Open Anomaly Detection Benchmark (OADB)

A benchmark for evaluation of anomaly detection algorithms

Muhammad Shah Zaib

Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2021

Open Anomaly Detection Benchmark (OADB)

A benchmark for evaluation of anomaly detection algorithms

Muhammad Shah Zaib

Open Anomaly Detection Benchmark (OADB)

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

There has been a significant rise in data generated and collected through various sources. Machine learning algorithms are used to find meaningful information from this data. Among many other applications of machine learning algorithms, an exciting task is anomaly detection. Anomaly detection is the process of finding observations that deviate significantly from the general distribution of data. Finding an unusual behavior in data is critical in various fields. Many algorithms were proposed in the literature and research is ongoing to develop new algorithms that are more efficient and accurate for detecting anomalies. A large number of algorithms has made choosing and evaluating algorithms significantly difficult. Existing benchmarks for evaluating machine learning algorithms lack the necessary capabilities, e.g., transparency, and proper visualization of results.

To make it easier to compare and choose anomaly detection algorithms, we propose a new benchmark called Open Anomaly Detection Benchmark (OADB). We identified requirements for a good anomaly detection benchmark, e.g., variety of datasets, extensibility, and in-depth visualisation of results. OADB is developed based on identified requirements. OADB fills gaps in existing benchmarks and helps get insight into state of the art algorithms for anomaly detection. A subset of popular machine learning algorithms for anomaly detection and a variety of numerical datasets are integrated into the benchmark for evaluation. OADB is available as part of DataBench Toolbox.

Multiple benchmarking experiments are performed using the OADB. OADB utilizes numerical datasets of varying number of dimensions and size. These datasets are collected from multiple sources and converted to a single format for usage in OADB. Results show that Windowed gaussian on univariate and Isolation forest on multivariate datasets perform better than other algorithms in terms of accuracy and computational complexity. However, performance of an algorithm varies significantly based on the dataset utilized. Selection of machine learning algorithms for anomaly detection should be according to problem domain and data available for utilization.

Acknowledgements

My sincere thanks to Dr. Arne J. Berre and Dr. Volker Hoffmann for providing invaluable guidance and help in various ways. Your availability for regular meetings has been beneficial for me. I am also thankful to Dr. Dumitru Roman for his feedback on the thesis.

I am profoundly grateful to SINTEF for providing me with the resources to work on my thesis and the University of Oslo (UiO) for providing the best learning environment.

At last, I would like to thank my parents and siblings for supporting me throughout the study.

Muhammad Shah Zaib
November 2021

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
CPU	Central Processing Unit
DL	Deep Learning
GPU	Graphics Processing Unit
IP	Internet Protocol
KNN	K Nearest Neighbors
ML	Machine Learning
NAB	Numenta Anomaly Benchmark
OADB	Open Anomaly Detection Benchmark
OC-NN	One-Class Neural Networks
OCSVM	One-Class Support Vector Machine
ODD	Outlier Detection DataSets
OS	Operating System
SKAB	Skoltech Anomaly Benchmark
SVM	Support Vector Machine
UCR	University of California, Riverside

Part I

Introduction and background

Chapter 1

Introduction

1.1 Motivation

"Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world. — Atul Butte"

World started moving toward digitalization after the invention of digital computer and birth of the World Wide Web (WWW). In the last decade, a large percentage of the human population connected with the internet for the first time, and internet-connected devices have seen a remarkable rise. Global IP traffic will increase threefold between 2017-2022, and the number of devices connected to the internet will be three times the human population by 2022 [21]. This trend has affected all the industries, and today almost every organization is moving toward digitalization. Each organization generates an enormous amount of data, which can be helpful in decision-making. This data is being collected with the help of various tools. Sensors have been a vital tool for collecting data. Different industries use sensors to monitor their infrastructure, e.g., in manufacturing industry temperature, vibration of manufacturing machines.

The introduction of new technologies gives companies a competitive advantage over their peers. These also help to reduce costs and to improve the quality of products. In some cases, the entire market is changed into something different with the introduction of new technologies. Keeping up with changes and improving has become even more critical to remain competitive. Organizations need to collect information about all their crucial tasks and processes to move in the right direction. With the increasing size of organization or complication of tasks, data generated and collected also increase in size. Often it becomes difficult to monitor everything through human eyes. A large part of generated data from various sources is not utilized for analysis. This data can be used for finding meaningful patterns in various fields. Cheap sensors and advanced machine learning techniques have made it easier for both small and big organizations. Finding anomalies in data is an important task that is often helpful in making processes run smoothly.

A data point significantly different from other data is called anomaly and it is of special interest. Information about these anomalies plays important role in decision making in various fields such as Finance [9], Health [66, 83],

Petroleum [50] and many more. With the increase in the amount of data available, complexity also increases, making it impossible to monitor systems through human observers or simplistic methods, e.g., threshold. Machine learning algorithms are used for monitoring and finding anomalies in large datasets. The performance of these algorithms varies based on different conditions. The number of algorithms presented as solutions to anomaly detection problem is already high and increasing.

Consequently, choosing the best machine learning algorithm for anomaly detection requires much effort and some time impossible. The solution to this problem requires a good benchmark for comparing the performance of machine learning algorithms for anomaly detection tasks. There are only a few benchmarks available with a focus on anomaly detection. These do not provide all the information required to choose the anomaly detection algorithm. Therefore there is need for a better anomaly detection benchmark.

1.2 Research question

Machine Learning is becoming increasingly popular for solving complex problems in many domains. Among all those problems, one is anomaly detection in numerical data. Many algorithms and commercial tools are available for finding anomalies among normal data. Their performance varies depending on many factors such as data quality, availability of training data, nature of the anomaly. The process of comparing all algorithms based on different attributes requires much work in the absence of a good benchmark. The lack of benchmarks focusing on anomaly detection makes choosing an appropriate algorithm for an anomaly detection task quite tricky. Existing benchmarks for anomaly detection suffer from many deficiencies and problems.

Considering the problem stated above, we have formulated following research questions.

- **Q1:** What are the requirements for a good anomaly detection benchmark?
- **Q2:** Is there any anomaly detection algorithm that outperforms all the others on numerical datasets in terms of accuracy and computational complexity?
- **Q3:** What is the impact of dataset size on the computational complexity of anomaly detection algorithms?

1.3 Objectives

Following are the objectives specified for this thesis based on our research question.

Objective 1: Find the relevant tools and algorithms for anomaly detection and review them based on their characteristics.

Objective 2: Analyze existing benchmarking tools relevant for comparing the performance of anomaly detection algorithms and identify their shortcomings.

Objective 3: Collect a variety of labeled public numerical datasets with anomalies. Analyze, format, and improve the quality of datasets for further use in benchmarking.

Objective 4: Identify shortcomings in existing anomaly detection benchmarks and define requirements for a better benchmark.

Objective 5: Design and develop a new benchmark, which fulfills the defined requirements.

Objective 6: Identify and integrate popular anomaly detection algorithms for numerical data in the newly developed benchmark.

Objective 7: Perform benchmarking experiments with different settings. Utilize visualizing techniques to analyze and interpret the results generated by the benchmark.

Objective 8: Conclude from results of benchmark and provide general recommendations about use of anomaly detection algorithms.

1.4 Research methodology and work plan

This thesis aims to evaluate anomaly detection algorithms and develop a concrete benchmark for evaluation. Our research is quantitative applied research and slightly modified version of technology research [76] methodology. Modified version is used to fit it in the context of benchmarking. Our research is divided into three stages. We repeat these steps if our requirements are not met in last step.

Problem formulation: This step includes understanding the domain and defining the research problem.

Propose solution: This step includes proposing a solution for the problem we formulated in the previous step.

Experimentation and evaluation: This step consists of experimentation and evaluation of the proposed solution.

1.5 Thesis structure

This thesis consist of three parts, introduction and background, Open Anomaly Detection Benchmark (OADB), evaluation. It is also organized into seven chapters. Following is a brief description of each chapter.

Introduction (Chapter 1): In this chapter, the motivation for this research, research question, thesis objectives and thesis structure are outlined.

Background (Chapter 2): This chapter reviews literature relevant to anomaly detection. It also explains how anomaly detection is vital for the scientific community and people from different fields of life. It also sheds light on challenges faced in building tools and algorithms for solving the problem.

Machine learning algorithms for anomaly detection (Chapter 3): This chapter sheds light on popularly used Machine Learning algorithms for anomaly detection and classification of these algorithms.

State of the art anomaly detection benchmarks (Chapter 4): This chapter includes an analysis of existing benchmarks for evaluating and comparing machine learning algorithms for anomaly detection. This chapter also identifies issues with existing benchmarks and summarizes publicly available numerical datasets with anomalies.

Design and Implementation (Chapter 5): This chapter explains the design and implementation details of the Open Anomaly Detection Benchmark (OADB). Requirements for OADB are also outlined.

Evaluation (Chapter 6): This chapter evaluates Open Anomaly Detection Benchmark (OADB) based on defined requirements and illustrates results generated by the benchmark.

Conclusion and Future Work (Chapter 7): This chapter presents the conclusion of this study. It outlines various directions for research work to expand the work done in this thesis. It also defines the scope and limitation of this study.

Chapter 2

Background

This chapter provides a brief knowledge about artificial intelligence and its subfields. Anomaly, its different types, and steps involved in finding anomalies using machine learning are explained. This chapter sheds light on terms that are essential to understanding this thesis.

2.1 Artificial Intelligence

Artificial Intelligence is defined by most people as a system that can do the tasks requiring human intelligence, e.g., writing books, playing sports, and decision making in different traffic situations. AI researchers community has taken inspiration from human intelligence and brain structure to develop software systems with similar intelligence as humans. However, there is no consensus on a single AI definition among researchers yet, which can be used by everyone [82]. A good working definition is given by Kaplan et al. [39]:

"*a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation*"

AI is being used by many companies for developing new products, services and also improving existing ones. Some of them are used almost every day by millions of users, e.g., recognizing faces in pictures by Facebook to tag them automatically, and suggesting routes by google maps with less probability of braking during the drive [1]. AI, based on its abilities, can be categorized into three levels such as Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI), Artificial Super Intelligence (ASI) [39]. Artificial Narrow Intelligence is domain-specific, cannot perform other than a few tasks autonomously. However, it outperforms humans in doing the job for which it is designed and trained. At the Artificial General Intelligence level, it can perform tasks independently of human assistance in multiple domains. Artificial Super Intelligence refers to the level of intelligence that outperforms humans in almost all disciplines. Currently, most AI applications are at a narrow intelligence level as they are more focused on one single task. It has not reached human-like intelligence yet and is quite far from the Artificial Super Intelligence level. AI is used to solve many domain-specific problems currently, as stated in the examples above. There is a growing focus on developing artificial intelligence with abilities spanning multiple domains and surpassing the Artificial Narrow Intelligence level.

AI is a broad field and combination of multiple disciplines from statistics to biology. AI has subfields machine learning and deep learning.

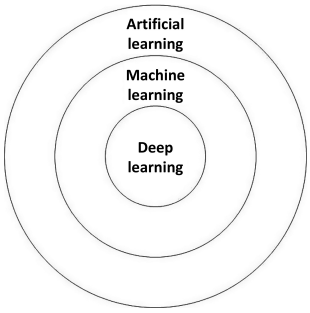


Figure 2.1: Overlap of artificial intelligence, machine learning and deep learning [38]

2.1.1 Machine learning

Machine learning has a narrow focus compared to AI as it does not give machines general intelligence. ML algorithms learn from data and find patterns in it. There has been a lot of research on ML, and a vital milestone in ML development was the first algorithm developed by Samuel in 1959 [68], to play checkers game and get better with experience. ML allows creating computer systems to do tasks that they are not explicitly programmed to do and get better at doing these tasks with experience. Different researchers have used different wording to define ML but it can be defined broadly as [53]

"*computational methods using experience to improve or to make accurate predictions*"

Most problems that ML is solving can be divided into two types, regression and classification. Regression is a problem type where the algorithm has to produce result as a real number. For example, predicting average temperature for the next 24 hours based on historical data. Classification is a problem where an algorithm classifies data instances in a dataset into different categories or classes. A simple example of classification problem can be detection of various animals in pictures. Each animal should be classified in its own class.

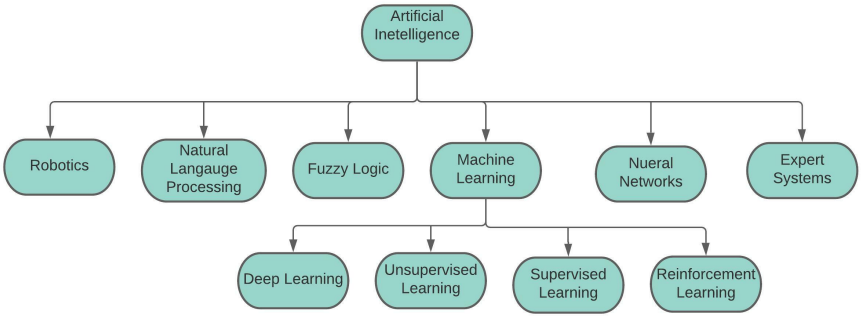


Figure 2.2: Artificial Intelligence and its subfields

Figure 2.2 shows classification of AI and Machine learning branches. Many of these branches overlap with each other.

2.1.2 Deep learning

Deep learning is a subfield of machine learning. Deep learning takes inspiration from the human brain and tries to replicate its structure in a computer system. Neurons are basic building blocks of the human brain and these perform most of the processing of information for us. Deep learning networks use several layers of artificial neurons[38]. It has three types of layers. The input layer represents input data. Hidden layers process and transform data and pass the results generated by a network of artificial neurons to the output layer. Each artificial neuron in hidden layers receives data from the neurons of the previous layer and passes it to neurons in the next layer after processing. The output layer represents the result produced by the neural network. These networks of artificial neurons are also called Artificial Neural Network (ANN). Figure 2.3 shows the basic structure of ANN and how artificial neurons interact with each other.

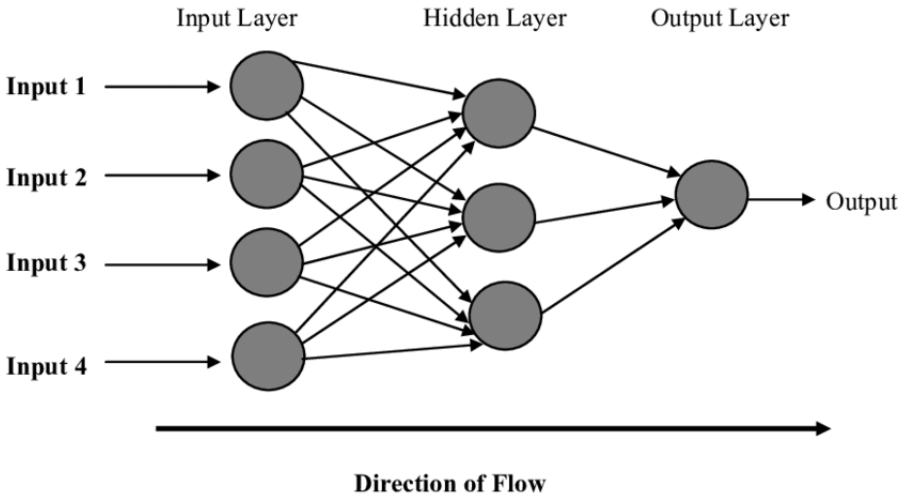


Figure 2.3: Example of a simple Artificial Neural Network (ANN) structure and interaction of various layers with each other [67]

Deep learning requires much more data to train than traditional machine learning algorithms to perform well, also requires special hardware such as GPUs to perform matrix calculations faster. Compared to traditional ML algorithms, one advantage is that there is no need for feature selection. Second, it performs well in the absence of labeled data [84]. Deep learning has made fantastic progress in Natural language processing, object recognition etc. in recent years.

2.2 Types of Machine learning

2.2.1 Unsupervised learning

Unsupervised learning algorithms classify unlabeled data into different clusters or groups based on similarities among them. All elements in one cluster have more similarity to each other and differences from elements of other clusters [75]. These algorithms do not require labeled training data for them to work. K-means is an unsupervised learning algorithm that is very

simple and widely used. Figure 2.4 shows a simple example of clustering done by K-means.

