Anomaly Detection Model Selection Using Super-AUC Scores

Gautam Dilip Hariharan
Matriculation Number: 230237

Academic Advisors:
Prof. Dr. Emmanuel Müller
Simon Klüttermann, M. Sc.

Master Thesis

Technical University of Dortmund
Department of Computer Science
Chair of Data Science and Data Engineering
https://ls9-www.cs.tu-dortmund.de/

Semester: Winter 2024/25

Contents

List of Figures

List of Tables

1 Introduction

With the ever increasing collection and usage of data, information security has become of vital importance. From business to healthcare, to law enforcement and sports, data is critical to their operations and data science plays a very vital role here [29]. This in turn leads us to one of the most interesting applications of data science—anomaly detection. Identifying and classifying anomalies takes precedence in fields where data collection and usage is high and vital to the operations of the company/industry, not only to ensure security, but also to ensure efficiency.

There are multiple anomaly detection algorithms available that help in identifying these anomalies and aiding in fixing or improving systems and architectures. But first, before understanding anomaly detection, it is important to understand what an anomaly is. An anomaly is a data point or suspicious event that deviates from the baseline pattern. Inconsistent or redundant data points define anomalous data in general[9].

Anomaly detection on its own is not enough though. It has to be effective and efficient. For example, an anomaly detection model may work great for a particular dataset but may not work very well for another application[36]. How do we decide how good an algorithm works for a particular dataset or application? This especially becomes a problem because metrics like the AUC score are very challenging to calculate for unlabeled data[19]. This leads us to alternative ways to evaluate performance of a model on non-labeled data.

Issues like these give way to investigating and evaluating new metrics that would help us achieve this. This study discusses, demonstrates and presents a metric called the Super AUC score. The purpose of this metric is to evaluate the performance of anomaly detection models for unlabeled data, which is something the AUC score cannot do.

What is the Super AUC score? The Super AUC score is an innovative metric that combines decision scores from both training and test sets of unlabeled datasets. Conventional evaluation metrics rely primarily on test data performance while the Super AUC score leverages information from both data partitions, giving us a better look into algorithm effectiveness.

This study makes use of multiple popular anomaly detection algorithms like K-Nearest Neighbors (KNN), Isolation forest (iForest), Gaussian Mixture Models (GMM) and a few others to test the efficiency and robustness of the Super AUC score. Metrics like the Spearman Correlation are used to investigate relationships between different evaluation metrics, thereby opening the door to more room for drawing inferences.

To sum up, the main goal of this thesis is to provide an accurate and reliable means for anomaly detection algorithm selection, specifically aimed at data sets without labels where traditional evaluation methods prove ineffective.

1.1 Research Question and Contributions

The main research question addressed in this thesis is whether the Super AUC score can serve as a reliable metric for selecting the most effective anomaly detection algorithm for unlabeled datasets. This investigation is geared towards understanding how the combination of decision scores from training and test sets, which is how the Super AUC score is evaluated, can overcome the limitations of traditional evaluation metrics in an anomaly detection setting.

To answer this question, a systematic method for evaluating anomaly detection algorithms using the Super AUC score is presented. The approach investigates algorithms with hyperparameters and without, and datasets that have different complexities holding different kinds of data.

Through a series of experiments conducted on all these datasets, the proposed methodology is analyzed and inferences are drawn from the results of these experiments. The results offer an insight into the practicality of the Super AUC score, highlighting it's advantages in identifying high performing models without relying on labeled data.

1.1.1 Key Contributions

The key contributions of this study include very primarily the introduction of the Super AUC score, which as already briefly introduced, is a score that leverages the combined test and training decision scores to evaluate the performance of the algorithm. It aims to benchmark multiple popular simple and advanced anomaly detection algorithms. Additionally, the relationships between various performance metrics and dataset characteristics are analyzed, which can eventually also open doors to other areas of research that would interest anomaly detection practitioners in general.

The goal is to effectively address the gap that exists in evaluating the performance of an anomaly detection algorithm for unlabeled data, thereby opening doors to other research areas in the field of unsupervised anomaly detection and data science in general.

1.2 Structure of this Thesis

To start with, Chapter 2 provides a little background into why the introduction of this metric was important. It also provides a little insight into other related research in this domain and how it can be connected to this study.

Chapter 3 discusses the basics and provides an actual introduction and base for the rest of the study. Anomaly Detection is introduced and discussed, along with popular performance metrics used in the industry to evaluate these algorithms with emphasis on whatever is being used in this study.

Chapter 4 explains the methodology of the study. It discusses the various frameworks being used and how all of this is applied in this study. It introduces the ADbench datasets that

are used throughout the course of this study and also discusses in detail the algorithms used in this study. It also discusses how the evaluation metrics are being used.

Chapter 5 discusses how this study is implemented. Derived from the methodology in Chapter 4, this chapter aims to paint a canvas and write a story about how everything is implemented. It talks about the objectives, scope of the experiments, the setup, the metrics and the workflow of the scripts and code. It also very briefly discusses the challenges faced and how these difficulties were handled.

Chapter 6 presents the results and analysis. It presents facts about the data obtained from running the models on the datasets and details the metrics that are used to analyze said data. This chapter also provides analysis into these metrics which eventually helps us draw conclusions. The results are presented in the form of visualizations and tables, which provide all of the information required to understand the outcome of the process.

Chapter 7 eventually concludes the project and provides insights obtained from this project. Future work is also discussed along with the strengths and limitations of the Super AUC score where it is compared to other metrics in general to see where it can be improved or if there is something about it that needs to be nerfed. Practical and ethical considerations for further research are also discussed here.

The final section is the Appendix which is split into two parts: Appendix A and Appendix B. Appendix A contains tables and Appendix B contains images.

2 Background and Literature Review

The concept of anomaly detection was already very briefly defined in Chapter 1. Anomaly detection is used widely in multiple industries across the globe, making it imperative to research, study and improve anomaly detection methods. For example, at the industrial level, anomaly detection from images captured using camera sensors is one of the mainstream applications. Specifically, it aids in maintenance and optimizing efficiency in production processes across various industrial tasks, including advanced manufacturing and aerospace engineering[4]. This here was just one example of the necessity of anomaly detection at the industrial level.

Apart from that, anomaly detection is widely used in finance, healthcare, manufacturing, telecommunications and most importantly in cybersecurity[52]. Traditional methods such as statistical techniques and machine learning algorithms have been widely used to identify anomalies.

In scenarios where labeled data is difficult to obtain or not available, evaluating the performance of anomaly detection algorithms poses a challenge which gives rise to the need for alternative and new evaluation methods because conventional metrics like precision and accuracy may not necessarily serve these needs.[19]

That is where the Super AUC score comes into play. Information from the test as well as training sets are leveraged to determine if an algorithm is suitable enough for a particular purpose.

2.1 Related Works

2.1.1 Self Supervised Ensembles for Anomaly Detection

Self-supervised learning (SSL) has gained a lot of traction in anomaly detection in recent years especially when combined with ensemble methods. SSL helps models learn useful patterns from unlabeled data by solving simple tasks, allowing them to understand the natural structure of the data. When used in ensemble models, SSL improves the reliability and accuracy of anomaly detection systems.

One notable approach is the Self-Supervised Learning Framework with Dual Ensemble Voting Fusion for Maximizing Anomaly Prediction in Time-series (S2DEVFMAP). This method uses five heterogeneous auto-encoder models, each learning unique representations of the data.[38] The dual ensemble fusion strategy merges these models through voting techniques, capturing different system behaviors and reducing false alarms. Tests on real industrial cooling system data showed its effectiveness, indicating its potential for other critical anomaly detection applications.

Another key contribution is the Self-supervised, Refine, Repeat (SRR) framework, which improves unsupervised anomaly detection by continuously refining data representations. SRR uses multiple one-class classifiers (OCCs), each trained on different data subsets. By lever-

aging the ensemble's agreement, it removes likely anomalies, progressively improving training data.[61] This enhances the model's ability to differentiate normal from anomalous data, achieving strong performance across various domains, including semantic, manufacturing, and medical applications.

The SSL frameworks discussed, place importance on learning from unlabeled data leading to an improvement in anomaly detection performance. This study similarly assesses anomaly detection models that operate in unsupervised settings, which means that self-supervised approaches could be a future consideration in optimizing algorithm selection based on AUC scores.

Unlike traditional SSL-based anomaly detection methods that learn data representations from unlabeled data, the Super AUC score focuses on evaluating and selecting the best anomaly detection models. By aggregating performance across multiple datasets, it offers a comprehensive, model-agnostic assessment rather than relying on a learning-based optimization approach.

2.1.2 Clustering: Evaluation Metrics

Clustering evaluation metrics assess the quality of clusters generated by algorithms. Internal metrics, like the Silhouette Coefficient and Davies–Bouldin Index, measure cluster compactness and separation without needing external labels. The Silhouette Coefficient quantifies how well an object fits within its cluster compared to others, ranging from -1 to 1, where higher values indicate better clustering[2]. The Davies–Bouldin Index calculates the average similarity ratio of each cluster with the cluster most similar to it, where a lower score signifies better-defined clusters[60].

External metrics, such as the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI), assess clustering quality by comparing results to ground truth labels[62].

Just as how these clustering evaluation metrics assesses model performance in unsupervised settings by evaluating structure quality (for example, Silhouette, Davies-Bouldin), the Super AUC score quantifies anomaly detection effectiveness through algorithm selection.

2.1.3 Quality Measure for Uncertainty Quantification

Uncertainty quantification (UQ) is crucial for evaluating the reliability of model predictions. Key UQ metrics include Expected Calibration Error (ECE), which measures how well predicted probabilities match actual outcomes, and Negative Log-Likelihood (NLL), which assesses the confidence of probabilistic predictions. These metrics help distinguish between epistemic (knowledge-based) and aleatoric (random) uncertainties.[21].

UQ metrics evaluate confidence in predictions while the Super AUC score measures anomaly detection effectiveness, potentially benefiting from UQ techniques to refine algorithm selection.

2.1.4 Outlier Exposure

Outlier Exposure (OE) is an anomaly detection technique that improves model performance by training on auxiliary datasets containing outliers. Introduced by Hendrycks et al. in their 2018 paper "Deep Anomaly Detection with Outlier Exposure,"[24] this method improves the model's ability to detect unseen anomalies, leading to improved detection across various tasks.

Expanding on this idea, Qiu et al. introduced "Latent Outlier Exposure"[46] in 2022 to improve anomaly detection in datasets containing unlabeled anomalies. Their method simultaneously infers binary labels and updates model parameters, leading to consistent improvements across various datasets.

While OE aims to improve anomaly detection performance by enhancing model robustness through auxiliary outliers, the Super AUC also focuses on improving anomaly detection performance by selecting the most effective anomaly detection algorithms.

3 Anomaly Detection and Performance Metrics

This chapter talks about the basics of anomaly detection: what it is, how it is performed and what purpose it serves. It also introduces and discusses the various performance metrics used in evaluating the performance of anomaly detection algorithms in general. The concept of the Super-AUC score is introduced and discussed as well.

3.1 What is Anomaly Detection?

Let's start at the beginning. To understand how anomaly detection works, what algorithms are anomaly detection algorithms and so on, it is necessary to understand what anomaly detection and anomalies actually are.

Anomaly detection is the process of identifying data points or events that fall out of a range considered normal. An anomaly is a data point that displays this behavior[33]. For example, if a trader wanted to buy a truckload of apples, he would only expect apples to be in his purchase. If he investigated the contents of the truck and found an orange or a peach among the apples as well, he would be upset and would remove what he would in that case consider anomalies. To confirm, anything apart from the apples in the shipment would be considered an anomaly or an outlier which is why in some cases this concept is also referred to as outlier detection or novelty detection.

Anomaly detection has always been an integral part of statistics which has been driven by analysts and scientists who spent a lot of time carefully studying charts to find elements that have stood out. With time, researchers have started automating anomaly detection using machine learning techniques in an effort to find more efficient ways to detect different types of anomalies.

In industry, anomaly detection is commonly used to detect suspicious events, unexpected opportunities or bad data buried in time series data. A suspicious event could constitute fraud, crime, disease, a network vulnerability or some faulty equipment. An unexpected opportunity could be finding a stock that's performing much better than others and should be investigated as a potential trading option.

For data scientists, it is important to identify these anomalies in order to be able to remove them from further analysis so as to not threaten the development of new algorithms.

3.2 How does Anomaly Detection work?

Anomalies can be detected by machine learning algorithms which can be trained in several ways. Supervised machine learning techniques are used when a labeled dataset exists that indicates normal vs. abnormal conditions. For example, a company that does information security auditing could develop a process for labeling important security vulnerabilities after those vulnerabilities have been detected/reported. People in the medical industry might label

images or data sets that could diagnose a potential disease[1, 33, 54]. This is where supervised methods are used to detect anomalies.

It is also possible that researchers start with outliers that were already discovered but suspect that other anomalies may also exist. In the scenario of the information security company, they may overlook some small vulnerability if properly disguised. Somebody who carefully goes over the report later can include these types of vulnerabilities to automatically label other similar vulnerabilities using semi-supervised machine learning techniques.

Supervised and semi-supervised techniques typically only detect known anomalies. In reality though, a large chunk of data is actually unlabeled. This is where unsupervised anomaly detection techniques come into play where anomalies can be automatically detected.

A popular example of unsupervised learning is customer segmentation in marketing where businesses aim to identify distinct groups within their customer base. By studying and understanding these segments, companies can design their marketing strategies to suit the needs of each group[39]. Unsupervised algorithms like k-means clustering and hierarchical clustering are usually used in this case.

Retail companies might use clustering to classify customers based on purchasing behavior or other attributes. By analyzing these attributes, companies can recognize high-value consumers and target specific classes with personalized deals and in turn optimize their marketing strategies to increase retention of customers[39].

3.3 Types of Anomalies and how to identify them

Anomalies are classified into different types and there are also different ways of identifying different types of anomalies. This section outlines a few types and ways to identify them. Figure 1 illustrates the different types of anomalies.

Point Anomalies (Outliers)

Point anomalies or outliers are data points that are significantly different from the rest of the dataset. There are two kinds of outliers:

- **Univariate Outliers:** The anomalies are observed within a single dataset. For example, in a customer database, an address of 'Planet Jupiter' would be considered a univariate outlier. Studying these individual data points and comparing them to the expected range of values within that dataset is how these anomalies can be identified.
- **Multivariate Outliers:** Considering multiple variables together is how these outliers are identified. For example, going to a store and buying a commonly bought item couple with an extremely rarely bought item would be a multivariate outlier. More complex analysis taking into account relationships and correlations between variables is required to detect these multivariate outliers.[5]

There are many statistical methods and visualization techniques used to identify point anomalies.

