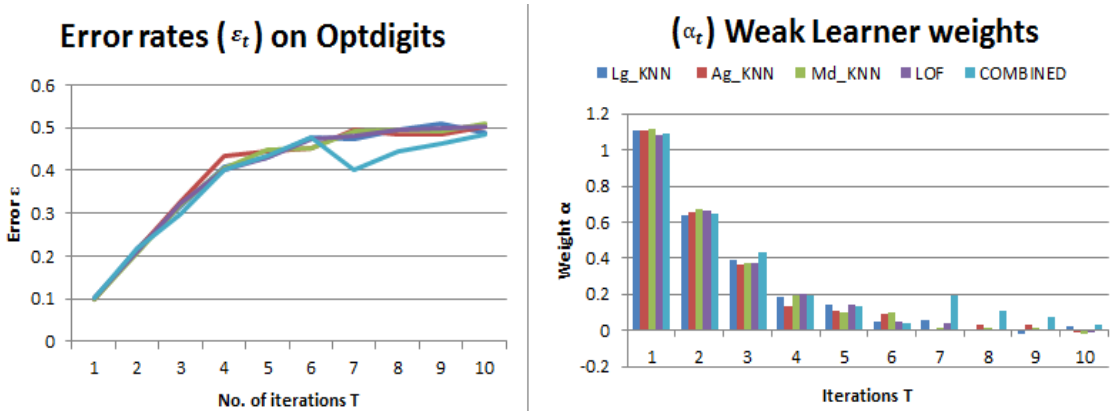

Figure 4.13: Comparison between Error Rates and Weights (Optdigits Dataset)

4.2.12 Summary of Selection of Weak Learners Based on Error Rates

From 50 weak learners, an average of 45.1, representing 90.2%, was selected based on their error rates being less than 0.5. These weak learners satisfied the third requirement of having better predictions than random guessing, as described in section 3.5.1. The highest number of weak learners at 48 (representing 96%) was selected when training using the Thyroid dataset. This could be attributed to the fact that this dataset was the smallest as it had the least number of dimensions (6) with only 2.74% outliers. It was also evident that this dataset's outliers were clustered as densities as Md_KNN, LOF, and COMB had all 10 weak learners predicting at least more than half the outliers correctly, and no weak learner was dropped. The lowest number of weak learners at 41 (representing 82%) was selected when training using the Breastw dataset. This is possible because this dataset had the highest number of outliers, as shown in table 3.1 (34.99%); hence, as the number of training iterations increased, harder examples increased, and more weak learners made more errors at every round. Only all 10 (100%) LOF weak learners could predict with error rates less than 0.5, and all were selected for future use. Md_KNNs were the least selected learners, as only 7 out of 10 were selected for this dataset. On average, 8.7 weak learners from three groups, Lg_KNN, Ag_KNN and Md_KNN, were selected in every dataset, as shown in Table 4.15. This represented 87% selection from each of the three groups.

Furthermore, on average, another 9.3 weak learners from the LOF category were selected for all datasets training. This represented 93% of LOF learners selected. Lastly, the highest selected learners by all datasets training rounds were the combination of all the heterogeneous weak learners (COMB) with 9.7 learners on average selection when using any dataset. This represented 97% selection when heterogeneous learners were used.

Table 4.15: Summary of Selected Weak Learners Based on their Error Rates

Dataset	Lg_KNN	Ag_KNN	Md_KNN	LOF	COMB	Total	Rep. %
Mnist	9	9	8	9	10	45	0.90
Letter	9	9	10	9	10	47	0.94
Cardio	8	8	8	10	10	44	0.88
Annthyroid	9	8	9	10	10	46	0.92
Pima	8	9	9	8	10	44	0.88
Vowels	9	9	8	9	10	45	0.90
Thyroid	9	9	10	10	10	48	0.96
Pendigits	9	8	9	9	10	45	0.90
Breastw	8	9	7	10	7	41	0.82
Optdigits	9	9	9	9	10	46	0.92
AVERAGE	**8.7 (87%)**	**8.7 (87%)**	**8.7 (87%)**	**9.3 (93%)**	**9.7 (97%)**	**45.1**	**0.90**

4.3 Comparison of the Proposed Fusion Methods with Generic Methods

Optimal learners were carefully selected for fusion based on their performances in their local domains, as described in sections 3.5.2 and 3.5.3. This process was followed by evaluating different fusion strategies for the qualified base learners for the ensemble formation. To this far, two more experiment groups were performed. One group compared existing generic (global) methods (as mentioned in Section 3.5.2) to the proposed variants, and the other group compared outlier score outcomes, which helped determine whether there were improvements.

4.3.1 Experiment 13: Comparing ADAHO_Avg and ADAHO_maxA vs. G_Avg

Table 4.16 shows the mean AUCROC scores for G_Avg compared with ADAHO_Avg and ADAHO_MaxA. ADAHO_MaxA had better results in at least 5 datasets, representing 50% of the overall performance. This was followed by G_Avg, which had better results in 3 datasets representing 30% in performance. ADAHO_Avg was best in only 2 datasets representing 20%. It was clear that averaging the group scores of the selected base learners and then taking the maximum from every group yielded better results. In terms of bias, this method reduced individual biases through averaging in the second level, unlike G_Avg or ADAHO_Avg, which only had one combination level.

Table 4.16: ROC Performances (Mean of 20 Trials, Highest Scores Bolded)

Dataset	G_Avg	ADAHO_Avg	ADAHO_MaxA
Mnist	0.8456	0.8475	**0.8522**
Letter	0.7824	0.7717	**0.7853**
Cardio	0.8669	0.8590	**0.8807**
Annthyroid	0.7583	0.7444	**0.7866**
Pima	0.6929	**0.6958**	0.6890
Vowels	0.9164	**0.9175**	0.9084
Thyroid	**0.9555**	0.9478	0.9523
Pendigits	0.8277	0.8137	**0.8555**
Breastw	**0.7261**	0.6452	0.6943
Optdigits	**0.8845**	0.8787	0.8618

4.3.2 Experiment 14: Comparing ADAHO_Avg and ADAHO_maxA vs. G_mov

Table 4.17 shows the mean AUCROC scores for ADAHO_Avg and ADAHO_MaxA compared with G_Mov. It was evident that both ADAHO_MaxA and G_MoV had similar performances, with each predicting better in at least 4 datasets representing 40% overall performance. ADAHO_Avg

was better in only 2 datasets representing 20%. In both ADAHO_MaxA and G_MoV, it was clear that by having two levels of fusion, these two models yielded better results in Mnist, Cardio, Annthyroid, Pendigits, and in Letter, Thyroid, Breastw and Optdigits, respectively.

Table 4.17: ROC Performances (Mean of 20 Trials, Highest Scores Bolded)

Dataset	G_MoV	ADAHO_Avg	ADAHO_MaxA
Mnist	0.8487	0.8475	**0.8522**
Letter	**0.7930**	0.7717	0.7853
Cardio	0.8764	0.8590	**0.8807**
Annthyroid	0.7768	0.7444	**0.7866**
Pima	0.6902	**0.6958**	0.6890
Vowels	0.9174	**0.9175**	0.9084
Thyroid	**0.9546**	0.9478	0.9523
Pendigits	0.8408	0.8137	**0.8555**
Breastw	**0.7039**	0.6452	0.6943
Optdigits	**0.8926**	0.8787	0.8618

4.3.3 Experiment 15: Comparing ADAHO_Max and ADAHO_maxA vs. G_Max

Table 4.18 shows the mean AUCROC scores for G_Max compared with ADAHO_Max and ADAHO_MaxA. ADAHO_MaxA had better results overall in at least 7 datasets, representing 70% of the datasets. This was followed by G_Max, which had better results in 2 datasets, i.e., Letter and Vowels. ADAHO_Max was last as it was best in only 1 dataset (Breastw). The performance of ADAHO_MaxA was attributed to the fact that it had two fusion levels. First, by averaging the group scores of the selected base learners, then taking the maximum score of every group. In terms of bias, this method greatly reduced individual learner biases through averaging in the first level, unlike G_Max and ADAHO_Max, which only had one level of fusion.

Table 4.18: ROC Performances (Mean of 20 Trials, Highest Scores Bolded)

Dataset	G_Max	ADAHO_Max	ADAHO_MaxA
Mnist	0.8248	0.7711	0.8522
Letter	0.8333	0.8260	0.7853
Cardio	0.8697	0.8390	0.8807
Annthyroid	0.7556	0.7460	0.7866
Pima	0.6629	0.6539	0.6890
Vowels	0.9212	0.9137	0.9084
Thyroid	0.9284	0.9312	0.9523
Pendigits	0.8387	0.7137	0.8555
Breastw	0.6489	0.7135	0.6943
Optdigits	0.8458	0.8424	0.8618

4.3.4 Experiment 16: Comparing ADAHO_Max, ADAHO_maxA vs. G_AoM

Table 4.19 shows the mean AUCROC scores for G_AoM compared with ADAHO_Max and ADAHO_MaxA. ADAHO_MaxA had better results overall in at least 60% of the datasets in this experiment. G_AoM and ADAHO_Max performed better in 2 datasets, i.e., Vowels and Optidigits with AUC values 0.9170 and 0.8662, respectively; and in Letter and Breastw with AUC 0.8260 0.7135, respectively. Once again, the best performance of ADAHO_MaxA could be attributed to the fact that it had two levels of fusion, i.e., by averaging group scores of the selected base learners and taking the maximum of the group averages after. This reduced ensemble bias and variance overall.

Table 4.19: ROC Performances (Mean of 20 Trials, Highest Scores Bolded)

Dataset	G_AoM	ADAHO_Max	ADAHO_MaxA
Mnist	0.8452	0.7711	0.8522
Letter	0.8209	0.8260	0.7853
Cardio	0.8802	0.8390	0.8807
Annthyroid	0.7538	0.7460	0.7866
Pima	0.6755	0.6539	0.6890
Vowels	0.9170	0.9137	0.9084
Thyroid	0.9409	0.9312	0.9523
Pendigits	0.8521	0.7137	0.8555
Breastw	0.6737	0.7135	0.6943
Optdigits	0.8662	0.8424	0.8618

4.3.5 Experiment 17: Comparing ADAHO_Max, ADAHO_MaxA versus G_mov

Table 4.20 shows the mean AUCROC scores for G_MoV compared with ADAHO_Max and ADAHO_MaxA. From the results, both ADAHO_MaxA and G_MoV had similar performances, with each predicting better in at least 4 datasets. ADAHO_Max was better in only 2 datasets, i.e., Letter and Breastw datasets with AUC ROC of 0.8260 and 0.7135, respectively. ADAHO_MaxA and G_MoV fused their outcomes in two levels, yielding better outcomes in most datasets. Overall, the AUC ROC scores for ADAHO_MaxA were consistently superior in most datasets. This was possible because its base learners were evaluated based on local domain competency (section 3.5.2) whereas G_MoV did not. The AUC values in Table 4.20 represent an average of 20 trials.

Table 4.20: ROC Performances (Mean of 20 Trials, Highest Scores Bolded)

Dataset	G_MoV	ADAHO_Max	ADAHO_MaxA
Mnist	0.8487	0.7711	**0.8522**
Letter	0.7930	**0.8260**	0.7853
Cardio	0.8764	0.8390	**0.8807**
Annthyroid	0.7768	0.7460	**0.7866**
Pima	**0.6902**	0.6539	0.6890
Vowels	**0.9174**	0.9137	0.9084
Thyroid	**0.9546**	0.9312	0.9523
Pendigits	0.8408	0.7137	**0.8555**
Breastw	0.7039	**0.7135**	0.6943
Optdigits	**0.8926**	0.8424	0.8618

4.3.6: Summary of Performances using Different Generic (Global) Fusion Methods Compared to the Proposed Method in Terms of AUCROC

A total of ten datasets from different domains were considered (section 3.3). The various baseline outcomes of the generic or global fusion methods like average-of-maximum, maximum-of-average, weighted averaging, and global threshold summation versus our proposed variants (ADAHO_Avg, ADAHO_Max, ADAHO_MaxA, and ADAHO_AvgM) were compared. Our variants yielded better results, particularly ADAHO_AvgM (table 4.21). This strongly indicated that the proposed method was better considering a data point's local domains. It achieved better results in at least 6 out of 10 datasets, as in table 4.21.

Overall, the performances of the selected baselines (Table 4.21), G_Max exhibited improvement in two datasets (letter and vowels) with high ROC of 0.8333 and 0.9212, respectively, in the first experiment. In contrast, G_AoM exhibited improvement in only one dataset (annthyroid) with 0.2415 in the second experiment of average precision (Table 4.22). The proposed (ADAHO variants) exhibited superior results for the other tests in both experiments.

Furthermore, regarding global methods, G_AoM and G_Max were superior to simple maximization or averaging. This is attributed to the fact that fusion considers a second dimension and yields stable outcomes, which is why ADAHO yielded higher scores. For G_Avg and G_Max, local competency assessment gave weaker results but lesser variance and bias based on global averaging. All outcomes should be used in the final fusion process to reduce bias. However, this could deteriorate ensemble performance because low-performance learners could be included. In contrast, selecting one optimal learner yields a smaller variance drop than averaging, which also deteriorates ensemble performance overall because of strong bias from a single selected learner. Using the maximum value to generate ground truth, global and ADAHO_Max became less stable. E.g., ADAHO_Max was better than G_AoM on letter and breast (0.8260 and 0.7135), respectively, in our first experiment (Table 4.21) and in letter dataset in our second experiment (Table 4.22) with an average precision of 0.3245.

Table 4.21: Summary of ROC Values (Highest Values Bolded) Between ADAHO Fusion Methods and Generic Fusion Methods

Dataset	G_Avg	G_MoV	G_Max	G_AoM	G_Wa	G_Ts	ADAHO_Avg	ADAHO_MaxA	ADAHO_Max	ADAHO_AvgM
Mnist	0.8456	0.8487	0.8248	0.8452	0.8462	0.8171	0.8475	0.8522	0.7711	**0.8532**
Letter	0.7824	0.7930	0.8333	0.8209	0.7807	0.7900	0.7717	0.7853	0.8260	0.7766
Cardio	0.8669	0.8764	0.8697	0.8802	0.8681	0.8729	0.8590	0.8807	0.8390	**0.8912**
Annthyroid	0.7583	0.7768	0.7556	0.7538	0.7664	0.7651	0.7444	0.7866	0.7460	0.7886
Pima	0.6929	0.6902	0.6629	0.6755	0.6936	0.6248	0.6958	0.6890	0.6539	**0.6960**
Vowels	0.9164	0.9174	0.9212	0.9170	0.9160	0.9198	0.9175	0.9084	0.9137	0.9098
Thyroid	0.9555	0.9546	0.9284	0.9409	0.9564	0.9543	0.9478	0.9523	0.9312	**0.9699**
Pendigits	0.8277	0.8408	0.8387	0.8521	0.8324	0.8447	0.8137	0.8555	0.7137	0.8643
Breastw	0.7261	0.7039	0.6489	0.6737	0.7352	0.6184	0.6452	0.6943	0.7135	**0.7743**
Optdigits	0.8845	0.8926	0.8458	0.8662	0.8852	0.8803	0.8787	0.8618	0.8424	**0.8884**

Table 4.22: Summary of Mean Average Precision Values (Highest Values Bolded) for ADAHO vs Generic Fusion Methods

Dataset	G_Avg	G_MoV	G_Max	G_AoM	G_Wa	G_Ts	ADAHO_Avg	ADAHO_MaxA	ADAHO_Max	ADAHO_AvgM
Mnist	0.3810	0.3840	0.3800	0.3795	0.3817	0.3735	0.3832	0.3873	0.3252	**0.3878**
Letter	0.2287	0.2372	0.3059	0.2766	0.2271	0.2317	0.2201	0.2295	**0.3245**	0.2306
Cardio	0.3415	0.3607	0.3565	0.3763	0.3434	0.3528	0.3274	0.3859	0.3096	**0.4016**
Annthyroid	0.2200	0.2294	0.2312	0.2415	0.2205	0.2176	0.2182	0.2274	0.2248	0.2352
Pima	0.4988	0.4953	0.4712	0.4819	0.4994	0.4498	0.4991	0.4944	0.4615	**0.5041**
Vowels	0.3682	0.3689	0.3659	0.3631	0.3683	0.3682	0.3812	0.3577	0.3381	0.3438
Thyroid	0.3944	0.4022	0.2749	0.3387	0.4029	0.3070	0.3443	0.3854	0.2537	**0.4550**
Pendigits	0.0676	0.0722	0.0732	0.0794	0.0679	0.0731	0.0608	0.0792	0.0524	**0.0843**
Breastw	0.4894	0.4748	0.4148	0.4476	0.4984	0.4265	0.4233	0.4665	0.4627	**0.5554**
Optdigits	0.3593	0.3559	0.3043	0.3286	0.3605	0.3537	0.3497	0.3209	0.3092	**0.3678**

It was evident that if only a single learner's maximum score was used, the overall ensemble had a high variance. However, applying a second fusion, such as averaging, mitigated this effect. This finding was also reported in (Zimek et al., 2013). Furthermore, to reduce the variance of an ensemble, ADAHO took advantage of the G_AoM effect by calculating the mean of outlier scores from subsets of optimal learners, implying ADAHO_AvgM further reduced the variance of the final ensemble compared to ADAHO_Max. To reduce ensemble bias, ADAHO_AvgM calculated an average in the second-level fusion, improving accuracy. This is evident in the experimental results as ADAHO_AvgM yielded higher scores in 6 datasets in terms of the ROC, namely mnist, cardio, pima, thyroid, breast, and optdigits (Table 4.21), and for seven datasets in terms of average precision, namely mnist, cardio, pima, thyroid, pendigits, breastw, and optdigits (Table 4.22). Based on these experiments, it was clear that calculating the maximum after the mean did not significantly improve classification results. This was evident for ADAHO_MaxA, which was not improved significantly by either global averaging or maximum-of-averaging. In summary, ADAHO_AvgM was a superior fusion strategy based on its ability to minimize both variance and bias, which answered the question regarding the best fusion strategy for outlier detection ensembles.