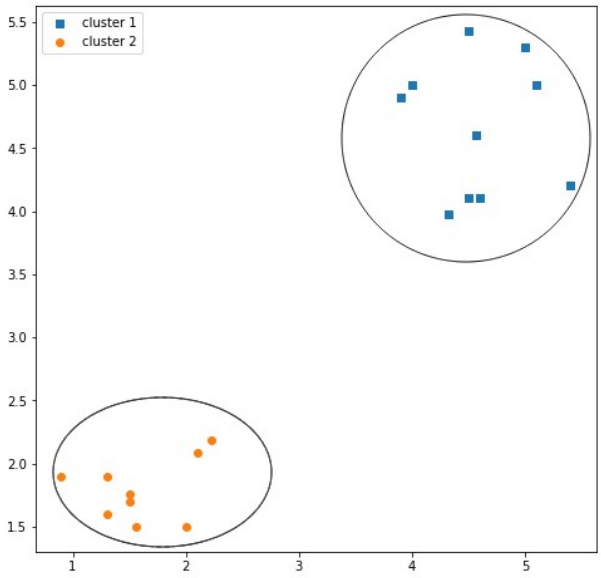


Figure 2.4: Example of data instances classified into two clusters by K-means algorithm

2.2.2 Supervised learning

Supervised learning algorithms require labeled data in contrast to unsupervised learning algorithms for training of model. Available data is divided into training and test data. The algorithm train on labeled data; for each input X there is output Y. Models fit to data during the training phase. The fitted model is then used to find output for test data inputs. Supervised learning algorithms help solve both classification and regression problems [75]. Supervised learning algorithms are prone to over-fitting the model to training data. Over-fitting leads to poor performance of algorithms on test data. Lack of labeled training data is a challenge in utilizing supervised learning.

2.2.3 Reinforcement learning

Reinforcement learning algorithms learn by doing different actions in an environment and learning from the output of these actions. Learning system acts as an agent which is rewarded and penalized based on its action in the environment. A learning agent, model of the environment and rewarding function are major components of a reinforcement learning algorithm. Goal of learning agent is to maximize rewards throughout its life span. Model of the environment should capture important features of a problem and should change depending on agent actions. Reward function defines what actions by the agent are good or bad in a given environment. Agent has to balance exploration and exploitation to try new actions in search of more rewarding actions and repeat previously done rewarding actions [78]. Reinforcement learning tries to replicate the natural learning process of a newborn baby, considering it as a learning agent.

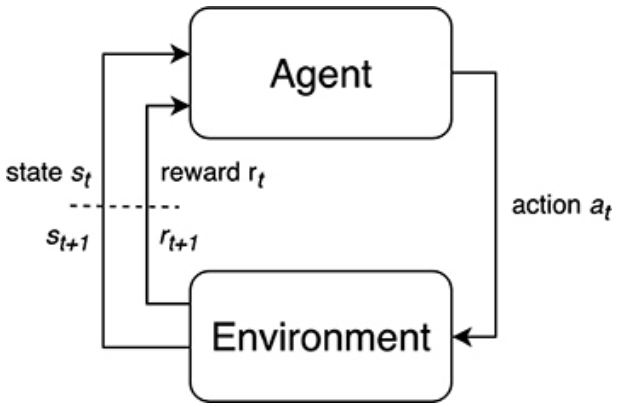


Figure 2.5: Interaction of learning agent with the surrounding environment in reinforcement learning [30]

In Figure 2.5 action a_t is action done by agent. In response to action, environment changes from state s_t to action s_{t+1} and gives reward r_t. Based on new state and reward of previous action, agent learns and performs next action. This loop continues as agent get better at performing most rewarding actions.

2.3 Anomaly and its types

Braei defined anomaly [15] as

" An anomaly is an observation or a sequence of observations which deviates remarkably from the general distribution of data. The set of the anomalies form a very small part of the dataset "

Anomalies have always been of particular interest in different fields, e.g., finance, gaming, medical etc., due to the significance of finding unusual behavior of a system. One should differentiate between noise and anomalies. Noise in data can be introduced due to reasons, e.g., seasonality, measurement tool malfunction that hinders our ability to find anomalies.

These anomalies can be positive or negative but have a high significance in all contexts. Anomalies can represent a high temperature than usual, high vibration etc. Anomalies detected at the right time can be quite helpful to avoid loss and potential issues [50]. For example, machines used in manufacturing industries are becoming more and more complex. Factories are using sensors to monitor the environment in the factories and also the performance of manufacturing equipment. Due to the massive amount of data coming from sensor data streams, humans cannot monitor and evaluate data to find meaningful patterns. Anomaly detection techniques can consume data from continuous data streams from sensors to detect anomalies and prevent any disaster in factories. It will increase the reliability of the diverse type of equipment and their lifetime. This one particular example shows the significance of anomalies, but this applies in almost all the fields.

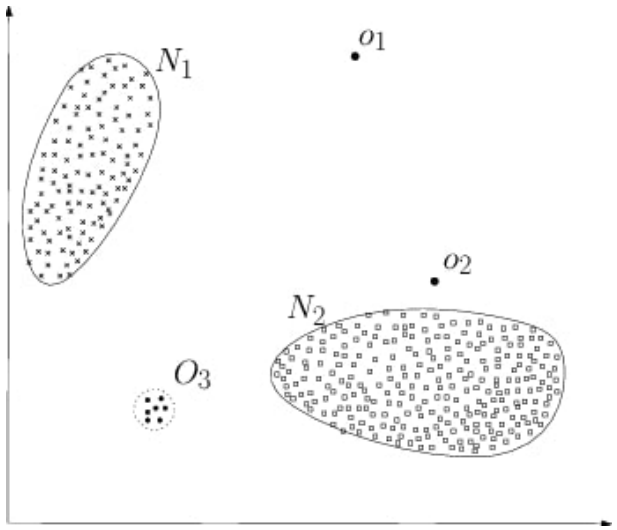


Figure 2.6: Illustration of anomalies in two-dimensional data [20]

In Figure 2.6 N_1 and N_2 represent values for normal observations and O_1,O_2 and O_3 are anomalies in two-dimensional space. Anomalies can be categorized mainly into three types [15].

Point anomalies: When a single data point varies significantly from the remaining data points in a dataset, it is a point anomaly as only one point shows abnormal behavior. For example, a sudden drop in car speed in temporal data for only one measurement through a journey represents a point anomaly.

Contextual Anomaly: A data point is normal in one context but can be labeled as an anomaly in another context, its contextual anomaly. For example, 5 centimeter snow depth in February is normal but same amount of snow in July in Oslo, Norway is an anomaly.

Aggregate/Collective Anomalies: When a set of data points deviates from normal pattern, not a single data point, this set of data points represent an aggregate anomaly. For example, 100 login API calls per second to a server from a single user are anomalous. However, a single API call is not an anomaly.

2.4 Process of finding anomaly

The usefulness of information about anomaly depends on when it is found and the type of problem. Sometimes it is helpful to detect anomaly immediately as it happens, but in some cases, it should be predicted before its occurrence. Process for finding anomaly consist of multiple steps from problem definition to detecting anomaly and performing appropriate measures to mitigate its effect.

2.4.1 Defining the problem

It is a process of figuring out what the problem is and how this problem can be solved. For example, small factories have a problem with machine failures, resulting in production delays and monetary loss. This problem can be solved by detecting the abnormal behavior of machines through continuously

measuring important data about machine status and preventing machine failure by intervention. It requires domain knowledge and understanding of the problem, knowledge about factors affecting the system. One has to perform multiple steps to understand the problem better and know objectives. Some of the steps and questions are:

- Identify how and what kind of data can be collected about this specific problem. Data will be available as data files after collection or continuous data streams coming from sensors.
- Identify elements that can affect data collection process or may introduce bias.
- Identify the variable which we want to find and discuss it with all stakeholders.
- Identify the possible causes of data noise and domain-specific characteristics of data, i.e., seasonality etc.
- Figure out when we should be able to detect the anomaly and when it is too late?

2.4.2 Data preprocessing

Data plays a vital role for any machine learning algorithm to perform better. With better quality and valuable data covering all aspects of the problem, there are more chances to get desired results. Most data available is not in a form to be used directly for ML algorithms. It needs to be prepossessed to make it suitable for training a ML algorithm to perform well. Data quality is reduced for various reasons, including issues in measuring devices, wrong understanding of the problem, and many more. Depending on the type of data and problem to be solved, it has to go through multiple steps. Some of the steps for improving quality of data are following.

Handling missing values

Dataset with missing values can seriously affect the performance of algorithms and introduce bias in results. Missing values in data can occur due to multiple reasons e.g., problems in data recording sensor, and incorrect measurement. Issues due to, missing values can be mainly avoided either by removing missing values or replacing missing values with estimated values. Multiple methods are used to estimate missing values such as mean substitution and K-means Clustering Imputation (KMI) [28]. Mean substitution replaces missing values with the mean of normal values in a dataset. This method will not be appropriate in some cases, i.e., when many values are missing. Different techniques are used for various problems and datasets.

Removing noise

Data collected from real-world usually contain irrelevant and distorted data called noise. Supervised algorithms are most affected by data noise. ML algorithms trained with noisy data can result in reduced accuracy. Data noise can be categorized into two types, class noise and attribute noise. Class noise represents the wrong labeling of data. Class noise can be introduced due to bugs in labeling software or entry error during manual labeling etc. Attribute noise arises by one or more corrupt data points [28].

Feature selection

Often a lot of data is collected but using all data for machine learning is not feasible. A higher number of features result in higher complexity of data, require high computational power and reduce the speed of ML algorithms. To solve this problem, we select important features and remove features that are dependent on other features. The semantics of data should remain intact. Feature selection helps in increasing accuracy and learning efficiency [28].

Feature scaling

Data sets contain many features, but not all are measured with the same scale. For example, if we have collected data temperature and vibration from the factory using sensors. Temperature and vibration measurement values will have considerable differences. Significant differences among features in a dataset are not ideal for the accuracy of the algorithm. Features with high magnitude will have more impact on calculations can result in biased results. The process of fixing removing significant differences among values is called feature scaling. Two techniques are commonly used for feature scaling. The min-max normalization converts all values between [0, 1] using the following formula.

$$z' = \frac{z - minimum.value}{maximum.values - minimum.value} \tag{2.1}$$

Z-score normalization use mean and standard deviation to convert values in range [0, 1] using the following formula [6].

$$z' = \frac{z - mean}{standard.deviation} \tag{2.2}$$

z' represents the new value after normalization and z represents the original value.

2.4.3 Anomaly detection

Anomalies can be detected in different ways depending on available data and the problem to be solved. A common way of detecting an anomaly is by visualization. It is done by creating dashboards and showing graphs of crucial features. Humans can detect unusual behavior and perform appropriate actions to avoid any failure/loss. As the number of features and complexity of data increases, it becomes difficult to visualize everything correctly and for the human brain to comprehend it. Sometimes thresholds are used and whenever a value increases from a certain threshold, an alert is sent to the relevant entity. Thresholds are static and are set based on the assumptions of an expert. They do not fluctuate based on changing situations and are prone to mistakes by experts who have set threshold values, resulting in false alerts. For example, an e-commerce website for buying gifts receives 50,000 orders per month and a threshold set for 70,000 to detect an unusual number of orders. Nevertheless, it may get 100,000 visitors during Christmas days as more people are shopping for gifts during this time of year. It can trigger a false alert.

The increased complexity of data requires a more complex solution for monitoring and detecting anomalies. In this regard, ML algorithms are pretty helpful because they consider all the features, labeled anomalies and learn from data overall. Data is divided into two parts training data and test data. Training

data is used for learning of model and test data is used to test the accuracy. Broadly ML techniques for anomaly can be divided into types supervised anomaly detection and unsupervised anomaly detection.

Supervised anomaly detection: Supervised algorithms are used when we have data that has labeled anomalies. These algorithms usually perform better than unsupervised and semi-supervised algorithms as they have more information available regarding anomalies. Anomalies are rare and undesirable events. It makes it challenging to gather a properly labeled dataset. Due to these challenges, these algorithms cannot be applied in many cases. Some of the supervised anomaly detection algorithms are Support Vector Machine (SVM), Decision trees, Bayesian Network, Supervised neural network, K-nearest neighbor [58].

Unsupervised anomaly detection: Unsupervised algorithms do not require labeled data and this makes them easier to use for anomaly detection. Unsupervised algorithms used for anomaly detection are K-means, One-Class Support Vector Machine (OCSVM) etc.

2.4.4 Anomaly prediction

Detecting anomalies before they occur is called anomaly prediction. ML techniques are used to predict next trends and events based on previous data [49]. Predictive maintenance for different systems has gained importance in the last decade because the cost attached to systems maintenance increased. Initially, most industries used time-based maintenance, but now with advances in Machine Learning and sensors, we can use data gathered from sensors to monitor the performance of systems and analyze unusual behavior. It can help in the early detection of problems. Predictive maintenance is the preferred maintenance method in 89% of cases, compared to time-based maintenance, which is preferred in only 11% of cases. Integrating the predictive maintenance techniques with the latest sensor technologies enable plants to avoid unnecessary equipment replacement, save costs, and improve process safety, availability, and efficiency [34]. Anomaly prediction is similarly helpful in various fields. Time horizon and explainability are two important factors for evaluating anomaly prediction technique [31].

Time horizon: These techniques aim to maximize the time duration between the prediction of anomaly and actual occurrence. Time duration before occurrence provides the opportunity to validate the alert for anomaly.

Explainability: If we act on all the anomaly predictions, then false positives can do the opposite effect. It can result in delays, increased costs and similar problems. It signifies the importance of explaining anomaly predictions to understand them and perform appropriate measures to avoid failure.

2.5 Challenges of anomaly detection

One faces many challenges during the process of anomaly detection. Some of them are following.

– The lack of enough labeled data is a significant challenge for anomaly

detection. In most cases, one does not have any or enough labeled data for the training of algorithms. Consequently, testing the performance of an algorithm for an actual problem is nearly impossible.

- The rarity of anomalies makes it challenging to detect them. It affects the pre-deployment phase as we do not have enough information about anomalies. It is leading to issues during the usage of anomaly detection solutions.
- Noisy data can pose a challenge in detecting anomalies and reducing the accuracy of the algorithm.
- False positive alarms are another challenge related to anomaly detection. They result in unnecessary disruptions. The algorithm used for anomaly detection should accommodate changes in data values to avoid false alarms.
- Anomalies are of multiple types broadly categorized as point anomaly, aggregate anomaly and contextual anomaly. It increases the complexity of anomaly detection.
- Anomalies change based on the size of data observation window. A data point may seem like an anomaly in the shorter window, but it can be a normal data point with a complete data view. Anomalies can be local and global. As shown in Figure 2.7 red dots show possible local anomalies and green dots show an example of global anomaly.
- Anomaly detection is time-critical in many scenarios. Its usefulness depends on the time difference between occurrence and detection. Often information is needed about anomalies before occurrence. It add time constraint for anomaly detection algorithms, making immediate anomaly detection a requirement.

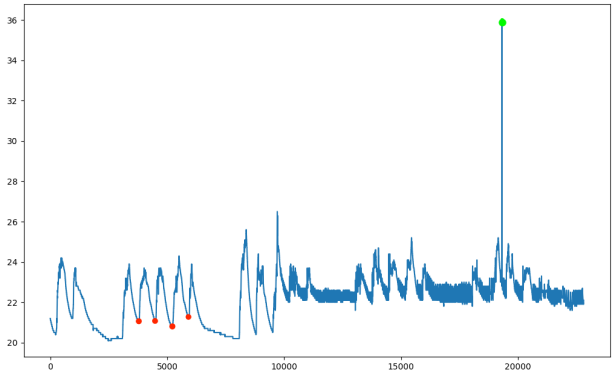


Figure 2.7: Examples of local and global anomalies

2.6 Applications of anomaly detection algorithms

Anomaly detection has applications in various fields where unseen events should be monitored and have adverse consequences if they go undetected.

Finance: Finding fraudulent activity in online financial transactions is crucial in the current banking system to make it more secure and trusted. ML techniques are applied to find anomalies among millions of transactions done every day. These techniques are also helpful in detecting issues in financial statements and anti-money laundering efforts [9].

Cyber security: Anomaly detection algorithms are applied for intrusion detection in a network, detecting unusual behavior by a malicious user, and finding a virus file in a data storage [84].

Manufacturing: Ensuring that each product produced in the factory is perfect is done by employing anomaly detection techniques to detect faults in each product. Another exciting application is monitoring the health of manufacturing machines in factories and avoiding halt of production lines [43].

City administration: Cities often deploy advanced equipment to gather data from air quality to the number of cars passing through a particular junction. This data can be used to find traffic congestion and to avoid it by using anomaly detection techniques [25]. Health care data collected from different facilities can also be used for a broader analysis of health care services, prediction of any overcapacity, and preventing the spread of any disease in society.

Medical: In the medical field, anomaly detection techniques are widely used from diagnosis of diseases from using X-ray images to the prediction of diseases and anomalies in heart rate data [4, 66].

Agriculture: Advanced equipment usage has allowed farmers to use anomaly detection algorithms on data collected from farms to tackle diseases and other problems in farming. A good example is detecting disease in different crops from images and weed detection, which can hinder the yield of crops. These techniques essentially provide food security for masses [46].