- **Z-Scores:** This is a useful method for identifying univariate outliers and helps in understanding how far the data point extends beyond the norm. Outliers that lie outside a specific range can be identified bny calculating standard deviations.
- **Inter-quartile Range (IQR):** This metric identifies the typical range of values and helps identify anomalies that lie beyond this range.
- **Boxplots:** These are visualizations that represent the distribution of data, where the data points outside the box whiskers are the outliers.
- **Scatterplots:** These plots helps in identifying data points that deviate from expected relationships and trends by plotting different variables against each other.

Contextual Anomalies

These anomalies refer to data that is inconsistent within a particular context.

- **Logical Inconsistencies:** For example, if an item is sent to a customer's house before it was even ordered. This violates a logical flow and generally indicates issues with data integrity or system processes.
- **Referential Integrity issues:** These are generally anomalies that arise when there are inconsistencies between related data entities and can cause errors in data processing or reporting.
- **Missing or incomplete data:** Data analysis and decision making can be hindered by incomplete or missing data.

Contextual anomalies are generally identified through the following ways:

- **Data Validation rules:** By enforcing validation rules based on requirements and quality standards, the introduction of anomalous data can be prevented.
- **Defining expected relationships between data points:** When different variables are defined by the relationships that exist between them, it becomes easier to identify anomalous data when they break said relationships.

Collective Anomalies:

These kinds of anomalies are groups of related data points that display unusual behavior when considered together.

Examples are:

- Sudden variations in time series data such as website traffic may indicate significant

events, system failures or changes in user behavior that may have to be investigated further.

- When unrelated metrics fluctuate together, like an increase in server load coinciding with a surge in user login attempts. Collective anomalies can reveal hidden relationships between unrelated variables, providing valuable information about performance or threats.

Collective anomalies can be identified by:

- **Real time anomaly detection algorithms:** These algorithms learn from historical data patterns.
- **Time series analysis:** Analyzing trends in time series data can help identify unexpected changes[5]. Moving averages, exponential smoothing, and anomaly detection models can be applied to time series data to discover collective anomalies.

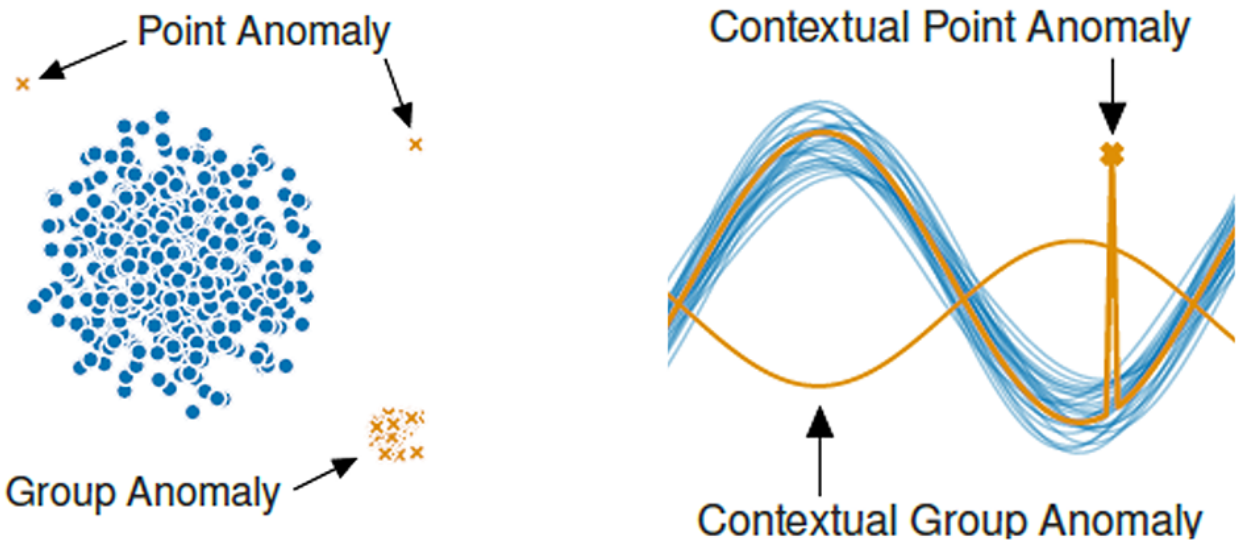

Figure 1: Types of Anomalies[15]

3.4 Challenges of Anomaly Detection

Since anomaly detection is unsupervised, it poses bigger challenges than other machine learning tasks. There are a few questions that arise in outlier detection such as, are the found outliers really outliers? Or was the model able to find all outliers in the data?

By matching predictions on the test data with the actual labels, we can check if the model is performing well in supervised learning. The same cannot be said of outlier detection because of the lack of ready-made labels classifying samples as inliers or outliers.

This highlights that one of the biggest challenges in anomaly detection is not knowing the contamination level.

Another challenge is the imbalance of data. Anomalies are generally rarer in occurrence to normal instances, which gives rise to an imbalance in data sets[58]. This imbalance can make

it hard to differentiate between actual anomalies and irregularities within the majority class.

Choosing suitable algorithms, hyperparameter tuning, feature selection and other methods can be used to address these issues.

3.5 Algorithms for Anomaly Detection

There are many algorithms that are tailored for anomaly detection. Here we introduce just a few popular ones.

Gaussian Mixture Models

These are probabilistic models that assume the data points in a dataset are generated from a mixture of Gaussian distributions. Anomaly detection using GMMs are done by fitting a GMM to the dataset and estimating the likelihood of each data point belonging to the learned model. The points whose likelihoods are significantly low are considered to be anomalous. These are useful when anomalies deviate from the usual distribution.

Isolation Forest

This algorithm is designed on the concept of isolating anomalies. It isolates anomalies by constructing a random forest of decision trees and recursively partitioning them until each instance is in its own leaf node. The idea is based on the fact that anomalies are more easily isolated than normal instances as they require lesser partitioning steps. The algorithm assigns an anomaly score to every data point where lower scores indicate a higher probability of that point being an anomaly. An isolation forest is illustrated in Figure 2.

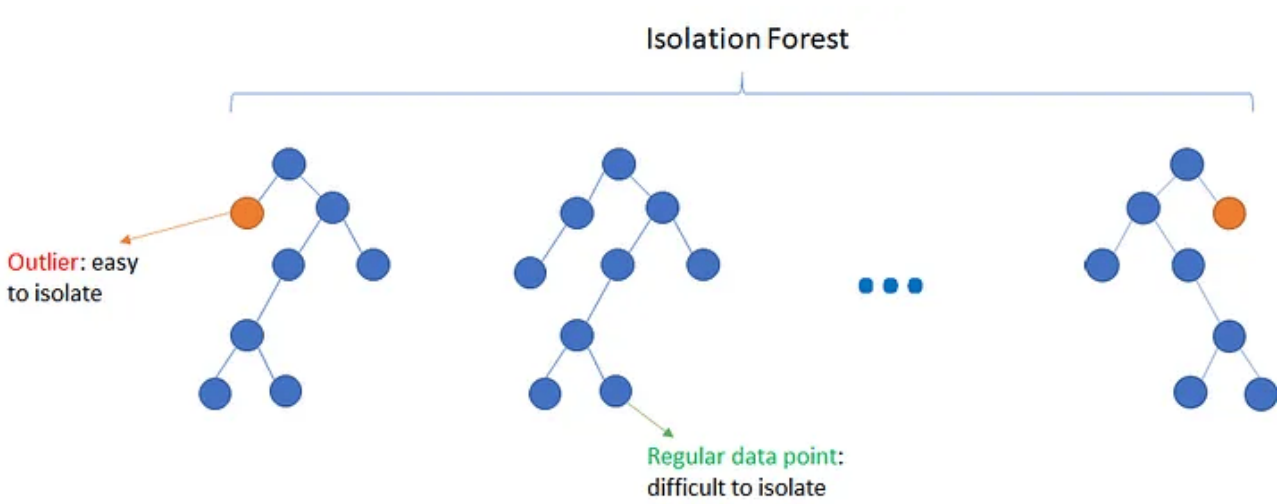

Figure 2: Isolation Forest[28]

One-Class Support Vector Machines

These are binary classifiers designed to identify anomalies in data. OCSVMs are trained only on normal instances unlike traditional SVMs. This works by assuming the anomalies are rare and

do not conform to the usual data distribution[6]. The data is mapped into a high-dimensional feature space and the algorithm then finds a hyperplane that separates the normal instances from the origin. The anomalies are considered to be the points that lie on the side of the hyperplane opposite to the origin.

3.6 Metrics for Evaluating Anomaly Detection Algorithms

To ensure effectiveness of anomaly detection algorithms, their performance must be evaluated. To achieve this, there are certain metrics that are designed to evaluate the performance of these algorithms, some of which will be discussed in this section.

3.6.1 Confusion Matrix-based Metrics

A confusion matrix is an important tool for evaluating classification performance, which can also be adapted to anomaly detection. The matrix divides the predictions into four categories:

- **True Positives (TP):** Correctly identified anomalies.
- **False Positives (FP):** Normal instances incorrectly labeled as anomalies.
- **True Negatives (TN):** Correctly identified normal instances.
- **False Negatives (FN):** Anomalies incorrectly labeled as normal.

From the confusion matrix, various metrics can be derived, including **precision**, **recall**, and **F1-score**, which are crucial for assessing anomaly detection performance.

- **Precision:** Precision answers the question: *What proportion of identified anomalies are true anomalies?*. It is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity):** Recall answers the question: *What proportion of true anomalies was identified?*. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** The F1-score combines both recall and precision using the harmonic mean and identifies the performance of the anomaly detection model[8]. It is defined as:

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

The structure of the confusion matrix is visually represented in Figure 3.

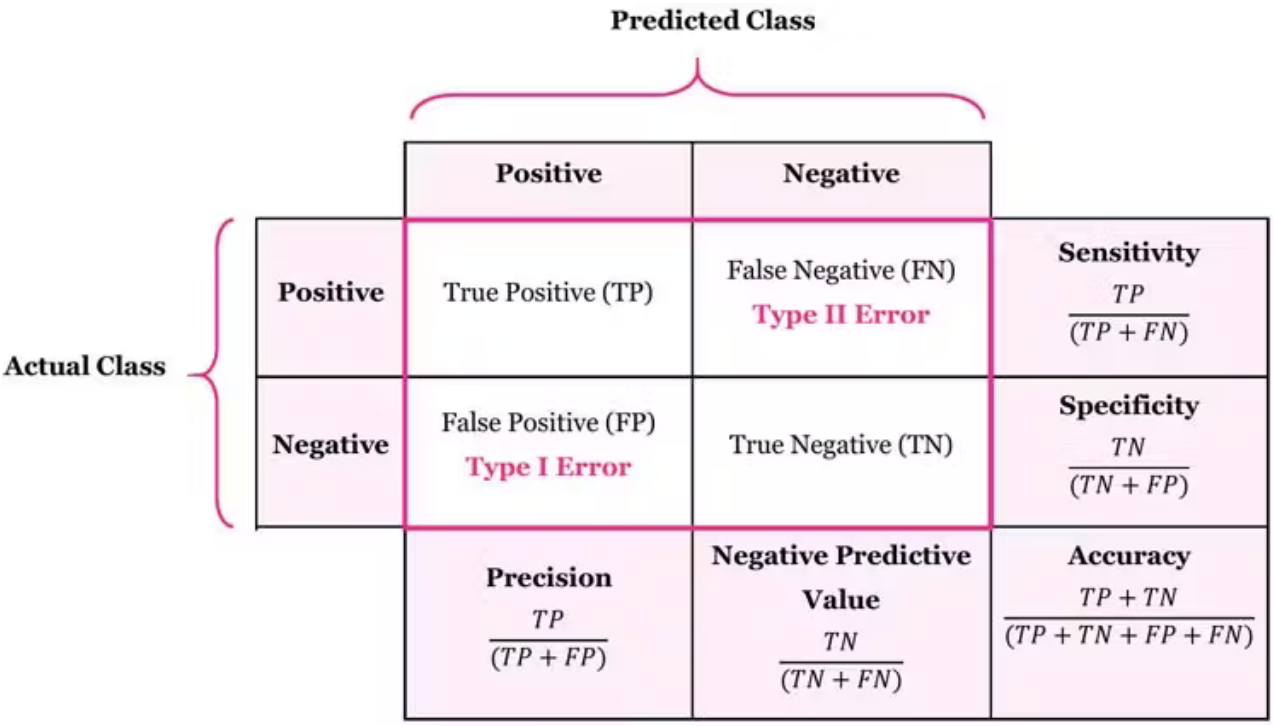

Figure 3: Confusion Matrix[10]

3.7 ROC Curve and AUC

If given a randomly chosen positive and negative example, the area under the ROC curve is the probability that the model will rank the positive higher than the negative.

It plots the recall (True Positive Rate (TPR)), against the False Positive Rate (FPR):

$$\text{FPR} = \frac{FP}{FP + TN}$$

An AUC value of 1 indicates perfect classification, while a value of 0.5 suggests that the model performs no better than random guessing. ROC curves are widely used for evaluating anomaly detection models due to their ability to illustrate performance across different threshold settings[50]. Figure 12 in Appendix B illustrates this.

4 Methodology

This chapter aims to introduce and explain the implementation. It also introduces and explains the Super AUC score in detail.

4.1 Super AUC Score: An Overview

One of the problems with unlabeled datasets are that evaluation of the performance of the dataset is very challenging with traditional metrics. For instance, the AUC score, while good on labeled data, cannot be applied to unlabeled data because the calculation of the AUC score depends on the outputs of the classification model and the actual class labels[42].

This is where the Super AUC score comes in. The traditional AUC score is calculated on the test set only but the Super AUC score leverages information from the training and the test set which makes it possible to evaluate the performance on a dataset without labels.

This is important because real world anomaly detection problems often deal with unlabeled or semi-labeled datasets, where separating normal from anomalous behavior poses unique challenges.

4.1.1 Super AUC: The Process

The Super AUC score is calculated by combining the decision scores of the training and test sets. The decision score is the degree of confidence associated with a given input sample.

Initially, the anomaly detection model is trained on the dataset. During training, the algorithms generates decision scores for every instance in the training set. The trained model is then applied to the test set and the decision scores are then calculated for the test instances. The decision scores are then combined into a single array. A corresponding label array is also created, assigning 0's to training scores and 1's to test scores. These combined decision scores and labels are then used to compute the AUC-ROC score. The value returned by this computation gives rise to the Super AUC score.