4.4 Comparing Performance of the Proposed Ensemble with Other Outlier Detection Ensembles using 10 Datasets

Additional experiments were conducted using other outlier detection ensembles to verify if the hypothesis enhanced anomaly detection and further examine the proposed method's performance. Therefore, its performance was compared against four other existing outlier detection ensemble formations (ALOI (Schubert et al., 2014), BASE (Micenkova et al., 2014), SELECT (Rayana & Akoglu, 2016)), which do not use distinct local data domains in their approach but use similar base models. The first two formations were symbolically referred to here as ALOI and BASE because of their relative design with respect to our proposed method. Both formations, similar to our proposed method ADAHO_OAAE, utilize LOF and KNN algorithms, with the only difference

being that they used a heuristic method instead of optimizing model parameters. It is important to note that while our proposed method weights its samples and its base models by adaptive boosting, these two apply feature bagging with only model weights being fused by summation or mean. In principle, ALOI and BASE follow our Algorithm 1, but their $\mathcal{D}_o$ values are empty. In addition, the two do not follow steps 1–4 of boosting but use equal weights $W_t = 1$ for all $t = 1, \dots, T$. ALOI and BASE set parameters for their base models in the same way as ours. Since the base models are similar, they utilize similar distance metrics as well as some of the fusion approaches like sum, median, average and maximum. Specifically, both approaches utilize summation and Euclidean distances. Our fusion, however, utilizes median for optimization and takes the final scores in two levels: by maximum of average and average of maximum.

The third formation, named SELECT (Rayana & Akoglu, 2016), presents two sub-formations: vertical-SELECT and horizontal-SELECT. We compared our method to horizontal-SELECT because, according to (Rayana & Akoglu, 2016), this formation achieves better overall performance. It works by converting anomaly scores into probabilities, representing the scores as samples from a combination of Gaussian and exponential distributions for anomalous and normal data, respectively. It bases its hypothesis on this assumption and creates base models based on how well each base model fit the pseudo-target. This ensemble's final scores emanate from a fusion of selected base models via robust rank aggregation.

The fourth and final baseline is the modified ADAHO (Bii et al., 2020), which partly learns the same base models as our proposed method ADAHO_OAAE and weights its samples and base models by adaptive boosting. The main difference is that it does not consider margin maximization between anomalous and normal data. Hence it does not loop through step 5 of algorithm 1 in the same way as ours.

We tested our proposed approach against these four baselines using the least known anomaly percentage. We considered that using the least percentage is the hardest test since very little prior information about the dataset was provided. If our proposed method outperformed these baselines, a significant improvement would have been achieved overall. Ensemble performance was measured using AUROC values. Results of the experiments were provided in the form of tables and figures.

4.4.1 Experiment 18: Comparing the Performance of Proposed Method with ALOI Ensemble using 10 Outlier Detection Datasets

Figure 4.14 and Table 4.23 depict results of the experiment. It indicates that while using Mnist, Letter, Annthyroid, Pima, Vowels, Pendigits, Breastw, and Optdigits, the proposed method had significant ROC (shaded green) than ALOI. The results revealed that our method edged marginally with ALOI using Cardio and underperformed in Thyroid. This was attributed to the fact that this dataset (Thyroid) had the least number of dim

Proposed Method verses ALOI

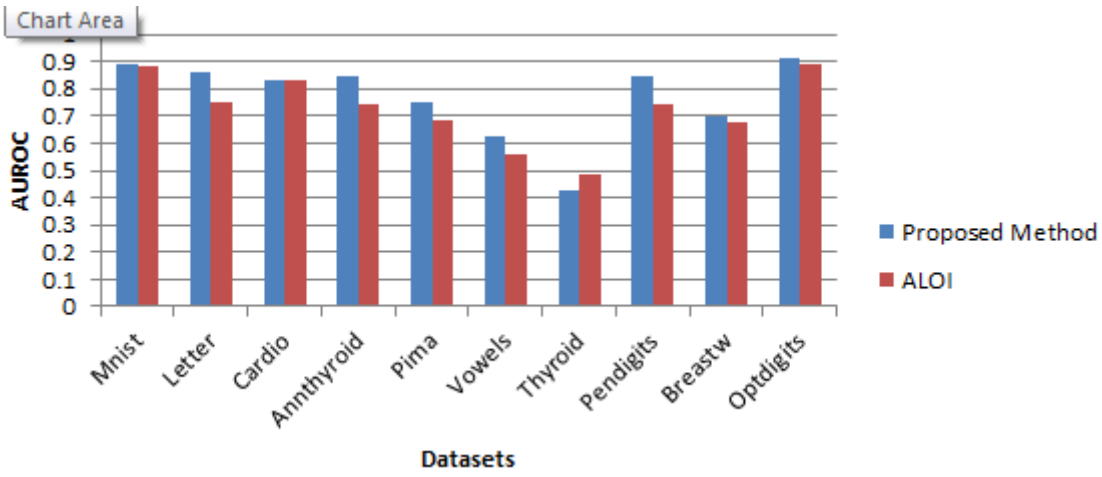

Figure 4.14: Proposed Method verses ALOI Ensemble using 10 Datasets

Table 4.23: Proposed Method vs. ALOI Ensemble using 10 Datasets

	Mnist	Letter	Cardio	Annthyroid	Pima	Vowels	Thyroid	Pendigits	Breastw	Optdigits
Proposed	**0.8916**	**0.8642**	0.828	**0.8449**	**0.7538**	**0.6238**	0.4283	**0.8451**	**0.7019**	**0.9114**
ALOI	0.8849	0.7529	**0.8304**	0.7435	0.6876	0.5622	**0.4827**	0.745	0.6734	0.8933
	0.0067	*0.1113*	*-0.0024*	*0.1014*	*0.0662*	*0.0616*	*-0.0544*	*0.1001*	*0.0285*	*0.0181*

Section 4.4 reveals that ALOI uses feature bagging to diversify its base models, unlike the proposed method that uses decision weights. The proposed method yield better results in 8 of 10 datasets (shaded green in Table 4.23) as harder samples were revised in every training round, unlike ALOI.

4.4.2 Experiment 19: Comparing the Performance of Proposed Method with BASE Ensemble using 10 Outlier Detection Datasets

Figure 4.15 and Table 4.24 show the results of the experiment, which indicate that while using Mnist, Letter, Annthyroid, Cardio, Pima, Thyroid, Pendigits, and Breastw datasets, the proposed method had better ROC than BASE. It further revealed that BASE slightly outperformed our method while using Vowels and Optdigits datasets (shaded yellow). Like ALOI, BASE utilized feature bagging, and a constant k-metric of 20; with base learner combination by summation and that explains the difference in performance. In 8 datasets out of 10 (shaded

green), our method demonstrated better improvement because of boosting and optimization.

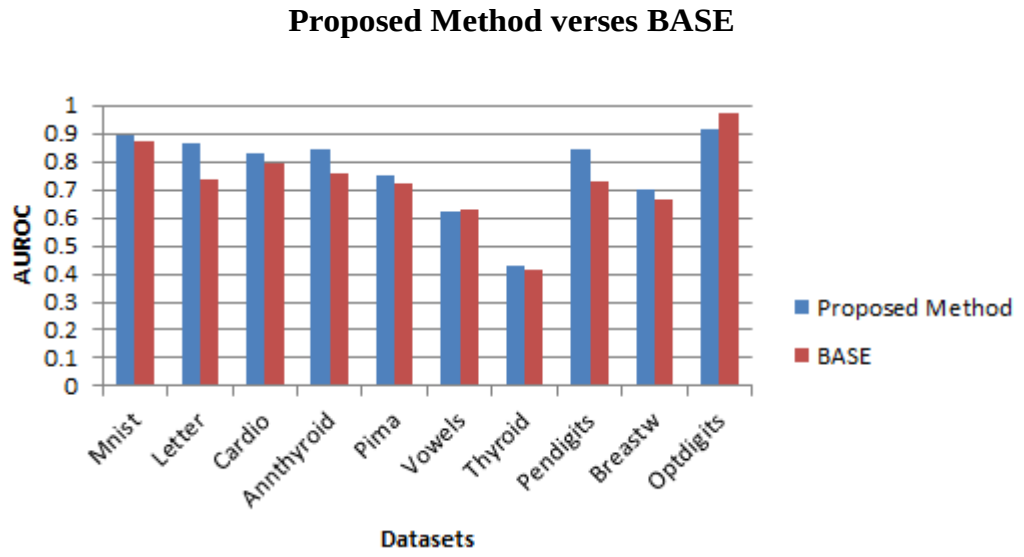

Figure 4.15: Proposed Method verses BASE Ensemble using 10 Datasets

Table 4.24: Proposed Method vs. BASE Ensemble using 10 Datasets

	Mnist	Letter	Cardio	Annthyroid	Pima	Vowels	Thyroid	Pendigits	Breastw	Optdigits
Proposed	**0.8916**	**0.8642**	**0.828**	**0.8449**	**0.7538**	0.6238	**0.4283**	**0.8451**	**0.7019**	0.9114
BASE	0.8703	0.7385	0.7907	0.7592	0.7213	**0.6298**	0.4177	0.7303	0.6671	**0.9735**
	0.0213	*0.1257*	*0.0373*	*0.0857*	*0.0325*	*-0.006*	*0.0106*	*0.1148*	*0.0348*	*-0.0621*

4.4.3 Experiment 20: Comparing the Performance of Proposed Method with Horizontal-SELECT Ensemble Using 10 Outlier Detection Datasets

Figure 4.16 and Table 4.25 show the results of the experiment, which indicate that our method performed well while using Mnist, Cardio and Vowels, and even performed better while using Letter, Annthyroid, Pima, Pendigits, and Breastw datasets (shaded green) compared to horizontal-SELECT. It also revealed that SELECT outperformed our method while using Thyroid and Optdigits datasets with ROC of 0.5149 and 0.9757, respectively (shaded yellow). SELECT did not boost its samples like our method but selected good learners in two levels like our method. That could explain the difference in performance. In 8 datasets out of 10, our method demonstrated significant improvement in scores because of the

adaptive boosting that revises harder examples and the optimization by margin maximization that gave a clear contrast between the inliers and outliers.

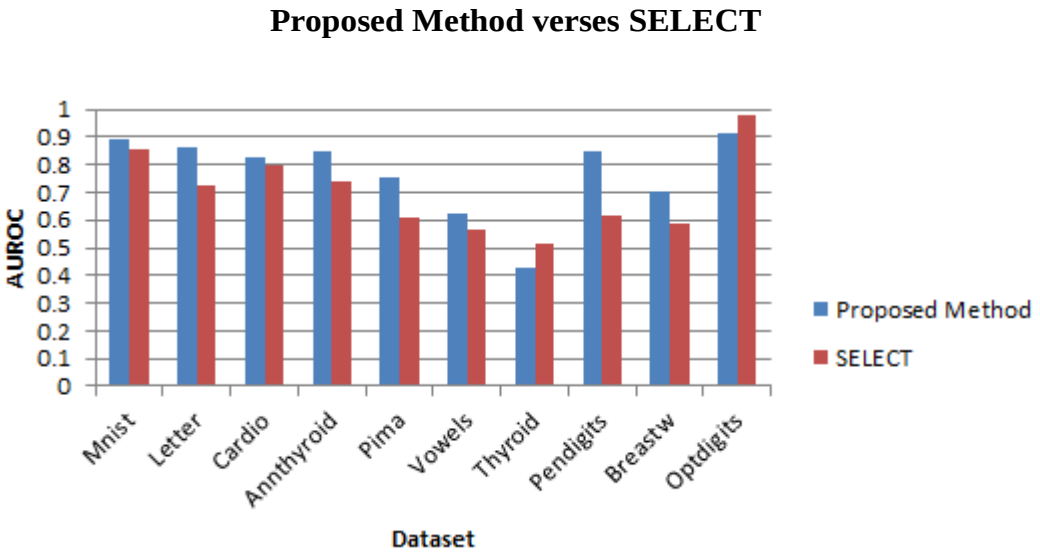

Figure 4.16: Proposed Method vs. Horizontal-SELECT Ensemble using 10 Datasets

Table 4.25: Proposed Method vs. Horizontal-SELECT Ensemble using 10 Datasets

	Mnist	Letter	Cardio	Annthyroid	Pima	Vowels	Thyroid	Pendigits	Breastw	Optdigits
Proposed	**0.8916**	**0.8642**	**0.828**	**0.8449**	**0.7538**	**0.6238**	0.4283	**0.8451**	**0.7019**	0.9114
SELECT	0.8572	0.7285	0.7979	0.7405	0.6093	0.5622	**0.5149**	0.6184	0.5877	**0.9757**
	0.0344	*0.1357*	*0.0301*	*0.1044*	*0.1445*	*0.0616*	*-0.0866*	*0.2267*	*0.1142*	*-0.0643*

4.4.4 Experiment 21: Comparing the Performance of Proposed Method with ADAHO Ensemble Using 10 Outlier Detection Datasets

Figure 4.17 and Table 4.26 show the results of the experiment, which indicate that our proposed method performed considerably well in at least 7 datasets (shaded green), i.e., Letter, Annthyroid, Cardio, Pima, Vowels, Thyroid, and Pendigits with ROC values 0.8642, 0.8280, 0.8449, 0.7538, 0.6238, 0.4283, and 0.8451, respectively. It also revealed that ADAHO outperformed our method using Mnist, Breastw, and Optdigits datasets with 0.9297, 0.7303, and 0.9396, respectively (shaded yellow). This could be credited to the fact that ADAHO

boosted its samples just like our method and selected good learners in two levels like our method but did not optimize models in iteration rounds by margin maximization as our method. In 7 out of 10 datasets, our method demonstrated improvement in scores (shaded green).

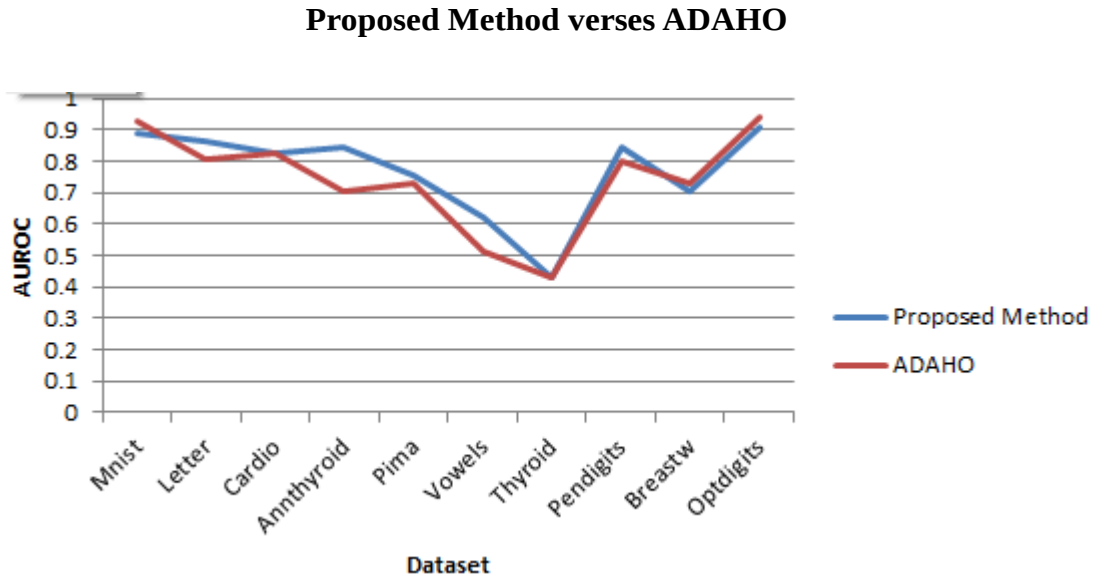

Figure 4.17: Proposed Method with Margin Maximization vs. ADAHO Ensemble

Table 4.26: Proposed Method with Margin Maximization vs. ADAHO

	Mnist	Letter	Cardio	Annthyroid	Pima	Vowels	Thyroid	Pendigits	Breastw	Optdigits
Proposed	0.8916	**0.8642**	**0.828**	**0.8449**	**0.7538**	**0.6238**	0.4283	**0.8451**	0.7019	0.9114
ADAHO	0.9297	0.8059	0.8248	0.7036	0.7279	0.5134	0.4278	0.7981	0.7303	0.9396
	-0.0381	*0.0583*	*0.0032*	*0.1413*	*0.0259*	*0.1104*	*0.0005*	*0.047*	*-0.0284*	*-0.0282*

Table 4.27 summarizes the analysis of the performance in terms of AUC scores of the various baseline outcomes of ALOI, BASE, SELECT and ADAHO (without margin maximization) techniques versus our method ADAHO_OAAE (with margin maximization optimization) over all the datasets in Table 3.1. When compared to the baselines, ADAHO_OAAE showed improvement in 4 datasets,

with substantial improvements noted in bold. This proposed method generated better results due to the optimization and the diverse set of its base models. As for the performance of the selected baselines, BASE showed improvement in one dataset (vowels) with an AUC score of 0.6298, while ALOI showed improvement in dataset cardio with 0.8304. SELECT, on the other hand, improved performance in two datasets, Thyroid and Optdigits, with AUC values of 0.5149 and 0.9757, respectively. ADAHO (without optimization) showed improvement in two datasets (Mnist and breastw) as well, with high AUC scores of 0.9297 and 0.7303, respectively. For the rest of the tests, ADAHO_OAAE (with optimization) exhibited superior results in four datasets (Letter, Annthyroid, Pima and Pendigits) with AUC scores of 0.8642, 0.8449, 0.7538, and 0.8451, respectively. In addition, with regards to the four baselines that either took the overall average score or the maximum score, our proposed method used that which produced better results between the average-of-maximum and the maximum-of-average and hence produced much better results. This was so because the score combination took place twice in two levels, one after the averaging and the other after the maximization and vice versa, aiming at a more stable result. It is worth noting that ADAHO_OAAE (with optimization) and ADAHO (without optimization) baseline performed almost similarly in terms of AUC scores, with a very close absolute average AUC difference of 0.0332, i.e., (0.7693-0.7361). The other baselines, ALOI, BASE, and SELECT differed significantly with absolute average score differences of 0.0437, 0.0395, and 0.0701, respectively. This was attributed to the fact that both ADAHO (without optimization) and ADAHO_OAAE (with optimization) similarly followed steps 1 to 4 of Algorithm 6, and only differed in score margin maximization. While ADAHO (without optimization) focused on only the local neighbourhood strengths of a weak learner like ADAHO_OAAE further focused on score margin maximization between the anomalous and normal data to build a contrast, which improved detection accuracy. The two methods dynamically weighted their samples as well as their weak learners before fusion, hence the close similarity in their AUC scores.