2.7 Benchmarking

2.7.1 What is benchmarking

Benchmarking is defined by Anand et al. [8] as

"search for the best industry practices which will lead to exceptional performance through the implementation of these best practices."

Every organization has a set of tasks that need to be done to run the organization smoothly. These tasks keep on changing with passing time and quite often new approach is introduced to do these tasks. Ensuring that the newly adopted approach to perform the task is better than the previous one or measuring how much it is better is essential for an organization. It allows them to compete with others and get better continuously. It is one of many cases where benchmarking fulfills the needs of an organization.

2.7.2 Benchmarking in machine learning

Benchmarking in machine learning refers to comparing machine learning algorithms based on different metrics and finding out which performs better un-

der given circumstances. Machine learning benchmark is a software tool used for benchmarking. There are some general-purpose benchmarks, although most benchmarks have a certain focus. MLPerf [51] and PMLB [60] are examples of general-purpose benchmarks.

Metrics: ML benchmarks are developed to compare algorithms based on metrics such as computing power requirement, accuracy, compatibility etc. NAB [44] and SKAB [41] are examples of benchmarks which use accuracy of algorithms for benchmarking.

Data: ML algorithms are run on particular data for evaluating their performance. These datasets are collected from various sources. Datasets can be broadly divided into three types real-world, synthetic and toy data.
Real-world data is collected from the real-world through different devices such as sensors measuring temperature in a factory. Synthetic data is generated artificially to simulate a real-world situation. Toy data is not best for benchmarking but used as a readily available dataset to experiment.

Focus of benchmark: As machine learning continues to grow as a field, it has become challenging to make a benchmark that can integrate and evaluate all ML algorithms. One reason is that often ML algorithm is developed to solve a specific problem. Another reason is that a fundamentally different approach is required depending on the nature of the problem. It results in benchmarks focusing on a specific application of ML algorithms. For example, NAB [44] has focused on anomaly detection on time series, DeepBench [24] has focused on deep learning and computing power needed by algorithms.

2.7.3 Architecture of a benchmark

Machine learning benchmark usually consists of three components data, algorithms and benchmark software. Software tool train and test algorithms on datasets. During this process, all the important metrics are measured. Later, measured data is visualized to compare the performance of algorithms. Figure 2.8 show example of common components in most of ML benchmarks and how they interact.

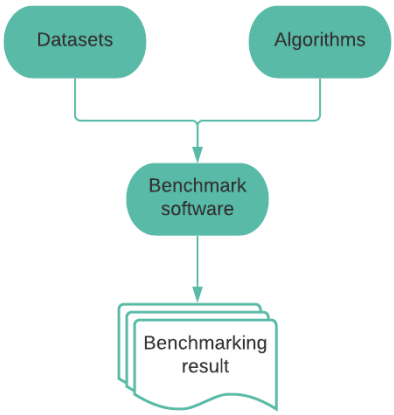


Figure 2.8: Components of a machine learning benchmark

2.7.4 Why ML benchmarks?

Machine learning as a field is growing very fast, new techniques and algorithms are introduced constantly. Availability of good tools to evaluate ML algorithms is important for both researchers and practitioners. Benchmark accelerate the process for evaluating of an algorithm and selection of an algorithm for a certain problem. They also provide a way to standardize the algorithms on basis of performance and can become a reference point.

Part II

Problem analysis

Chapter 3

Machine learning algorithms for anomaly detection

This chapter provides an overview of popular approaches used for anomaly detection and ML algorithms for anomaly detection. Along with a brief introduction to commercial tools available for anomaly detection, a comparison of deep learning and machine is also part of the chapter.

3.1 Anomaly detection approaches

Numerous algorithms are available for anomaly detection and each one has a different implementation. These differences can be of two types. Either they are completely different due to distinct underlying principles used or variants based on the same approach for detecting an anomaly. These approaches work based on some assumptions related to anomalies. Approaches in ML algorithms for anomaly detection can be categorized into distance-based, clustering-based and model-based approach [52].

3.1.1 Distance-based approach

This approach measures and utilizes distance between all data points in n-dimensional space to identify anomalies among normal data. Number n represents dimensions in a dataset. The simplest solution is to calculate the distance of each data point to all other, sum them and use it to detect anomalies. High dimensionality has negative effect on distance based algorithms called the curse of dimensionality. With increase in features, size of feature space is also increased significantly. This results in sparse distribution of data in feature space and distance measure is affected [3]. The distance and similarity measurement methods play a critical role in the algorithm's accuracy.

Distance calculation method: Euclidean, Minkowski, and Mahalanobis [52] are various methods utilized by ML algorithms to measure the distance between two data points. Euclidean distance is the simplest method as shown in equation 3.1, d(x, y) represents the distance between two points x and y in n-dimensional space.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

Similarity measurement method: Similarity measurement answers the question, if two data points are similar or not? Calculating distance to all data points, distance to k-nearest neighbors, median or average distance to k-nearest neighbors are various approaches to measure similarity.

Assumption: It is assumed that the anomaly data instance has a relatively long distance from other data instances.

3.1.2 Clustering-based approach

In this approach, data points in multi-dimensional space are grouped based on density and assigned cluster labels. The rule for assigning data points to a cluster differs among various algorithms. It differs in the following ways; a point can be part of multiple clusters, clusters can overlap, or they are assigned non-binary values in 0 to 1 range. Figure 2.6 shows data visualized in 2-dimensional space with clusters.

Assumption: It is assumed that data points away from cluster centers are more likely anomalous than data points closer to the cluster center.

3.1.3 Model-based approach

The model-based approach attempts to create an ML model that can represent the underlying characteristics of a system. The model tries to understand the relationship among data features. It is trained using available data to learn from it. Linear regression is an example of an algorithm with model-based approach.

Assumption: It is assumed that a relationship among different data features exists and affects the target variable.

3.2 Popular ML anomaly detection algorithms

3.2.1 Supervised algorithms

K-Nearest Neighbors (KNN)

KNN is a supervised non-parametric algorithm that is simultaneously quite simple and effective for classification. All the instances from labeled data are populated in the feature space. K number of nearest neighbors in feature space are selected based on distance from new observation to find the class of new observation. The class to which most numbers of neighbors belong is assigned to a new observation. The value of K significantly impacts performance as low value results in over-fitting and very high value increases computational complexity. It has a short training time but requires high computation power and memory in test phase. It is due to the computation of distance to K number of observations in the dataset for finding nearest neighbors. Either value for K is optimized by experimenting with different values or special methods applied

for finding it [13]. Figure 3.1 demonstrate effect of K value on assignment of class to new instances. When K value is 2, a new instance is assigned class 2. With the increase of K value to 3, the new instance is assigned class 1.

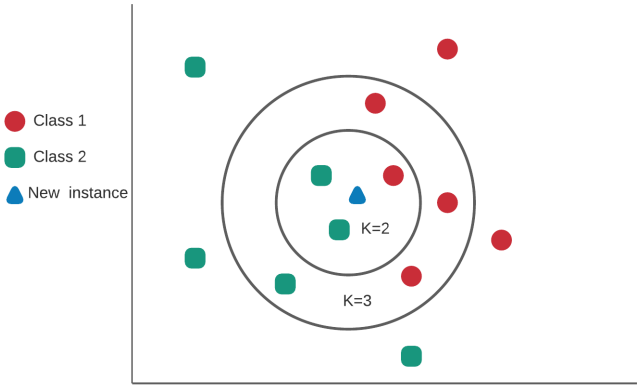


Figure 3.1: Effect of K value for KNN algorithm

Some KNN variants are quite effective for anomaly detection, e.g., Transductive Confidence Machines for K-Nearest Neighbors (TCM-KNN) [45] for intrusion detection, Mahalanobis-squared distance and one-class kNN based algorithm for health data monitoring for anomalies [69], and Weighted KNN for DoS attack detection [77].

Bayesian Network

A Bayesian network is a directed acyclic graph representing probabilistic relationships among a set of variables of interest. All the variables are conditionally independent of each other. The graph structure and variable probabilities are learned from training data. Probabilistic inference is used to calculate the probability for variables of interest. It is well suited for the problem where variables are interdependent and helps in learning about causal relationships [35].

Bayesian networks are effective in detection of anomalies in various fields, e.g., detection of disease outbreaks [83], network intrusion detection [40], etc.

Decision Tree

Decision trees are commonly used for the decision-making process. Classification Tree, Regression Tree, and Decision Forest Tree are multiple variants of decision trees used to solve different types of problems. Decision tree internal nodes are different variables and branches connecting these nodes are tests performed on the value of variables. Leaf nodes represent categories or different possible outcomes. There is a danger of overfitting that is reduced by pruning of decision tree [57]. Decision Trees are used for anomaly detection, also sometimes in conjunction with other algorithms such as Support Vector Machine [73]. Figure 3.2 shows a simple Decision Tree for the process of decision making for buying a car.

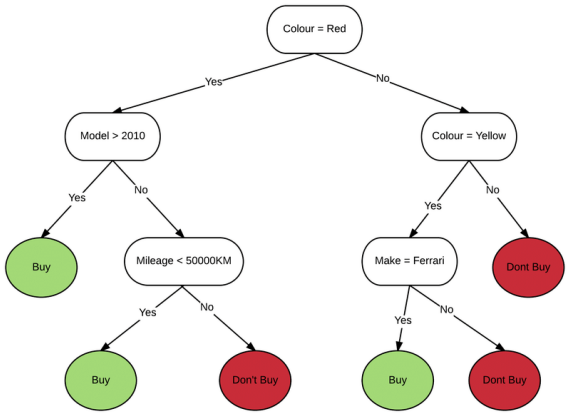


Figure 3.2: Simple decision tree showing process of buying car and how different factors impact the decision [54]

3.2.2 Unsupervised algorithms

K-means

K-means is an unsupervised non-parametric clustering algorithm. It tries to divide data points into K number of clusters. In the first step, clusters are assigned centroids randomly. The data points become part of a cluster of the nearest centroid. In the next step, center point of each cluster is considered new centroid. The last two steps are repeated until centroids do not change. The value of K significantly affects the final clusters. Currently, different methods are used for finding optimal K values, such as Elbow method, The Gap statistic algorithm, The Silhouette Coefficient Algorithm etc. [86]. In Figure 2.4, an example of K-means is shown for the two-dimensional dataset.
There are many improved versions of K-means algorithms; some of them are used for anomaly detection in different domains [48, 85].

One-class Support Vector Machine (OCSVM)

OCSVM is a particular type of Support Vector Machine (SVM) that is effective for anomaly detection. It creates a feature space with data points and finds an optimal hyperplane separating normal data points from anomalies. It maximizes the margin of hyperplane from data points of both classes. Sometimes no hyperplane can separate data points of two classes, so a particular possibility of error should be allowed[74]. OCSVM is commonly used for anomaly detection and has applications in many fields. The algorithm focuses on finding anomalies instead of finding normal behavior, as done in many other algorithms. Figure 3.3 show working of the hyperplane. OCSVM aims to find a hyperplane that can distinguish anomalies and normal data.

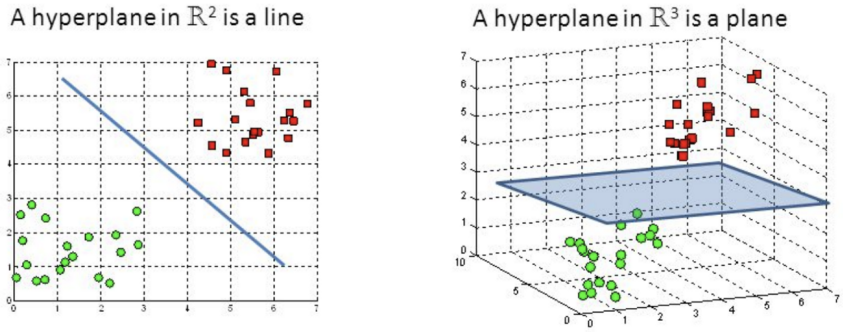


Figure 3.3: Hyperplane in 2-dimensional and 3-dimensional space [81]

Anomaly detection in telecommunication data [87], Fault detection in photovoltaic power plants [33] are two examples of OCSVM usage for anomaly detection among many other its applications.

Isolation Forest

Isolation forest assumes that anomalies are very few and have distinct features from normal data points. It is based on Isolation trees. A tree is partitioned until each instance is isolated or all items in the tree have the same value. A set of trees is built from data and referred to as isolation forest. Anomalies get isolated at the early stage of partitioning. Tree nodes with the shortest average path length are labeled as anomalies. No need to calculate distance among data instances makes isolation forest less computationally expensive compared to distance-based algorithms[47].

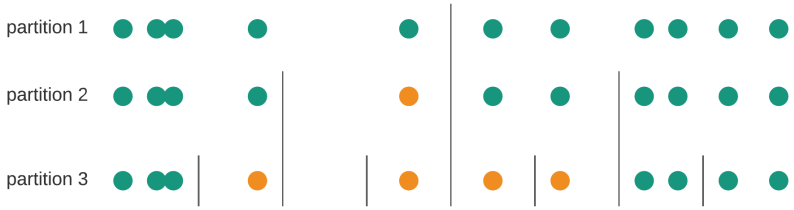


Figure 3.4: Isolation tree partitioning process on one-dimensional data

3.2.3 Deep learning algorithms

DeepAnt

DeepAnt is an unsupervised deep learning algorithm purposed especially to solve anomaly detection problem. It consists of two parts time-series predictor and anomaly detector. A time-series predictor is used to predict the next timestamp value based on data from a historical window of time. This predictor relies on Convolutional Neural Network (CNN) for prediction. The predicted value, along with the original corresponding value, is passed to the anomaly detector. Euclidean distance is calculated between the original value

and predicted value and is used as anomaly score. Higher distance means higher chances of this instance of being an anomaly. The algorithm performs well with smaller datasets and detection of different types of anomalies, i.e., point anomaly, contextual anomaly, etc., even with seasonality in data [56].

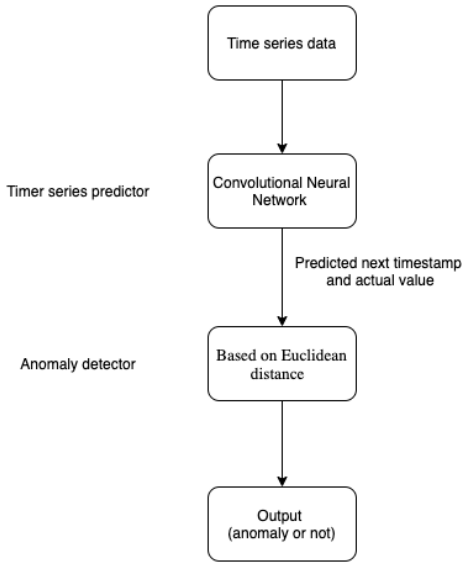
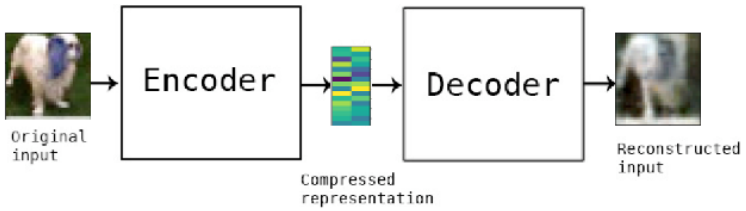


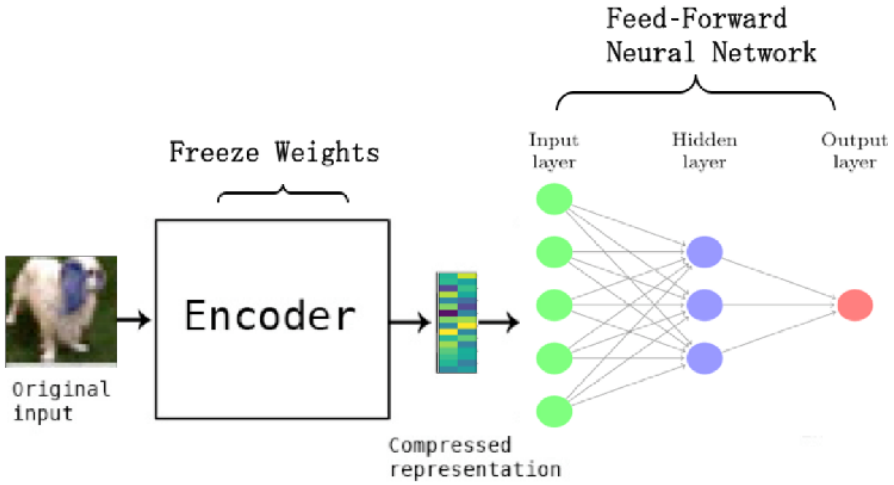
Figure 3.5: Working of DeepAnt algorithm for anomaly detection

One-class Neural Networks (OC-NN)

One-class Neural Networks (OC-NN) uses Autoencoder for feature extraction and provides this data as input for Feed-Forward Neural Network with Linear or Sigmoid activation function, one hidden layer and one output node. It is different from other algorithms which use Deep Learning for feature extraction, then feed data to traditional ML algorithms for anomaly detection because hidden layer data representation is customized for anomaly detection. It is helpful to find anomalies in complex high dimensional data [19]. Figure 3.6 represents simplified version of OC-NN.