4.1.2 Super AUC: The Advantages and Applications

The Super AUC score is computed by calculating the AUC score after combining the test and training decision scores. This steers away from dependence on test labels, which is a requirement for computation of the AUC score, resulting in a useful metric for unlabeled and semi-labeled datasets. It also avoids overemphasizing training or test performance separately, thereby presenting a more holistic view of the capabilities of a model.

In domains like cybersecurity and fraud detection, anomalies are generally rare and ground truth labels are often unavailable or difficult to obtain. In situations like these, the Super AUC score shows strength as it can evaluate the ability of an algorithm to detect anomalous data across historical data and new data.

4.2 The Datasets

The datasets used in this study were taken from the ADBench benchmark which comprises of 57 benchmark datasets[22]. For the purpose of this study though, only 44 of these datasets were used. The list of datasets used in this study are listed in Table 9 in Appendix A. The datasets that are chosen out of the 57 are 47 classical datasets but only 44 were used because training the Cover, Census and Cardio datasets were computationally very expensive. These datasets were selected for their ability to cover a wide range of anomaly detection scenarios. It also provides a robust foundation for comparing different anomaly detection algorithms because of its diversity, quality and relevance to real-world applications.

4.2.1 Dataset Overview

The datasets of ADBench are quite diverse: they consist of datasets of varying sizes, anomaly ratios and feature dimensionalities, offering a wide scope for research for multiple different purposes in the field of Anomaly Detection. Some prominent datasets are introduced below, just to get an idea of what they're like:

- **Fraud:** A dataset that focuses on fraud detection, commonly used in anomaly detection and classification tasks.[20]
- **ALOI:** The Amsterdam Library of Object Images is a collection of color images of 1000 small objects[17].
- **Vowels:** Involves Vowel recognition tasks[12].
- **Pima:** Based on the Pima Indians diabetes dataset, typically used for medical diagnosis tasks taken from the National Institute of Diabetes and Digestive and Kidney Diseases[34].
- **Optidigits:** It is a collection of handwritten digits available from the UCI machine learning repository[3].
- **Skin:** This dataset is collected by randomly sampling B, G, R values from images of faces of people of different ages, races and genders[7].
- **Glass:** This is a glass identification dataset, often used for classification problems[16].
- **Thyroid:** A medical dataset related to thyroid disorder detection[47].
- **MNIST:** A well known dataset for handwritten digit recognition[59].
- **SpamBase:** A dataset that classifies E-Mail as spam or not spam[25].

A summary with some basic information about these datasets such as number of features and number of instances can be found in Table 1.

Dataset	Number of Instances	Number of Features
fraud	283823	29
ALOI	46518	27
vowels	1356	12
Pima	232	8
optdigits	4916	64
skin	143339	3
glass	196	7
thyroid	3586	6
mnist	6203	100
SpamBase	849	57

Table 1: Dataset statistics

4.3 The Algorithms

To evaluate and compare the performance of anomaly detection methods, a diverse set of algorithms was selected. This section provides an overview of each algorithm and the key hyperparameters used in the experiments.

4.3.1 k-Nearest Neighbors (KNN)

KNN-based anomaly detection identifies anomalies by analyzing the distances between data points. Anomalies are expected to lie far from their nearest neighbors. Variants of KNN used in this study include:

- **KNN (Default)**: Computes distances using the mean of k-nearest neighbors.
- **KNN (Median)**: Computes distances using the median distance to k-nearest neighbors.

Hyperparameters:

- `n_neighbors`: Number of nearest neighbors to consider (e.g., 4, 6, 7).
- `method`: Method for computing distances (mean or median)[32].

An illustration of KNN can be found in Figure 4.

4.3.2 Isolation Forest (IForest)

Isolation Forest isolates anomalies based on the principle that anomalies are more susceptible to isolation than normal points. It builds a tree structure, splitting data randomly.

Hyperparameters:

- `max_samples`: Number of samples used to build each tree (e.g., 2, 3, 4).
- `max_features`: Number of features considered for splits (e.g., 2, 3).[26]

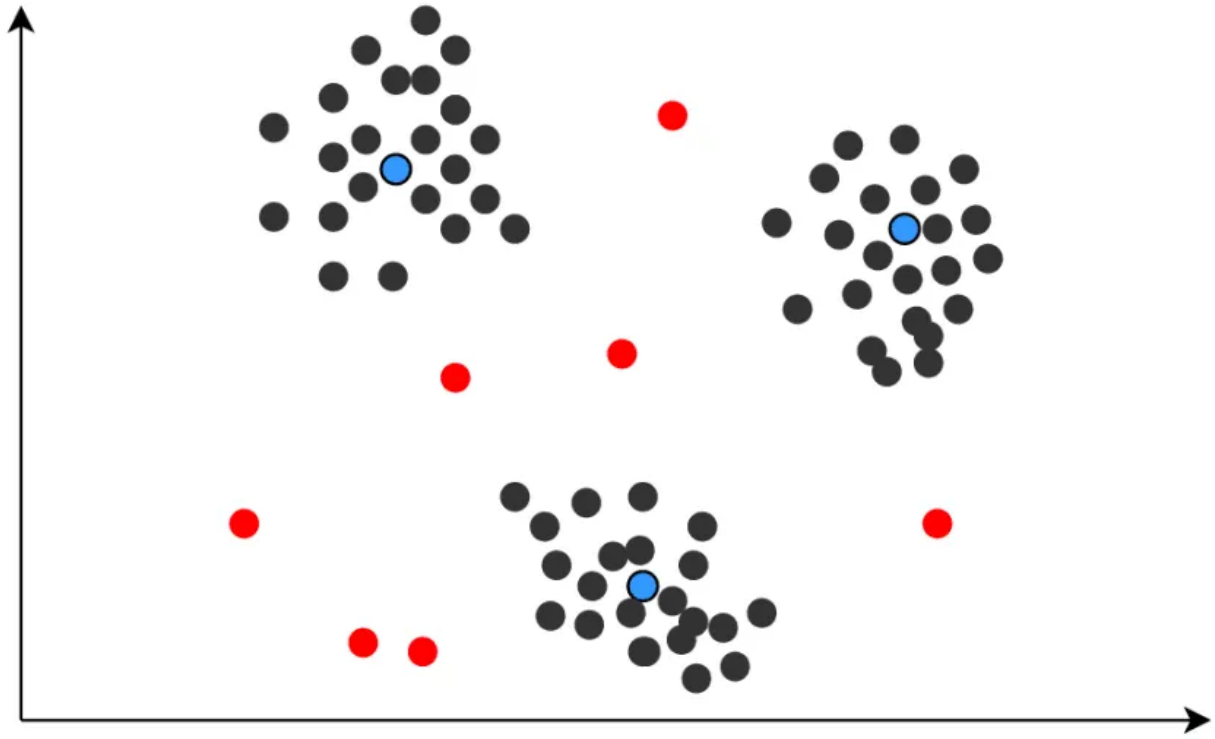

Figure 4: K-Nearest Neighbors[57]

4.3.3 COPOD (Copula-Based Outlier Detection)

COPOD is based on empirical copula models and is parameter free and a highly interpretable outlier detection algorithm.[11]

4.3.4 Gaussian Mixture Model (GMM)

GMM assumes data is generated from a mixture of multiple Gaussian distributions. The algorithm fits the data to these distributions and calculates the likelihood of each point belonging to a Gaussian. Low likelihood values indicate anomalies.

Hyperparameters:

- `n_components`: Number of Gaussian components (e.g., 2, 3, 4).
- `covariance_type`: Type of covariance matrix (e.g., tied, diagonal, spherical).
- `init_params`: Initialization method (random or k-means)[18].

An illustration of a complex distribution represented by a Gaussian Mixture Model can be found in Figure 5.

4.3.5 Histogram-Based Outlier Detection (HBOS)

HBOS uses histograms to model the data distribution feature-wise. Anomalies are identified by comparing the density of data points within bins.

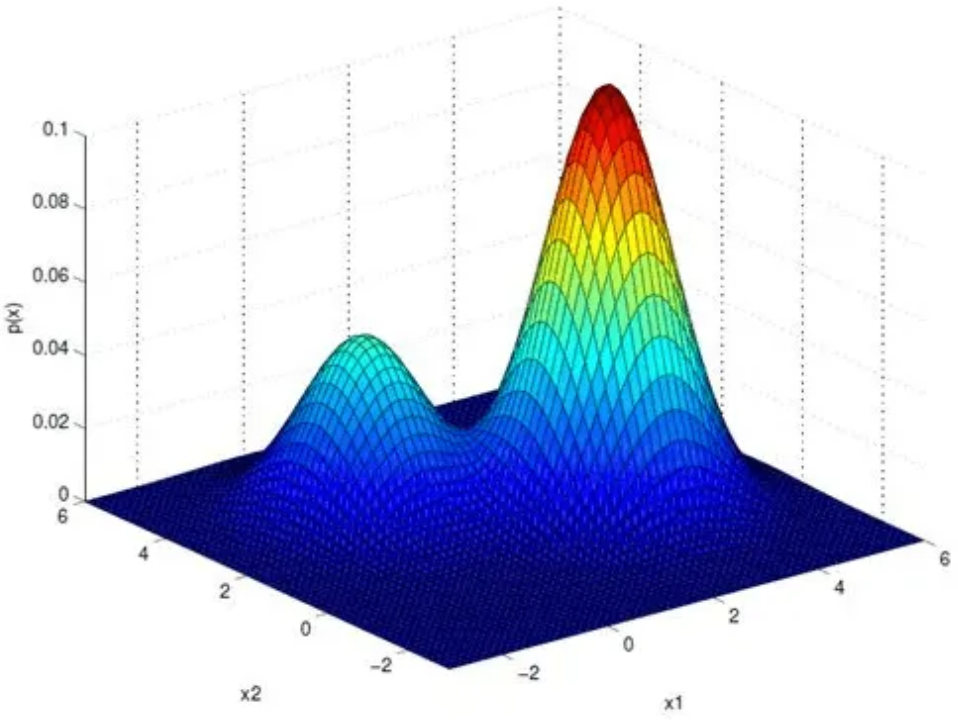

Figure 5: Gaussian Mixture Model[30]

Hyperparameters:

- `n_bins`: Number of bins in the histogram (e.g., 5, 15, 20)[23].

4.3.6 Empirical-Cumulative Outlier Detection (ECOD)

ECOD is a cumulative density-based approach. It computes the empirical cumulative distribution function (ECDF) for each feature and detects anomalies based on empirical CDF functions[13].

4.3.7 SEAN (Shallow Ensemble ANomaly detection)

SEAN is a novel anomaly detection algorithm in the field of predictive maintenance for real-time applications. It reduces computational costs and performs competitively by leveraging an ensemble based approach.[31]

4.3.8 Summary of Hyperparameters

Each algorithm was tested with different configurations of hyperparameters to ensure robust comparisons. A few examples of the hyperparameters used are:

- **KNN**: `n_neighbors = {4, 6, 7}`, `method = {mean, median}`
- **IForest**: `max_samples = {2, 3, 4}`, `max_features = {2, 3}`
- **COPOD**: Default parameters
- **GMM**: `n_components = {2, 3, 4}`, `covariance_type = {tied, diag, spherical}`
- **HBOS**: `n_bins = {5, 15, 20}`
- **ECOD**: Default parameters
- **SEAN**: Default parameters

The above information is better represented in Table 2.

Table 2: Summary of Algorithms and Hyperparameters

Algorithm	**Key Hyperparameters**
KNN	$n\_neighbors = \{4, 6, 7\}$, $method = \{mean, median\}$
IForest	$max\_samples = \{2, 3, 4\}$, $max\_features = \{2, 3\}$
COPOD	Default parameters
GMM	$n\_components = \{2, 3, 4\}$, $covariance\_type = \{tied, diag, spherical\}$
HBOS	$n\_bins = \{5, 15, 20\}$
ECOD	Default parameters
SEAN	Default parameters

4.4 Evaluation Methodology

The evaluation of anomaly detection algorithms has been a vital part of this study, considering it is meant to evaluate the performance of a metric called the Super AUC score. The metrics used in this study are introduced and explained in this section.

4.4.1 Performance Metrics

A few important metrics were used to derive conclusions and make analyses about the Super AUC score to be able to draw inferences about the metric. They are listed and explained below.

- **AUC:** This metric is explained in Section 3.7. This metric was calculated specifically for the test dataset. The test AUC was the comparison metric that was used to study how well the Super AUC score actually worked by studying the correlations between the two metrics.
- **Super AUC:** This is the AUC score calculated by combining the decision scores of the training and test sets.
- **Spearman Correlation:** This metric is an alternative to the Pearson's correlation. It is generally used for ordinal data and data that follows monotonic, curvilinear relationships[14].

4.5 Implementation Details

It is important to understand the implementation details in order to reproduce the experiments and ensure reliability of the results. The metrics, datasets and algorithms have already been described in the earlier parts of this chapter. The rest of the chapter will discuss how the study was implemented. Libraries, hardware and other implementation details will be discussed in the remainder of this chapter.

4.5.1 The Libraries

The code was written in Python on the Visual Studio Code IDE. Multiple libraries were used to implement the code. These are listed below with a description of what they provide. Every single library used to enable the functioning of this study is listed and explained.

Standard Libraries

- json: This library is used to handle JSON data. Using this library, JSON strings can be parsed into Python objects or Python objects can be serialized into JSON[27].
- `numpy`: Numpy is one of the most important libraries in Python for numerical computing. It provides powerful array and matrix structures and a wide range of mathematical operations[40].
- `pandas`: An important library for handling and analyzing structured data. It is built on top of Numpy[44].
- `os`: Commonly used for file and directory manipulation, path management and environment variable handling[43].
- `sys`: Provides access to system specific parameters and functions[55].
- `random`: Provides functionality to generate random numbers and perform random sampling[48].

Third Party Libraries

- `scipy`: A library that is built on Numpy and offers functionality for mathematics, statistics, optimization and more[56].
- `scikit-learn (via sklearn.metrics)`: A popular machine learning library that offers functions like `roc_auc_score` which is used to calculate the AUC-ROC score[53].
- `pyod`: This library provides a lot of the anomaly detection models that were used in the course of this study. The KNN, iForest and COPOD models are some such examples. These algorithms are explained in Section 4.3[45].
- `sean`: Provides access to the SEAN algorithm explained in Section 4.3.7[31].

- `matplotlib`: This library is very popular for creating visualizations. A wide range of plot types can be produced using this library. It is very useful for exploratory data analysis and presenting results[37].
- `scipy.stats.rankdata`: This is a part of the scipy library and is a function that assigns ranks to data elements. It is used where ranks are of more interest than absolute values in non-parametric statitics[49].

Where and how were these libraries used?

The easiest way to understand what each of these libraries contribute to this study would be to start at the top of the code and work our way down.