In terms of bias and variance reduction, the tests revealed that combining only a few well-performing base models led to more biased results. To reduce bias, it was preferable if all base models participated in the final decision; however, this was risky because poor models could have degraded overall ensemble performance. Since ADAHO_OAAE combined scores of optimized yet diverse base models at two levels, this effect was considerably reduced, as evidenced by higher AUROC scores in the test results of four datasets, namely Letter, Annthyroid, Pima, and Pendigits, at 0.8642, 0.8449, 0.7538, and 0.8451, respectively. It was also clear that combining only a base model's maximum score yields a high-variance ensemble, but using a second level combination, such as averaging, mitigated this effect (similarly to (Zhao et al., 2019). As for ADAHO_OAAE, rather than take the highest anomaly score, it computes the average from the selected subsets of optimized models, thereby reducing the overall ensemble's variance.

Table 4.27: Summary of AUC scores of OAAE, ALOI, BASE and SELECT (Highest Values Bolded)

Dataset	ALOI	BASE	SELECT	ADAHO	OAAE
Mnist	0.8849	0.8703	0.8572	0.9297	0.8916
Letter	0.7529	0.7385	0.7285	0.8059	0.8642
Cardio	0.8304	0.7907	0.7979	0.8248	0.8280
Annthyroid	0.7435	0.7592	0.7405	0.7036	0.8449
Pima	0.6876	0.7213	0.6093	0.7279	0.7538
Vowels	0.5622	0.6298	0.5622	0.5134	0.6238
Thyroid	0.4827	0.4177	0.5149	0.4278	0.4283
Pendigits	0.7450	0.7303	0.6184	0.7981	0.8451
Breastw	0.6734	0.6671	0.5877	0.7303	0.7019
Optdigits	0.8933	0.9735	0.9757	0.9396	0.9114
AVERAGE	*0.7256*	*0.7298*	*0.6992*	*0.7401*	*0.7693*

Table 4.28 shows the mean of test results of ALOI, BASE, SELECT, ADAHO, and OAAE with the highest values bolded. The diagonal values show each technique's average AUC values. The other values off diagonal (shaded) reflect the average differences in the AUC values between these techniques over all the selected datasets. Since the paired t test produced p values less than 0.05, the average difference in AUC values of our proposed algorithm OAAE versus the baselines ALOI, BASE, SELECT and ADAHO are statistically significant, i.e., 0.0095, 0.0204, 0.0166, and 0.0476, respectively as depicted and bolded in Table 4.29 with confidence level at 95 percent.

Table 4.28: OAAE, ALOI, BASE, SELECT, and ADAHO Mean AUC Values and Average Differences over the Datasets in Table 3.1.

Versus	**ALOI**	**BASE**	**SELECT**	**ADAHO**	**OAAE**
ALOI	0.7256	0.0042	-0.0264	0.0105	0.0437*
BASE		0.7298	-0.0306*	0.0063	0.0395*
SELECT			0.6992	0.0369	0.0701*
ADAHO				0.7361	0.0332*
OAAE					0.7693

In figure 4.18, the Area Under Receiver Operating Characterstic Curve (AUC-ROC) values represent the densities for each method against the selected datasets. The short vertical lines below the line curves show AUC values, representing each method's Kernel Density Estimates (KDE). Figure 4.18 demonstrates that outliers in at least half the datasets were easier to detect while others were not. It's also important to note that the AUCROC values of ALOI and BASE had near-similar kernel density estimates, which affirms the fact that there was not much significance in terms of the AUC difference even in the paired t-test between the two (p-value of 0.3859, which is higher than 0.05) and therefore not statistically significant.

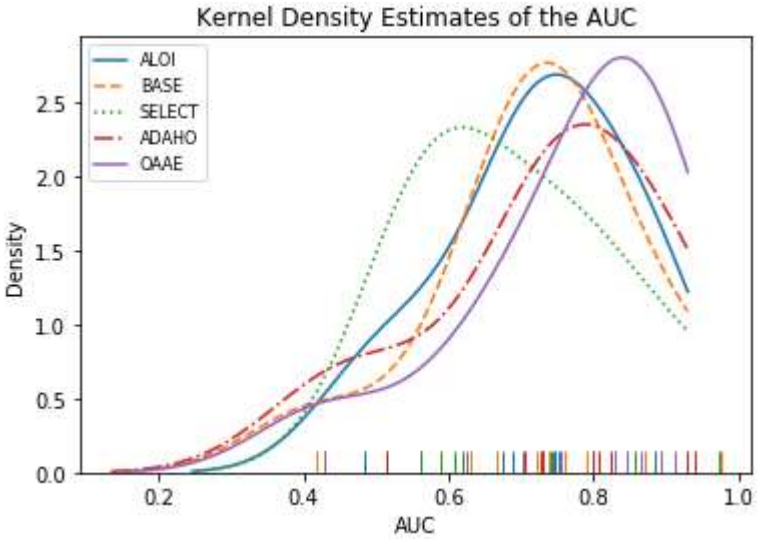

Figure 4.18: Kernel Density Estimates of ALOI, BASE, SELECT, ADAHO and ADAHO_OAAE

Lastly, in figure 4.18, it is also worth mentioning that the mass of the KDE of the AUC values of OAAE is drifting more towards the right as compared to ALOI, BASE, SELECT, and ADAHO, which implies that OAAE achieved higher AUCROC values overall. Table 4.29 of p-values validates this observation.

Table 4.29: The p-values for Average Differences of the Methods from Table 4.27

Paired_Test	***p values***
OAAE vs. ALOI	0.0095
OAAE vs. BASE	0.0204
OAAE vs. SELECT	0.0166
OAAE vs. ADAHO	0.0476
ADAHO vs. ALOI	0.1690
ADAHO vs. BASE	0.3036
ADAHO vs. SELECT	0.0864
SELECT vs. ALOI	0.0922
SELECT vs. BASE	0.0736
BASE vs. ALOI	0.3859

Table 4.29 shows the associated p values for the paired t-test. The statistically significant values, i.e., $p < 0.05$, are emphasized in bold. With p values of 0.0095, 0.0204, 0.0166, and 0.0476, these tests revealed that OAAE (with optimization) outperformed ALOI, BASE, SELECT, and ADAHO on a statistically significant level. This was not the case with ADAHO versus ALOI, BASE, and SELECT, or SELECT versus ALOI, and BASE, or BASE versus ALOI, as their tests produced p-values above 0.05, i.e., 0.169, 0.3036, 0.0922, 0.0736, and 0.3859, respectively. Compared to others, ALOI had the least significant result at 0.0095.

Figure 4.19 shows the AUC difference between OAAE and ADAHO while using only one known outlier example in both ensembles. The average confidence interval at the center is 0.95, indicating that most datasets were classifiable. Furthermore, at the very extreme ends are the AUC difference points where each method is superior in terms of the kernel density estimates, specifically in MNIST and letter datasets (marked red), respectively. The short vertical pointers along the x-axis and at density zero define AUC scores of each dataset, with scores beyond zero indicating datasets where OAAE outperformed ADAHO.

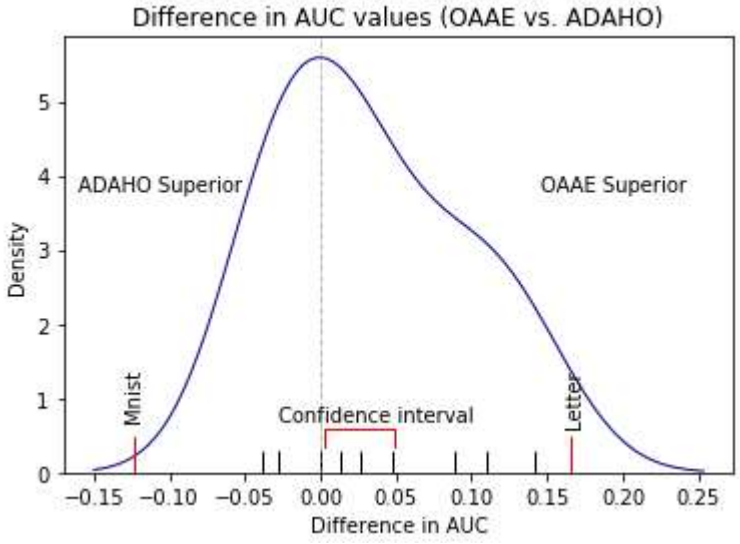

Figure 4.19: Differences in AUC between OAAE and ADAHO

4.5 Model Improvement by Reduction of Ensemble Bias and Variance

To improve the performance of the base detectors, we used the ensemble technique with adaptive boosting, as discussed in section 3.5. In order to lower the overall ensemble variance, the selected weak learners must be uncorrelated, as indicated in sections 2.8 and equations in 3.5.4.2. This was achieved by our model diversity (using heterogeneous base detectors) to reduce average error of bias and variance. In the case of KNN, the three variants used the median, the average, and the largest k values. Furthermore, different k distances and distance metrics were chosen. The re-weighting of samples at each training round introduced sample diversity. The choice of different training algorithms introduced model diversity, and the use of uncorrelated learner scores for model combination introduced score diversity, which guaranteed overall variance reduction. We calculated the bias and variance for every benchmark ensemble model with varying number of base detectors and the outcomes are presented in Table 4.30.

Table 4.30: Comparison Between the Benchmark Ensembles vs the Proposed Ensemble in Terms of Overall Bias -Variance. Expected error = (Bias^2 + Variance)

Dataset	Ensemble	Complexity (No. of learners)	Bias^2 $(E(\hat{y}) - y)^2$	Variance $E\left[(\hat{y} - E(\hat{y}))^2\right]$	Expected Error $E[(\hat{y} - y)^2]$
Mnist	Benchmark(SELECT)	1	0.0326	0.0540	0.0866
	ADAHO_OAAE1	50	0.0322	0.0503	0.0825
	ADAHO_OAAE2	100	0.0321	0.0483	**0.0804**
	ADAHO_OAAE3	150	0.0322	0.0489	0.0810
	ADAHO_OAAE4	200	0.0323	0.0489	0.0811
	ADAHO_OAAE5	250	0.0325	0.0488	0.0814
Letter	Benchmark(SELECT)	1	0.1281	0.0948	0.2229
	ADAHO_OAAE1	50	0.1275	0.0900	0.2175
	ADAHO_OAAE2	100	0.1266	0.0895	0.2161
	ADAHO_OAAE3	150	0.1271	0.0849	**0.2120**
	ADAHO_OAAE4	200	0.1278	0.0871	0.2149
	ADAHO_OAAE5	250	0.1280	0.0898	0.2178

From Table 4.30, we note that the ensemble with 100 weak learners performed better using the MINST dataset, with the least bias-variance at 0.0804 (bold), while for the LETTER dataset, the ensemble with 150 weak learners was better, having the least bias-variance at 0.2120 (bold). The main reason is that these datasets with high dimensionality are a little harder to classify; secondly, compared to the benchmark ensembles (i.e., using single homogeneous models as weak learners), the proposed ensemble model reduces bias and variance better. Still, the bias drop is less than the variance drop, suggesting that the ensemble may successfully reduce the overall variance. Thirdly, of all the ensembles, the most complex one does not necessarily yield the best performance, as revealed by the experiment, implying that an ensemble model should have a suitable or appropriate complexity, that is, neither too complex nor too basic. Lastly, all the ensemble models performed better by reducing the expected error, which implies that the ensemble technique is an effective modeling technique for improving detection accuracy and performance in outlier detection.

4.6 Test of Generalizability of the Proposed Method OAAE

Since outliers (anomalies) are not always the same, it is highly improbable that the known anomalies cover every type of possible anomaly (Ruff et al., 2019). In light of this, we tested ADAHO_OAAE's applicability to detecting a broader range of previously unknown anomalies. To complete this test, we utilize four datasets containing different classes, namely MNIST, LETTER, THYROID, and OPTDIGITS. In each dataset, we first sample the normal data from the most recurring data points to form the normal class and, secondly, we used the remainder of the data points in two ways: (i) by randomly sampling to create the anomalous class, (ii) by letting the rest of the data to be used as previously unknown anomalous cases. In addition, we further split the normal data into a training set consisting of sixty percent of normal data and ten percent of the known anomalies. Lastly, using the remaining normal data, we formed two testing sets: one containing a portion of these normal data along with the anomalies and the other containing a portion of these normal data along with the previously unknown anomalies, each at a ratio of 60-40, respectively.

We then fitted the models on the training set and tested them using the two sets containing known and previously unknown anomalies. The model that consistently attained near-similar scores over both tests passed the generalizability test. This test was repeated five times and computed the mean so that each model generated two scores from the two independent testing sets.

Table 4.31 summarizes the outcomes of these tests, with each outcome having two values: the first is from the test that contained known anomalies, and the second, placed inside the brackets, is from the test that contained the previously unknown anomalies. The highest values are bolded and italic, respectively. It was evident from the outcomes that OAAE (optimized), ADAHO (without optimization), and SELECT attained near-similar scores and that they could often discern previously unseen or even novel data anomalies. It was not the case for the BASE as unknown or novel cases of anomalies appeared to deteriorate their ability to detect overall, especially with dataset thyroid that yielded 0.590 (0.732) with a single known anomaly and 0.402 (0.646) with 10% known anomalies.

It was also apparent from these findings that the more known anomalies are in a training set, the better the detection accuracy overall. This was the case when having 10% of known anomalies in OAAE (MNIST improved from 0.958(0.978) to 0.962(0.986)), in ADAHO (OPTDIGITS improved from 0.937 (0.966) to 0.939 (0.974)), in SELECT (THYROID improved from 0.631 (0.644) to 0.634 (0.664)), and in ALOI (THYROID improved from 0.600(0.751) to 0.606(0.757)) as shown in Table 4.31. The AUC averages generally moved up from as low as 0.001 to 0.1; for example, OAAE's AUC average moved up from 0.868 to 0.871, a difference of 0.003, and ADAHO's average moved up from 0.851(0.878) to 0.854(0.880), a difference of 0.003(0.002). The known anomalies positively contributed to the margin maximization contrast of outliers vs. inliers because they effectively increased the detection accuracy of weak learners.