(a) Autoencoder.



(b) One-class neural networks.

Figure 3.6: One-class Neural Network steps for anomaly detection [19]

3.3 Comparison of traditional machine learning and deep learning for anomaly detection

There has been much research and ongoing to improve anomaly detection techniques. However, there are still some unresolved issues due to complexities associated with anomalies. Rarity, anomalies with different characteristics, and unknown until occurrence are some complexities that make anomaly detection difficult. Deep learning successfully solves different ML problems in various fields and is also employed for anomaly detection. Deep learning algorithms for anomaly detection can broadly in three categories based on their usage.

One is feature extraction, where deep learning converts high dimensional data to low dimensional data while keeping the most semantic-rich features intact. This data can be further used for anomaly detection using desired techniques. Employing readily available pre-trained deep learning models makes the process quicker. Second is learning feature representation of normality, and the third is end-to-end anomaly score learning. Traditional ML algorithms struggle to perform optimally for anomaly detection for various reasons, including high dimensional data, lack of labeled data, noisy data, complex anomalies, and interdependent features. Due to better heterogeneous data and intricate relationship learning ability, deep learning algorithms can perform better in these situations. They can be used as anomaly explanation algorithms to figure out why an instance is marked as anomaly [62].

DL performance is dependent on the availability of data and requires large datasets for understanding it correctly. Their accuracy increases with an increase in dataset size, While traditional ML algorithms can perform better even with smaller datasets. Another difference is that deep learning algorithms consume much more computing power for training and take a long time. Deep Learning algorithms can take weeks for training, while traditional ML algorithms usually finish training in a few minutes to hours on the same data. On the other hand, deep learning takes a shorter time for testing [84].

3.4 Hybrid algorithms

In some cases, one type of ML algorithms cannot efficiently and accurately solve a particular problem. It can be due to multiple reasons, e.g., high dimensional data, noisy data, etc. Researchers combine different techniques such as deep learning for feature extraction and traditional algorithms for anomaly detection. Hybrid algorithms have performed better results in some cases. Algorithms produced by deep learning and traditional machine learning combination are called deep hybrid algorithms. Deep learning for feature selection help in reducing the number of features and results in better performance by traditional algorithms. However, deep learning does not take into consideration our goal for anomaly detection, which may result in removing essential features for the anomaly detection process [18]. Combining multiple ML algorithms to enhance performance is called ensemble learning. There are multiple strategies for combining algorithms. Average-based strategies for numerical outputs and vote-based strategies for classification are popular. In an average-based strategy, the average is calculated of output from all algorithms and the final output is based on it. In vote-based strategy, all algorithms vote for a specific output and output with a majority vote is selected result [89].

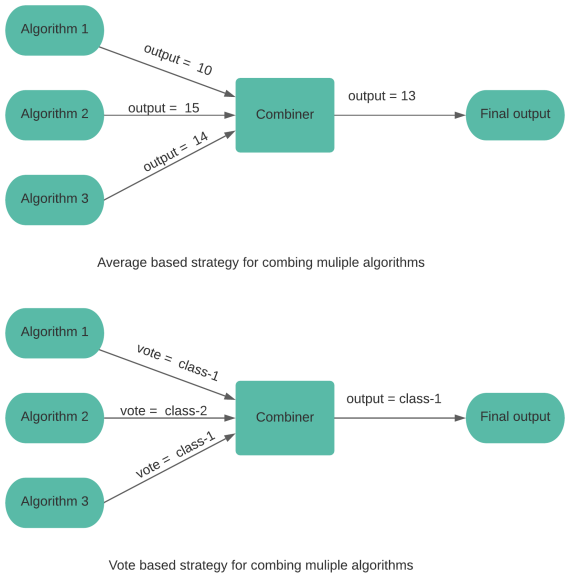


Figure 3.7: Ensemble learning strategies for combining algorithms

3.5 Commercial tools for anomaly detection

Aside from many open-source algorithms, companies have also worked on developing tools and algorithms for anomaly detection. A subset of them is open-source and available for free usage. The rest of them are products and sold by companies to their customers.
Seasonal Hybrid ESD (S-H-ESD) [72] by Twitter and Extensible Generic Anomaly Detection System (EGADS) [27] by Yahoo are examples of open source anomaly detection algorithms developed by companies. Anomaly detector [11] as part of Microsoft Cognitive Services and Anodot [10] are examples of paid commercial solutions for autonomous anomaly detection without any extra work. These claim to perform very well, but most of them are not benchmarked on public datasets.

3.6 Ways to categorize algorithms

3.6.1 Supervised vs Unsupervised learning

In supervised learning, algorithms learn from labeled data and based on this learning; predictions are made for new data instances. It is tough to collect or label data correctly in many situations, making supervised learning unsuitable. Unsupervised learning is helpful in these situations as labeled data is not required.

3.6.2 Eager vs Lazy learning

Eager learning algorithms work in two phases, the training and testing phase. In the training phase, algorithms learn from training data instances and create a model. While in the testing phase, test data instances are assigned labels. Lazy learning, aka Instance-based algorithm, usually do not have a clear separation between training and testing phases. It results in more computational and storage resources required because creating a model for each testing phase is computationally expensive. Despite being expensive, eager learning algorithms can perform better in some instances due to a better understanding of data complexities in comparison to eager learning algorithms [2].

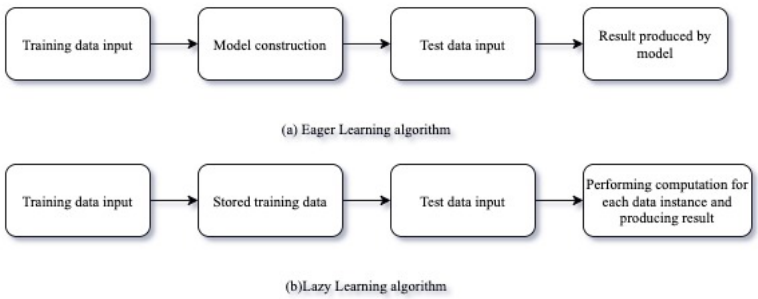


Figure 3.8: Lazy vs Eager learning algorithm steps

3.6.3 Parametric vs Non-Parametric learning

The number of parameters provided to the model is fixed and independent of data instances in parametric learning. In contrast, for non-parametric learning, the number of parameters can change based on data instances. In contrast to non-parametric models, parametric models make assumptions about data, thus making them helpful in cases where data for training is scarce. When enough data is available, one can use non-parametric models and avoid making strong assumptions [32].

Chapter 4

State-of-the-art anomaly detection benchmarks

An overview of public datasets suitable for evaluating anomaly detection algorithms and existing anomaly detection benchmarks is part of this chapter. This chapter also describes shortcomings in existing anomaly detection benchmarks.

4.1 Public datasets with labeled anomalies

4.1.1 Yahoo Labeled Anomaly Detection Dataset

This dataset consists of 67 real-world data files collected from Yahoo services and 300 artificially generated data files. All the anomalies are labeled and data can be downloaded and used for research by requesting the Yahoo team to access it. Dataset is widely used for benchmarking algorithms [65].

4.1.2 NAB Dataset

It has real-world and artificial data. Real-world data is collected from various sources such as online advertising data, AWS server metrics with features, e.g., CPU utilization, Disk read operations, Network data transfer, Traffic data, etc. Further meta-information about the dataset is in Section 4.2.1.

4.1.3 UCI KDD Archive

This dataset was released as part of Knowledge Discovery and Data Mining (KDD) Tools Competition 1999 for network intrusion detection. The goal for competitors was to detect cyber attacks among normal connections. The dataset contains 5 million network connections data which is collected over seven weeks. Each instance is labeled as normal or attack and has 41 features. It contains 24 attack types, including Denial of Service attack, User to Root attack, Remote to Local attack and Probing attack [12]. Tavallaee et al. [79] has created an improved dataset NSL-KDD [59] after removing redundant, duplicate data.

4.1.4 UCI Machine learning repository

It is a popular data repository among researchers to evaluate the ML algorithms, but most data is suited for classification problems. Some of it is used for the evaluation of anomaly detection algorithms after downsampling, but downsampling affects natural distribution anomalies [17]. It has 591 datasets donated by different researchers and organizations over a long period, as the first donation dates back to 1988. These datasets are diverse both in terms of subject area and type of data. For example, datasets contain information from health records to internet advertisements and types are text to images. A subset of datasets, approximately 30 datasets, are suitable for anomaly detection problems without downsampling or any other operation which can affect the semantics of data [26].

4.1.5 NASA valve dataset

NASA has collected this dataset from fuel control valves of rocket propulsion systems through sensors. It is time-series data with anomalies [14].

4.1.6 ADRepository datasets

This data repository consists of numerical, categorical and image datasets collected from various sources. Data is normalized for usage [61].

Dataset	Data size	Dimensionality
donors	619326	10
census	299285	500
fraud	284807	29
celeba	202599	39
backdoor	95329	196
campaign	41188	62
thyroid	7200	21

Table 4.1: Summary of numerical datasets in ADRepository

Dataset	Data size	Dimensionality	Anomaly class
bank	41188	10	yes
census	299285	33	50K+
AID362	4279	114	active
w7a	49749	300	yes
CMC	1473	8	child>10
APAS	12695	64	train
CelebA	202599	39	bald
Chess	28056	6	zero
AD	3279	1555	ad.
Solar-flare	1066	11	F
Probe	64759	6	attack
U2R	60821	6	attack
R10	12897	100	corn
CoverType	581012	44	cottonwood

Table 4.2: Summary of classification datasets in ADRepository

4.1.7 Outlier Detection DataSets (ODDS) library

ODDS library has a collection of 31 datasets from various sources as mentioned in Section 4.1.3 and UCI repository 4.1.4. Original datasets are processed to make them suitable for the anomaly detection task. Processing of datasets includes different methods such as downsampling, merging categories etc.

4.1.8 Summary of Datasets

Table 4.3 provides brief summary of public datasets for anomaly detection.

Repository	Datasets	Features	Subject Area	Data source	Anomaly types
Yahoo dataset	367	1	Online traffic	Real world, Artificial	Point, Aggregate
NAB	58	1	Diverse	Real world, Artificial	Point
UCI KDD-99	1	42	Online traffic	Artificial (Simulated)	Point, Aggregate
UCI ML repository	591	1-*	Diverse	Real world, Artificial	Point, Aggregate
ADRepository	21	6-1555	Diverse	Real world, Artificial	Point, Aggregate
Exathlon data	10	2283	Application traces	Real world	Point

Table 4.3: Summary of public datasets with labeled anomalies

4.2 Existing anomaly detection benchmarks

4.2.1 Numenta anomaly benchmark (NAB)

The Numenta Anomaly Benchmark (NAB) is developed to benchmark algorithms for online anomaly detection in time-series data. It has a unique scoring function that takes into account the early anomaly detection capability of the algorithm. The scoring function gives positive scores to the algorithm based on accuracy, early anomaly detection ability, fewer false positive alarms, real-time anomaly detection ability and adaptability to new datasets automatically. In figure. 4.1, example of scoring function shown that is used to determine how early an anomaly is detected [44]. New datasets and algorithms can be incorporated for benchmarking in NAB.

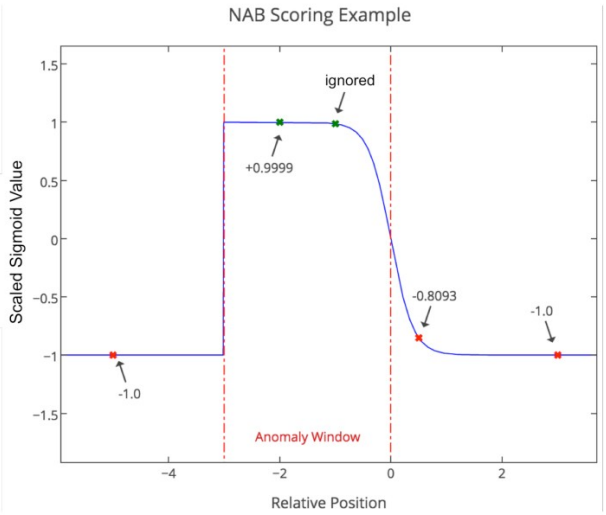


Figure 4.1: NAB scoring function used for assigning score to an algorithm for its ability to detect anomaly early [44]

Datasets

Data module of the benchmark consists of 58 data files collected from both real-world and artificially generated data. All data files contain univariate time series, each with 1000-22000 data instances, all hand-labeled.

Results

According to results, Hierarchical Temporal Memory (HTM) outperforms other algorithms, including Twitter ADVec, Etsy Skyline and KNN-CAD, with a 64.7 score out of a perfect score of 100.

Detector	Standard Profile	Reward Low FP	Reward Low FN
Perfect	100.0	100.0	100.0
Numenta HTM	70.5-69.7	62.6-61.7	75.2-74.2
CAD OSE	69.9	67.0	73.2
earthgecko Skyline	58.2	46.2	63.9
KNN CAD	58.0	43.4	64.8
Relative Entropy	54.6	47.6	58.8
Random Cut Forest	51.7	38.4	59.7
Twitter ADVec v1.0.0	47.1	33.6	53.5
Windowed Gaussian	39.6	20.9	47.4
Etsy Skyline	35.7	27.1	44.5
Bayesian Changepoint	17.7	3.2	32.2
EXPoSE	16.4	3.2	26.9
Random	11.0	1.2	19.5
Null	0.0	0.0	0.0

Table 4.4: Numenta anomaly benchmark results for accuracy of algorithms

4.2.2 Skoltech Anomaly Benchmark (SKAB)

The Skoltech Anomaly Benchmark (SKAB) [41] is developed to especially evaluate algorithms for their efficiency in the detection of point anomalies and change point in data. It has leader boards for the performance of different algorithms with Jupyter notebooks for reproducing results. SKAB has borrowed anomaly window-based scoring function from NAB 4.2.1 to give scores to algorithms based on their ability for early anomaly detection.

Datasets

SKAB has 34 datasets, and each contains multivariate time-series data coming from sensors in the testbed. It is labeled data with features such as temperature, pressure, Voltage etc. Testbed setup is shown in figure 4.2 that is used for generating data.

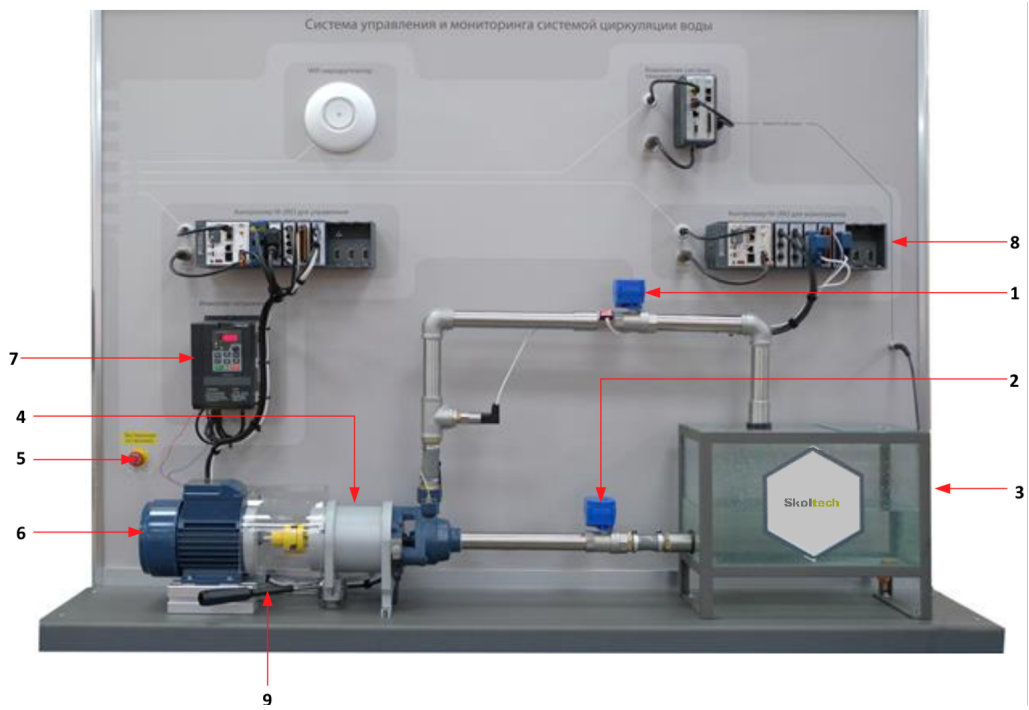


Figure 4.2: Testbed used to generate data for SKAB benchmark [41]

1,2 - solenoid valve; 3 - a tank with water; 4 - a water pump; 5 - emergency stop button; 6 - electric motor; 7 - inverter; 8 - compactRIO; 9 - a mechanical lever for shaft misalignment. Not shown parts - vibration sensor, pressure meter, flow meter, thermocouple.