The `numpy` library loads the .npz datasets for input features and labels. `numpy` provides functions that combines the predictions. Owing to the large sizes of the datasets, to reduce time spent in training every model, the process was parallelized and the order of the datasets and algorithms were shuffled in order using the `random` library. The library `os` is used to manipulate and access directories and files to retrieve, store and edit data. The datasets are retrieved, worked on and the results are stored in a JSON file.

The `pyod` library gives provides access to all of the anomaly detection algorithms that were used in this study, except for the **SEAN** algorithm which was used through the `sean` import. `scipy.stats` provides access to the function which calculates the Spearman correlation and `sklearn.metrics.roc_auc_score` is used to calculate the the AUC-ROC score of the combined data as well as the test AUC.

Additionally, `json` was used to load the JSON data that was stored as a result of the computations and save the processed results into another CSV file which was used for further analysis. This analysis is done mostly in the form of plots which derives its data from a structured DataFrame created using the `pandas` library. The plots are created using the `matplotlib` library which gives us a visual representation of our analysis in the form of critical difference plots, bar charts and other plots that aids the analysis which in turn leads to drawing inferences from the data. Finally, the `sys` library was used to interact with the user to choose the best algorithm for comparison that would then be used in the data for analysis.

4.5.2 Computational Resources

The experiments were performed using a high-performance computing environment to handle the computational demands of training and evaluating algorithms across 44 datasets. Key specifications included:

- **Processor:** AMD Ryzen 9 5900HX, with multiple cores to support parallel processing.
- **Memory:** 16 GB of RAM to efficiently load and process large datasets.
- **GPU Acceleration:** NVIDIA RTX 3060 for deep learning models, facilitating faster

training and inference.

4.5.3 Code Organization

To ensure maintainability and scalability, the codebase was organized into modular components:

- **Data Loader:** Functions to preprocess datasets, handle missing values, and split them into training and testing sets.
- **Algorithm Wrapper:** A unified interface for initializing, training, and evaluating anomaly detection models, irrespective of their library or implementation details.
- **Evaluation Module:** Code to compute performance metrics, including AUC-ROC and Super AUC, and perform statistical tests for result analysis.
- **Visualization Tools:** Scripts to generate plots, including ROC curves and heatmaps, for visual representation of results.
- **Result Aggregator:** Utilities to collate and summarize experimental outcomes for reporting purposes.

4.5.4 Challenges and Solutions

Several challenges arose during implementation:

- **Handling Imbalanced Data:** Many datasets exhibited significant class imbalance, with anomalies constituting a small fraction of the data.
- **Computational Overhead:** Training multiple algorithms across 44 datasets required significant computational resources. Parallel processing and efficient data pipelines were implemented to reduce runtime.
- **Algorithm Compatibility:** Certain algorithms like **SEAN** required specific data formats or preprocessing steps. A standardized data pipeline was developed to ensure compatibility across all models.
- **Result Reproducibility:** To ensure consistent results, random seeds were set for all random number generators in Python and third-party libraries.

5 Implementation

5.1 The Objectives

Before diving into how the study was implemented, it is important to revisit the objectives of this study.

The experiments were designed to address the following research questions:

- Which algorithms consistently score high on the Super AUC scale?
- How does adding hyperparameters to the algorithms affect the performance of these algorithms?
- How does the Super AUC score compare to traditional metrics like the Test AUC in representing algorithm performance? Is there a positive correlation between the two metrics?
- Is the Super AUC score dependent on the types of datasets?
- Is the Super AUC score a reliable metric to gauge the performance of an algorithm on an unlabeled dataset?

These are some of the main questions the study aims to answer. The answers to these questions can be found in Chapter 6 in the form of data, plots and tables.

5.2 Scope of the Experiments

The scope of the experiments covers:

- **Algorithm Selection:** A diverse set of traditional and modern anomaly detection algorithms as well as their hyperparameter variants were evaluated.
- **Dataset Coverage:** The ADBench benchmark includes a wide range of datasets which cover the different types of anomalies that were explained in Section 3.3.
- **Evaluation Metrics:** The Super AUC score was the main metric that was focused on for this study. The test AUC was used for comparative analysis.

5.3 Experimental Setup

The experimental setup involved the evaluation of various anomaly detection algorithms across multiple datasets using the Super AUC score. This section describes the key components of the workflow.

5.3.1 Datasets

- **Source:** The datasets were taken from ADBench benchmark. A whole list of datasets used can be found in Table 9.
- **Structure:** Every dataset had the same structure.
  - `x`: Training data
  - `tx`: Testing data
  - `ty`: Ground truth labels for the testing data
- **Randomization:** The datasets and algorithms were not executed in any particular order. The execution was processed in random to avoid biases.

5.3.2 Algorithms

A diverse set of algorithms from the PyOD [45] library were used. **SEAN**[31] was the only algorithm among these that was not taken from PyOD. A whole list of algorithms used along with the hyperparameters used can be found in Table 3.

5.4 Evaluation Metrics

Two main metrics are used for evaluation.

- **Super AUC:** The AUC score computed over the concatenated training and testing predictions against their combined labels.
- **Test AUC:** The AUC score computed solely on the test set predictions against the ground truth labels.

5.5 Workflow

- **Loading and Preprocessing**
  - Each dataset is loaded, and training (`x`), testing (`tx`), and testing labels (`ty`) are extracted.
  - The datasets are shuffled for randomness.
- **Model Training and Scoring**
  - Each algorithm is initialized and trained using `x`.
  - Predictions are generated for training and test data. For all models, the decision function is used to calculate decision scores.
  - Super AUC and test AUC are calculated using the `roc_auc_score` metric

Abbreviation	Algorithm with Hyperparameters
KNN2-median	`KNN(method='median')`
KNN4	`KNN(n_neighbors=4)`
KNN6	`KNN(n_neighbors=6)`
KNN7	`KNN(n_neighbors=7)`
KNN4-mean	`KNN(n_neighbors=4, method='mean')`
KNN4-median	`KNN(n_neighbors=4, method='median')`
KNN6-mean	`KNN(n_neighbors=6, method='mean')`
KNN6-median	`KNN(n_neighbors=6, method='median')`
iForest12	`IForest(max_samples=2)`
iForest13	`IForest(max_samples=3)`
iForest14	`IForest(max_samples=4)`
iForest2	`IForest(max_features=2)`
iForest3	`IForest(max_features=3)`
iForest22	`IForest(max_features=2, max_samples=2)`
iForest23	`IForest(max_features=2, max_samples=3)`
iForest24	`IForest(max_features=2, max_samples=4)`
iForest32	`IForest(max_features=3, max_samples=2)`
iForest33	`IForest(max_features=3, max_samples=3)`
iForest34	`IForest(max_features=3, max_samples=4)`
GMM2	`GMM(n_components=2)`
GMM3	`GMM(n_components=3)`
GMM4	`GMM(n_components=4)`
GMM-tied	`GMM(covariance_type='tied')`
GMM-diag	`GMM(covariance_type='diag')`
GMM-spherical	`GMM(covariance_type='spherical')`
GMM-random	`GMM(init_params='random')`
GMM-tied-random	`GMM(covariance_type='tied', init_params='random')`
GMM-diag-random	`GMM(covariance_type='diag', init_params='random')`
GMM-spherical-random	`GMM(covariance_type='spherical', init_params='random')`
GMM2-random	`GMM(n_components=2, init_params='random')`
GMM3-random	`GMM(n_components=3, init_params='random')`
GMM4-random	`GMM(n_components=4, init_params='random')`
HBOS5	`HBOS(n_bins=5)`
HBOS15	`HBOS(n_bins=15)`
HBOS20	`HBOS(n_bins=20)`

Table 3: Algorithms and Their Hyperparameter Configurations

- **Metric Computation**
  - **Combined Scores:** Training and testing scores are combined and the Super AUC is calculated against the combined labels.
  - **Correlations:** Spearman correlations are calculated between the Super AUCs and test AUCs across all algorithms for each dataset.
  - **Percent Above Threshold:** The percentage of algorithms where the Spearman correlations between the Super AUC and Test AUC were greater than 0.5 is tabulated

and plotted.

- **Results Storage:**
  - Results per dataset is stored and saved in JSON files. These included Super AUC, Test AUC scores and Spearman correlation.

5.6 Execution Challenges and Computational Limitations

Despite the robust design of the experimental pipeline, several execution challenges and computational limitations were encountered during the course of this study.

- The inclusion of numerous hyperparameter variants, especially for models like KNN, Isolation Forest, and GMM, increased the complexity of implementation. Each variant required to be checked to ensure compatibility with the dataset and scoring procedures.
- The evaluation involved a wide range of models with multiple hyperparameter configurations, such as KNN with varying neighbor counts and scoring methods, or GMM with different covariance types and initialization parameters. This significantly increased the total computation time.
- Processing large datasets with high dimensionality increased the computational requirements. Some algorithms, such as KNN and GMM, were particularly sensitive to dataset size, resulting in longer execution times due to their dependence on pairwise distance calculations or iterative parameter estimation.
- Variations in the number of instances and features across datasets led to uneven computational loads. Certain datasets required significantly more time for fitting and scoring, especially with resource-intensive models.
- While the algorithms were shuffled and processed sequentially to avoid bias, this approach limited opportunities for parallel execution. Parallelizing across datasets or algorithm variants could improve efficiency but was constrained by hardware resources and model-specific dependencies.
- Spearman correlation calculations were sensitive to datasets where the combined or test AUCs were uniformly distributed, leading to low statistical significance in the correlation values.
- The reliance on third-party libraries such as PyOD introduced potential reproducibility challenges due to differences in library versions or platform-specific dependencies.
- The experiments were conducted on a single machine, which constrained the scale of simultaneous executions. Computational resources such as CPU cores, memory, and disk I/O often approached their limits during peak loads.
- Evaluating the entire range of algorithms across all datasets was very time consuming,

with some configurations taking significantly longer than others. This necessitated frequent intermediate saving of results to prevent data loss in case of interruptions, which in the beginning stages, occurred quite frequently.

6 Results and Analysis

6.1 Introduction

This chapter showcases the results obtained from the experiments performed in the previous chapters. This will be through various plots and tables obtained throughout the course of this thesis.

6.2 The Results

6.2.1 For Models without Hyperparameters

To start with, models without any hyperparameters were trained on the datasets. The results these returned were the Spearman's correlations between the Super and the test AUCs.

The AUC scores enabled the ranking of algorithms by their effectiveness for each dataset. The correlations helped in assessing the consistency between the Super-AUC and the test AUC for each dataset, hence helping us determine if the Super AUC is a suitable metric to evaluate the performance of an unlabeled dataset.

Seven algorithms (KNN, iForest, COPOD, GMM, ECOD, HBOS and sean) were run on 44 different datasets without any hyperparameters attached to them to see how simple algorithms perform on these datasets.

Overall Algorithm Performance

Table 4 shows the average test auc scores and Super AUC scores across all 44 datasets for all 7 algorithms mentioned above.

- **Top-Performing Algorithms (Based on Average `test_auc` Across Datasets):**
  - **KNN:** Consistently achieves high performance, with high AUC scores in datasets such as `fraud`, `optdigits`, and `Lymphography`.
    - `fraud`: 0.972
    - `optdigits`: 0.999
    - `Lymphography`: 1.000

    Average `test_auc`: ~ 0.846.
  - **IForest:** Performs well on many datasets, including `fraud` and `WDBC`.
    - `fraud`: 0.954
    - `WDBC`: 1.000

    Average `test_auc`: ~ 0.798.

- **Algorithms Struggling on Multiple Datasets:**
  - **COPOD:** Frequently has the lowest `test_auc` on datasets like `vertebral` and `yeast`.
    - `vertebral`: 0.336
    - `yeast`: 0.382

    Average `test_auc`: ~ 0.612.
  - **ECOD:** Often has low `test_auc`, particularly for datasets like `vertebral` and `Wilt`.
    - `vertebral`: 0.411
    - `Wilt`: 0.382

    Average `test_auc`: ~ 0.615.
- **Dataset-Specific Algorithm Success:**
  - `Lymphography`**:**
    - All algorithms achieved perfect `test_auc` (1.000). This indicates the dataset is highly separable for anomaly detection.
    - `COPOD`: 1.000
    - `ECOD`: 1.000
    - `KNN`: 1.000
    - `IForest`: 1.000
  - `vertebral`**:** Most algorithms performed poorly on this dataset with GMM achieving the highest AUC score of 0.532.
    - `COPOD`: 0.336
    - `ECOD`: 0.411
    - `GMM`: 0.532

Dataset Insights

- **Easiest Datasets (Highest `test_auc` Across Algorithms):**
  - `WBC`**:** Most algorithms achieve `test_auc` near 1.0.
    - **Average `test_auc`:** 0.98
    - Spearman Correlation: -0.1497

Table 4: Combined Table of Average Test and Combined AUCs

Algorithms	Average Test AUC	Average Super AUC
KNN	0.846414	0.702326
GMM	0.824685	0.686921
iForest	0.798297	0.658159
sean	0.778965	0.655006
HBOS	0.769171	0.645483
COPOD	0.738202	0.612484
ECOD	0.731145	0.614575

  - `Lymphography`: All algorithms achieve a perfect `test_auc` of 1.0.

- **Most Challenging Datasets:**
  - `vertebral`: None of the algorithms achieve a `test_auc` above 0.532 by `GMM`.
    - **Average `test_auc`:** 0.430
    - Spearman Correlation: 0.6429

Correlation Analysis

- **Datasets with High Spearman Correlation (1.0):**
  - Perfect correlation was observed for datasets like `vowels`, `InternetAds`, `letter`, `magic`, `celeba`, `landsat` and `satellite`.
- **Datasets with Low or Negative Spearman Correlation:**
  - `WPBC` had a Spearman correlation of -0.5000, and `WBC` had a Spearman correlation of -0.1497, indicating no meaningful relationship between `combined_auc` and `test_auc`.
    - Negative correlation suggests that higher combined AUC values are associated with lower test AUC values.

Overall Analysis

Firstly from these results, as can also be seen in Figure 6, about 82% of the datasets show a positive correlation of greater than 0.5 between the Super AUC and the Test AUC scores which already means that there is a strong alignment between the Test and Super AUC scores for most datasets. This also shows that the Super AUC score is quite reliable in evaluating the performance of an anomaly detection algorithm on most datasets, without having to add hyperparameters to the algorithms.

A table representing this data can be found in Table 7 of Appendix A under the column 'Without Hyperparameters'.