Table 4.31: Summary of AUC Outcomes from the Generalizability Test

a) Summary of AUC in generalizability test with a single known anomaly example					
Dataset	ALOI	BASE	SELECT	ADAHO	OAAE
Mnist	0.748 (0.988)	0.750(0.910)	0.949 (0.882)	0.925 (0.934)	0.958 (0.978)
Letter	0.913 (0.976)	0.921 (0.978)	0.952 (0.918)	0.926 (0.943)	0.925 (0.966)
Thyroid	0.600(0.751)	0.590 (0.732)	0.631 (0.644)	0.614 (0.670)	0.639 (0.770)
Optdigits	0.922 (0.971)	0.914 (0.967)	0.978 (0.954)	0.937 (0.966)	0.950 (0.973)
Average	0.796(0.922)	0.794(0.897)	0.878(0.850)	0.851(0.878)	0.868(0.922)
b) Summary of AUC outcomes in the generalizability test with 10% known anomalies					
Mnist	0.767 (0.989)	0.730(0.927)	0.955 (0.878)	0.934 (0.930)	0.962 (0.986)
Letter	0.913 (0.944)	0.929 (0.979)	0.951 (0.912)	0.928 (0.944)	0.925 (0.942)
Thyroid	0.606(0.757)	0.402 (0.646)	0.634 (0.664)	0.616 (0.670)	0.639 (0.774)
Optdigits	0.928 (0.970)	0.922 (0.950)	0.979 (0.961)	0.939 (0.974)	0.959 (0.973)
Average	0.804(0.915)	0.746(0.876)	0.880(0.854)	0.854(0.880)	0.871(0.919)

4.7 Chapter summary

In this chapter, a novel optimized adaptive outlier detection ensemble was presented together with the experiments conducted. The ensemble adaptively boosts the performance of its weak predecessor learners by using decision weights and optimizing model parameters. It uses a set of heterogeneous weak learners to induce diversity at the model level, and then it re-weights training samples to induce diversity at the sample level. Express optimization of parameters in adaptive anomaly detection ensembles is novel. It is an attempt to tune the parameters of the base models to ensure diverse but accurate results. The proposed ensemble was tested on ten benchmark datasets and found superior to the baselines selected. Experiments depicted an improvement in results, even when the least of known anomalies, single cases up to 10%, were used. This means that the proposed ensemble was effective even with a limited number of anomalies.

CHAPTER FIVE

SUMMARY AND DISCUSSION

5.1 Introduction

This study's primary goal was to create a model for outlier detection utilizing a heterogeneous hybrid ensemble of weak learners to improve performance while prioritizing minimization of bias, variance, and order of base learners. The specific objectives were:

(i) To find out what classifiers constitute weak learners for constructing the base (detectors) for the outlier detection ensemble.
(ii) To identify different combination sequences or fusion strategies (order) from the selected base learners for the outlier detection ensemble.
(iii) To create a model for outlier detection that utilizes multiple weak learners in a hybrid ensemble structure to provide improved performance and accuracy while prioritizing minimization of bias, variance, and classifier fusion order.
(iv) To evaluate the developed ensemble model for outlier detection accuracy.

5.2 Objectives re-examination

The overall objective of this study was to create a model for outlier detection utilizing a heterogeneous hybrid ensemble of weak learners to provide improved performance. This research's particular objectives are listed in section 1.2.2. The first objective was to provide a literature review on existing and state-of-the-art detection models used in outlier detection ensembles. The study reviewed the ensemble concept in section 2.1, ensemble construction methods in section 2.2, selecting ensemble learners in 2.3, combining ensemble learners in 2.4, outlier detection overview in section 2.5, aspects of outlier detection in section 2.6, ingredients of outlier detection in section 2.7, bias-variance trade-off in outlier detection including

reduction methods in sections 2.8 and 2.9, respectively. Section 2.10 provided a critique, and section 2.11 provided the research gap from closely related work in outlier detection using different ensembles.

Using the review findings of objective 1, the research gap in section 2.11 was identified. The literature review showed that no standard or best method for outlier detection problems reduced bias and variance or focused on the order of weak learners. The advantages and disadvantages of each technique varied hence the need to develop an ensemble for outlier detection with a focus on minimizing bias, variance, and order of base learners. Although there were several techniques for detecting outliers for a given dataset, not one technique was deemed to be the universal choice. An outlier detection ensemble was proposed. The method used heterogeneous weak learners to reduce ensemble bias and variance. Existing literature did not show much work on outlier detection using heterogeneous weak learners, whose performances were assessed based on their regions of competency, nor improved by margin optimization, hence the opportunity to develop an outlier detection ensemble that took advantage of the base learners' areas of expertise in different datasets to improve detection accuracy.

The second objective intended to identify different combination sequences or fusion strategies (order) from the selected base learners for outlier detection. The objective aimed at harnessing the strength of selection as well as alleviating single learner weaknesses. The literature in sections 2.2, 2.3, 2.4, 2.8, 2.9, and 2.10 informed the development of the proposed hybrid outlier detection ensemble method in section 3.5

The third objective was to create a model for outlier detection that utilized multiple weak learners in a hybrid ensemble to provide better overall performance and accuracy while prioritizing the minimization of bias, variance, and order of base learners. This objective was intended to address the issues identified in section 2.11 using the proposed method in section 3.5. Experiments for creating the models, selecting the models, and improving the models' performance were conducted in sections 4.1 to section 4.3.6.

The fourth objective was to evaluate the developed ensemble model for outlier detection accuracy. The objective intended to evaluate the performance of the developed model. Performance evaluation metrics for data mining tasks were reviewed in section 2.7.6. Experimental tests for assessing and comparing the performance of the proposed ensemble with existing ensembles were conducted in sections 4.4, 4.5 to 4.6.

5.3 Selection of Base Learners by Error Rates and Local Domain Competence

This study proposed a heterogeneous adaptive boosting ensemble for outlier detection. Different base learners were selected since it was deemed effective when base learners of dissimilar types were used. This finding was also true and in line with the conclusions drawn by (Rayana et al., 2017). Through their differences, unique properties in data are discovered or learned. It was clear that when base learners of the same type (homogeneous) were used, the advantage of learner fusion was lost unless different data subsamples for training, different data features, or tuned learner parameters were used. This work focused on unsupervised outlier detection, techniques that assigned a score to individual data points and allowed ranking those points based on their outlierness score. Distance and density-based methods were the chosen unsupervised methods for the task of outlier detection. This choice was influenced by the idea that outliers were by themselves observations that deviate so much from other observations (Hawkins, 1980). The distance-based algorithm, i.e., *k*-NN, was selected to detect global outliers, and the density-based algorithm, i.e., LOF, was selected to detect local outliers. For *k*-NN, the distance of individual instances was used to generate outlierness score, while for LOF, the local deviation of a given instance with respect to its neighbours was used to generate outlierness score.

Furthermore, this work was motivated by the importance of data locality and dynamic learner fusion from DCSO (Zhao & Hryniewicki, 2018; Wang & Mao, 2019) and the concept of heterogeneous detector formation by SELECT (Rayana & Akoglu, 2016). Our method used same but weighted training versions of the data as opposed to random subsampling or boot-strapping. The same training set was

repeatedly used, so it did not need to be as large as other methods required. Successive weak learners were trained using re-weighted versions of the training data, re-weighted according to the misclassification (error) by the previous weak learners. This allowed weak learners to focus on outliers that the previous ones did not detect well. The ensemble first selected optimal base learners by their error rates in every iteration, hence eliminating the weak learners that did not predict at least half the datasets correctly. The testing instances focused on local domains or regions within the training datasets. Our method assessed the capability or competency of each base detector before fusion, and since most outlier data had no actual labels or ground truth, a simulated ground truth called 'target' was created using maximization or averaging of scores of the selected weak learners. Both maximization and averaging methods generated scores for training data, unlike those of generic/global methods that generated scores for the test data. Avg*k*NN was used for setting the local domain, such that, for every test instance, its local domain was derived as a set of its k nearest training objects by Euclidean distance.

In addition to the first selection, that was based on the error rate of each weak learner, a second selection was performed by obtaining the local simulated label $target^{\ell_j}$ for every test instance, where the values of $target$ with respect to the local domain ℓ_j were used. The local training outlier scores $O(\ell_j)$ were obtained from the previously generated training score matrix. To determine the competence of each base detector in the local domain, a Pearson Correlation similarity measure between the base detector score $b_r\left(X_{train}^{\ell_j}\right)$ and the simulated label $target^{\ell_j}$was taken. This method was considered more reliable in outlier detection as it took a similarity measure in evaluating detectors instead of absolute accuracy, as most outlier datasets had no ground truth and were unpredictable and imbalanced in most cases. This method was also earlier used by (Isadora et. al., 2016). The base detector b_r^* with the

highest similarity measure was chosen as the optimal base detector, with its outlier score $b_r^*\left(X_{test}^{(j)}\right)$ retained as an intermediate result for later use.

5.4 Model Diversity and Optimization by Margin Maximization

Each detector scores separated the potential outliers from the rest of the data, with high scores assigned to the outliers and low scores to the inliers. A contrast between the two scores distinguished the outliers from the other data in a dataset. However, these scores did not represent a clear contrast between the outliers and the rest of the data, so score optimization was required. This method created a clear contrast by maximizing outlier scores and minimizing inlier scores, which improved outlier detection. (Clark, Liu, & Japkowicz, 2018) utilized adaptive score threshold to separate outliers from other data, and their method would have been ideal but not applicable in our case since outlier scores did not always match. Our method conforms to other methods like (Cervantes et al., 2020) about margin maximization hyperplane, which are effective and generalize data better.

The score margin was defined as the difference between the known and unknown or unseen outliers present in the already analysed dataset. For the unknown or unseen outliers not to affect this difference, the median value of the score distributions was introduced, which took the 50th percentile of the distribution. Outliers were few (rare), so using the median method provided a robust measurement of the outlier scores as it was not greatly affected by the unknown or unseen outliers in the dataset. For every selected base model, the optimization maximization was based on the values of the parameters that maximized the distance between the medians of the scores. In most outlier detection ensembles, only well-performing base models were selected for fusion (Xu et al., 2019); however, in our work, instead of only selecting the well-performing base models, we further optimized their parameters and then adaptively trained them to detect anomalies before fusion.

The base model optimization by score margin maximization only improved the detection accuracy of the model but did not guarantee that the scores of base models

were diverse. Their errors needed to differ so that when fused into one model, it addresses the shortcomings of the individual base models. This technique in this study conforms with the study by (Zimek et al., 2014), who utilized varying errors to improve their model performance. In our case, this was intended to reduce the overall model's biases further. For the diversity between base models' scores, we adjusted the optimization to reduce the correlation between two score vectors similar to (Reunanen et al., 2020). In a nutshell, the known outliers from the analyzed dataset were utilized in two ways: first, to create a contrast between the outliers and other data, and second, to obtain diverse outcomes from the base models before final fusion.

5.5 Fusion of Scores of the Heterogeneous Base Learners

Our final result was a fusion of carefully selected results from the individual learners. As base detectors were heterogeneous, their scores varied in range and interpretation, and fusing them directly would have been erroneous. An agreement was needed within the ensemble. From chapter 2, section 2.7.5, agreement methods were grouped into rank-based and score-based. In rank-based, detector scores are ordered into ranked lists then aggregation is performed where they are merged into a single ranked list.

On the other hand, score-based methods convert outlier scores into probabilities either by exponential or Gaussian scaling, by posterior probabilities, regularization, or normalization, to make outlier scores across detectors comparable, then take a final score by either averaging or maximization. Other score-based methods like mixture modeling convert outliers using a model where scores are samples from a combination of exponential (representing inliers) and Gaussian (representing outliers) distributions, and one can convert scores into probabilities and then provide binary classes for the instances with probabilities greater than half getting value 1, i.e., outliers and 0 otherwise, i.e., inliers. Our work adopted score-based probabilities and took the maximum-of-average of top h performing detectors in relation to their target, or average-of-maximum of h chosen detector subgroups in relation to their target as the subgroup's score; then obtained the final score by picking the maximum

among all subgroups' scores. These two methods were utilized to reduce bias, which reduced the risk of only picking one best-performing base detector. Biasness of the ensemble was greatly reduced by the fact that only top *h* performing base detectors in relation to target were selected and for with which *h* detectors did not increase variance overall.

In general, four fusion variants were created at different levels: maximization and averaging in the first level, and maximum-of-average and the average-of-maximum of base learners' scores in the second level. The selected optimal base learners' outcomes were carefully fused to reduce variance and bias while improving overall ensemble accuracy. Our method was tested on ten benchmark datasets and showed improved results compared to existing baselines using global scores (section 5.6). The fusion by the average-of-maximum method showed promising results compared to the other three variants and was deemed a better fusion strategy.

5.6 Performance Assessment of the Proposed Method

This research sought to determine the performance of the proposed heterogeneous ensemble method in relation to other existing algorithms. The first comparative study was done using various baselines' outcomes of generic or global fusion by averaging (G_Avg), maximization (G_Max), average-of-maximum (G_AvgM), maximum-of-average (G_MaxA), weighted averaging (G_Wa), and global threshold summation (G_Ts) versus the proposed methods' variants for fusion by local domain averaging (ADAHO_Avg), maximization (ADAHO_Max), maximum-of-average (ADAHO_MaxA) and average-of-maximum (ADAHO_AvgM). The second comparative study was done using various outlier detection ensembles appreviated as ALOI, BASE, ADAHO and SELECT, because of their relative design with respect to our method. ALOI and BASE, similar to our proposed method, used LOF and KNN algorithms, with the only difference being that they used a heuristic method instead of optimizing model parameters. In principle, ALOI and BASE followed our Algorithm 6 but their $\mathcal{D}_o$ values as described in section 3.5.4.1 were empty. In addition, the two did not follow steps 2–5 of Algorithm 6 but used equal weights

$W_t = 1$ for all $t = 1, \ldots, T$. The two baselines set parameters for their base models in

the same way as our proposed method and since the base models were similar, they utilized similar distance metrics as well as some of the fusion approaches like summation, median, averaging and maximization. Specifically, both approaches utilized summation and Euclidean distances. Our method however, utilized the median during optimization and took the final score in two levels, that is, by maximum of average and by average of maximum. SELECT presented two sub-formations referred to as vertical-SELECT and horizontal-SELECT. We compared our method to horizontal-SELECT because, according to (Rayana & Akoglu, 2016), it achieved better performance. ADAHO (Bii et al., 2020), which partly learned similar base models as our proposed method and weighted its samples as well as its base models by adaptive boosting, was also compared. The main difference was that it did not consider margin maximization between outlier data and normal data. Hence it did not loop through the last part of step 7 of algorithm 6. The proposed method was tested against these baselines using the least known outlier percentage. It was assumed that using the least percentage of outliers was the hardest test since very little prior information about the dataset was provided, which meant, if the proposed method outperformed baselines, a major improvement was achieved overall.

The proposed method was tested on ten benchmark datasets and was superior to the baselines. Experiments showed an improvement in results, even when the least of known outliers, single cases up to 10%, were used, which meant that our method was effective even where there was a limited number of outliers. The strength of our method was in three pillars: adaptive boosting, sample and detector weighting, and parameter optimization. We ignored base models whose error rates were more than half. In every round of training, samples gained new weights based on previous model errors, and base model parameters were adjusted to optimize the margin maximization between outliers and the inliers. Classification performance metrics of recall (REC), false positive rate (FPR), receiver operating characteristics curve (ROC), and the area under ROC (AUROC) were used. These metrics are widely used in anomaly detection (Zhao & Hryniewicki, 2018). According to (Campos et al., 2016), a recall metric performance measure is strongly recommended for binary and

multiclass datasets. Thus this study used Recall (TPR) as one of the evaluation performance metrics for testing the performance of the proposed method, as discussed in chapter 2. Scores above a threshold (T = 0.5) were considered outliers and vice versa. Furthermore, a value of T inversely affected both REC and FPR values as a higher T value caused low REC and FPR values, and the reverse was true; that is, T was a tradeoff. ROC graph was used with different T values to evaluate the proposed model's efficiency with FPR plotted on the horizontal axis, and REC plotted on the vertical axis. AUROC values were computed ranging (0, 1) from the ROC graph so that an optimal AUC value was close to 1.

The AUC values were compared to determine the overall performance of the model. Other related works, such as (Isadora et al., 2016), utilized 1-10% of the anomalies without adaptive learner boosting, while (Micenkova et al., 2014) utilized half of the anomalies. The strength of the proposed method was in three pillars: adaptive boosting, point weighting based on the local domain, and score optimization. The method ignored weak learners whose error rates were more than half, which could deteriorate the final ensemble. In every iteration, samples gained new weights based on previous errors and parameters that were adjusted. To decide if two outcomes contained significant differences, the paired t-test was utilized to statistically analyse results of experiments (Rietveld & Van Hout, 2017), which measured the average difference between paired samples. In this test, the null hypothesis H0 is zero, given that the difference (z) in AUC scores of two diverse base models is zero. If the p-value was less than 0.05 the test was declared statistically significant.

The study findings of this work showed that the proposed method had statistically significant performance compared with the baseline ensembles in outlier detection. Further observation revealed that there was no uniform performance with the datasets. Thus the choice of datasets may affect the performance of base detectors. Literature in chapter 2 echoed that heterogeneous learners had the advantage of generating different errors in different domains, which improves the ensemble's stability and outlier detection. This study supports the claim by (Zhao et al., 2019), who empirically showed that ensembles often produce better results when there is a substantial disparity among the base learners.

The similarity measure between the base learner outcomes and the simulated ground truth determined the evaluation of weak learner competency. It was, however, clear that not much difference was observed when a Friedman test was performed on both Euclidean distance and Person correlation about the Receiver Operating Characteristic and the average Precision as the performance variance was so minimal, that is, less than one percent. Furthermore, the weight assignment to base learners did affect the proposed model positively in terms of performance, as it helped detect harder examples in every iteration. The setting of the local domain for every test instance proved to be computationally expensive. To lessen computational cost, other measures, like the determination of the value of *k* for the size of the local domain, could be normalized as long as it does not affect performance, even when outliers are within dense domains. In this study, the Euclidean distance between the two became the most effective when the size of *k* was set to a large value so that the local target and the detector outcomes were normalized.