Results

Overall results from the benchmark are shown as two leaderboards for point anomaly and change point detection performance of algorithms. The relevant repository contains Jupyter notebooks as well for the reproduction of results. Convolutional-Auto Encoder (Conv-AE) performed best for point anomaly detection and Isolation forest performed better for change point detection.

4.2.3 Exathlon

Exathlon benchmark has focused on explainable anomaly detection. This benchmark makes the comparison of anomaly detection and explanation discovery techniques on large datasets collected from real-world [37].

Datasets

High dimensional data is collected from real-world traces of 10 applications in Apache Spark. This large amount of data is collected specially for anomaly detection evaluation.

Metric Type	Spark UI Driver	Spark UI Executor	OS (Nmon)
# of Metrics	243	5 x 140 = 700	4 x 335 = 1340
Total	2,283		

Frequency	1 data item per second
Data Items	2,335,781
Duration	649 hours
Total Size	24.6 GB

Figure 4.3: Summary of datasets in Exathlon benchmark [37]

4.2.4 Summary of benchmarks

There are only a few benchmarks available with a focus on anomaly detection. Table 4.3 provides a summary of benchmarks for evaluating anomaly detection algorithms.

Name	No. of algorithms	No. of datasets	Data source	Allow new algorithm/dataset
NAB	12	58	Real world, Artificial	Yes
SKAB	13	32	Real world	Yes
Exathlon	3	10	Real world	Yes

Table 4.5: Summary of existing machine learning benchmarks

4.3 Shortcomings in existing benchmarks

- Datasets used in benchmarks are not complex enough to represent real-world problems and do not have enough variety, e.g., NAB has only univariate data for evaluation. Anomalies are easy to detect in these datasets [22].
- Labeling of datasets on which benchmarks operate is not accurate in some cases. It makes results from these benchmarks also less accurate [22].
- Adding new datasets or algorithms for evaluation is quite difficult and make benchmark expansion difficult. For example, SKAB has implementation in Jupyter notebooks without standard guidelines.
- Evaluating commercial solutions for anomaly detection is not possible with existing benchmarks.
- Small size and lack of diversity in datasets for evaluation hinder existing benchmarks from being applicable in many situations.
- Some important metrics are not measured, e.g., time complexity, computation power needed for anomaly detection, and the effect of training dataset size on algorithm accuracy.

- Researchers often create benchmarks with datasets and evaluate their proposed algorithm, e.g., NAB. It can potentially introduce bias in the benchmark towards their own proposed anomaly detection algorithm.
- Benchmarks do not visualize results for each algorithm against each dataset file. It increases the difficulty in understanding how algorithms performed and behaved on the different types of data. It reduces the ability of researchers to get insight and analyze results from the benchmark.
- Benchmarks do not provide information about how they performed on different types of data, e.g., univariate, and multivariate data.
- No benchmark is available for the comparison of deep learning algorithms and traditional machine learning algorithms for anomaly detection.

Part III

Open anomaly detection benchmark (OADB)

Chapter 5

Design and Implementation

This chapter includes a brief overview of general requirements for an ideal anomaly detection benchmark and requirements defined for our proposed benchmark (OADB). Further, this chapter provides detail about the OADB implementation, integrated datasets, algorithms and process for extension of the OADB.

5.1 General requirements for an ideal anomaly detection benchmark

Ideal machine learning benchmark should have following characteristics [36, 42].

Transparency The benchmark should have good documentation for details about its working and implementation. Therefore, users can quickly get insight into the working of benchmark and understand results produced by the benchmark.

Reproducibility Results produced by the benchmark should be reproducible under the same configuration.

Usability Setting up and using benchmark should be a short and straightforward process. It should be designed and developed to be used by both technical and non-technical users.

Fairness The benchmark should ensure all the algorithms are tested under the same conditions and not biased toward a set of algorithms. Benchmark should not have constraints that affect algorithms differently.

Expandability The benchmark should be expandable with the integration of new algorithms and datasets. The expansion process should be reasonably straightforward.

Portability The benchmark should work on different types of operating systems, e.g., Windows, MacOS, etc., and underlying hardware.

5.2 Requirements for Open Anomaly Detection Benchmark (OADB)

Following are the requirements which should be fulfilled by a anomaly detection benchmark to be reasonably good. These requirements are based on the needs of researchers and practitioners for using anomaly detection benchmarks.

- **R1: Accuracy** The benchmark should be able to compare performance of algorithms based on their accuracy on given datasets.
- **R2: Computational complexity** The benchmark should measure the time utilized by various algorithms for detecting anomalies in given datasets under given conditions.
- **R3: Extensibility** The benchmark should allow extension by adding new algorithms and datasets in the future.
- **R4: Visualization** The result from benchmark should be visualized so that users can visualize the result of each algorithm against each dataset individually.
- **R5: Documentation** The benchmark should have good documentation about its working and implementation details. Process for extension of the benchmark with new dataset or algorithm should be documented as well.
- **R6: Configuration** The benchmark should allow configuration of benchmark through parameters e.g., specifying train and test split size, etc.
- **R7: Real-world data** The benchmark should have datasets collected from the real world. Real-world datasets give legitimacy to benchmarks to be used by researchers and practitioners.
- **R8: Datasets variety** The benchmark should have a variety of datasets for anomaly detection. Ideally, datasets should vary in the number of features from one to hundreds and similarly in size.
- **R9: Training dataset size/Accuracy trade-offs** The benchmark should provide insight into trade-offs between training dataset size and accuracy of the algorithm on given datasets.
- **R10: Anomaly types** The benchmark should accommodate benchmarking of algorithms for various types of anomalies. It should also show how algorithms performed on the different types of anomalies.
- **R11: Deep learning** The benchmark should allow the integration of deep learning algorithms for comparison with other machine learning algorithms.

5.3 Evaluation of existing benchmarks based on defined requirements

A significant amount of research is being done to improve existing anomaly detection techniques and develop new anomaly detection algorithms [7, 19, 29, 56]. Researchers worked on evaluating and comparing various techniques for anomaly detection [5, 55]. However, developing an open-source benchmark

was not the main focus. A small number of benchmarks are available with a focus on anomaly detection problems.

There are a few anomaly detection benchmarks developed as mentioned in section 4.2 and all of them suffer from various deficiencies as described in section 4.3. These benchmarks are essential for researchers and practitioners to find appropriate algorithms for their anomaly detection problems. The unavailability of reliable benchmarks increases the difficulty in selecting an algorithm and benchmarking a new algorithm.

Table 5.1 shows evaluation of existing anomaly detection benchmarks based on our requirements. All the requirements fulfillment is checked and a relevant total score is assigned to benchmarks.

✓ represents that benchmark fulfills the requirement and assigned 1 score.
✗ represents that benchmark does not fulfill the requirement and assigned 0 score.
* represents that benchmark partially fulfills the requirement and assigned 0.5 score.

Benchmark	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	Total score
NAB	✓	✗	✓	*	*	✗	✓	*	✗	*	✗	5.0
SKAB	✓	✗	*	✗	✗	✗	✓	✗	✗	✓	*	4.0
Exathlon	✓	✗	✗	*	✗	✗	✓	✗	✗	*	✗	3.0

Table 5.1: Evaluation of popular anomaly detection benchmarks based on defined requirements

Evaluation of existing benchmarks shows the obvious shortcomings and overall lack of anomaly detection benchmarks. They lack a good variety of datasets and are not transparent. This analysis of state-of-the-art anomaly detection benchmarks suggests the need for a new anomaly detection benchmark. The new benchmark should address existing benchmarks' issues and give insight into state-of-the-art anomaly detection algorithms.

5.4 Problem and proposed solution

As mentioned in section 4.3 and 5.3, there are only a few benchmarks developed to evaluate anomaly detection algorithms and most of them are plagued with various issues. Consequently, these benchmarks are problematic to use for any benchmarking purpose. Consequently, hinder the ability to get insight into the status of anomaly detection algorithms performance. There is a need for a transparent and expandable benchmark to meet practitioners' and researchers' various requirements.

We present a new anomaly detection benchmark that fulfills most of the basic requirements to solve this problem. The new benchmark is named as Open Anomaly Detection Benchmark (OADB). It is transparent enough to give a good insight into datasets, algorithms and their performance on datasets. Some of the existing benchmarks contain certain biases [22], OADB minimizes

bias toward any specific algorithm. It will make research anomaly detection algorithms more transparent and verifying progress in the field relatively easy.

Some benchmarks have only one type of datasets, e.g., Numenta benchmark has only univariate datasets. However, OADB has datasets collected from various sources and includes recently published public datasets.

5.5 Design and architecture

Benchmark architecture is designed to be modular. Code is divided into Detector, Data, Preprocessor, Core, Helper and Visualizer modules. As shown in Figure 5.1 data module reads datasets from files and it is passed through different modules and processed accordingly. In the final phase, benchmarking result is saved in the result folder. This saved result file is used to visualize it for the user.

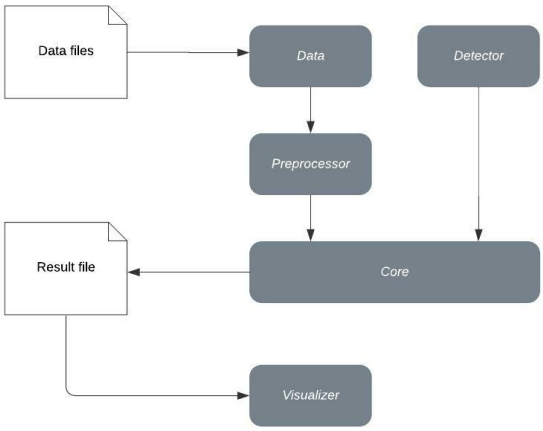


Figure 5.1: OADB modules and interaction of them with each other

5.6 Summary of datasets utilized in OADB

Getting quality labeled data for anomaly detection benchmarks is difficult. There are not many public datasets that contain anomalies and are labeled. We have have collected data from Numenta anomaly benchmark (NAB) [44], Yahoo [65], ODDS library [64] and UCR Time Series Anomaly Archive [22]. By combining these datasets, a data repository is created that has a good variety of data and is well suited for benchmarking process. These datasets are transformed to use in our benchmark based on our requirements. Table 5.2 provides summary of datasets integrated in OADB for benchmarking use.

Dataset	Type	No. of datasets
NAB	Univariate	58
Yahoo	Univariate	367
UCR time series	Univariate	250
ODDS library	Multivariate	31

Table 5.2: Overview of OADB datasets

5.7 Implementation

Python language version 3.9, PyCharm IDE and some popular open-source libraries are used to develop benchmark. Figure 5.2 present the OADB organization of modules..

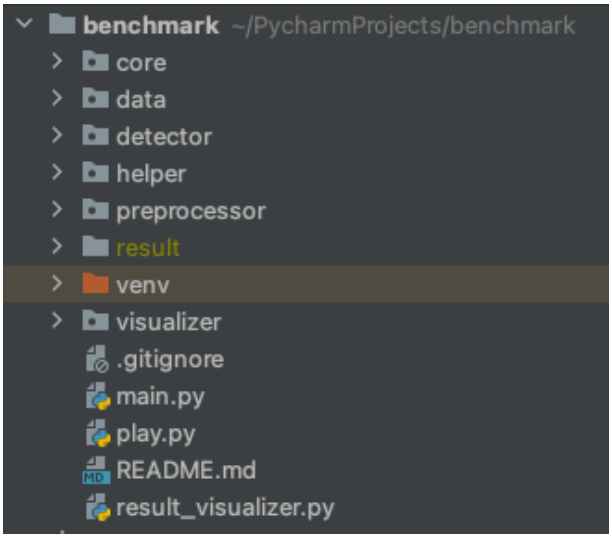


Figure 5.2: Organization of OADB modules

5.7.1 Modules

Data module: This module contains all the datasets used for benchmarking. Each folder name represents a different repository from where data is sourced. This module collects all CSV files divided into different folders and transfers them to other modules as a dictionary.

Detector module: This module contain implementation of all algorithms integrated in benchmark. Implementation for these algorithms is sourced from Anomaly Detection Toolkit (ADTK) [11], PyOD [88], NAB [44] and Scikit-learn [71] open source libraries. Instances of all algorithms are passed to core module for evaluation. Table 5.3 provides overview of all algorithms integrated in OADB.

Preprocessor module: This module performs data cleaning and makes data ready for algorithms consumption. It also splits data into test and training datasets.

Algorithm	Data support	Implementation source
Relative entropy	Univariate	NAB
Windowed gaussian	Univariate	NAB
Bayesian change point detector	Univariate	NAB
EXPoSE (EXPected Similarity Estimation)	Univariate	NAB
Contextual Anomaly Detector - Open Source Edition	Univariate	NAB
KNN CAD	Univariate	NAB
Isolation forest	Multivariate	Scikit-learn
Elliptic envelope	Multivariate	Scikit-learn
Local outlier factor	Multivariate	Scikit-learn
Generalized ESD test	Univariate	ADTK
Principal component analysis	Univariate	ADTK
Angle based outlier	Multivariate	PyOD
Clustering based local outlier factor	Multivariate	PyOD
One class SVM	Multivariate	PyOD
Autoencoder	Multivariate	PyOD

Table 5.3: Overview of OADB algorithms

Core module: This module receives data from the Preprocessor module and algorithms from the Detector module. Runs each algorithm against each dataset and saves the result in a file for each combination.

Visualize module: This module reads result data from result files. All data is combined into a single data frame to calculate required metrics such as accuracy and time complexity. This data is visualized as a heat map to provide overview of result. Each cell represents the performance of an algorithm on a data folder. These cells can be selected to see the performance of an algorithm on subfolders and individual dataset. It allows users to view each detail of datasets and result of benchmark.

5.7.2 Benchmarking metrics

Metrics selected for benchmarking are essential in determining its success and usability for benchmarking. There are many metrics to evaluate the performance of an algorithm, such as Accuracy, Relevance, Time complexity, Computation power needed etc. Following are two main performance metrics selected for the OADB benchmark to evaluate and compare the performance of various algorithms.

Accuracy:

There are various methods available to get an insight of machine learning algorithm accuracy on a given dataset, e.g., confusion matrix, and Area Under Curve (AUC). OADB, during benchmarking process, saves the dataset, actual anomaly labels and detected anomalies by algorithm in the result file. OADB

visualizes accuracy result in different ways to provide deeper detail about the performance of algorithm.

Computational complexity:

Computational complexity of the algorithm refer to the computing power used by algorithms for training and detecting anomalies in a dataset. OADB measures the time used by the algorithm in training and test phase under given computing power. The Visualize module uses this data to present it to user in an understandable form.

5.8 Extensions

5.8.1 Dataset integration

New dataset integration is quite simple. Data should be in the required format as mentioned in the benchmark repository. By simply copying into a new folder in the datasets folder, it will be integrated into benchmark.

5.8.2 Algorithm integration

It is a two-step process. First, create a new class inheriting BaseDetector class and then add this new class as an element in the algorithms dictionary.

5.8.3 Customization

OADB allows customization of algorithms implementation. Algorithms integrated in the benchmark require different parameters to create a model. These parameters affect the algorithm's performance in detecting anomalies. They can be adjusted in the Detector module to meet specific needs or to make improvements. The number of algorithms to be used for benchmarking can be adjusted by changing the dictionary in the detector module that contains all algorithms. OADB also allows setting training and test data size as a percentage of the complete dataset through parameters.

Chapter 6

Evaluation

In this chapter, results generated from Open Anomaly Detection Benchmark (OADB) are analyzed. First, information about environment setup is provided. Further, we analyze datasets used in the benchmark and results produced through different aspects. Afterward, Open Anomaly Detection Benchmark (OADB) is evaluated based on a set of requirements and its capabilities are described.

6.1 Experimental Setup

Underlying hardware and software setup affects the performance of machine learning algorithms in terms of time complexity. A larger amount of computing power results in faster processing of data for anomaly detection. We have used the same setup for benchmarking to avoid any negative impact on the benchmark results.