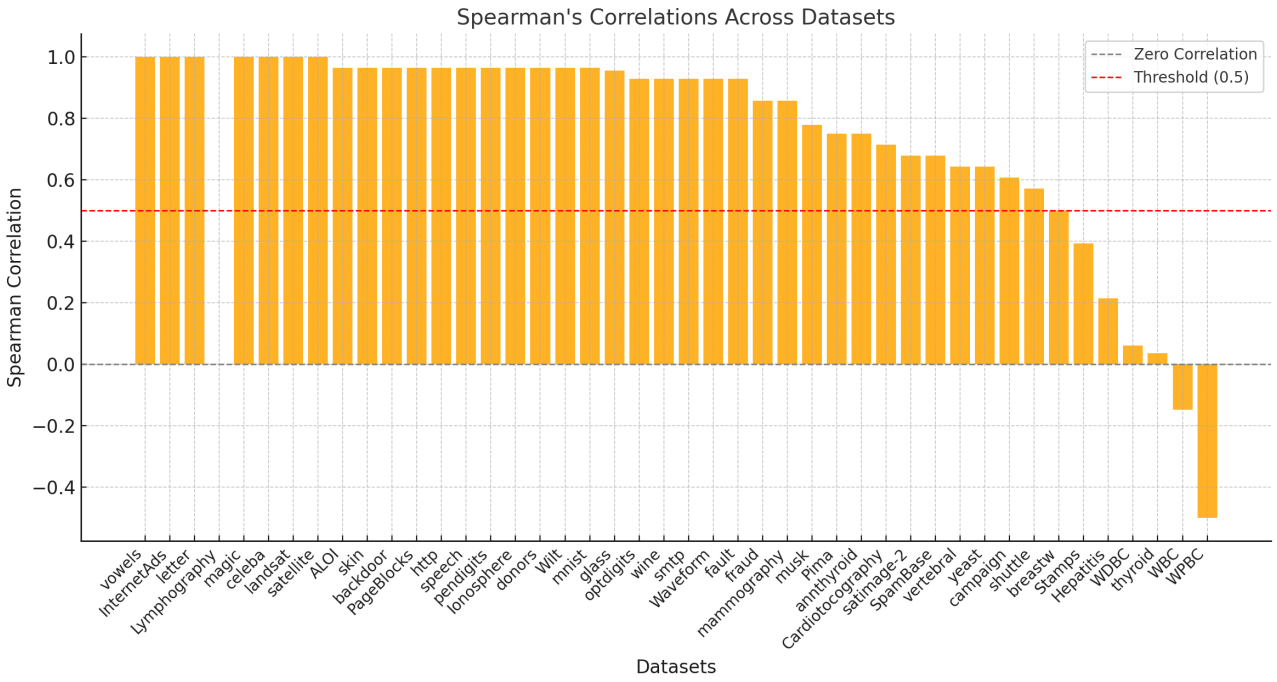

Figure 6: Spearman's Correlations Across Datasets

Additionally, analysis was done where the Test AUC of the algorithm with the highest Super AUC score for each dataset was compared with an algorithm that had the highest Test AUC score for a majority of the datasets. The algorithm was also given a rank by test AUC to see where it stood. As can be seen in Table 5 in Appendix A, the average rank of the chosen algorithm was 1.9.

Referring to the same table (Table 5), the sizes of the test sets of every dataset was also compared to see if it had a correlation to the rank. The results conclude that the test set size was not a contributing factor and as can be seen in the plot in Figure 7, there is no significant relationship between the rank and the test set size. Upon further calculation, the correlation coefficient was found to be -0.13.

Table 5: Algorithm AUC Table

Dataset	Best Alg (KNN) Test AUC	Avg Test AUC (All Algs)	Max Super AUC Test AUC	Chosen Algorithms	Avg Rank by Test ROC	Test Set Size
fraud	0.972367	0.958480	0.972367	KNN	1.000000	984.000000
ALOI	0.701848	0.571572	0.701848	KNN	1.000000	3016.000000
vowels	0.970400	0.734971	0.970400	KNN	1.000000	100.000000
Pima	0.694169	0.674583	0.694169	KNN	4.000000	536.000000
optdigits	0.999267	0.758771	0.999267	KNN	1.000000	300.000000

Continued on next page

Table 5: Algorithm AUC Table

Dataset	Best Alg (KNN) Test AUC	Avg Test AUC (All Algs)	Max Super AUC Test AUC	Chosen Algorithms	Avg Rank by Test ROC	Test Set Size
skin	0.998292	0.715383	0.998292	KNN	1.000000	101718.000000
glass	1.000000	0.818342	1.000000	KNN	1.000000	18.000000
vertebral	0.410000	0.424444	0.410000	KNN	5.000000	60.000000
backdoor	0.950717	0.839872	0.950717	KNN	2.000000	4658.000000
thyroid	0.970979	0.976463	0.970979	KNN	6.000000	186.000000
WPBC	0.552739	0.514519	0.410593	GMM	7.000000	94.000000
InternetAds	0.819330	0.713294	0.914188	GMM	1.000000	736.000000
wine	0.990000	0.860000	0.990000	KNN	1.000000	20.000000
breastw	0.995404	0.991125	0.993575	ECOD	5.000000	478.000000
letter	0.882400	0.662586	0.882400	KNN	1.000000	200.000000
Cardiotocography	0.764996	0.715284	0.764996	KNN	3.000000	932.000000
annthyroid	0.780944	0.811334	0.921562	iForest	1.000000	1068.000000
Lymphography	1.000000	1.000000	1.000000	ECOD	1.000000	12.000000
PageBlocks	0.655271	0.873079	0.949550	GMM	1.000000	1020.000000
campaign	0.744015	0.766859	0.744015	KNN	5.000000	9280.000000
magic	0.843036	0.747544	0.843036	KNN	1.000000	NaN
celeba	0.623989	0.741560	0.808745	GMM	1.000000	9094.000000
http	0.999642	0.992163	0.999642	KNN	1.000000	4422.000000
mammography	0.863365	0.825682	0.903129	ECOD	1.000000	520.000000
Hepatitis	0.532544	0.733728	0.757396	GMM	4.000000	26.000000
shuttle	0.996973	0.991681	0.996973	KNN	1.000000	7022.000000
speech	0.505241	0.501056	0.531309	GMM	1.000000	122.000000
WBC	0.990000	0.974286	0.990000	KNN	3.000000	20.000000
musk	1.000000	0.984301	1.000000	KNN	2.000000	194.000000
pendigits	0.998397	0.930726	0.998397	KNN	1.000000	312.000000
Ionosphere	0.937453	0.850637	0.972474	sean	1.000000	252.000000
donors	0.999994	0.851332	0.999994	KNN	1.000000	73420.000000
satimage-2	0.998810	0.984612	0.998810	KNN	1.000000	142.000000
WDBC	1.000000	0.991429	1.000000	KNN	1.000000	20.000000
smtp	0.945556	0.875556	0.945556	KNN	1.000000	60.000000
landsat	0.767888	0.559984	0.767888	KNN	1.000000	2666.000000
Wilt	0.818877	0.537341	0.818877	KNN	1.000000	514.000000
yeast	0.453633	0.426693	0.453633	KNN	1.000000	1014.000000

Continued on next page

Table 5: Algorithm AUC Table

Dataset	Best Alg (KNN) Test AUC	Avg Test AUC (All Algs)	Max Super AUC Test AUC	Chosen Algorithms	Avg Rank by Test ROC	Test Set Size
mnist	0.937384	0.842534	0.937384	KNN	1.000000	1400.000000
satellite	0.873558	0.769427	0.873558	KNN	1.000000	4072.000000
Waveform	0.805200	0.697114	0.805200	KNN	1.000000	200.000000
Stamps	0.945890	0.915713	0.899063	HBOS	6.000000	62.000000
fault	0.798776	0.630779	0.798776	KNN	1.000000	1346.000000
SpamBase	0.752873	0.752118	0.788134	GMM	3.000000	3358.000000
Average	0.846414	0.783840	0.866520		1.954545	

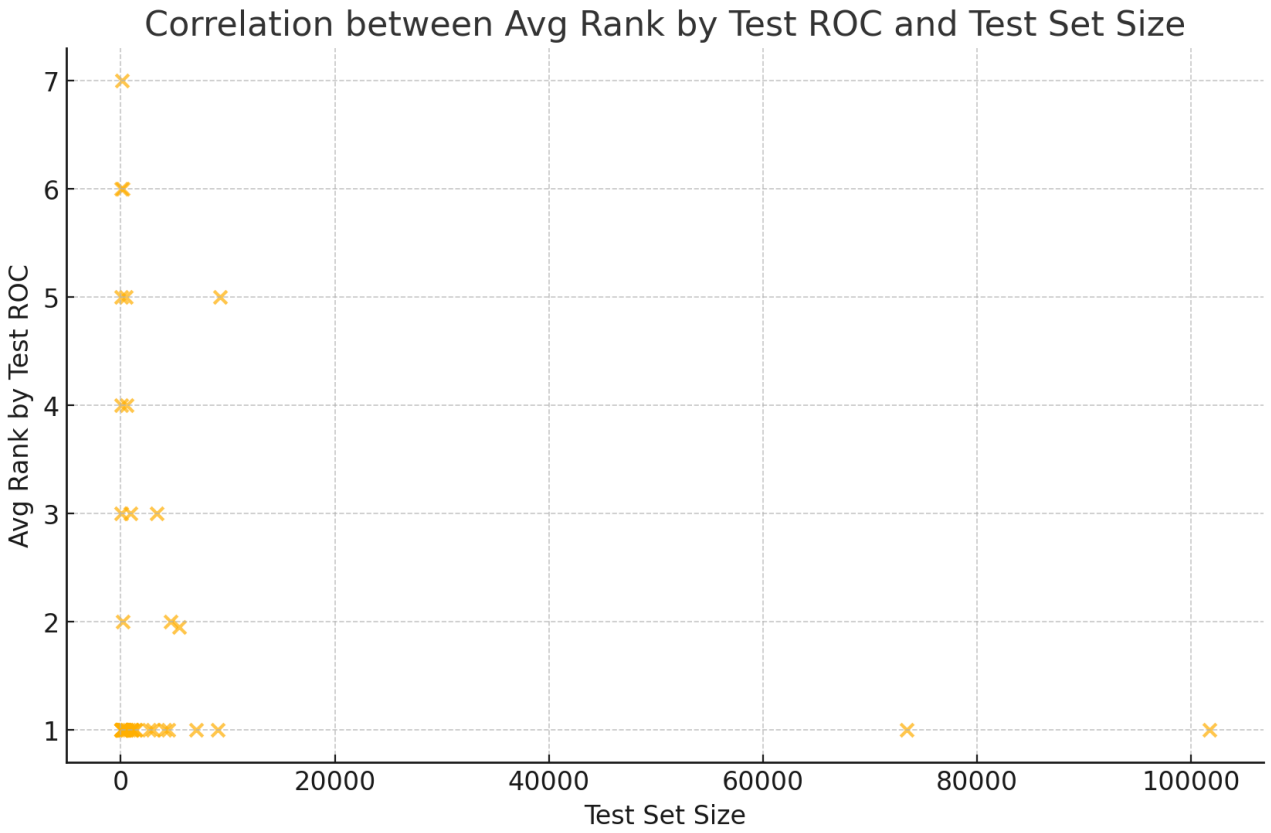

Figure 7: Correlation Between Rank and Test Set Size

Based on the calculations of ranks, a critical difference plot as can be seen in Figure 13 in Appendix B was also plotted to investigate if there was a significant difference between the ranking of the Test AUCs of `KNN` which was chosen as the best algorithm against the Test AUCs of the algorithms which had the highest Super AUC scores. The plot shows that the intervals overlap, which means that the difference is not statistically significant. This also shows that

KNN had a higher average rank as compared to the test AUC of the algorithms with the highest Super AUC score. A lower rank is considered better.

6.2.2 For Models with Hyperparameters

A similar analysis was done with models with hyperparameters to see if the inclusion of hyperparameters affected the effect of the Super AUC score. The same seven algorithms as listed in the section above were used, but hyperparameters were added to the KNN, iForest, GMM and HBOS algorithms which then resulted in the usage of a total of 42 algorithms.

For better clarity, the expansions for the shorthand of the algorithms with hyperparameters mentioned in the section can be found in Table 3 in the Appendix A.

Overall Algorithm Performance

Table 6 shows the average Test and Super AUC scores across all 44 datasets for all 7 algorithms and the variants of the 7 algorithms with hyperparameters.

- **Top-Performing Algorithms (Based on Average test_auc Across Datasets):**
  - **KNN Variants:** Consistently achieve high performance, with high AUC scores in datasets such as fraud, glass, and optdigits.
    - fraud: 0.973 (KNN6-mean)
    - glass: 1.000 (KNN6-mean, KNN4-median)
    - optdigits: 1.000 (KNN4, KNN6-median)

    Average test_auc: ~ 0.85.
  - **GMM Variants:** Performs well on datasets like thyroid and backdoor.
    - thyroid: 0.986 (GMM2-random)
    - backdoor: 0.975 (GMM4)

    Average test_auc: ~ 0.81.
- **Algorithms Struggling on Multiple Datasets:**
  - **COPOD:** Frequently has the lowest test_auc on datasets like vertebral and vowels.
    - vertebral: 0.336
    - vowels: 0.451

    Average test_auc: ~ 0.52.

  - **iForest Variants:** Some variants report consistently low performance (e.g., `iForest32`, `iForest12`).
    * `fraud`: 0.5 (`iForest32`)
    * `ALOI`: 0.5 (`iForest12`)

    Average `test_auc`: ~ 0.66.

- **Dataset-Specific Algorithm Success:**
  - `glass`:
    * Many algorithms achieved perfect `test_auc` (1.000), showcasing high separability for anomaly detection.
    * `KNN6-mean`: 1.000
    * `KNN4-median`: 1.000
    * `GMM4`: 1.000
  - `vertebral`: Most algorithms performed poorly on this dataset, with `GMM4-random` achieving the highest `test_auc` of 0.554.
    * `COPOD`: 0.336
    * `ECOD`: 0.411
    * `GMM4-random`: 0.554

Dataset Insights

- **Easiest Datasets (Highest `test_auc` Across Algorithms):**
  - `glass`: Most algorithms achieve `test_auc` near 1.0, with KNN variants consistently achieving perfect scores.
    * Spearman Correlation: 0.8070
  - `optdigits`: Several algorithms achieve perfect `test_auc` (1.0), particularly KNN variants.
    * Spearman Correlation: 0.9760
  - `vowels`: High-performing dataset where many algorithms achieve near-perfect scores.
    * Spearman Correlation: 0.9719
- **Most Challenging Datasets:**

- `vertebral`: None of the algorithms achieve a `test_auc` above 0.554 which is by `GMM4-random`.
  - Spearman Correlation: 0.5174
- `ALOI`: Algorithms struggle to achieve high `test_auc`, with the highest being 0.739 by `KNN4-mean`.
  - Spearman Correlation: 0.9589
- `fraud`: While the best algorithms achieve high scores (e.g., 0.973), several algorithms under-perform, leading to variability.
  - Spearman Correlation: 0.8512

Correlation Analysis

- **Datasets with High Spearman Correlation:**
  - Strong positive correlation was observed for datasets like `optdigits`, `vowels`, `fraud`, `ALOI`, and `backdoor`.
    - `optdigits`: Spearman Correlation = 0.9760
    - `vowels`: Spearman Correlation = 0.9719
    - `fraud`: Spearman Correlation = 0.8512
    - `ALOI`: Spearman Correlation = 0.9589
    - `backdoor`: Spearman Correlation = 0.9578
- **Datasets with Low or Negative Spearman Correlation:**
  - `vertebral` had a relatively low Spearman correlation of 0.5174, suggesting weak alignment between `combined_auc` and `test_auc`.
    - `vertebral`: Spearman Correlation = 0.5174

Overall Analysis

As shown in Figure 8 the Spearman's correlation between the Test and Super AUCs are greater than 0.5 for around 89% of the datasets which is a larger percentage compared to the 82% for the algorithms without hyperparameters. This also shows a strong relationship between the Super and Test AUC scores for the datasets.