5.7 Chapter summary

A discussion of the research study findings has been provided. The discussion compared the study findings with the literature provided in chapter 2. The discussion has shown that the proposed method outperformed most of the existing ensembles in terms of accuracy in outlier detection because of optimization and reduced bias and variance. The next chapter provides conclusions and recommendations for future work.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter provides knowledge contributions that were realized in the course of the study. A conclusion for the study is provided. The chapter ends by providing a list of the various journals and publications produced and conferences presented based on this research study.

6.2 Knowledge contributions

The major contribution of the research was the development of an optimized adaptive boosting model of heterogeneous ensembles for outlier detection. The model utilized well-known distance-based and density-based algorithms as its weak base learners. Thus the technique applied many strategies and ensemble techniques in developing the model.

This study provides a set of contributions. In this research study, well-performing or optimal heterogeneous base learners were first selected based on their ability to detect at least half the data instances correctly, that is, by their error rates being less than half, and secondly by assessing their capability in reference to their local domains or areas of expertise before optimization; this was because every outlier detection technique performed best within a specific domain in the entire space. This selection positively impacted the performance of base models and made the method empirically testable, justifiable, reliable, and stable. We have no knowledge of any previous similar work.

In this study, distance-based and density-based outlier detection algorithms formed the weak base learners that enabled the discovery of both local and global outliers. This was in line with Hawkins's classical definition of outliers. The study demonstrated that outliers could be found by assessing the distances of the neighborhood of every data point from other data points; and that heterogeneous base models could be combined into a single function for evaluating overall outlierness,

where each base model performed differently in certain spaces and produced outlier scores of different types and scales. Furthermore, outliers can be found in dense or less dense (distant) neighborhoods, and finding all outliers is critical.

The study showed that decision weights could be used adaptively to boost the outcomes of predecessor base models. In this study, the proposed method utilized decision weights to adaptively boost the outcomes of its predecessor base models in the first phase, and then optimized the parameters that maximized the score margins of its base models in the second phase. Finally, it fused selected heterogeneous well-performing base models in the last phase to achieve better predictions. The margin maximization created a clear contrast between outliers and normal data samples by maximizing the outliers' scores and minimizing the rest of the data. This score contrast was critical in optimizing outlier detection and improving overall detection accuracy by reducing bias and variance. We have no knowledge of any previous similar work.

The study also demonstrated how bias and variance could be reduced by combining models in a bi-level structure that guaranteed diversity at different levels. The method in this study used different kinds of base models and re-weighted training samples in every training round, thereby not only inducing diversity at the model level (through model heterogeneity) but also at the sample level (through re-weighting samples) and at the score level, which made it effective even where there was a limited number of outliers. The order of base learners showed that the first detector in the ensemble had priority to decide about the outlierness of a given instance. The higher the accuracy of the first detector, the fewer the number of training iterations. The study produced a model that depicted better performance in outlier detection. We have no knowledge of any previous similar work.

Finally, the study findings reaffirmed that ensemble learning of several weak learners over similar tasks results in better performance than any individual learner.

6.3 Conclusion

The primary research's goal was to create a model for outlier detection that utilizes multiple weak learners in a hybrid ensemble structure to improve performance and accuracy while prioritizing the minimization of bias and variance. An adaptive boosting heterogeneous ensemble model for outlier detection was developed using distance-based and density-based algorithms as the base classifiers. It selects optimal base learners in relation to their local domains and fuses their outputs with the aim of reducing variance and bias. The model effectively assessed learners' prediction capability through error rates. The score margin maximization technique was applied to increase the contrast between classes, so as to increase the predictability of outlier classes. Since poor-performing learners degrade the performance of the ensemble, they are eliminated. The proposed model was tested on ten benchmark datasets and found to be superior in performance to the baselines using Recall, Precision, and ROC performance measures. The research study demonstrated that the more diverse the base learners' errors are, the better the combination power. The results of experiments were presented as cross-tabulations, with detailed explanations and interpretations.

6.4 Future work

This study determined a test instance's local domain using nearest neighbors. Some areas that can be investigated include setting local domains to reduce the time taken establishing nearest neighbors. Also, a way of defining the value of k should be considered as features keep changing, and k must be dynamic. This method is extensively applicable to other kinds of ensembles that not only lean toward outlier detection. It unlocks endless possibilities for research, including furthering investigations into the choice of base models, dynamic parameter adjustments, bias-variance supervision frameworks, and adaptive evolutionary methods that could accelerate margin optimization, among others, that could reduce the cost of computation overall. Finally, as regards the proposed method's scalability, deep learning techniques could further extend its application to ultrahigh-dimensional spaces.

6.5 Publications and Conferences

1st Publication (SCI-Indexed):

Paper: Adaptive Boosting in Ensembles for Outlier Detection: Base Learner Selection and Fusion by Local Domain Competence

Important dates: Submitted: 12 Apr 2019 | Revised: 29 Oct 2019 | Accepted: 12 Dec 2019. First published: March 30, 2020

Journal: *ETRI Journal, Volume 42, Issue 6 December 2020 Pages 886-898*

DOI: 10.4218/etrij.2019-0205

Link: https://onlinelibrary.wiley.com/doi/full/10.4218/etrij.2019-0205

2nd Publication (EI-Indexed):

Paper: OAAE: Optimized Adaptive Anomaly Detection Ensemble: Base Model Boosting By Parameter Optimization

Important dates: Submitted: 19 Mar 2021 | Revised: 13 Jul 2021 | Accepted: 27 Jul 2021. First published: August 22, 2021

Peer Review link: https://publons.com/publon/49146156/

Journal: *Engineering Reports, Volume 4, Issue 2 February 2022*
DOI: 10.1002/eng2.12449

Link: https://onlinelibrary.wiley.com/doi/10.1002/eng2.12449

Conference: Bii, J., Rimiru, R., & Waweru, M. (2018, September). Improved adaptive boosting in heterogeneous hybrid ensembles for outlier detection: prioritizing minimization of bias, variance and order of base learners. In *Proceedings of the 2018 4th Annual International Maasai Mara University Conference (pp. 40 - 41).* Narok, Kenya — September 11- 13, 2018

REFERENCES

Aggarwal, C. C. (2021). *Outlier Analysis* (2nd ed.). London: Springer International Publishing.

Aggarwal, C. C., & Sathe, S. (2015). Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, *17*(1), 24-47.

Aggarwal, C. C., & Sathe, S. (2017). *Outlier Ensembles: An Introduction*. London: Springer.

Aggarwal, P., Gonzalez, C., & Dutt, V. (2020). HackIt: A real-time simulation tool for studying real-world cyberattacks in the laboratory. In B. Gupta, G. Perez, D. Agrawal, & D. Gupta (Eds.), *Handbook of Computer Networks and Cyber Security* (pp. 667-680). London: Springer.

Akaike, H. (1970). Statistical predictor information. *Annals of the Institute of Statistical Mathematics*, 22, 203–217.

Alpaydin, E. (2020). *Introduction to Machine Learning* (3rd ed.). Massachusetts: MIT Press.

Allam, V. (2019). Detecting fraudulent credit card transactions using outlier detection. *International Journal of Scientific & Technology Research*, *8*, 630-637.

Amil, P., Almeira, N., & Masoller, C. (2019). Outlier mining methods based on graph structure analysis. *Frontiers in Physics*, *7*, 194.

Angelin, B., & Geetha, A. (2020). Outlier detection using clustering techniques – K-means and K-median. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 373-378). IEEE.

Angiulli, F., Basta, S., Lodi, S., & Sartori, C. (2010). A distributed approach to detect outliers in very large data sets. In M. L. Gavrilova, C. J. Kenneth Tan,

& M. I. Lourdes (Eds.), *Transactions on Computational Science XI* (pp. 439-461). Springer. https://doi.org/10.1007/978-3-642-15277-1_32

Barai (Deb), A., & Dey, L. (2017). Outlier detection and removal algorithm in K-Means and hierarchical clustering. *World Journal of Computer Application and Technology, 5*(2), 24-29.

Barnett, V., & Lewis, T. (2021). *Outliers in statistical data* (4th ed.). New York: Wiley.

Beckman, R. J., & Cook, D. (1983). Outliers with discussion. *Technometrics, 25*, 119-149.

Bhattacharyya, S., & Kalita, J. K. (2019). Malware classification: A survey. *Computers & Security, 83*, 81-105.

Bii, J. K., Rimiru, R., & Mwangi, R. W. (2020). Adaptive boosting in ensembles for outlier detection: Base learner selection and fusion via local domain competence. *ETRI Journal, 42*(1), 1-13.

Bouguessa, M. (2012). *A probabilistic combination approach to improve outlier detection*. In Proceedings of the IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 666–673).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (1999). *Using adaptive bagging to debias regressions* (Statistics Department Technical Report), Berkeley: University of California.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Breuning, M., Kriegel, H., Ng, R. T., & Sander, J. (2000). LOF: Identifying density based local outliers. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 1-12).

Breve, F. A., Ponti-Jr., M. P., & Mascarenhas, N. D. A. (2007). Multilayer perceptron classifier combination for identification of materials on noisy soil science multispectral images. In *Proceedings of the 20th Brazilian Symposium on Comp Graphics and Image Processing* (pp. 239-244). MG: IEEE.

Brownlee, J. (2019). How to Develop a Skillful Ensemble of Machine Learning Models. *Machine Learning Mastery*. Retrieved from https://machinelearningmastery.com /ensemble-machine-learning-algorithms-python-scikit-learn/

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in CNNs. *Neural Networks, 106*, 249-259.

Burnaev, E., Erofeev, P., & Smolyakov, D. (2015). Model selection for anomaly detection. In *International Conference on Machine Vision*.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenkova, B., Schubert, E., Assent, I., & Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, *30*(4), 891-927.

Carter, P. (2019). Database consistency. Retrieved from https://doi.org/10.1007/978-1-4842-5089-1_9.

Cervantes, J., Garcia-Lamont, F., Rodriguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing, 405*, 43-67.

Chander, B., & Kumaravelan, G. (2022). Outlier detection strategies for WSNs: A survey. *Journal of King Saud University - Computer and Information Sciences*, *34*(8), 5684-5707.

Chen, G., Du, L., & B. (2020). An ordinal outlier algorithm for anomaly detection of high-dimensional data sets. In *2020 Chinese Control and Decision Conference (CCDC)* (pp. 5356-5361). IEEE.

Chen, T., Zhang, J., & Xue, J. H. (2019). Heterogeneous ensembles for outlier detection: Leveraging strengths of multiple models. *Knowledge-Based Systems, 166*, 115-128.

Christy, A., Meeragandhi, G., & Vaithyasubramanian, S. (2015). Cluster based outlier detection algorithm for healthcare data. *Procedia Computer Science, 48*, 368-375.

Clark, J., Liu, Z., & Japkowicz, N. (2018). Adaptive threshold for outlier detection on data streams. In *IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 76-85). IEEE.

Cousineau, D., & Chartier, S. (2010). Outliers detection and treatment: A review. *International Journal of Psychological Research*, *3*(1), 58–67.

Das, S., Wong, K., Dietterich, T., Fern, A., & Emmott, A. (2016). Incorporating Expert Feedback into Active Anomaly Discovery. In *Proceedings of the IEEE International Conference on Data Mining* (pp. 853-858).

Feldbauer, R., & Flexer, A. (2019). A comprehensive empirical comparison of hubness reduction in high-dimensional spaces. *Knowledge and Information Systems*, *59*(1), 137-166.

Foorthuis, R. (2021). On the nature and types of anomalies: a review of deviations in data. *International Journal of Data Science and Analytics*, *12*, 1-35.

Fowke, K. R., Nagelkerke, N. J., & Kimani, J. (1996). Genital ulcer disease in female sex workers in Nairobi: Results of a cohort study. *The Lancet, 348*(9033), 1347-1352.

Freund, Y., & Schapire, R. E. (1997). A decision–theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

Freund, Y., & Schapire, R. E. (1996). Game theory, online prediction and boosting. In *Proceedings of the 9th Annual Conference on Computational Learning Theory* (pp. 325–332). ACM press.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Bari, Italy.

Fumera, G., & Roli, F. (2005). A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*, 924–956.

Gao, J., & Tan, P.N. (2006). Converting Output Scores from Outlier Detection Algorithms to Probability Estimates. In *Proceedings of the Sixth International Conference on Data Mining (ICDM)*.

Ghorbani, H. (2019). Mahalanobis distance and its application for detecting multivariate outliers. *Facta Universitatis Series Mathematics and Informatics, 34*(4), 583-593.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics, 11*(1), 1-21.

Gupta, P., Krishnamurthy, V., & Nasraoui, O. (2019). Deep learning for anomaly detection: A survey. *ACM Computing Surveys (CSUR), 51*(3), 1-36.

Guo, F., Shi, C., Li, X., He, J., & Xi, W. (2018). Outlier Detection Based on the Data Structure. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-6).

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). New York: Morgan Kaufmann.

Hansen, L. K., & Salamon, P. (1990). Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(10), 993-1001.

Haque, M., Berretta, R., & Moscato, P. (2016). Heterogeneous Ensemble Combination Search Using Genetic Algorithm for Class Imbalanced Data Classification. *PLoS One, 11*(1), e0146116.

Hawkins, D. (1980). *Identification of Outliers*. London, UK: Chapman and Hall.

Huang, B., Dong, J., Mao, Y., Ji, X., & Liu, Z. (2019). A survey of ensemble learning in deep neural networks. *Information Fusion, 52*, 1-16.

IEEE International Conference on Data Engineering. (2012). In *Proceedings of the Conference on Data Engineering (ICDE)* (pp. 1037-1048). Washington, DC, USA.

Isadora, N., Pavlos, P., Brandon, S., & Wesley, C. (2016). Ensemble Learning Method for Outlier Detection and Its Application to Astronomical Light Curves. *The Astronomical Journal, 152*, 71.

Janssens, J. H. M., Huszár, F., Postma, E. O., & van den Herik, H. J. (2012). Stochastic outlier selection. *Tilburg centre for Creative Computing, tech. report, 1*, 2012.

Lopez-de-Lacalle, J. (2017). Detection of Outliers in Time Series. Package outliers, Ver. 0.6-6. Retrieved from https://jalobe.com

Jaw, E., & Wang, X. (2021). Feature Selection and Ensemble-Based Intrusion Detection System: An Efficient and Comprehensive Approach. *Symmetry, 13*(10), 1764.

Jiang, H., Zhang, K., Wang, J., Wang, X., & Huang, P. (2020). Anomaly Detection and Identification in Satellite Telemetry Data Based on Pseudo-Period. *Applied Sciences*, *10*(1), 103.

Jiang, S., & An, Q. (2008). Clustering-based outlier detection method. In: Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery, *FSKD '08, 2008, 2,* 429–433.

Jiang, Z., Zhang, F., Xu, H., Tao, L., & Zhang, Z. (2022). MEOD: A Robust Multi-stage Ensemble Model Based on Rank Aggregation and Stacking for Outlier Detection. In *Knowledge Science, Engineering and Management. KSEM 2022* 238-251. Springer, Cham.

Johnson, A. (2022). Evaluating outlier detection performance using mean squared error. *Journal of Machine Learning Research*, *23*(1), 45-62.

Kaur, K., & Garg, A. (2016). Comparative Study of Outlier Detection Algorithms. *International Journal of Computer Applications*, *147*(9), 1-5.

Kalinichenko, L., Shanin, I., & Taraban, I. (2014). Methods for anomaly detection: A survey. In *CEUR workshop proceedings* (Vol. 1297, p. 2025).

Kamalov, Firuz & Leung, Ho-Hon. (2020). Outlier Detection in High-Dimensional Data. *Journal of Information and Knowledge Management*, *19,* 2040013.

Kandhari, A., Aggarwal, C. C., & Das, K. (2018). Outlier ensemble detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *12*(4), 1-35.

Keller, F., Muller, E., & Bohm, K. (2012). Hics: High contrast subspaces for density-based outlier ranking. In *Proceedings of the 2012 IEEE 28th International Conference*.

Khullar, D., Jha, A. K., & Jena, A. B. (2015). Reducing diagnostic errors- why now. *New England Journal of Medicine*, *373*, 2491-2493.

Kieu, L. M., & Nguyen, T. T. (2020). A Comparative Study of Machine Learning Techniques for Fraud Detection in Banking Sector. *Journal of Business Research*, *122*, 298-308.

Klementiev, A., Roth, D., & Small, K. (2007). An unsupervised learning algorithm for rank aggregation. In *Proceedings of the 18th European Conference on Machine Learning (ECML'07)* (pp. 616-623). Springer.

Knorr, E. M., & Ng, R. T. (1998). Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)* (pp. 392-403). ACM Press.

Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: Algorithms and applications. *VLDB Journal*, *8*(3-4), 237-253.

Kriegel, H., Kröger, P., Schubert, E., & Zimek, A. (2009). Outlier detection in axis-parallel subspaces of high dimensional data. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (pp. 831-838). Springer.

Kumar, S., Kaur, P., & Gosain, A. (2022). A comprehensive survey on ensemble methods. In 2022 *International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 1357-1362). IEEE.

Larson, E. R., & Moore, J. A. (2022). Spatial data analysis of ecological data using machine learning techniques. *Ecological Modelling*, *488*, 109460.

Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 157-166).

Lei Cao, Di Yang, Qingyang Wang, Yanwei Yu, Jiayuan Wang, & Elke A Rundensteiner. (2014). Scalable distance-based outlier detection over high-volume data streams. In *Data Engineering (ICDE), IEEE 30th International Conference on* (pp. 76–87).