6.1.1 Hardware setup

All the benchmarking experiments are done in the Google Cloud Platform (GCP). Benchmark is run on three Virtual Machine (VM) instances, each with specifications as described Table 6.1.

Item	Specification
Cloud provider	Google Cloud Platform (GCP)
CPU platform	Intel Broadwell
vCPUs	8
Memory	32GB
Boot disk	50GB
OS	Debian GNU/Linux 10 (buster)
Machine type	e2-standard-8
Service	Compute engine

Table 6.1: Hardware specifications of experimental setup

6.1.2 Software setup

Table 6.2 provides detail about software libraries used in OADB and their versions.

Library	Version
Python	3.9
Scikit-learn	0.24.2
PyOD	0.9.4
Pandas	1.3.3
Numpy	1.19.5
Matplotlib	3.4.3
ADTK	0.6.2
Tensorflow	2.60

Table 6.2: Software libraries used for development of OADB

6.2 Evaluation of OADB datasets

As described in table 5.2, we have sourced datasets from four data repositories. These datasets are organized into four folders nab, odd, yahoo and ucr. One of four folders contain multivariate datasets and the rest consist of univariate datasets.

Figure 6.1 provides insight into datasets used in OADB. Bar chart (a) provides a summary of the number of dataset files in OADB. Bar chart (b) shows that Yahoo datasets have 4683 data instances per dataset on average. This number increases for NAB, UCR and ODDS, reaching 440699 data instances per dataset. It is evident from (a) and (b) that datasets have a good variety in terms of dataset size. Thus, allowing algorithms to be tested on various amounts of data. Bar chart (c) presents information about the number of features in multivariate datasets. Datasets have features from a few to a few hundred. It is evident that datasets also have a wide variety in terms of the number of features.

This analysis of datasets through visualization makes it evident that OADB datasets have great variety in terms of size and dimensions. Hence, these datasets are well suited for the evaluation of anomaly detection algorithms.

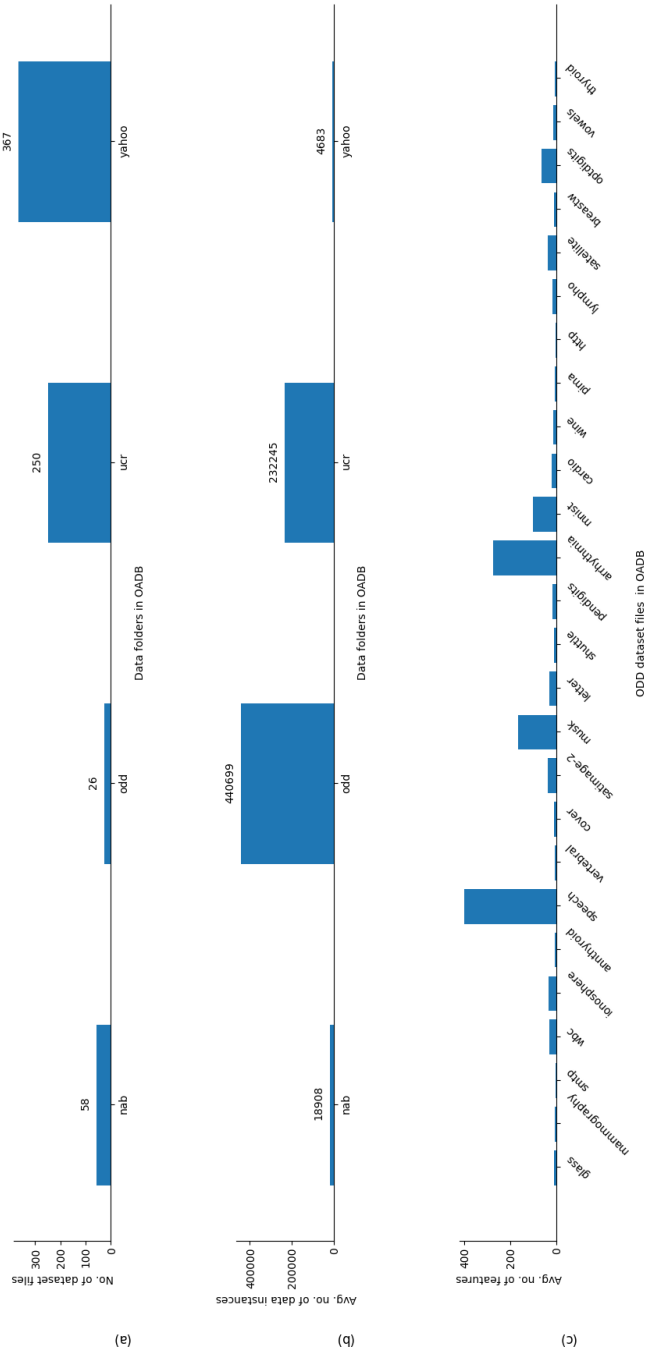


Figure 6.1: Analysis of datasets in OADB through different aspects

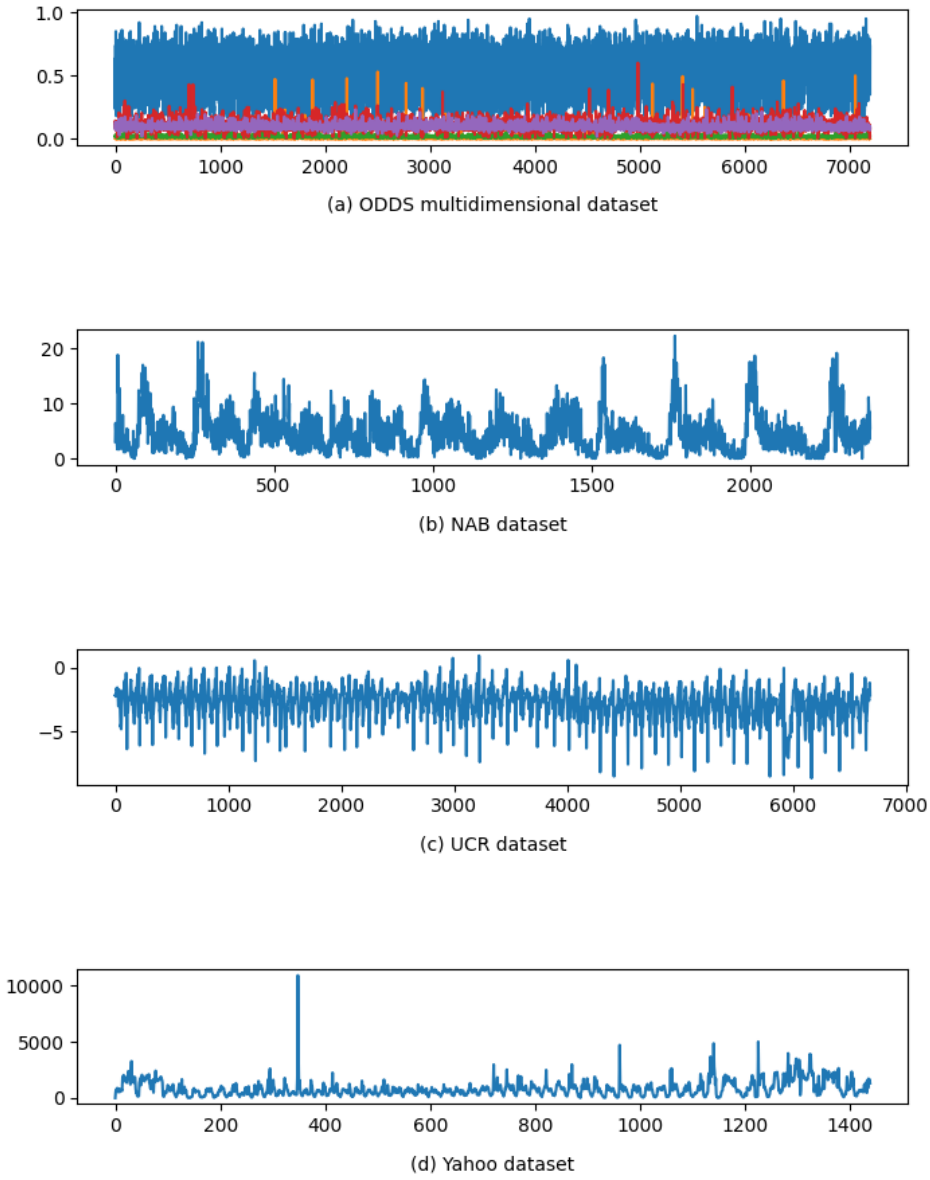


Figure 6.2: Visualization of datasets individually for getting insight into data patterns

Figure 6.2 visualize randomly selected dataset files as examples and provide information about their characteristics. All the datasets have different patterns in them. Some datasets have obvious and easy to detect anomalies, e.g. as shown in (d) Yahoo dataset. At the same time, others contain anomalies that are difficult to detect e.g. as shown in (c) UCR dataset. Datasets contain various kinds of anomalies e.g. density changes, a sudden increase in values, change in cyclical pattern etc.

It is evident from the analysis that OADB datasets have the qualities required to evaluate anomaly detection algorithms and are well suited to be used in anomaly detection benchmarks.

6.3 Analysis of ML algorithms performance for anomaly detection

In this section, we will perform multiple benchmarking experiments to evaluate the performance of machine learning algorithms for anomaly detection. We aim to learn patterns in the performance of algorithms for anomaly detection by comparing them.

6.3.1 Methods for measuring accuracy

Accuracy of an algorithm can be measured through various methods. Some of these methods suffer from different type of issues and do not shed light on all aspects. Eventually, leading researchers into wrong directions. A subset of popular methods for measuring accuracy are described below.

Confusion matrix: Confusion matrix [80] is two-dimensional matrix and consists of four values True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). These values can be explained in terms of anomaly detection as

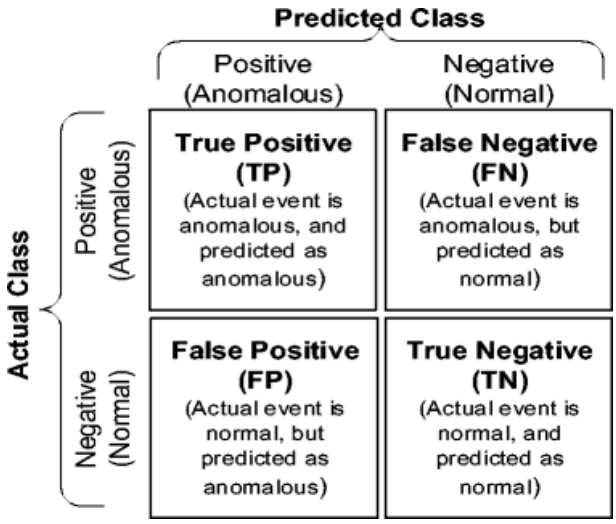


Figure 6.3: Description of the confusion matrix for anomaly detection [63]

True positive is a data instance that is correctly labeled as an anomaly by the

algorithm.
True negative is a normal data instance that is also labeled as normal by the algorithm.
False positive is a data instance that is incorrectly labeled as an anomaly by the algorithm.
False negative is a data instance labeled as normal by the algorithm, but it is an anomaly.

Confusion matrix provides information about both failures and successes of the algorithm. However, it is not suitable to get an insight into the accuracy of an algorithm on multiple datasets. Figure. 6.3 show example of confusion matrix for anomaly detection algorithm.

Recall: represents proportion of anomalies detected by the algorithm out of all actual anomalies.

$$Recall = \frac{TP}{TP + FN} \tag{6.1}$$

Precision: represents proportion of correctly detected anomalies.

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

Classification accuracy: represents the percentage of data instances correctly labeled.

$$Classification\ accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.3}$$

F1 score: is measured by the equal contribution of both precision and recall as shown in equation 6.5 and defined as [70]

"a harmonic mean of precision and recall"

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6.4}$$

Receiver Operating Characteristic (ROC) curve: ROC curve is generated by plotting False positive rate on the X-axis and True positive rate on Y-axis. It provides insight into performance of algorithm under different thresholds for labeling an instance as an anomaly. Area Under Curve (AUC) is the measurement of area under the ROC curve and helps to compare the performance of algorithms. Higher AUC means better performance of the algorithm.

Precision-recall curve: The precision-recall curve is generated by plotting precision on Y-axis and recall on X-axis at various thresholds. This curve provides insight into the trade-off between precision and recall.

Average precision score: The average precision score describes the precision-recall curve, but it is not the area under curve. The increase in recall from the preceding threshold is used as weight and the weighted average of precision values is calculated.

$$Average\ precision\ score = \sum_{n}(R_n - R_{n-1})P_n \tag{6.5}$$

What are the best metrics for anomaly detection results evaluation?

A confusion matrix can accurately summarize the accuracy of an anomaly detection algorithm. However, a confusion matrix cannot be efficiently used for comparing performance of multiple algorithms. Classification accuracy measurement allows comparison. However, it is flawed for comparing accuracy in imbalanced data. For example, if an algorithm labels all data instances in the dataset as normal and 5% data instances are actual anomalies. It will result in 95% classification accuracy of the algorithm. This behavior illustrates wrong picture about the accuracy of algorithm.

Precision and recall provide better information about the accuracy of algorithm. However, precision and recall are measured at one threshold and dependent on the threshold value. F1 score summarizes the precision and recall values by an equal contribution from both values in the calculation. Usually, algorithms assign anomaly scores to data instances based on the probability of that instance being an anomaly. ROC curve and precision-recall curve are used to analyze the accuracy of an algorithm at varying thresholds. Precision-recall provides a better view for imbalanced data e.g. in the case of anomaly detection due to rarity of anomalies. Further, the Average precision score provides a way to summarize the precision-recall curve.

It is evident from analysis that one should analyze the performance of algorithms using multiple metrics and look for a more relevant metric for his needs. It is also obvious that Precision, Recall, F1 score, Precision-recall curve and Average precision score are better metrics for evaluating the performance of anomaly detection algorithms.

6.3.2 Accuracy analysis of anomaly detection algorithms

These results are generated using OADB with 30-70 split for training and test data. Thresholds for declaring a data instance as anomaly are defined and optimized internally i.e. using contamination values, by algorithms for results shown in figure. 6.4, 6.5 and 6.6. ODD datasets are multivariate and Yahoo, NAB, UCR datasets are univariate.

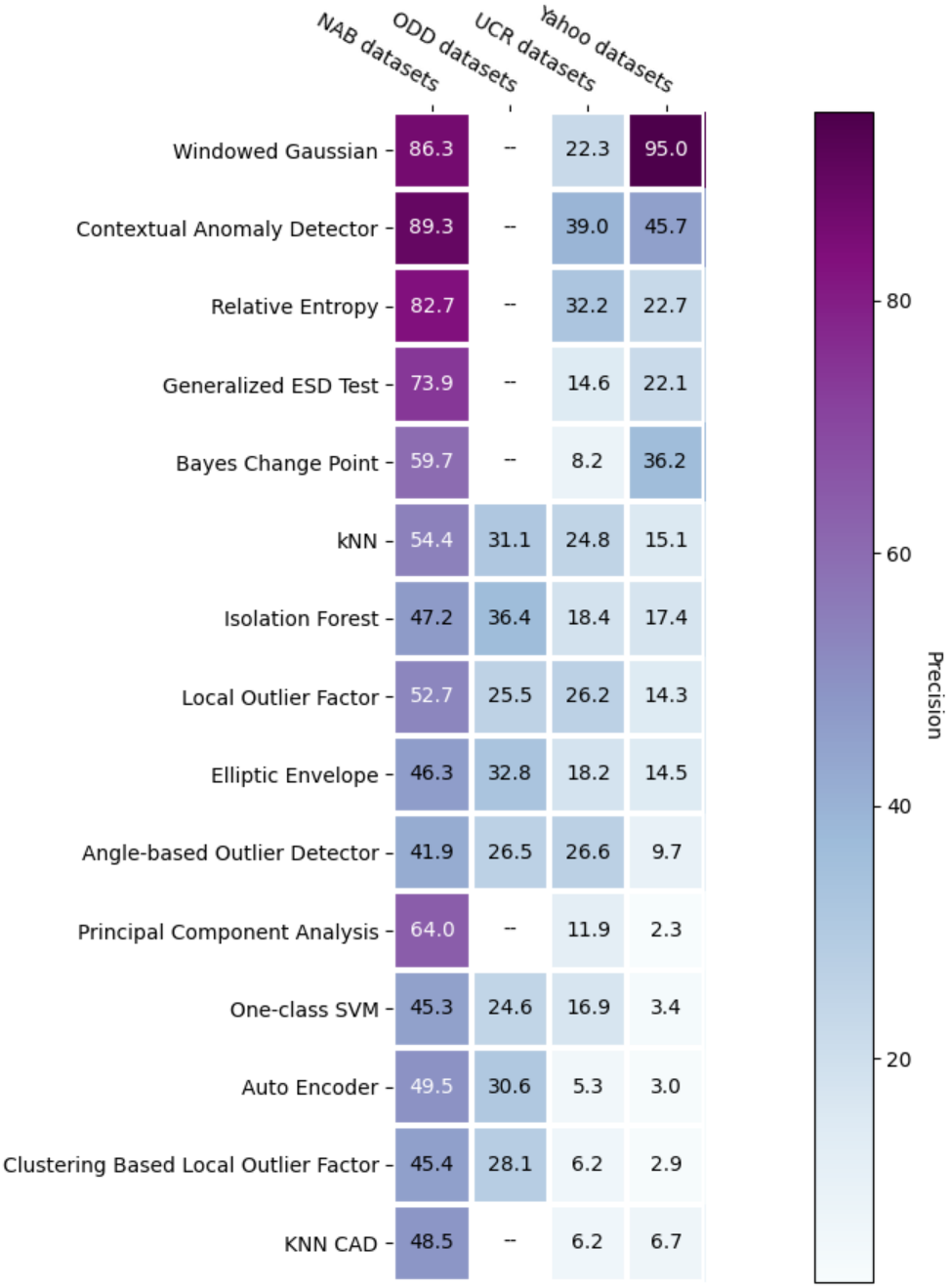


Figure 6.4: Heat map showing the precision score of each algorithm against each data repository in OADB

* Heat map cells with (–) symbol show algorithm is no evaluated against this data repository due to its implementation limitations. As some algorithms perform only on univariate data.