The relevant table for this data can be found in Table 7 in the Appendix A under the column 'With Hyperparameters'. This table also shows a comparison between the values of the

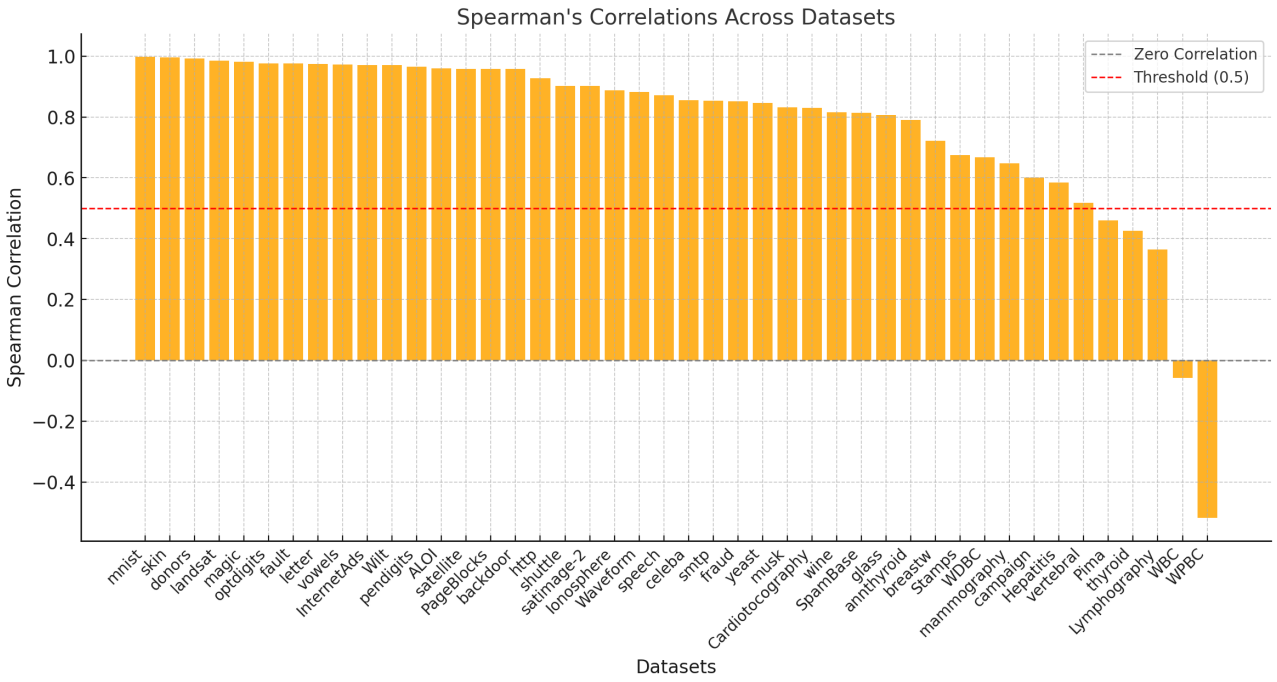

Figure 8: Correlation Between Super and test AUC

Spearman Correlations between the Test and Super AUC scores calculated from running the algorithms on the datasets with hyperparameters and without hyperparameters.

As shown in Table 8, the average rank of the chosen algorithms (algorithms with highest Super AUC and ranked by Test AUC) was 10.7 which was not as promising as the rank obtained by the algorithms without hyperparameters which was 1.95. But the average highest Super AUC was still slightly better than the average Test AUC of the best selected algorithm, which in this case was `KNN4-mean`.

A critical difference plot was constructed for this as well that is shown in Figure 9. This shows that there was barely any difference between the Test AUCs of the algotihms with the highest Super AUC and the best chosen algorithm, which in our case here was `KNN4-mean`.

6.2.3 KNN and its variants

As can be seen in the results above, standard `KNN` and some of its variants showed the highest performance in general among all the other algorithms included in this study. This section simply lists some observations about KNN and its variants to understand it better.

- The standard `KNN` algorithm performs strongly across datasets. For example, in the `fraud` dataset, `KNN` achieved a Super AUC score of 0.743 and a test AUC of 0.972 but in datasets like ALOI, the Super AUC is much lower (0.616) and test AUC of 0.702.
- On the other hand, on the ALOI dataset, a variant of `KNN` (`KNN4-mean`) is the best performing algorithm among all algorithms with Super and test AUC scores of 0.693 and 0.739 respectively. For reference, the scores for the ALOI dataset can be found in Table 10 in Appendix A.

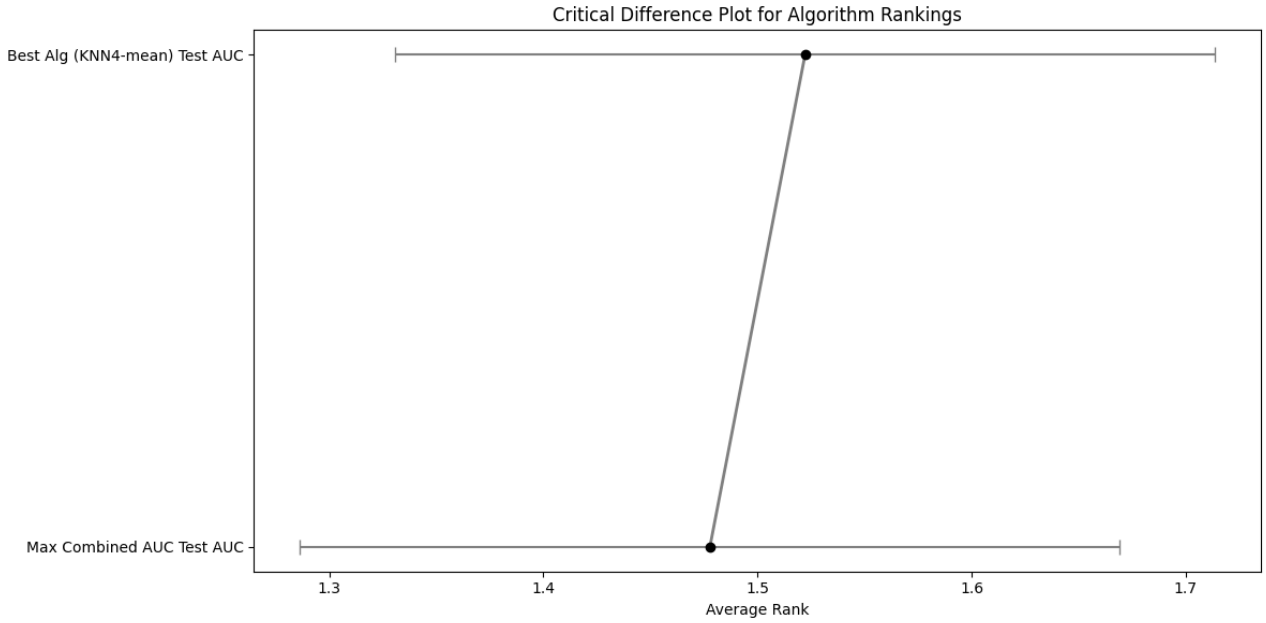

Figure 9: Critical Difference Plot (KNN4-mean Test AUC vs Test AUC of Highest Super AUC)

- `KNN6-mean` and `KNN4-mean` often perform better than standard `KNN`. In the `fraud` dataset `KNN6-mean` has a higher Super AUC of 0.775 while KNN4-mean has the highest Super AUC score of 0.794. Similary in the `vowels` dataset, `KNN6-mean` has an outstanding Super AUC of 0.862, performing much better than the baseline KNN. Overall, KNN4-mean and KNN4-median tend to display consistently high AUC scores, suggesting that mean/method-based neighbor calculations improve robustness.

- Higher neighbor counts seem to show mixed results. For example, in the `fraud` dataset, KNN7 scores high in Super AUC score with 0.737, slightly lower than `KNN6-mean` and KNN4-median. In the `optidigits` dataset on the other hand, it achieves a near perfect Test AUC score of 0.9994 but a much lower Super AUC of 0.746.

- KNN based methods seem to dominate the `fraud` dataset in terms of performance where anomaly detection is absolutely critical. `KNN6-mean`, KNN4-median, and KNN4-mean display the best results in terms of combined AUC (0.775–0.794) and test AUC (0.972–0.973). The median-based KNN models (KNN2-median, KNN6-median, KNN4-median) also consistently achieve high scores, suggesting that alternative distance measures also perform well in fraud detection.

- Some datasets show low performance for KNN. In ALOI, the Super AUC score for KNN is 0.616 which indicates poor separation between normal and anomalous data points. In the `vertebral` dataset, the Super AUC for KNN drops to 0.503 which is as good as random guessing. This shows that KNN may struggle when feature distributions are highly overlapping or when anomalies are not clearly separated from normal data.

- Compared to other models like iForest, GMM and COPOD, KNN variants consistently outperforms these in datasets like fraud and vowels. But in datasets like ALOI and

vertebral, iForest and GMM perform competitively or better than KNN, showing that the type of datasets that these algorithms are applied to also plays a role in performance.

A plot comparing the performance of KNN variants across datasets is displayed in Figure 10.

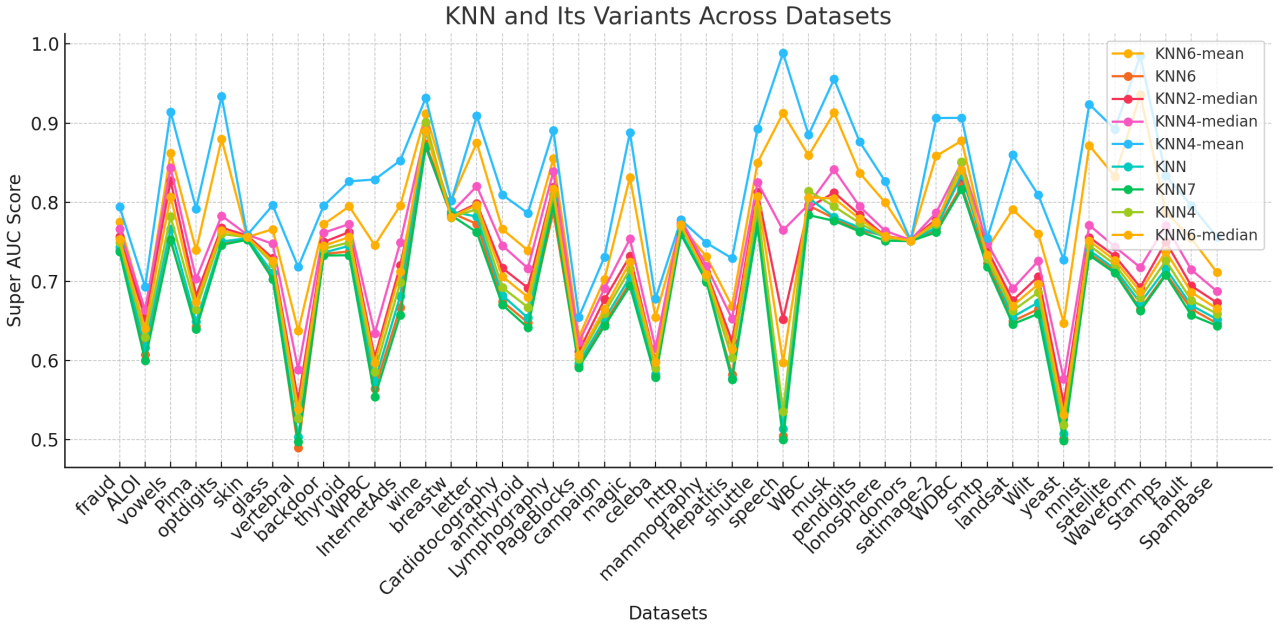

Figure 10: Performance of KNN variants across datasets

6.2.4 One-Class Support Vector Machines

One-Class Support Vector Machines (OCSVM) was not listed in the final results but was included in the original python script of algorithms without hyperparameters. It was considered in early experiments but was dropped in the extended analysis.

OCSVM is a kernel-based method that models normal data and detects outliers based on deviations from learned hyperplanes[41]. The performance of OCSVM is highly dependent on dataset characteristics.

OCSVM generally works well for high-dimensional datasets where anomalies are not well clustered and do not follow a simple distribution. The reason it was exluded from the analysis is because it was computationally very expensive on large datasets. It was also outperformed by tree based models like iForest and distance based models like KNN.

An illustration of OCSVM can be seen in Figure 11.

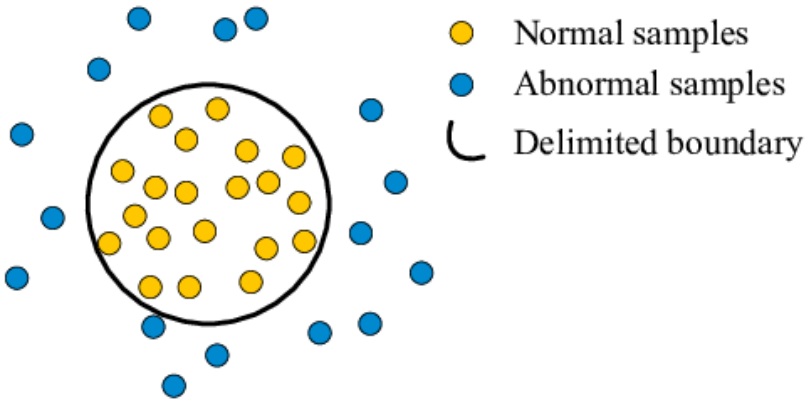

Figure 11: OCSVM[35]

7 Conclusion and Discussions

7.1 Conclusion

This study introduced a novel methodology for selecting anomaly detection models for unlabeled datasets by leveraging the Super AUC score. By applying this metric across multiple datasets, we compared many variants of many anomaly detection algorithms like KNN, iForest, GMM, HBOS and some others. The results (as seen in Chapter 6) demonstrated that the Super AUC effectively evaluated models for unlabeled datasets by combining decision scores from both the test and the training datasets.