Li, X., Guo, L., Zhang, X., & Jin, X. (2021). Deep learning for anomaly detection: A review. *Pattern Recognition, 110,* 107638.

Lichman, M. (2013). UCI machine learning repository. Retrieved from http://archive.ics.uci.edu/ml.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, 16(3), 30-57.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2018). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD), 12*(1), 1-39.

Liu, Sophia, Ngan, Henry Y.T., Ng, M., & Simske, Steven. (2018). Accumulated relative density outlier detection for large scale traffic data. *Electronic Imaging*, *2018*(9), 1-10.

Ma, Y., Zhao, X., Zhang, C., Zhang, J., & Qin, X. (2021). Outlier detection from multiple data sources. *Information Sciences*, *580*, 819-837.

Merza, E., & Mohammed, N. (2021). Fast Ways to Detect Outliers. *Journal of Techniques*, *3*, 66-73.

Micenkova, B., McWilliams, B., & Assent, I. (2014). Learning outlier ensembles: the best of both worlds-supervised and unsupervised. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, ODD '14*.

Muller, E., Assent, I., Steinhausen, U., & Seidl, T. (2008). Outrank: ranking outliers in high dimensional data. In *ICDE Workshops* (pp. 600-603).

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Murty, M. N., Jain, A., & Mukhopadhyay, S. (2021). Machine learning: a review. *Artificial Intelligence Review*, *54*(2), 1035-1078.

Nguyen, H., Ang, H., & Gopalkrishnan, V. (2010). Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *Proceedings of the 15th*

International Conference on Database Systems for Advanced Applications, DASFAA'10 (pp. 368–383). Springer, Berlin, Heidelberg.

Niemann, F.; Ludtke, S.; Bartelt, C.; ten Hompel, M. (2022). Context-Aware Human Activity Recognition in Industrial Processes. *Sensors*, *22*(1), 134.

ODDS Library. (2016). Retrieved from http://odds.cs.stonybrook.edu/.

Otero Gomez, D., Agudelo, S. C., Patiño, A. O., & Lopez-Rojas, E. (2021). *Anomaly Detection applied to Money Laundering Detecion using Ensemble Learning* (No. f84ht). Center for Open Science.

Papadimitriou, S., Kitagawa, H., Gibbons, P., & Faloutsos, C. (2003). LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the International Conference on Data Engineering (ICDE)* (pp. 315–326). IEEE Computer Society.

Pawar, A. M., & Mahindrakar, M. S. (2015). A comprehensive survey on online anomaly detection. *International Journal of Computer Applications*, *119*, 17.

Pintelas, P. E., & Livieris, I. E. (Eds.). (2020). *Ensemble algorithms and their applications*. Basel: MDPI-Multidisciplinary Digital Publishing Institute.

Prasada, A., Mahapatra, A., Nanda, A., Mohapatra, B., Padhy, A., & Padhy, I. (2020). Concept of outlier study: The management of outlier handling with significance in inclusive education setting. *Asian Research Journal of Mathematics*, *16*, 7-25.

Rayana, S., & Akoglu, L. (2014). An Ensemble Approach for Event Detection in Dynamic Graphs. In *Proceedings of the KDD ODD2 Workshop*.

Rayana, S., & Akoglu, L. (2016). Less is More: Building Selective Anomaly Ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD), 10*(4), 1-33.

Rayana, S., Zhong, W., & Akoglu, L. (2017). Sequential ensemble learning for outlier detection: A bias-variance perspective. In 2017 *IEEE International Conference on Data Mining (ICDM)* (pp. 1167-1172). IEEE.

Reunanen, N., Raty, T., & Lintonen, T. (2020). Automatic optimization of outlier detection ensembles using a limited number of outlier examples. *International Journal of Data Science and Analytics*, *10*(2), 203-218.

Rietveld, T., & van Hout, R. (2017). The paired t test and beyond: Recommendations for testing the central tendencies of two paired samples in research on speech, language and hearing pathology. *Journal of Communication Disorders*, 69, 44-57.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1-2), 1-39.

Roy, A., & Garg, A. (2022). A Comprehensive Study of Various Outlier Detection Approaches. *ECS Transactions*, *107*, 8561.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K. R., & Kloft, M. (2019). Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

Sabzevari, M., Martinez-Munoz, G., & Suarez, A. (2022). Building heterogeneous ensembles by pooling homogeneous ensembles. *International Journal of Machine Learning and Cybernetics*, *13*(2), 551-558.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197-227.

Schapire, R. E. (2004). The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification* (pp. 149-171). Berkeley: Springer.

Schubert, E., Wojdanowski, R., Zimek, A., & Kriegel, H.-P. (2013). On Evaluation of Outlier Rankings and Outlier Scores. *SDM*, 1047-1058.

Shaikh, S. A., & Kitagawa, H. (2012). Distance-based outlier detection on uncertain data of Gaussian distribution. In *Web Technologies and Applications: 14th Asia-Pacific Web Conference, APWeb 2012, Kunming, China, April 11-13, 2012. Proceedings 14* (pp. 109-121). Springer Berlin Heidelberg.

Shwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*, 461–464.

Skelsey, P. (2021). Forecasting Risk of Crop Disease with Anomaly Detection Algorithms. *Phytopathology, 111*(2), 321-332.

Sun, J., Xiong, Y., Zhu, M., & Zhang, W. (2019). Anomaly detection using One-Class SVM and MAE-Based Resampling. IEEE Access, 7, 138074-138085.

Tang, A., Sethumadhavan, S., & Stolfo, S. J. (2014). *Research in Attacks Intrusions and Defenses*. New York: Springer.

Tang, G. (2015). New methods in outlier detection. Simon Fraser University. Retrieved from http://summit.sfu.ca/system/files/iritems1/15321/etd8992GTang.pdf

Tasaki, S., Gaiteri, C., Mostafavi, S., & Wang, Y. (2020). Decoding differential gene expression. *bioRxiv, 2020.01*.10.894238.

Thudumu, S., Branch, P., Jin, J., Guo, J., & Kulkarni, P. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, *7*(1), 42.

Tsoumakas, G., Angelis, L., & Vlahava, I. (2014). *Selective fusion of heterogeneous classifiers.* Greece: Aristotle University of Thessaloniki,.

Tumer, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science, 8*(3-4), 385-204.

Ur Rehman, M., & Khan, D. (2020). Local neighborhood-based outlier detection of high dimensional data using different proximity functions. *International Journal of Advanced Computer Science and Applications*, *11*, 255-260.

Varun, A., & Bhatia, S. (2020). An overview of outlier detection techniques: State-of-the-art, challenges, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *10*(1), e1349.

Wang, B., & Mao, Z. (2019). Outlier detection based on a dynamic ensemble model: Applied to process monitoring. *Information Fusion*, *51*, 244–258.

Wang, J., & Sun, H. (2018). A novel ensemble outlier detection algorithm based on neighborhood density and distance. *Expert Systems with Applications*, *95*, 43-54.

Wu, E., Liu, W., & Chawla, S. (2010). Spatio-temporal outlier detection in precipitation data. In *Knowledge Discovery from Sensor Data: Second International Workshop, Sensor-KDD 2008, Las Vegas, NV, USA, August 24-27, 2008, Revised Selected Papers* (pp. 115-133). Springer Berlin Heidelberg.

Xu, H., Zhang, L., Li, P., & Zhu, F. (2022). Outlier detection algorithm based on Knn-LOF. *Journal of Algorithms & Com Technology*, *16*, 17483026221078111.

Xu, M., Li, X., Zhang, Z., Luo, J., & Zeng, W. (2020). Hybrid ensemble of autoencoders and k-nearest neighbors for credit card fraud detection. *IEEE Transactions on Information Forensics and Security*, *15*, 1076-1090.

Xu, X., Liu, H., & Yao, M. (2019). Recent progress of anomaly detection. *Complexity*, *2019*, 1-11.

Yan, K., You, X., Ji, X., Yin, G., & Yang, F. (2016). A hybrid outlier detection method for health care big data. In *2016 IEEE International Conference on Big Data and Cloud Computing, Social Computing and Networking,*

Sustainable Computing and Communications (BDCloud-SocialCom-SustainCom) (pp. 157-162).

Zhang, J. (2008). *Towards outlier detection for high-dimensional data streams using projected outlier analysis strategy*, Unpublished PhD dissertation, Dalhousie: Dalhousie University.

Zhang, J., Li, Z., & Chen, S. (2020). Diversity aware-based sequential ensemble learning for robust anomaly detection. *IEEE Access, 8*, 42349-42363.

Zhang, K., Hutter, M., & Jin, H. (2009). A new local distance-based outlier detection approach for scattered real-world data. *Computing Research Repository*. Retrieved from http://arxiv.org/abs/0909.4928

Zhao, Y., & Hryniewicki, M. K. (2018). DCSO: Dynamic combination of detector scores for outlier ensembles. Retrieved from https://doi.org/10.13140/RG.2.2.11165.77288

Zhao, Y., & Hryniewicki, M. K. (2018). XGBOD: Improving supervised outlier detection with unsupervised representation learning. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8).

Zhao, Y., Nasrullah, Z., Hryniewicki, M. K., & Li, Z. (2019). LSCP: Locally Selective Combination in Parallel Outlier Ensembles. In *Proceedings of the 2019 SIAM International Conference on Data Mining* (pp. 585-593). Society for Industrial and Applied Mathematics.

Zhu, C., Kitagawa, H., Papadimitriou, S., & Faloutsos, C. (2004). OBE: Outlier by Example. *Lecture Notes in Computer Science, 3056*, 222-234.

Zimek, A., Campello, R. J. G. B., & Sander, J. (2014). Data perturbation for outlier detection ensembles. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management (SSDBM '14)* (pp. 13:1-13:12). ACM.

Zimek, A., Campello, R. J. G. B., & Sander, J. (2014). Ensembles for unsupervised outlier detection: Challenges and research questions. *ACM SIGKDD Explorations, 15*(1), 11-22.

Zimek, A., Gaudet, M., Campello, R. J., & Sander, J. (2013). Subsampling for efficient and effective unsupervised OD ensembles. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 428-436). ACM.

APPENDICES

Appendix I: Details of Datasets and Scatter Plots

Table APX1 1.0: Description for MNIST Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	100(numeric)	7603	700 (9.21)	2	None

Additional information

This modified MNIST dataset has digits (0s) from the original MNIST dataset, with (6s) added as outliers. A random sample of 100 features out of the initial 784 pixels was taken. It comprises 100 columns and 7603 rows, which means that there are many features for a limited number of observations. It also has a high outlier fraction of 9.2 %.

Table APX1 1.1: Sample MNIST Dataset

x91	x93	x94	x96	x97	x98	x99	Class
188.0556	-4.46997	158.3814	27.13142	-2.27463	-0.00065	-12.3513	0
186.0556	-4.46997	123.3814	157.1314	-2.27463	-0.00065	-12.3513	0
149.0556	-4.46997	-93.6186	-75.8686	-2.27463	-0.00065	-12.3513	0
-64.9444	-4.46997	94.89937	-93.8686	-2.27463	-0.00065	-12.3513	1
-64.9444	-4.46997	-137.101	-53.8686	-2.27463	-0.00065	-12.3513	1
-64.9444	-4.46997	23.89937	-93.8686	-2.27463	-0.00065	125.6487	1
-64.9444	-4.46997	-9.10063	-93.8686	-2.27463	-0.00065	-12.3513	1

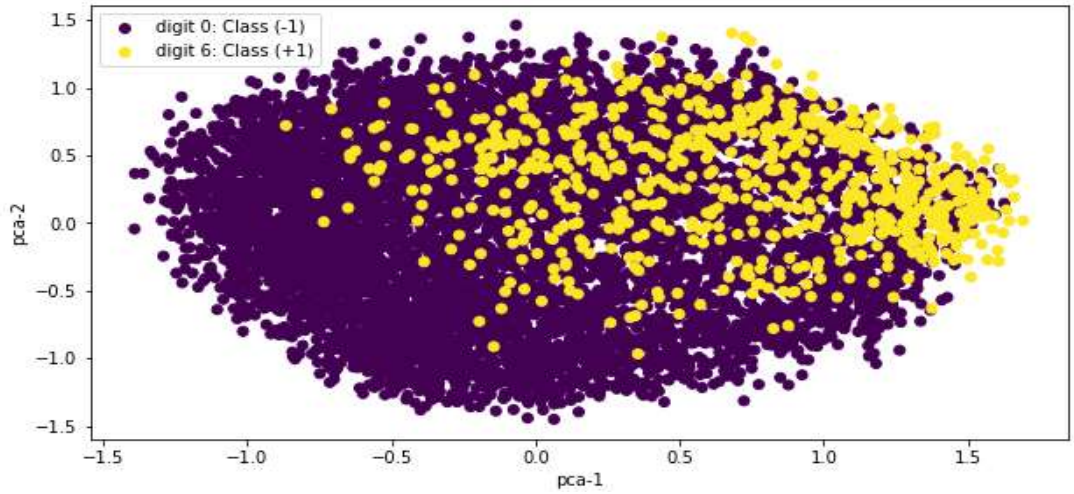

Figure APX1 1.0: Scatter Plot for the MNIST Dataset

Table APX1 2.0: Description for Letter Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	32 (numeric)	1600	100 (0.0625)	2	None

Additional information

The original letter dataset included 26 capital letters in the English alphabet represented in 16 dimensions. To convert it for outlier detection, 3 letters of data are sampled to create the inliers, and then their pairs are randomly joined to double their dimensionality. A few letters that are not inliers are selected at random to make up the outlier class. A total of 1600 data points were sampled with a positive class fraction of 6.250%

Table APX1 2.1: Sample Letter Dataset

	0	1	2	3	4	5	6	7	8	9	...	23	24	25	26	27	28	29	30	31	y
0	6	10	5	6	3	10	6	4	6	14	...	0	8	14	6	6	0	10	2	7	0
1	0	6	0	4	0	7	7	4	4	7	...	0	7	13	6	8	0	8	1	7	0
2	4	7	5	5	3	7	8	2	7	7	...	1	7	7	6	8	0	8	2	8	0
3	1	6	1	4	2	7	7	0	7	7	...	1	8	7	6	8	0	8	3	8	0
4	1	2	1	3	1	7	7	1	7	7	...	1	8	7	6	9	0	8	3	8	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1595	1	6	0	4	0	7	7	4	4	7	...	10	4	7	12	8	2	10	0	8	1
1596	5	8	6	7	6	6	9	5	7	8	...	4	14	9	6	8	0	8	8	8	1
1597	4	9	4	4	2	7	10	2	5	13	...	7	5	10	8	7	3	8	3	8	1

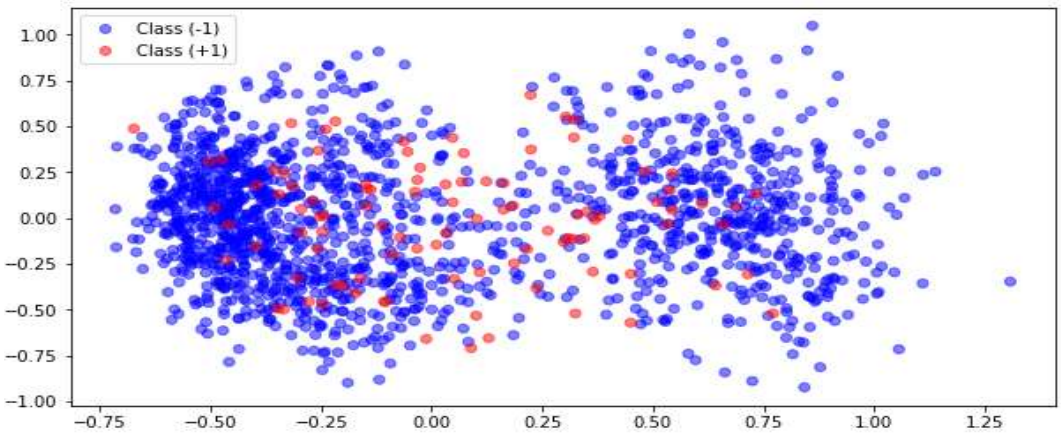

Figure APX1 2.0: Scatter Plot for the Letter Dataset

Table APX1 3.0: Description for Cardio Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	21(numeric)	1831	176 (0.0961)	2	None

Additional information

Expert obstetricians divided fetal heart rate and uterine contraction scans into 3 groups in the Cardio dataset: pathogenic, normal, and suspect. The pathologic class of 176 (9.61%) was converted into outliers for the purpose of outlier detection, the

normal class was transformed into inliers, and the suspect class was discarded. The dataset has 21 attributes, 2 classes, and 1831 instances.