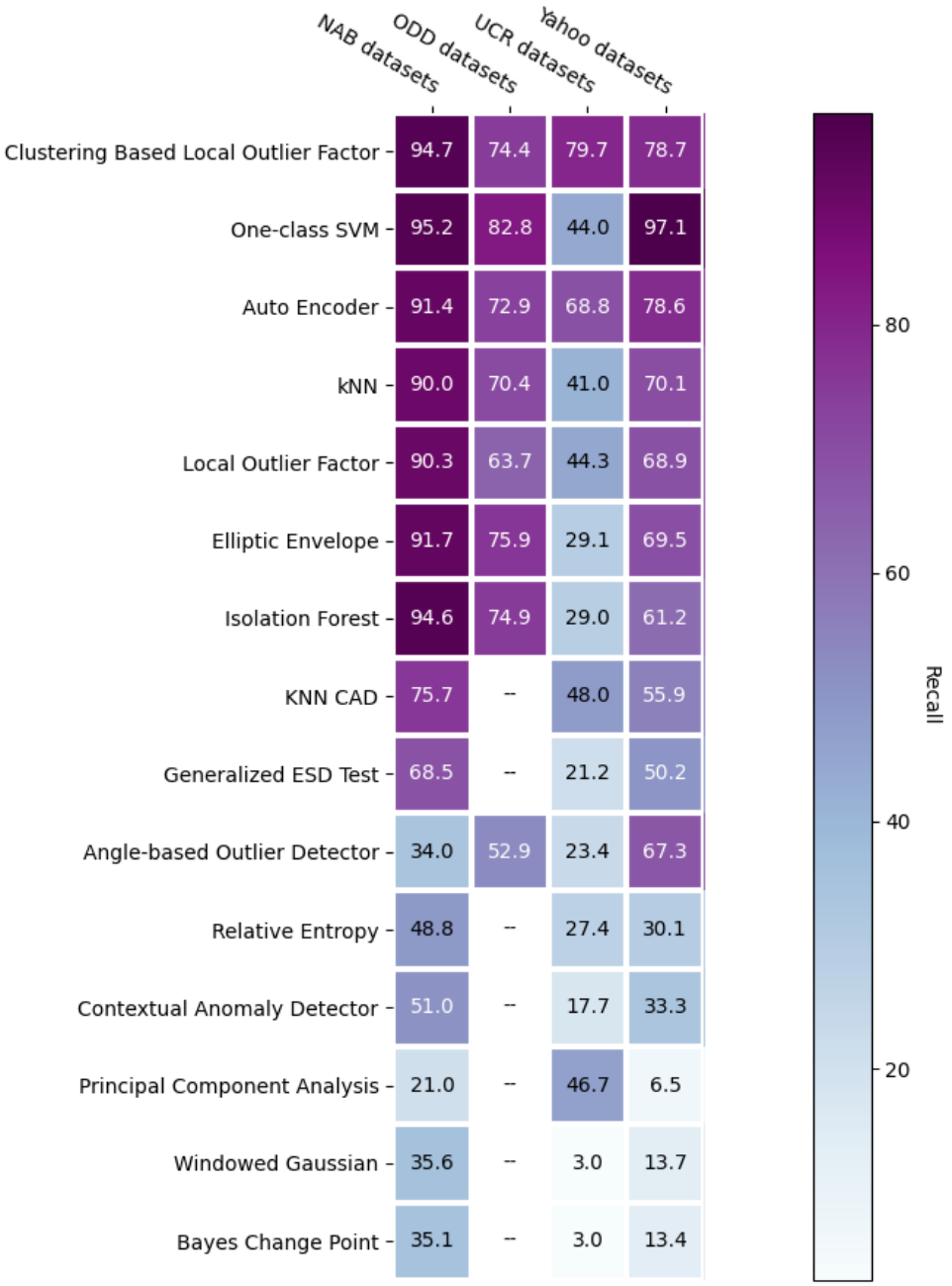


Figure 6.5: Heat map showing recall score of each algorithm against each data repository in OADB

* Heat map cells with (–) symbol show algorithm is no evaluated against this data repository due to its implementation limitations.

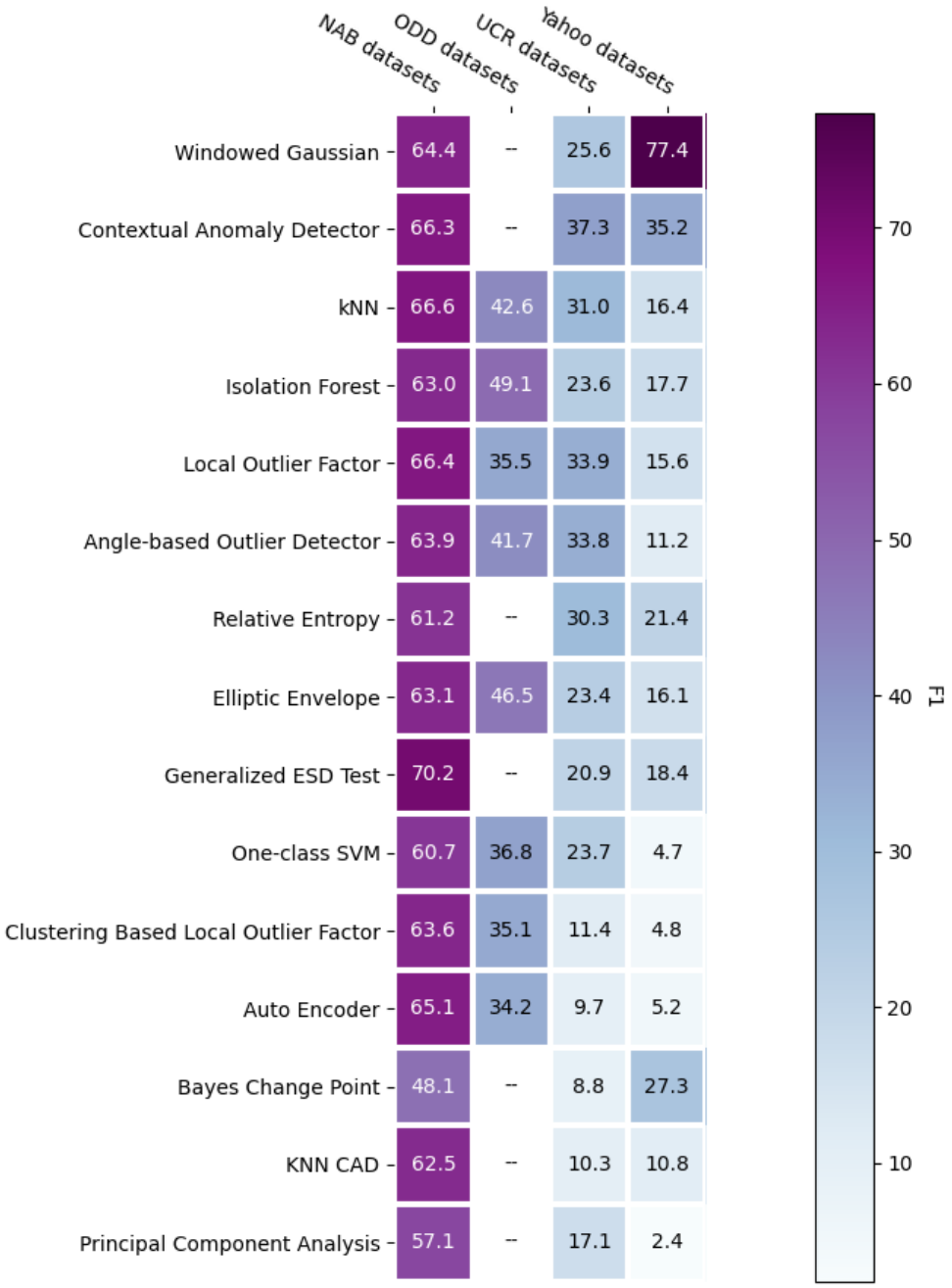


Figure 6.6: Heat map showing average F1 score of each algorithm against each data repository in OADB

* Heat map cells with (–) symbol show algorithm is no evaluated against this data repository due to its implementation limitations.

Figures 6.4, 6.5 and 6.6 show that all algorithms performed better on NAB data repository in comparison to other univariate data repositories. It is due to the higher complexity of datasets in UCR and Yahoo, making it difficult to detect anomalies. It can be observed that the Windowed Gaussian algorithm has a relatively higher precision and f1 score for univariate datasets. It is also visible that the isolation forest algorithm performs better on multivariate datasets than other algorithms in terms of precision and f1 score. These precision, recall and F1 score heat maps summarize the performance of algorithms on a wide variety of datasets. Clustering based local outlier factor has highest recall score relative to other algorithms. Algorithms often perform with some assumptions about the data. For example, Isolation forest assume that anomalies are rare and elliptic envelope requires data to be Gaussian distributed. These assumptions may not hold in some of our datasets and result in low accuracy. It can be solved to some extent by analyzing individual datasets available for evaluation.

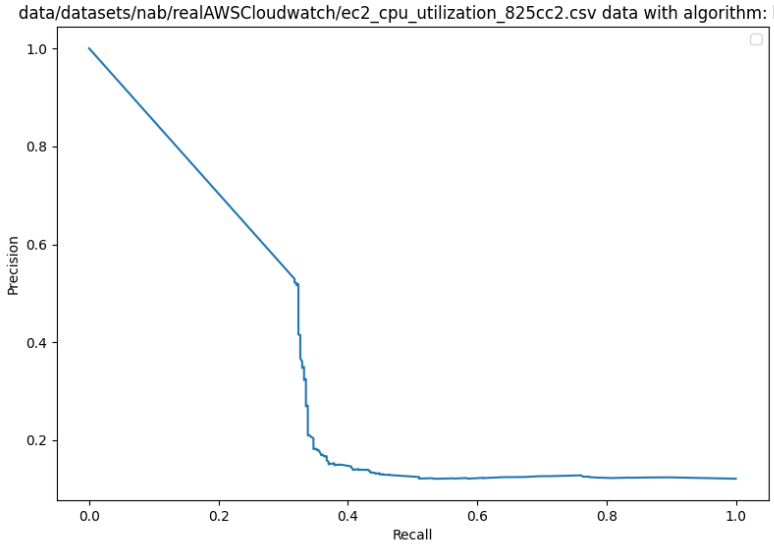
Precision and recall curve analysis

Precision, recall and f1 score are entirely dependent on the threshold. The threshold value is used for labeling a data instance as an anomaly. However, it is not easy to find the threshold that maximizes the accuracy of an algorithm when true anomalies are unknown. Another way to evaluate the accuracy of the algorithm is by utilizing the precision-recall curve. The average precision score summarizes the precision-recall curve as explained in Section 6.3.1.

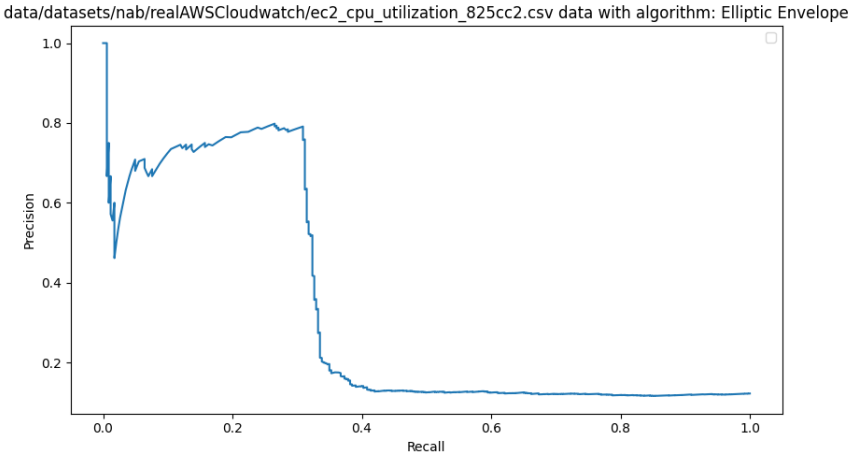
Figure 6.7 show precision-recall curves of KNN, Elliptic envelope, Windowed gaussian and Isolation forest on a single dataset. It can be observed that all algorithms have different precision-recall curves and accuracy.

Figure 6.8 show precision-recall curves of Isolation forest algorithm on a variety of datasets. Figures (a) and (b) show high algorithm accuracy and good precision and recall trade-offs with increasing anomaly thresholds. Figures (c) show the worst performance and slightly better accuracy in figure (d). It can be observed that precision-recall curves vary significantly based on the dataset utilized for evaluation.

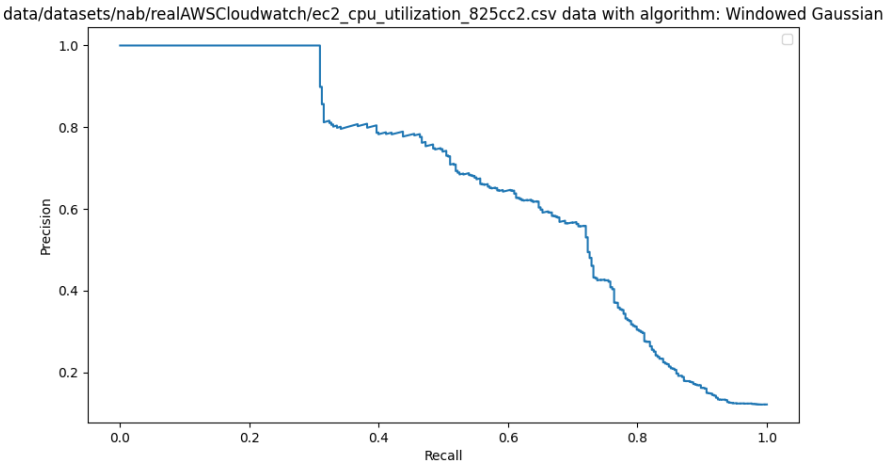
It is evident from figures 6.7 and 6.8 that algorithms accuracy and precision-recall curves vary depending on qualities possessed by dataset. We can evaluate algorithms on a wide variety of datasets and get an overview of their performance. However, they behave differently on various datasets and one needs to evaluate performance on datasets relevant for them.


data/datasets/nab/realAWSCloudwatch/ec2_cpu_utilization_825cc2.csv data with algorithm: kNN
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

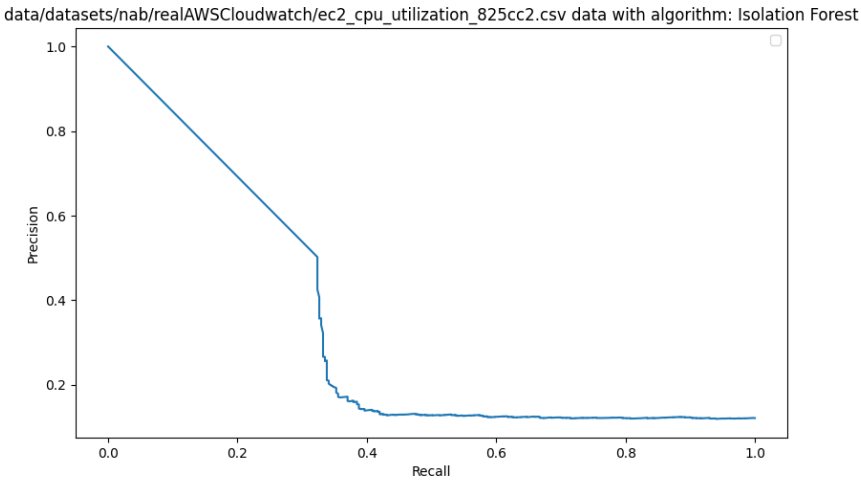
(a)


data/datasets/nab/realAWSCloudwatch/ec2_cpu_utilization_825cc2.csv data with algorithm: Elliptic Envelope
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