To reiterate, some key findings from this study were:

- KNN-based models consistently performed better than most other models on many datasets, which was especially obvious in datasets like Vowels and Backdoor where Super AUC scores of 0.914 and 0.795 were respectively achieved.
- Models like GMM and HBOS also showed competitive performance but displayed variability across datasets, suggesting their dependence on data characteristics such as dimensionality and class imbalance.
- A strong overall Spearman correlation between the test AUC and Super AUC, where the Spearman correlation value exceeded 0.85 in many cases, validated the reliability of the Super AUC score.

7.2 Ethical Considerations

- Some datasets may contain inherent biases such as imbalanced class distributions. If these biases are not addressed, the evaluated algorithms could amplify the effect of thee biases. Ensuring that the datasets are diverse and representative could minimize the risk of unfair conclusions regarding the performance of anomaly detection methods across different practical scenarios.
- To protect individual privacy, the datasets must be anonymized and handled in compliance with data protection laws as anomaly detection is usually applied to sensitive domains such as cybersecurity, medical diagnostics or fraud detection. Storing or processing datasets when they include personal information requires secure infrastructure to prevent unauthorized access.
- The results and metrics, including the Super AUC score must be interpretable and transparent to ensure that, when reproduced, the evaluation process can be trusted.
- The findings from this project could be applied in very sensitive domains such as fraud detection, where incorrect anomaly detection could lead to sever consequences. Researchers and practitioners should emphasize the limitations of each algorithm and caution against over-reliance on any single method without thorough domain-specific validation.

7.3 Practical Considerations

- The experimental setup involves a large number of algorithms and datasets, which presents computational scalability challenges. Future practitioners may need to scale up resources or optimize execution strategies. In real-world applications, the algorithms must be tested for scalability on production-level data volumes and velocities.

- The datasets used in this study may not fully represent all real-world scenarios. Practical implementation should involve further testing on domain-specific datasets to ensure generalizability.

- The Super AUC (Combined AUC) metric provides a unique perspective by combining training and testing decision scores. However, practitioners should consider if this metric aligns with their specific domain requirements, as different contexts may prioritize other performance metrics (e.g., precision, recall, or F1 score). The choice of correlation metrics (Spearman, Pearson) assumes a linear or monotonic relationship, which may not fully capture more complex dependencies between metrics.

- Resource-intensive algorithms such as KNN and GMM may pose challenges for deployment in environments with constrained computational power, such as IoT devices or edge computing systems. Practical applications should consider lightweight alternatives or optimizations to reduce computational costs while maintaining acceptable performance.

- Some domains, such as medical diagnostics or critical infrastructure monitoring, demand extremely low false positive and false negative rates. The tolerance for error in such applications should guide the choice of algorithms and their parameter configurations. During evaluation, it is essential to log and analyze algorithm failures to understand their root causes and mitigate potential risks in deployment.

- Anomaly detection is often embedded in larger systems with human decision-makers, such as fraud analysts or system administrators. Practical implementation should involve collaboration with these stakeholders to ensure the algorithms provide actionable insights that align with operational workflows.

7.4 Future Work

Based on the results and limitations that we have explored in this study, there are several avenues for future work as listed below:

- Future studies can explore more anomaly detection algorithms, perhaps even deep learning based methods, to assess their performance using the Super AUC score.

- It would be useful to explore the performance of the Super AUC score in real-time anomaly detection systems.

- Studying why some algorithms perform very well on some kinds of datasets and why some

algorithms perform very badly could help in trying to zero in on a proper hyperparameter selection for a particular algorithm on a given dataset.

- Extending the methodology to automatically tune hyperparameters in order to maximize the Super AUC score would lead to better performing algorithms.
- Evaluating the scalability of models that have high performance on larger and more complex datasets ensures that they can be applied to real world anomaly detection problems.

References

[1] Michael M (https://stats.stackexchange.com/users/30351/michael-m). *Is Anomaly Detection Supervised or Un-supervised?* Cross Validated. eprint: https://stats.stackexchange.com/q/474663. URL: https://stats.stackexchange.com/q/474663.

[2] Samhita Alla. *Evaluation metrics for machine learning models*. Apr. 2022. URL: https://blog.paperspace.com/ml-evaluation-metrics-part-2/.

[3] E. Alpaydin and C. Kaynak. *Optical Recognition of Handwritten Digits*. UCI Machine Learning Repository. 1998. URL: https://doi.org/10.24432/C50P49.

[4] Abdelrahman Alzarooni et al. *Anomaly Detection for Industrial Applications, Its Challenges, Solutions, and Future Directions: A Review*. 2025. arXiv: 2501.11310 [cs.CV]. URL: https://arxiv.org/abs/2501.11310.

[5] Anomalo. *Data anomaly: What is it, common types and how to identify them*. Jan. 2025. URL: https://www.anomalo.com/blog/data-anomaly-what-is-it-common-types-and-how-to-identify-them/.

[6] Harshini Bhat. *What is anomaly detection? types, algorithm and applications*. Aug. 2023. URL: https://www.almabetter.com/bytes/articles/anomaly-detection.

[7] Rajen Bhatt and Abhinav Dhall. *Skin Segmentation*. UCI Machine Learning Repository. 2009. URL: https://doi.org/10.24432/C5T30C.

[8] Julia Bohutska. *Anomaly detection-how to tell good performance from bad*. Aug. 2021. URL: https://towardsdatascience.com/anomaly-detection-how-to-tell-good-performance-from-bad-b57116d71a10.

[9] Schuyler Brown. *What is anomaly detection? algorithms, examples, and more*. June 2024. URL: https://www.strongdm.com/blog/anomaly-detection.

[10] *Confusion Matrix*. URL: https://encord.com/glossary/confusion-matrix/.

[11] *COPOD*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.copod.

[12] David Deterding, Mahesan Niranjan, and Tony Robinson. *Connectionist Bench (Vowel Recognition - Deterding Data)*. UCI Machine Learning Repository. 1988. URL: https://doi.org/10.24432/C58P4S.

[13] *ECOD*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.ecod.

[14] Jim Frost. *Spearman's correlation explained*. Apr. 2024. URL: https://statisticsbyjim.com/basics/spearmans-correlation/.

[15] Ivan Gavrilan et al. *Anomaly detection with quantum machine learning – identifying cybersecurity issues in datasets*. Jan. 2024. URL: https://www.cybersecurity.blog.aisec.fraunhofer.de/en/anomaly-detection-with-quantum-machine-learning-identifying-cybersecurity-issues-in-datasets/.

[16] B. German. *Glass Identification*. UCI Machine Learning Repository. 1987. URL: https://doi.org/10.24432/C5WW2P.

[17] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W.M. Smeulders. "The Amsterdam Library of Object Images". In: *International Journal of Computer Vision* 61.1 (Jan. 2005), pp. 103–112. ISSN: 0920-5691. DOI: 10.1023/b:visi.0000042993.50813.60. URL: http://dx.doi.org/10.1023/B:VISI.0000042993.50813.60.

[18] *GMM*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.gmm.

[19] Nicolas Goix. *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?* 2016. arXiv: 1607.01152 [stat.ML]. URL: https://arxiv.org/abs/1607.01152.

[20] Prince Grover et al. *Fraud Dataset Benchmark and Applications*. 2023. arXiv: 2208.14417 [cs.LG]. URL: https://arxiv.org/abs/2208.14417.

[21] Stephen Guth, Alireza Mojahed, and Themistoklis P. Sapsis. "Quality measures for the evaluation of machine learning architectures on the quantification of epistemic and aleatoric uncertainties in complex dynamical systems". In: *Computer Methods in Applied Mechanics and Engineering* 420 (2024), p. 116760. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.116760. URL: https://www.sciencedirect.com/science/article/pii/S0045782524000161.

[22] Songqiao Han et al. *ADBench: Anomaly Detection Benchmark*. 2022. arXiv: 2206.09426 [cs.LG]. URL: https://arxiv.org/abs/2206.09426.

[23] *HBOS*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.hbos.

[24] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. *Deep Anomaly Detection with Outlier Exposure*. 2019. arXiv: 1812.04606 [cs.LG]. URL: https://arxiv.org/abs/1812.04606.

[25] Mark Hopkins et al. *Spambase*. UCI Machine Learning Repository. 1999. URL: https://doi.org/10.24432/C53G6X.

[26] *iForest*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.iforest.

[27] *JSON*. URL: https://docs.python.org/3/library/json.html#module-json.

[28] Nigar Esra K305;n. *What are isolation forests?* June 2022. URL: https://engineering.teknasyon.com/what-are-isolation-forests-151d8e98ef5f.

[29] Stylianos Kampakis. *What is anomaly detection, and why you need it.* Mar. 2024. URL: https://thedatascientist.com/anomaly-detection-why-you-need-it/.

[30] Mr Farkhod Khushaktov. *Gaussian mixture model by example in Python*. Aug. 2023. URL: https://medium.com/@mrmaster907/gaussian-mixture-model-by-example-in-python-f3891f51eccd.

[31] Simon Klüttermann et al. "Towards Highly Efficient Anomaly Detection for Predictive Maintenance". In: *International Conference on Machine Learning and Applications (ICMLA)*. 2024.

[32] *KNN*. URL: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.knn.

[33] George Lawton. *What is anomaly detection? an overview and explaination*. July 2024. URL: https://www.techtarget.com/searchEnterpriseAI/definition/anomaly-detection.

[34] UCI Machine Learning. *Pima Indians Diabetes Database*. Oct. 2016. URL: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

[35] Chenyang Li et al. "Lifelong Condition Monitoring Based on NB-IoT for Anomaly Detection of Machinery Equipment". In: *Procedia Manufacturing* 49 (Jan. 2020), pp. 144–149. DOI: 10.1016/j.promfg.2020.07.010.

[36] Jiyan Salim Mahmud and Imre Lendak. "Enhancing One-Class Anomaly Detection in Unlabeled Datasets Through Unsupervised Data Refinement". In: *2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*. 2024, pp. 000497–000502. DOI: 10.1109/SISY62279.2024.10737577.

[37] *Matplotlib*. URL: https://matplotlib.org/stable/users/getting_started/index.html.

[38] Sarala Naidu and Ning Xiong. *S2DEVFMAP: Self-Supervised Learning Framework with Dual Ensemble Voting Fusion for Maximizing Anomaly Prediction in Timeseries*. 2024. arXiv: 2404.16179 [cs.LG]. URL: https://arxiv.org/abs/2404.16179.

[39] Andrew Nailman. *Real-life examples of unsupervised machine learning*. May 2024. URL: https://machinelearningmodels.org/real-life-examples-of-unsupervised-machine-learning/.

[40] *Numpy*. URL: https://numpy.org/doc/stable/user/absolute_beginners.html.

[41] *OCSVM*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html.

[42] Orkun Orulluolu. *Area under the curve (AUC): A robust performance measure of classification models*. Aug. 2023. URL: https://medium.com/@bayramorkunor/area-under-the-curve-auc-a-robust-performance-measure-of-classification-models-cbfc3549d8c6.

[43] *OS*. URL: https://docs.python.org/3/library/os.html.

[44] *Pandas*. URL: https://pypi.org/project/pandas/.

[45] *pyod*. URL: https://pyod.readthedocs.io/en/latest/index.html.

[46] Chen Qiu et al. *Latent Outlier Exposure for Anomaly Detection with Contaminated Data*. 2022. arXiv: 2202.08088 [cs.LG]. URL: https://arxiv.org/abs/2202.08088.

[47] Ross Quinlan. *Thyroid Disease*. UCI Machine Learning Repository. 1986. URL: https://doi.org/10.24432/C5D010.

[48] *Random*. URL: https://docs.python.org/3/library/random.html.

[49] *Rankdata*. URL: https://docs.scipy.org/doc/scipy-1.13.0/reference/generated/scipy.stats.rankdata.html.

[50] *ROC and AUC*. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[51] *ROC AUC*. URL: https://www.evidentlyai.com/classification-metrics/explain-roc-curve.

[52] Frank Sacthesw. *Key use cases of anomaly detection in various industries*. Jan. 2025. URL: https://futuremachinelearning.org/key-use-cases-of-anomaly-detection-in-various-industries/.

[53] *scikit*. URL: https://scikit-learn.org/stable/getting_started.html.

[54] Srikant Shenoy. *What is the difference between supervised and unsupervised anomaly detection?* Sept. 2023. URL: https://www.linkedin.com/advice/0/what-difference-between-supervised-unsupervised.

[55] *sys*. URL: https://docs.python.org/3/library/sys.html.

[56] PythonGeeks Team. *Python scipy tutorial for beginners*. Nov. 2021. URL: https://pythongeeks.org/python-scipy-tutorial-for-beginners/.

[57] The Educative Team. *Get started with anomaly detection algorithms in 5 minutes*. Dec. 2022. URL: https://learningdaily.dev/get-started-with-anomaly-detection-algorithms-in-5-minutes-225691660e5f.

[58] Bex Tuychiev. *A comprehensive introduction to anomaly detection*. Nov. 2023. URL: https://www.datacamp.com/tutorial/introduction-to-anomaly-detection?dc_referrer=https%3A%2F%2Fduckduckgo.com%2F.

[59] Unknown. *MNIST Database of Handwritten Digits*. 1998. DOI: 10.24432/C53K8Q. URL: https://archive.ics.uci.edu/dataset/683.

[60] Kay Jan Wong. *7 evaluation metrics for clustering algorithms*. Oct. 2023. URL: https://towardsdatascience.com/7-evaluation-metrics-for-clustering-algorithms-bdc537ff54d2.

[61] Jinsung Yoon et al. "Self-supervise, Refine, Repeat: Improving Unsupervised Anomaly Detection". In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: https://openreview.net/forum?id=b3v1UrtF6G.

[62] Eugenio "Jay" Zuccarelli. *Performance metrics in machine learning-part 3: Clustering*. Jan. 2021. URL: https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6.