Table APX1 3.1: Sample Cardio Dataset

	0	1	2	3	4	5	6	7	8	9
0	0.004912	0.693191	-0.203640	0.595322	0.353190	-0.061401	-0.278295	-1.650444	0.759072	-0.420487
1	0.110729	-0.079903	-0.203640	1.268942	0.396246	-0.061401	-0.278295	-1.710270	0.759072	-0.420487
2	0.216546	-0.272445	-0.203640	1.050988	0.148753	-0.061401	-0.278295	-1.710270	1.106509	-0.420487
3	0.004912	0.727346	-0.203640	1.212171	-0.683598	-0.061401	-0.278295	-1.710270	1.106509	-0.420487
4	-0.100905	0.363595	1.321366	1.027120	0.141359	-0.061401	-0.278295	-0.992364	-0.051613	-0.420487
...	...	...	...	...	...	...	...	...	...	...
1826	-0.418356	-0.919988	-0.161178	0.829564	0.953023	-0.061401	3.060819	1.221178	1.338133	-0.420487
1827	-0.418356	-0.919988	-0.171055	0.796630	0.823510	-0.061401	3.565299	1.281003	1.453945	-0.420487
1828	-0.418356	-0.919988	-0.164635	0.952396	1.120470	-0.061401	2.788995	1.221178	1.222321	-0.420487

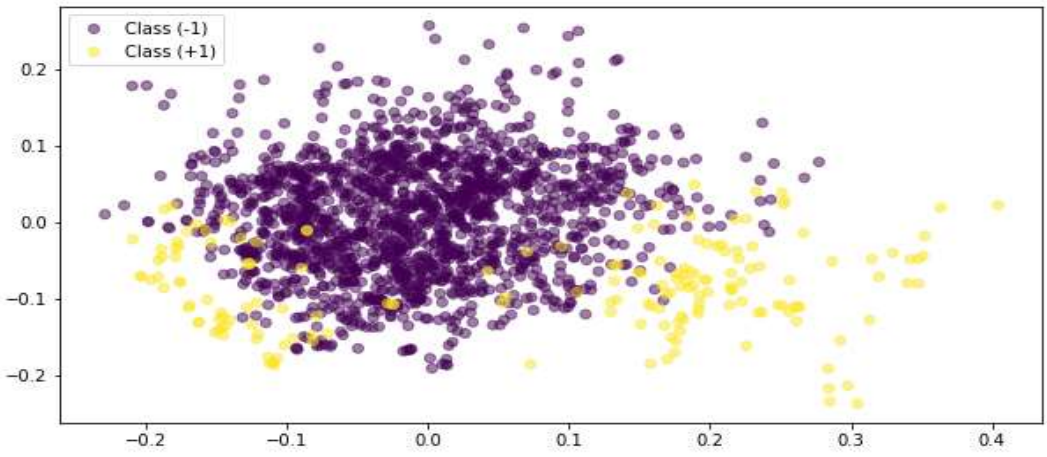

Figure APX1 3.0: Scatter Plot for the Cardio Dataset

Table APX1 4.0: Description for Annthyroid Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	6 (numeric)	7200	534 (0.0742)	2	None

Additional information

The original thyroid disease (ann-thyroid) dataset has 3772 training and 3428 testing instances. The issue is determining whether or not a patient who has been referred to the clinic is hypothyroid. Hence, three classes were created: normal, hyperfunction, and subnormal functioning. For outlier detection, both training and testing instances are used. The normal class is used as inliers, while the hyperfunction and subnormal classes are considered outliers. The dataset has 6 real attributes, 2 classes, and 7200 instances.

Table APX1 4.1: Sample Annthyroid Dataset

	0	1	2	3	4	5	y
0	0.73	0.00060	0.0150	0.120	0.082	0.1460	0
1	0.24	0.00025	0.0300	0.143	0.133	0.1080	0
2	0.47	0.00190	0.0240	0.102	0.131	0.0780	0
3	0.64	0.00090	0.0170	0.077	0.090	0.0850	0
4	0.23	0.00025	0.0260	0.139	0.090	0.1530	0
...	...	...	...	...	...	...	...
7195	0.59	0.00250	0.0208	0.079	0.099	0.0800	0
7196	0.51	0.10600	0.0060	0.005	0.089	0.0055	1
7197	0.51	0.00076	0.0201	0.090	0.067	0.1340	0

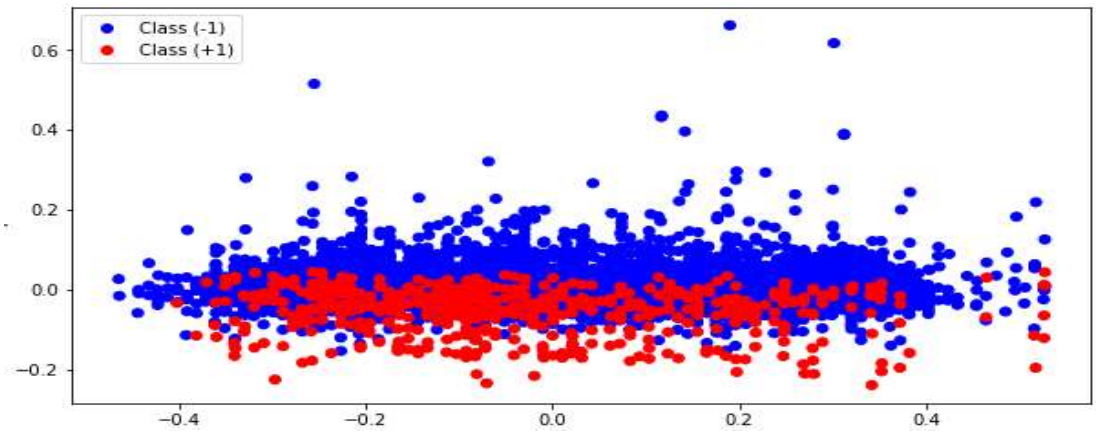

Figure APX1 4.0: Scatter Plot for the Annthyroid Dataset

Table APX1 5.0: Description for Pima Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	8(numeric)	768	268 (0.3490)	3	None

Additional information

The original Pima dataset for diabetes is a dataset for binary classification. The selection of examples from the wider database was subject to a number of restrictions including; all patients being female, at least 21 years of age, and of Pima heritage. It has 8 attributes, 3 classes, and 768 instances, of which 268 (34.9%) are considered outliers. The objective is to foresee if a patient has diabetes.

Table APX1 5.1: Sample Pima Dataset

	0	1	2	3	4	5	6	7	y
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	1
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0	0
764	2.0	122.0	70.0	27.0	0.0	36.8	0.340	27.0	0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.0	0
766	1.0	126.0	60.0	0.0	0.0	30.1	0.349	47.0	1
767	1.0	93.0	70.0	31.0	0.0	30.4	0.315	23.0	0

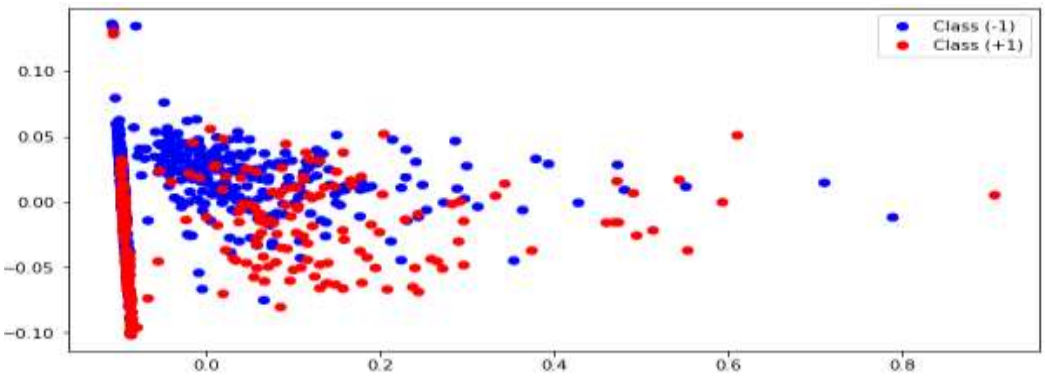

Figure APX1 5.0: Scatter Plot for the Pima Dataset

Table APX1 6.0: Description for Vowels Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	12(numeric)	1456	50 (0.0343)	2	None

Additional information

The original Vowels dataset is a multivariate time series data with 9 male speakers' utterances of two Japanese vowels, 'a' and 'e.' A single utterance gives a time series range of lengths 7-29, with every point consisting of 12 features. For the purpose of outlier detection, each frame in the training set is treated as a separate data point. 50 (3.43%) outliers (Class 1-Speaker) are included in the sample. Classes 6, 7, and 8 are regarded as inliers, while other classes are disregarded.

Table APX1 6.1: Sample Vowels Dataset

	0	1	2	3	4	5	6	7	8	9	10	11	y
0	0.580469	-0.902534	0.617899	-0.997942	-2.463799	-0.846455	2.349849	0.375400	-0.649334	1.604637	-0.623060	-0.383125	0.0
1	0.784375	-1.077366	0.615781	-0.921911	-2.388553	-0.638047	2.106684	0.361018	-0.714317	1.260236	-0.423339	-0.287791	0.0
2	0.791292	-1.086242	0.669773	-0.806112	-2.260781	-0.538491	2.053282	0.266492	-0.842815	1.081797	-0.267201	-0.172203	0.0
3	1.217306	-1.083425	0.855483	-0.724879	-2.155552	-0.101879	1.768597	0.303151	-1.044710	0.655290	0.214298	-0.341840	0.0
4	1.065352	-1.030178	0.773297	-0.452289	-1.955907	0.248205	1.530474	0.253740	-0.968961	-0.208287	0.331578	0.007288	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1451	0.286951	0.898280	0.898001	-0.489441	-0.127830	-0.674159	0.506261	2.655082	0.282258	-1.496882	-0.225170	1.195592	1.0
1452	1.229374	0.371318	0.507031	0.439482	0.449693	0.048680	-0.167676	0.374878	0.326678	-0.059276	-2.724481	0.365011	1.0
1453	0.947076	0.358108	0.274725	0.167539	-0.359012	-0.794370	0.628138	2.067813	-0.109592	-1.088328	0.327126	1.692834	1.0

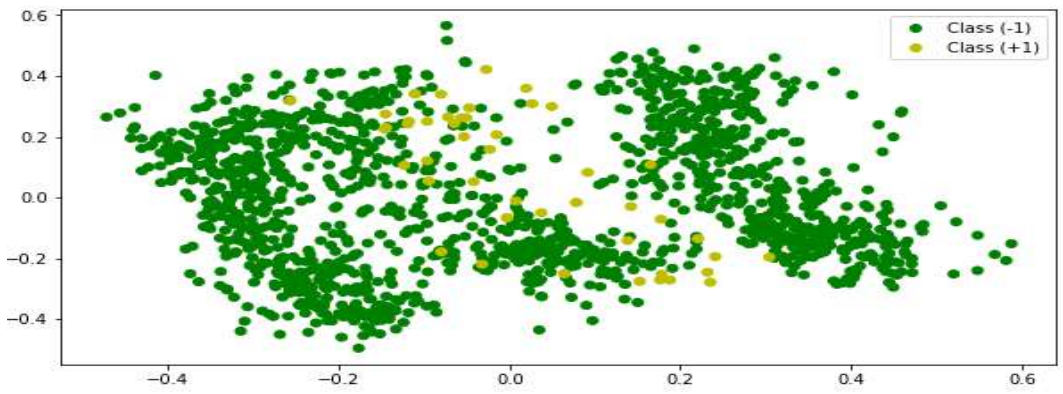

Figure APX1 6.0: Scatter Plot for the Vowels Dataset

Table APX1 7.0: Description for Thyroid Dataset

Type	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	6(numeric)	3772	93 (0.0247)	3	None

Additional information

The original thyroid (ann-thyroid) dataset has 3772 training and 3428 testing instances. The issue is determining whether or not a patient is hypothyroid. Hence, three classes are created: normal, hyperfunction, and subnormal functioning. For outlier detection, only the training instances are used. The normal and subnormal classes are used as inliers, while the hyperfunction class with 93 instances (2.466%) was considered outliers. The dataset has 6 real attributes, 3 classes, and 3772 instances.

Table APX1 7.1: Sample Thyroid Dataset

	0	1	2	3	4	5	y
0	0.774194	0.001132	0.137571	0.275701	0.295775	0.236066	0.0
1	0.247312	0.000472	0.279886	0.329439	0.535211	0.173770	0.0
2	0.494624	0.003585	0.222960	0.233645	0.525822	0.124590	0.0
3	0.677419	0.001698	0.156546	0.175234	0.333333	0.136066	0.0
4	0.236559	0.000472	0.241935	0.320093	0.333333	0.247541	0.0
...	...	...	...	...	...	...	...
3767	0.817204	0.000113	0.190702	0.287383	0.413146	0.188525	0.0
3768	0.430108	0.002453	0.232448	0.287383	0.446009	0.175410	0.0
3769	0.935484	0.024528	0.160342	0.282710	0.375587	0.200000	0.0

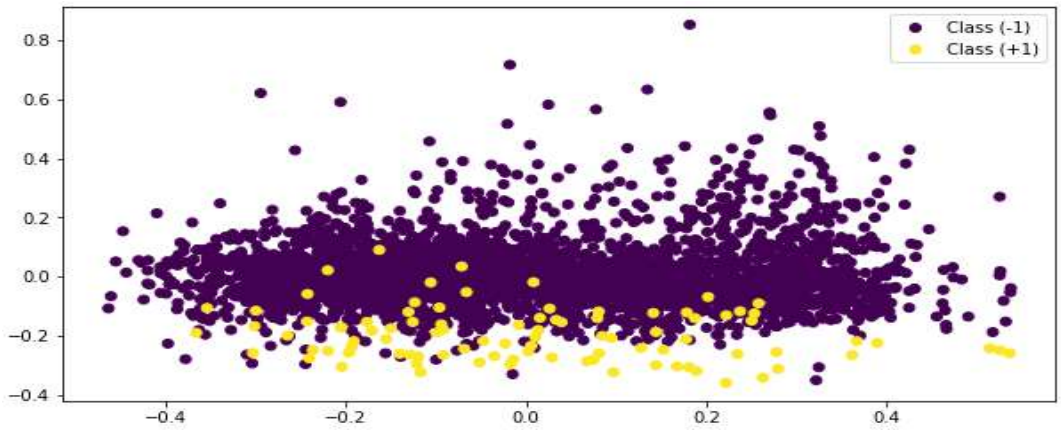

Figure APX1 7.0: Scatter Plot for the Thyroid Dataset

Table APX1 8.0: Description for Pendigits Dataset

Type (Int.)	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	16(Integers)	6870	156 (0.0227)	2	None

Additional information

The original Pen-Based Recognition of Handwritten Digits (pendigits) dataset has 16 features and classes 0 to 9. A collection of 250 samples from 44 writers, of which samples from 30 writers are taken as training and cross-validation data while the rest are for testing. For outlier detection, the original collection of handwritten samples is reduced to 6,870 points, of which 156 are outliers. The quantity of objects in one class, digit-0, is decreased by 10% because all classes have similar frequencies.

Table APX1 8.1: Sample Pendigits Dataset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	y
0	0.489770	1.000000	0.270463	0.812819	0.578246	0.371700	0.261600	0.000000	0.000000	0.234127	0.565463	0.533163	1.000000	0.901008	0.402439	9.827949e-01	0
1	0.000000	0.887156	0.275691	1.000000	0.421130	0.758004	0.296085	0.449351	0.150986	0.150846	0.376991	0.000000	0.689886	0.019072	1.000000	5.741760e-02	0
2	0.000000	0.568410	0.321768	0.682326	0.729609	0.903012	1.000000	1.000000	0.758613	0.752239	0.504585	0.505997	0.285181	0.255833	0.162748	0.000000e+00	0
3	0.000000	1.000000	0.070717	0.918846	0.047145	0.686851	0.189419	0.457426	0.863797	0.345193	1.000000	0.456065	0.753983	0.231130	0.660026	0.000000e+00	0
4	0.000000	0.671233	0.482427	0.626872	1.000000	1.000000	0.819488	0.802209	0.596662	0.603270	0.398628	0.403493	0.320968	0.200536	0.458826	0.000000e+00	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6865	0.000000	0.814390	0.098480	0.586115	0.567467	0.340996	0.407665	0.000000	0.101024	0.295027	0.035742	0.677206	0.423051	0.968106	1.000000	1.000000e+00	0
6866	0.484535	1.000000	0.000000	0.704414	0.243141	0.556696	1.000000	0.650909	0.861467	0.856085	0.439344	0.765738	0.207771	0.386698	0.053207	0.000000e+00	0
6867	1.000000	0.978963	0.611034	1.000000	0.245273	0.966617	0.040000	0.579466	0.347293	0.506535	0.586561	0.256205	0.370375	0.000000	0.000000	4.374904e-02	0
6868	0.588021	0.650253	0.908701	1.000000	0.842337	0.950497	0.722174	0.501346	0.504278	0.080702	0.000000	0.000000	0.453246	0.016379	1.000000	3.069548e-16	0
6869	0.000000	0.790930	0.294442	1.000000	0.947813	0.962214	0.700276	0.484855	0.431244	0.109819	0.325768	0.000000	0.254882	0.356541	1.000000	4.009105e-01	0

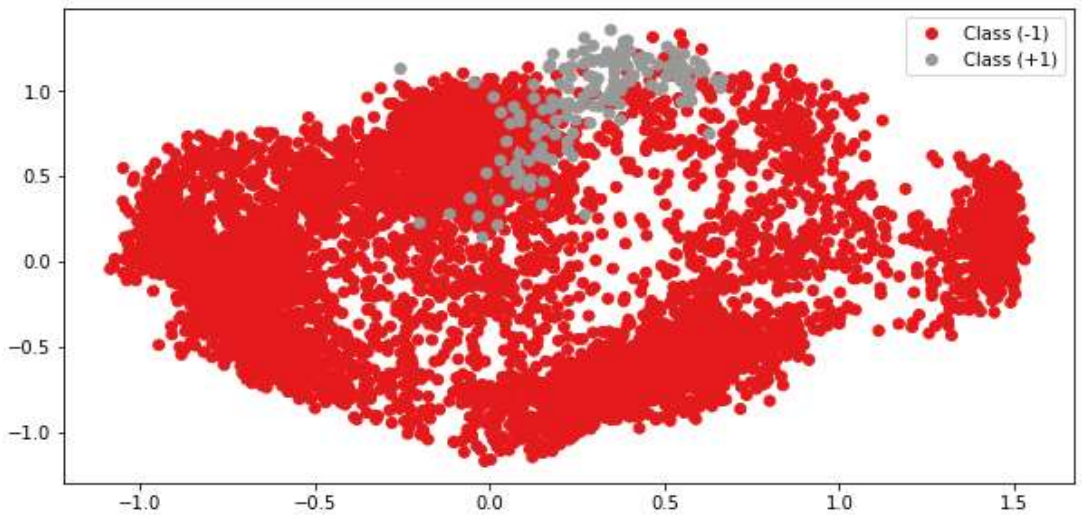

Figure APX1 8.0: Scatter Plot for Pendigits Dataset

Table APX1 9.0: Description for Breastw Dataset

Type (Int.)	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	9(Integers)	683	239 (0.3499)	2	None

Additional information

Breast Cancer Wisconsin (breastw) contains records of measurements for breast cancer cases. It has two classes: benign and malignant. It is a 9-dimensional dataset containing 683 instances, of which 239 represent malignant tumors here taken as outliers.

Table APX1 9.1: Sample Breastw Dataset

	0	1	2	3	4	5	6	7	8	y
0	5	1	1	1	2	1	3	1	1	0
1	5	4	4	5	7	10	3	2	1	0
2	3	1	1	1	2	2	3	1	1	0
3	6	8	8	1	3	4	3	7	1	0
4	4	1	1	3	2	1	3	1	1	0
...	...	...	...	...	...	...	...	...	...	...
678	3	1	1	1	3	2	1	1	1	0
679	2	1	1	1	2	1	1	1	1	0
680	5	10	10	3	7	3	8	10	2	1
681	4	8	6	4	3	4	10	6	1	1

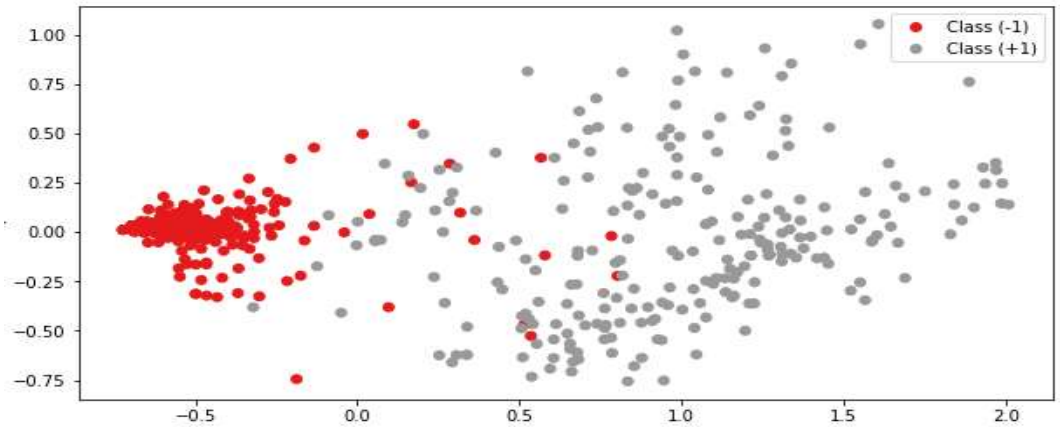

Figure APX1 9.0: Scatter Plot for the Breastw Dataset

Table APX1 10.0: Description for Optdigits Dataset

Type (Int.)	Features	Instances	Outliers (%)	Classes	Missing Values?
Outlier detection	64(numeric)	5216	150 (0.0286)	2	None

Additional information

The original Optical Recognition of Handwritten Digits (Optdigits) is a multi-class classification dataset that includes inliers made up of the instances of digits 1 through 9, and outliers made up of the instances of digit 0, which are down-sampled to 2.86%.

Table APX1 10.1: Sample Optdigits Dataset

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60	61	62	63	y
0	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	16.0	10.0	0.0	0.0	0.0
1	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	11.0	16.0	9.0	0.0	0.0
2	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	13.0	9.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	16.0	4.0	0.0	0.0	0.0
4	0.0	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	9.0	16.0	16.0	10.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5211	0.0	0.0	3.0	15.0	4.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	4.0	10.0	15.0	8.0	0.0	0.0	1.0
5212	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	16.0	14.0	6.0	0.0	0.0	1.0
5213	0.0	0.0	6.0	15.0	12.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	5.0	15.0	12.0	3.0	0.0	0.0	1.0
5214	0.0	0.0	5.0	15.0	3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	16.0	14.0	8.0	0.0	0.0	1.0
5215	0.0	0.0	11.0	15.0	8.0	0.0	0.0	0.0	0.0	5.0	...	0.0	0.0	0.0	11.0	16.0	12.0	0.0	0.0	0.0	1.0

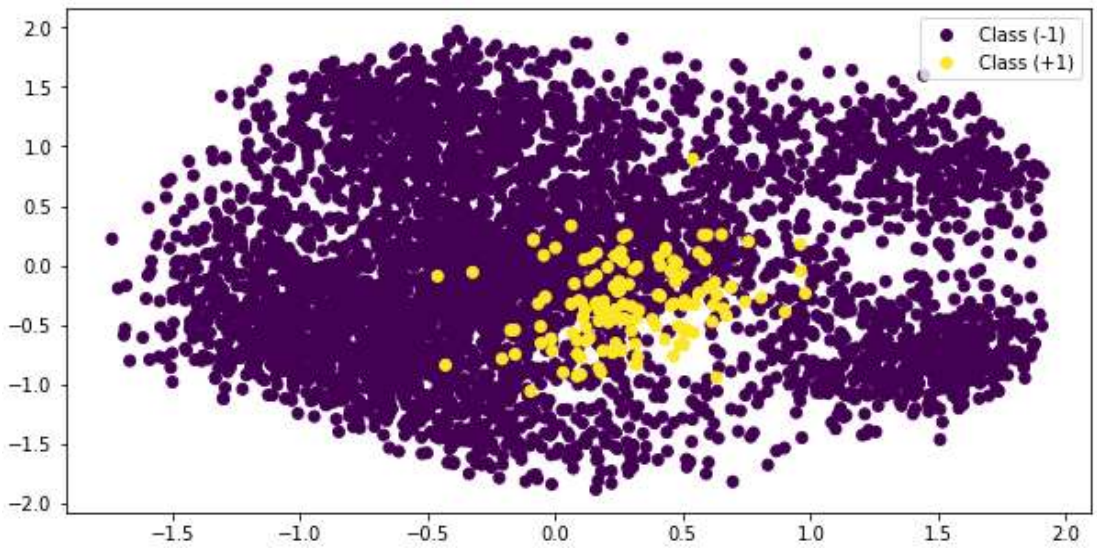

Figure APX1 10.0: Scatter Plot for Optdigits Dataset

Appendix II: Experiment screenshots

Step 1: IMPORT LIBRARIES (necessary)

In [1]:
```
from __future__ import division, print_function
import warnings
with warnings.catch_warnings():warnings.filterwarnings('ignore',category=DeprecationWarning)
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd

    #import sys
import progressbar
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib.cm as cmx
import matplotlib.colors as colors
```

Step 2: READ DATASETS(Raw)

DESCRIPTION:

MNIST
MNIST dataset contains the zeros (0's) as INLIERS from the original MNIST dataset, with sixes (6's) added as OUTLIERS.
100 features of the original 784 (number of pixels) were randomly sampled.
NEXT DATASET

In [187]:
```
    # Read data
mat_file = 'mnist.mat'#Load .mat version
mat = loadmat(os.path.join('data', mat_file))
    # seperate features and labels
dataset_raw_features=pd.DataFrame(data=mat['X']) # X features only
dataset_raw_target = pd.DataFrame(data=mat['y']) # y target/label only
dataset_raw_full = pd.concat((pd.DataFrame(data=mat['X']), pd.DataFrame(data=mat['y'])), axis=1)
    # send file to csv - file before pre-processing(whole view)
dataset_raw_full.to_csv("data/data_processing/data_adaho/01_dataset_full_unprocessed.csv")
dataset_raw_full
```

Out[187]:

	0	1	2	3	4	5	6	7	8	9	...	91	92	93	94
0	-0.0096	-73.804153	198.205963	0.0	-13.124617	-1.1501	0.0	-0.141633	179.249390	114.661163	...	-0.107867	-4.469967	158.381409	-137.100632
1	-0.0096	-73.804153	197.205963	0.0	-13.124617	-1.1501	0.0	-0.141633	179.249390	-44.338833	...	-0.107867	-4.469967	123.381416	-137.100632

Standardizing dataset

In [189]:
```
start_time = timer(None)
dataset_cleaned_features_scaled = standardizer(dataset_cleaned_features)#.fit_transform(dataset_cleaned_features)
dataset_cleaned_scaled_full = pd.concat((pd.DataFrame(data=dataset_cleaned_features_scaled), pd.DataFrame(data=dataset_raw_target
# show array by: Rb_X
dataset_cleaned_scaled_full.to_csv("data/data_processing/data_adaho/03_dataset_scaled_standardscaler.csv", index=None)
timer(start_time)
dataset_cleaned_scaled_full
```

Time taken: 0 minutes and 3.91 seconds.

Out[189]:

	0	1	2	3	4	5	6	7	8	9	...	69	70	71	72
0	-0.919459	0.948998	-0.208549	-0.06349	-0.026254	0.863589	0.981005	-0.413877	0.924327	0.612606	...	-0.556656	1.040383	-0.311593	0.940542
1	-0.919459	0.939662	-0.208549	-0.06349	-0.026254	0.863589	-0.696434	-0.413877	0.905941	0.879825	...	-0.556656	1.022176	-0.311593	0.614736
2	-0.919459	-1.353140	-0.208549	-0.06349	-0.026254	-1.543515	-0.696434	3.108485	0.924327	-0.494452	...	-0.556656	1.040383	-0.311593	0.931234

Step 3: DATA PRE-PROCESSING

Cleaning:

Converting categorical features if any to numerical; Remove null columns; Remove columns where all data points have the same values; Remove identical columns where all data points have the same value;

In [171]:
```
start_time = timer(None)
print('\n# Reading and Processing Data')

print('\n# Initial DataSet Matrix Dimensions: %d x %d' % (dataset_raw_features.shape[0], dataset_raw_features.shape[1]))
features_len = len(dataset_raw_features)

    #check missing data
print('\n# Checking missing values')
if sum(dataset_raw_full.isnull().sum()) == 0:
    print('-- No missing values')
else: dataset_raw_full.isnull().sum()

    # Sort out numerical and categorical features
print('\n# Checking for categorical data')
numeric_feats = dataset_raw_features.dtypes[dataset_raw_features.dtypes != 'object'].index
categorical_feats = dataset_raw_features.dtypes[dataset_raw_features.dtypes == 'object'].index
if categorical_feats.size > 0:
    for i, col_name in enumerate(categorical_feats):
        print(' Converting %s' % col_name)
        temp_df = pd.get_dummies(dataset_raw_features[col_name])
        new_features = temp_df.columns.tolist()
        new_features = [col_name + '_' + w for w in new_features]
```

```
# Reading and Processing Data

# Initial DataSet Matrix Dimensions: 7603 x 100

# Checking missing values
-- No missing values

# Checking for categorical data
-- No categorical data; only numeric

# Checking for features/columns with same values
 Column 3 is identical to 6. Removing 6
 Column 3 is identical to 21. Removing 21
 Column 3 is identical to 37. Removing 37
 Column 3 is identical to 61. Removing 61
 Column 3 is identical to 72. Removing 72
 Column 3 is identical to 78. Removing 78
 Column 3 is identical to 86. Removing 86
```

```
 Column 88 is identical to 99. Removing 99

 -- Number of columns after cleaning: 91

# Checking for data points with same values

 -- Number of columns before cleaning: 91
 Column 0 removed. All data points have same value
 Column 3 removed. All data points have same value
 Column 26 removed. All data points have same value
 Column 28 removed. All data points have same value
 Column 40 removed. All data points have same value
 Column 50 removed. All data points have same value
 Column 52 removed. All data points have same value
 Column 53 removed. All data points have same value
 Column 60 removed. All data points have same value
 Column 70 removed. All data points have same value
 Column 87 removed. All data points have same value
 Column 89 removed. All data points have same value
 Column 91 removed. All data points have same value

 -- Number of columns after cleaning: 78

# Final Matrix Dimensions: 7603 x 78
 Time taken: 0 minutes and 2.14 seconds.
```

CONVERT target y class '0' to -1(i.e INLIERS) and class '1' to +1 (i.e OUTLIERS)

In [192]:
```
data=np.array(dataset_cleaned_scaled_full)
target = data[:, -1].astype(int, copy=False)

X = data
y = target

digit1 = 0
digit2 = 1
idx = np.append(np.where(y == digit1)[0], np.where(y == digit2)[0])
y = target[idx]

  # Change labels to {-1, 1} (.e CONVERT target-class y data '0' to -1(i.e INLIERS) and class '1' to +1 (i.e OUTLIERS)
y[y == digit1] = -1
y[y == digit2] = 1
X = data[idx]
```

Step 4: LOAD CLEANED DATASET as {(X, y)}

In [193]:
```
#describe cleaned scaled dataset
X_min_value = X.min().min()
X_max_value = X.max().max()

print("New Dataset shape: ",X.shape)
print("Labels shape: ",y.shape)
print ("Min: ",X_min_value)
print("Max: ",X_max_value)
data
```

```
New Dataset shape:  (7603, 79)
Labels shape:  (7603,)
Min:  -1.8498380086356423
Max:  87.18944890294928
```

BEGIN ALGORITHM: Algorithm 1

In [185]:
```
class Ada_Base:
    def __init__(self, n_clf=1): #classifier #AdaBoost
        #self.classifier = classifier
        self.n_clf = n_clf
        self.classifiers = []
        self.alpha = []
        self.weights = []
        self.predictions = []

    def set_weights(self, incorrect_rows, N):
        alpha_clf = self.alpha[-1]
        for i in range(N):
            if i in incorrect_rows:
                self.weights[i] *= math.e**alpha_clf
            else:
                self.weights[i] *= (math.e**((-1) * alpha_clf))
        sum_weights = sum(self.weights)
        for i in range(len(self.weights)):
            self.weights[i] /= sum_weights # normalize sample weights to add up to 1

    def fit(self, df_train):
            # Initialize the weights
        N = int(len(df_train)/2)  # sample size;
```

```
Boosting algorithm:  Ag_KNN
clf_name_list ['Lg_KNN', 'Ag_KNN']
Iteration: 0, Missed: [662, 611]
Accuracy:0.8995, Alfa:0.681, Error: 0.2039
Confusion matrix: [[5282  240]
 [ 371  189]]
Boosting algorithm:  Md_KNN
clf_name_list ['Lg_KNN', 'Ag_KNN', 'Md_KNN']
Iteration: 0, Missed: [662, 611, 606]
Accuracy:0.9004, Alfa:0.3466, Error: 0.3333
Confusion matrix: [[5301  221]
 [ 385  175]]
Boosting algorithm:  LOF1
clf_name_list ['Lg_KNN', 'Ag_KNN', 'Md_KNN', 'LOF1']
Iteration: 0, Missed: [662, 611, 606, 651]
Accuracy:0.893, Alfa:0.2683, Error: 0.369
Confusion matrix: [[5254  268]
 [ 383  177]]
```

```
    clfs_preds = []
    clfs_miss = []
    clfs_lowest_err =[]
    sum_errors_all = []
    clfs_weights = []
    clf_names_list=[]
    accuracies=[]
    error_rates=[]
        # weaklearners #KNeighborsClassifier(n_neighbors = 78),#LocalOutlierFactor(n_neighbors = 78)#
    weak_classifiers = {"Lg_KNN": KNN(n_neighbors = 87, method='largest', metric = 'euclidean', contamination=0.092),
                        "Ag_KNN": KNN(n_neighbors = 87, method='mean', metric = 'euclidean', contamination=0.092),
                        "Md_KNN": KNN(n_neighbors = 87, method='median', metric = 'euclidean', contamination=0.092),
                        "LOF1": LOF(n_neighbors=87, metric = 'euclidean', contamination=0.092),
                        "LOF2": LOF(n_neighbors=10, metric = 'euclidean', contamination=0.092)
                       }
        # adaptive boosting
    for idx, (clf_name, clf) in enumerate(weak_classifiers.items()):
    #for idx, seque_weaklearner in enumerate (seque_weaklearners):
        print("Boosting algorithm: ", clf_name) #seque_weaklearner)
        # Iterate through SELECTED classifier N-times
        for _ in range(self.n_clf):
                # sample a training set using weights
            ts = df_train.sample(n=N, weights=self.weights)
                #ts.reset_index(drop=True, inplace=True)

            X_train = ts.drop('y', axis=1) #df_train.drop('y', axis=1)#
            y_train = ts['y'] #df_train['y']#
```