A Appendix A

Algorithm	Avg Super AUC	Avg Test AUC
KNN4-mean	0.8266	0.8546
KNN4-median	0.7439	0.8526
KNN6-mean	0.7868	0.8518
KNN2-median	0.7254	0.8494
KNN6-median	0.7191	0.8488
KNN6	0.6958	0.8386
KNN7	0.6927	0.8371
KNN4	0.6853	0.8363
KNN	0.6836	0.8354
GMM3	0.6778	0.8222
GMM4	0.6761	0.8211
GMM-tied	0.6745	0.8187
GMM	0.6736	0.8175
GMM2	0.6679	0.8123
GMM4-random	0.6625	0.8098
GMM-diag	0.6548	0.8056
GMM3-random	0.6529	0.8012
GMM2-random	0.6495	0.7989
GMM-spherical	0.6452	0.7933
iForest3	0.6389	0.7885
HBOS15	0.6346	0.7854
iForest	0.6317	0.7812
HBOS20	0.6298	0.7785
HBOS5	0.6236	0.7721
HBOS	0.6215	0.7703
iForest2	0.6192	0.7654
iForest13	0.6148	0.7629
COPOD	0.6072	0.7571
ECOD	0.6031	0.7544
sean	0.5996	0.7508
iForest14	0.5943	0.7461
iForest34	0.5906	0.7427
iForest24	0.5841	0.7354
iForest23	0.5798	0.7303
iForest33	0.5723	0.7238
iForest32	0.5000	0.5000
iForest12	0.5000	0.5000
iForest22	0.5000	0.5000

Table 6: Average Super AUC and Test AUC for All Algorithms Across All Datasets

Dataset	With Hyperparameters	Without Hyperparameters
ALOI	0.958945	0.964286
Cardiotocography	0.830264	0.714286
Hepatitis	0.585562	0.214286
InternetAds	0.971278	1.000000
Ionosphere	0.886897	0.964286
Lymphography	0.365393	NaN
PageBlocks	0.957972	0.964286
Pima	0.460446	0.750000
SpamBase	0.813063	0.678571
Stamps	0.675079	0.392857
WBC	-0.057506	-0.149696
WDBC	0.666991	0.059108
WPBC	-0.516069	-0.500000
Waveform	0.881334	0.928571
Wilt	0.971116	0.964286
annthyroid	0.790183	0.750000
backdoor	0.957809	0.964286
breastw	0.722353	0.500000
campaign	0.600974	0.607143
celeba	0.855091	1.000000
donors	0.991559	0.964286
fault	0.975335	0.928571
fraud	0.851197	0.857143
glass	0.807037	0.954994
http	0.926342	0.964286
landsat	0.984097	1.000000
letter	0.973225	1.000000
magic	0.981339	1.000000
mammography	0.648357	0.857143
mnist	0.998377	0.964286
musk	0.832369	0.778312
optdigits	0.975991	0.928571
pendigits	0.965232	0.964286
satellite	0.958621	1.000000
satimage-2	0.901741	0.678571
shuttle	0.902312	0.571429
skin	0.995781	0.964286
smtp	0.852958	0.928571
speech	0.871968	0.964286
thyroid	0.426369	0.035714
vertebral	0.517404	0.642857
vowels	0.971927	1.000000
wine	0.814752	0.928571
yeast	0.846304	0.642857

Table 7: Comparison of Spearman correlations across datasets

Table 8: Algorithm AUC Table (With Hyperparameters)

Dataset	Best Alg (KNN4-mean) Test AUC	Avg Test AUC (All Algs)	Max Super AUC Test AUC	Chosen Algorithms	Avg Rank by Test ROC	Test Set Size
fraud	0.972499	0.908421	0.972499	KNN4-mean	2.000000	984.000000
ALOI	0.739353	0.575843	0.739353	KNN4-mean	2.000000	3016.000000
vowels	0.995600	0.751729	0.995600	KNN4-mean	1.000000	100.000000
Pima	0.684061	0.677361	0.684061	KNN4-mean	30.000000	536.000000
optdigits	1.000000	0.758234	1.000000	KNN4-mean	6.000000	300.000000
skin	0.998582	0.763039	0.998536	KNN4-median	2.000000	101718.000000
glass	1.000000	0.835097	1.000000	KNN4-mean	7.000000	18.000000
vertebral	0.396667	0.450847	0.396667	KNN4-mean	31.000000	60.000000
backdoor	0.954064	0.803709	0.954064	KNN4-mean	4.000000	4658.000000
thyroid	0.971673	0.923191	0.971673	KNN4-mean	21.000000	186.000000
WPBC	0.525124	0.511845	0.437302	GMM4-random	33.000000	94.000000
InternetAds	0.864429	0.716505	0.919785	GMM4-random	1.000000	736.000000
wine	0.990000	0.859524	0.910000	GMM4-random	20.000000	20.000000
breastw	0.995326	0.927661	0.993575	ECOD	19.000000	478.000000
letter	0.905900	0.695655	0.905900	KNN4-mean	2.000000	200.000000
Cardiotocography	0.799656	0.709033	0.799656	KNN4-mean	9.000000	932.000000
annthyroid	0.799548	0.790894	0.799548	KNN4-mean	22.000000	1068.000000
Lymphography	1.000000	0.915344	1.000000	KNN4-mean	19.000000	12.000000
PageBlocks	0.663395	0.801835	0.949550	GMM	2.000000	1020.000000
campaign	0.721488	0.722359	0.721488	KNN4-mean	30.000000	9280.000000
magic	0.848740	0.751108	0.848740	KNN4-mean	3.000000	NaN
celeba	0.625412	0.702128	0.625412	KNN4-mean	31.000000	9094.000000
http	0.999652	0.948194	0.999668	KNN4-median	7.000000	4422.000000
mammography	0.866753	0.824115	0.866753	KNN4-mean	18.000000	520.000000
Hepatitis	0.520710	0.660609	0.636095	GMM4-random	24.000000	26.000000
shuttle	0.997965	0.935310	0.997965	KNN4-mean	5.000000	7022.000000
speech	0.673206	0.517494	0.673206	KNN4-mean	1.000000	122.000000
WBC	0.990000	0.949881	0.990000	KNN4-mean	17.000000	20.000000
musk	1.000000	0.884371	1.000000	KNN4-mean	11.000000	194.000000
pendigits	0.999753	0.898558	0.999753	KNN4-mean	1.000000	312.000000
Ionosphere	0.941484	0.828582	0.960884	GMM4-random	7.000000	252.000000
donors	1.000000	0.861180	1.000000	KNN4-median	3.000000	73420.000000
satimage-2	0.998611	0.924413	0.998611	KNN4-mean	5.000000	142.000000

Continued on next page

Table 8: Algorithm AUC Table

Dataset	Best Alg (KNN4-mean) Test AUC	Avg Test AUC (All Algs)	Max Super AUC Test AUC	Chosen Algorithms	Avg Rank by Test ROC	Test Set Size
WDBC	1.000000	0.942381	1.000000	KNN4-mean	10.000000	20.000000
smtp	0.923333	0.835503	0.923333	KNN4-mean	8.000000	60.000000
landsat	0.780215	0.593968	0.780215	KNN4-mean	1.000000	2666.000000
Wilt	0.844010	0.611804	0.844010	KNN4-mean	7.000000	514.000000
yeast	0.462441	0.435998	0.462441	KNN4-mean	6.000000	1014.000000
mnist	0.941016	0.806670	0.941016	KNN4-mean	1.000000	1400.000000
satellite	0.877039	0.759115	0.877039	KNN4-mean	2.000000	4072.000000
Waveform	0.797700	0.686870	0.797700	KNN4-mean	16.000000	200.000000
Stamps	0.965661	0.887543	0.965661	KNN4-mean	4.000000	62.000000
fault	0.813790	0.636786	0.813790	KNN4-mean	1.000000	1346.000000
SpamBase	0.759257	0.729040	0.759257	KNN4-mean	20.000000	3358.000000
Average	0.854639	0.766131	0.861609		10.727273	

Datasets
fraud
ALOI
vowels
Pima
optdigits
skin
glass
vertebral
backdoor
thyroid
WPBC
InternetAds
wine
breastw
letter
Cardiotocography
annthyroid
Lymphography
PageBlocks
campaign
magic
celeba
http
mammography
Hepatitis
shuttle
speech
WBC
musk
pendigits
Ionosphere
donors
satimage-2
WDBC
smtp
landsat
Wilt
yeast
mnist
satellite
Waveform
Stamps
fault
SpamBase

Table 9: List of Datasets

Table 10: ALOI Dataset Results

Algorithm	Super AUC	Test AUC
KNN4-mean	0.693307	0.739353
KNN4-median	0.664084	0.740266
KNN6-mean	0.659340	0.723441
KNN2-median	0.654323	0.728066
KNN6-median	0.640689	0.720273
KNN4	0.629474	0.713049
KNN	0.616852	0.701848
KNN6	0.607512	0.692963
KNN7	0.600429	0.685597
GMM-tied	0.523586	0.570095
GMM-random	0.523586	0.570095
GMM	0.523586	0.570095
GMM-tied-random	0.523586	0.570095
GMM2-random	0.521690	0.578981
GMM4-random	0.521010	0.568908
GMM2	0.520734	0.577164
GMM-diag-random	0.519631	0.558083
GMM-diag	0.519631	0.558083
sean	0.518138	0.556728
GMM4	0.517202	0.573068
GMM3	0.516768	0.572575
GMM3-random	0.516535	0.570515
iForest	0.513015	0.555041
HBOS	0.512621	0.542240
HBOS15	0.511766	0.546020
HBOS5	0.510841	0.533380
HBOS20	0.509999	0.544485
ECOD	0.507077	0.544227
GMM-spherical-random	0.506020	0.522995
GMM-spherical	0.506020	0.522995
iForest3	0.504028	0.540371
iForest12	0.500000	0.500000
iForest32	0.500000	0.500000
iForest22	0.500000	0.500000
COPOD	0.499143	0.530825
iForest2	0.497549	0.523709
iForest13	0.489301	0.493707
iForest34	0.485141	0.495569
iForest23	0.483936	0.487636
iForest24	0.483895	0.494490
iForest14	0.480401	0.486639
iForest33	0.475468	0.481742

B Appendix B

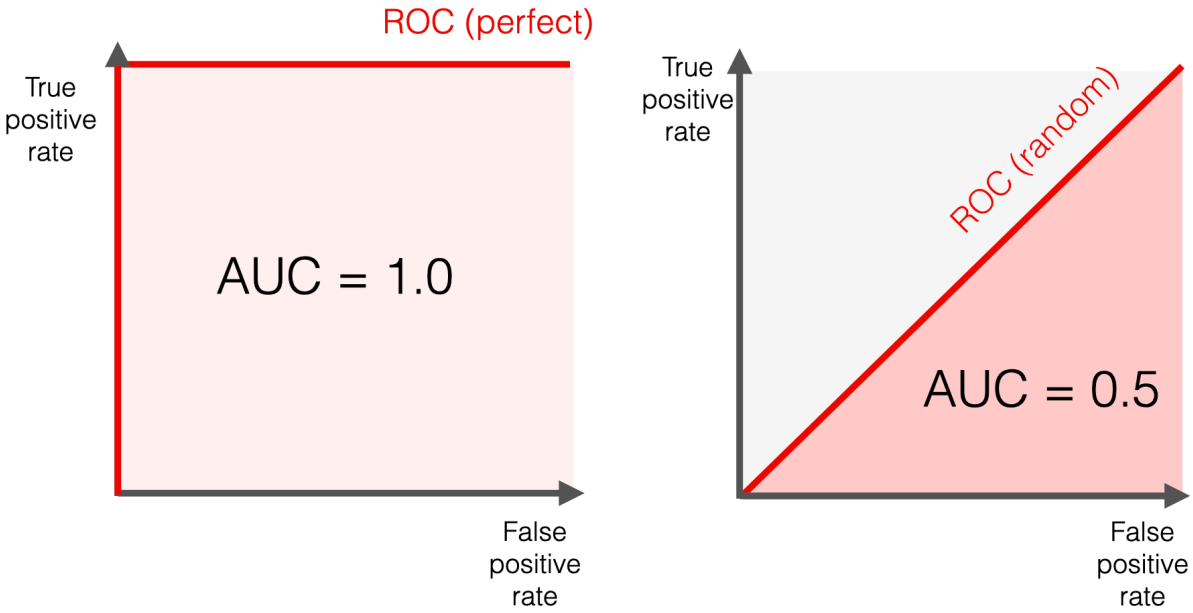

Figure 12: ROC AUC[51]

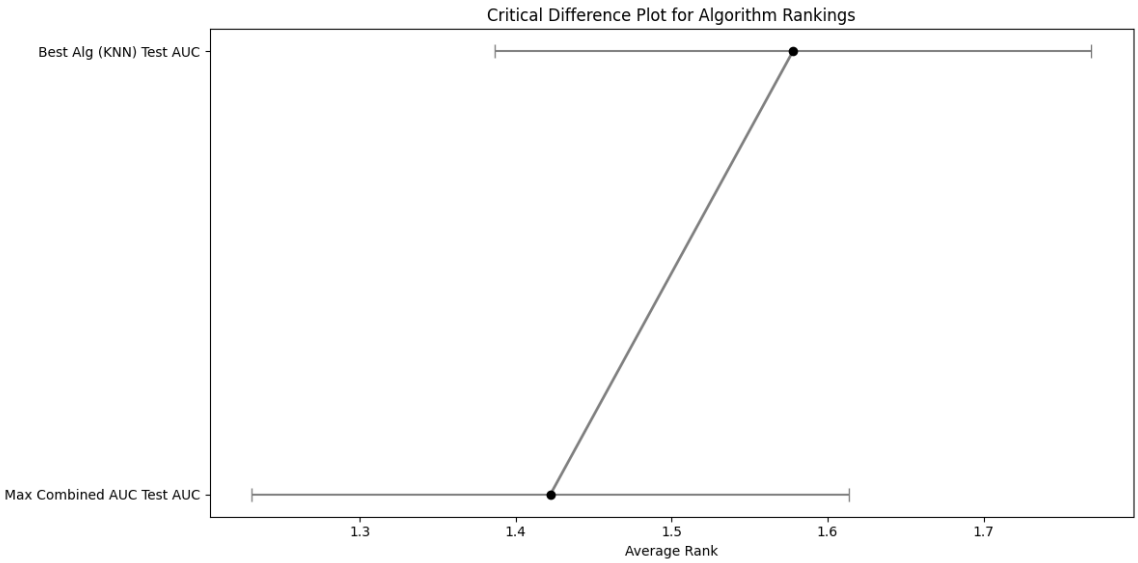

Figure 13: Critical Difference Plot (KNN Test AUC vs Test AUC of Highest Super AUC)

Eidesstattliche Versicherung

(Affidavit)

Gautam Dilip Hariharan

Name, Vorname
(surname, first name)

230237

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Anomaly Detection Model Selection Using Super-AUC Scores

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 02.02.2025

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 02.02.2025

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**