(b)

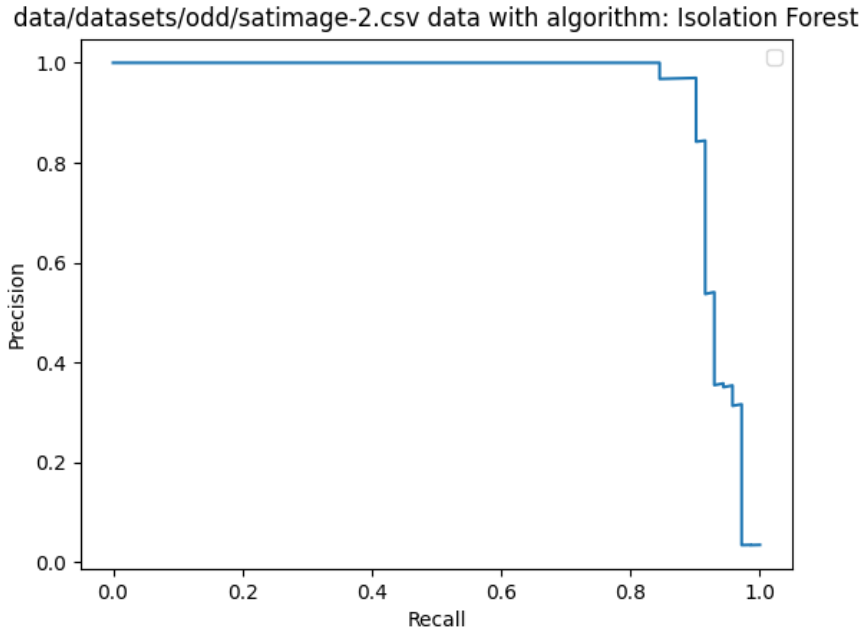

data/datasets/nab/realAWSCloudwatch/ec2_cpu_utilization_825cc2.csv data with algorithm: Windowed Gaussian
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

(c)

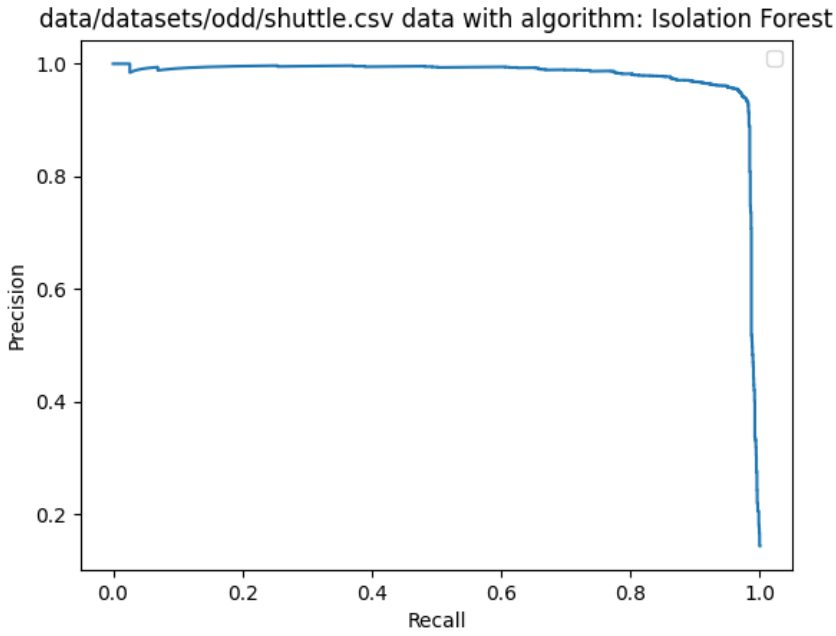


(d)

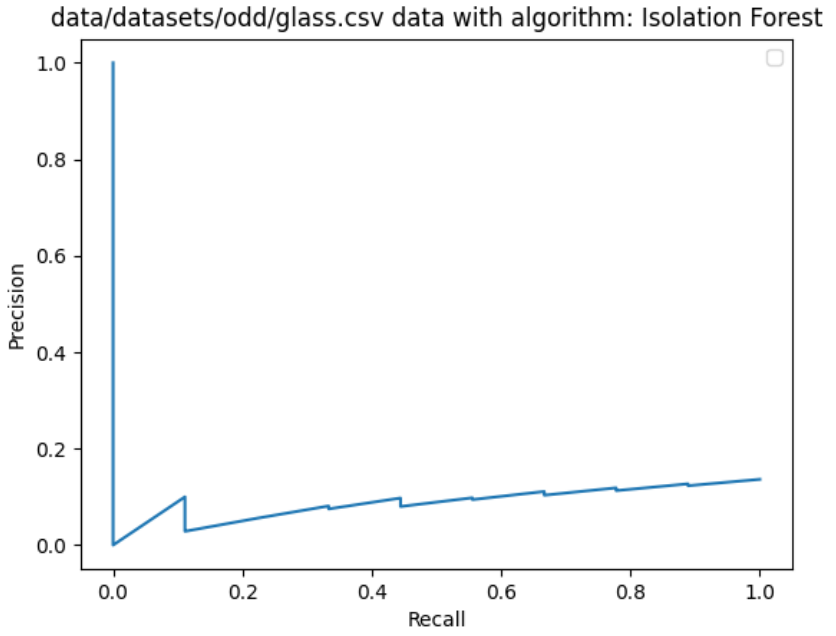
Figure 6.7: Precision-recall curves of KNN, Elliptic envelope, Windowed gaussian and Isolation forest on single dataset



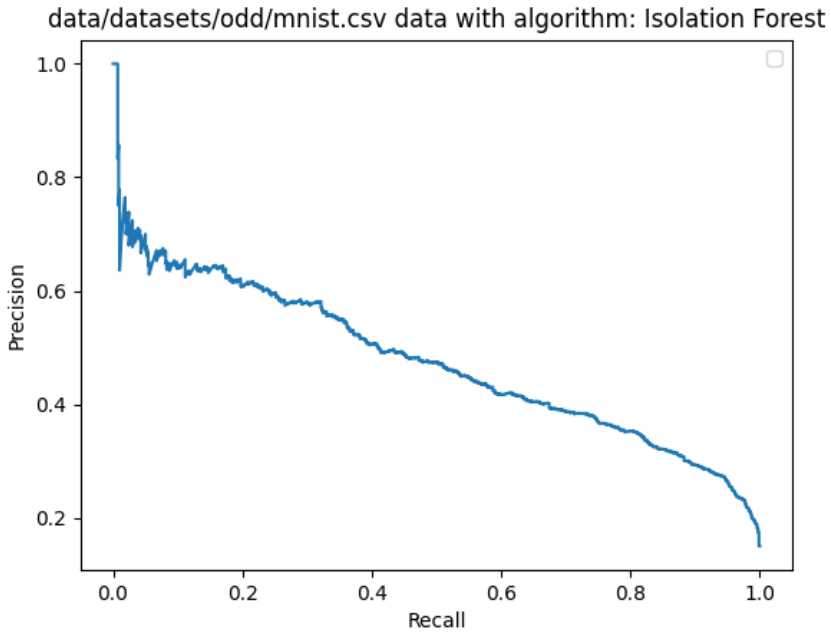
(a)


data/datasets/odd/shuttle.csv data with algorithm: Isolation Forest
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

(b)


data/datasets/odd/glass.csv data with algorithm: Isolation Forest
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

(c)



(d)

Figure 6.8: Precision-recall curves of Isolation forest algorithm on multiple datasets

Figure 6.9 show average precision score of algorithms and it can be observed that elliptic envelope has best average precision score followed by Windowed guassian. Isolation forest has best average precision score on multivariate datasets.

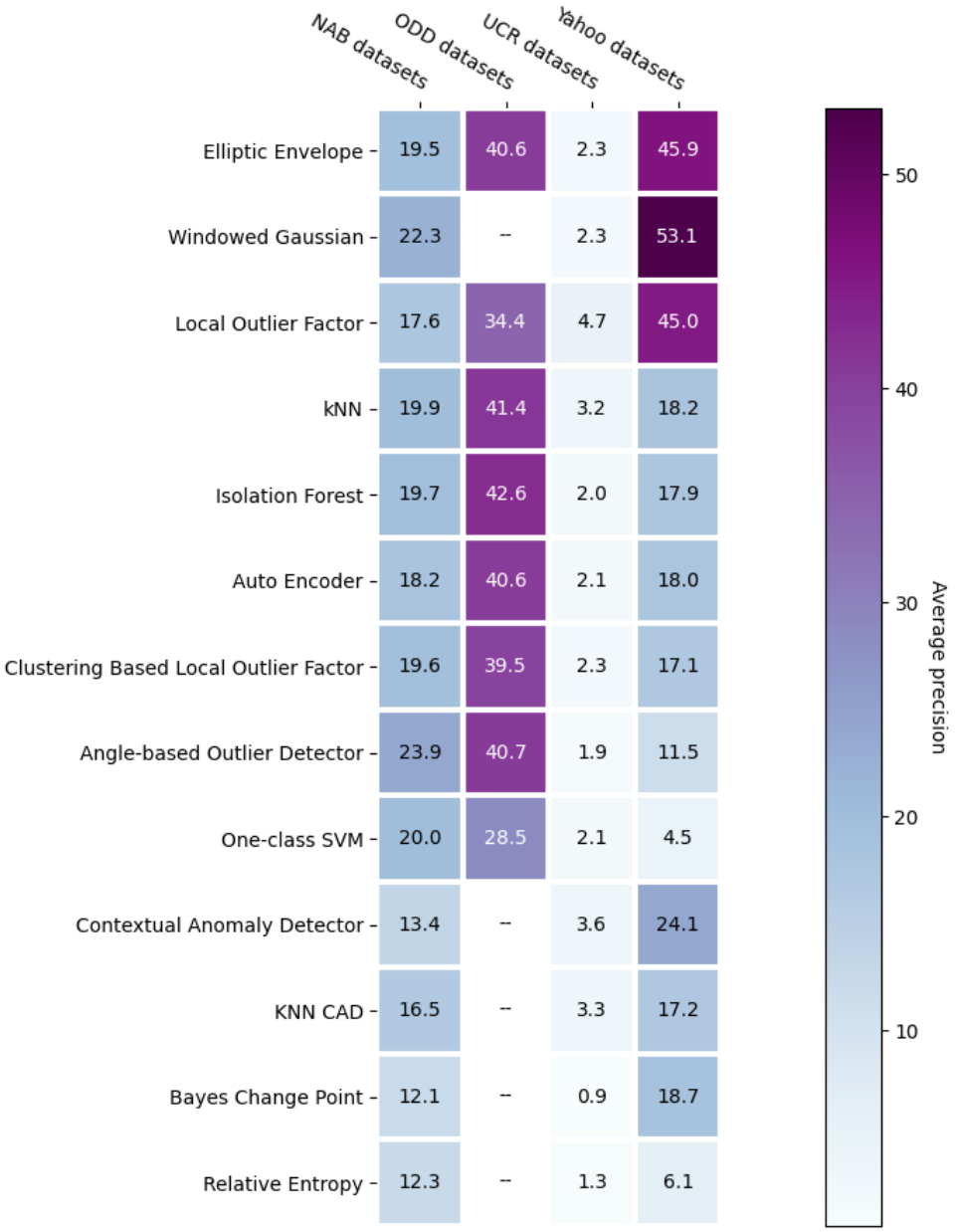


Figure 6.9: Heat map showing average precision score of each algorithm against each data repository

* Heat map cells with (–) symbol show algorithm is no evaluated against this data repository due to its implementation limitations.

6.3.3 Computational complexity analysis of anomaly detection algorithms

The computational complexity of algorithms for anomaly detection plays a vital role in utilizing them for a particular problem. It is also vital how the performance of algorithms change with the increasing size of the dataset. In many domains, anomaly detection requires the algorithm to detect anomalies

in a short time frame with limited computing power.

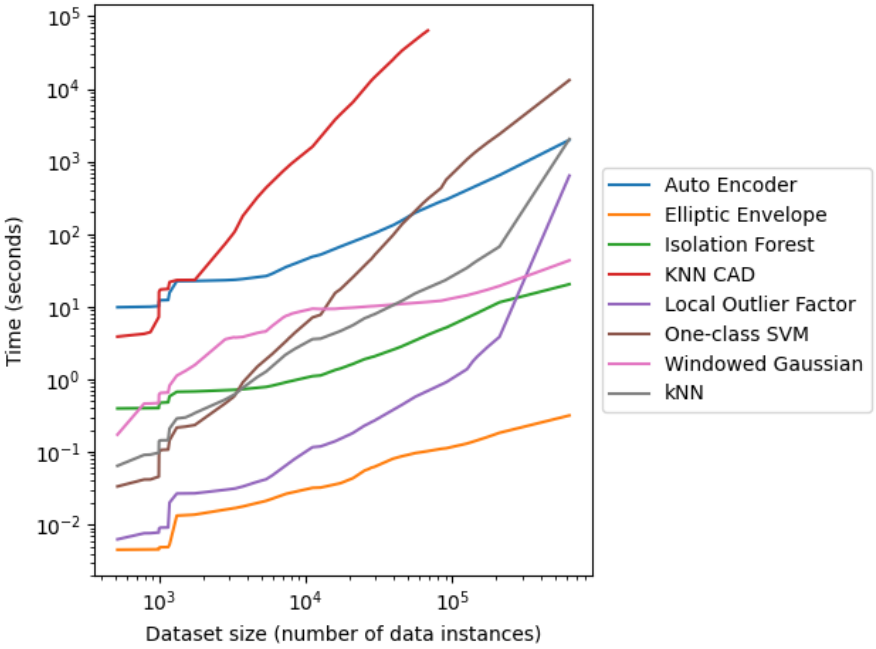


Figure 6.10: Showing time taken by algorithms for training and test phase (combined) on univariate datasets with increasing dataset size

Figure. 6.10 shows the time taken by algorithms to train and detect anomalies on univariate datasets. It can be observed that KNN CAD [16] is the worst performer in terms of computational complexity. KNN CAD calculates dot product and inverse of training data matrix for assigning anomaly score to each new observation. These mathematical operations are computationally expensive and result in bad performance. Relative entropy algorithm also performs poorly after KNN CAD with the increasing size of the dataset. It is observed that Elliptic envelope has lowest computational complexity. Windowed Gaussian and Isolation forest scale much better with increasing size of dataset in comparison to other algorithms.

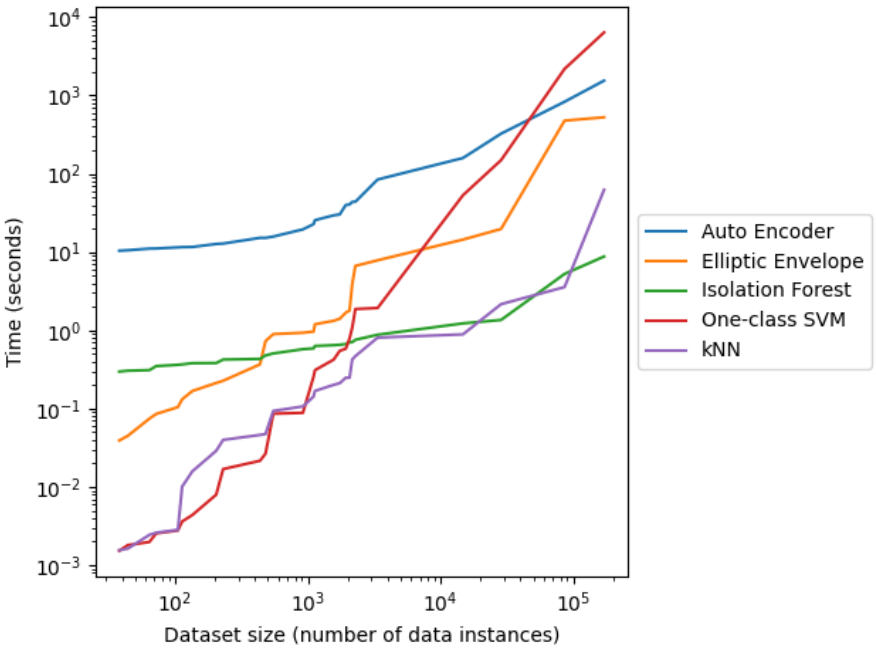


Figure 6.11: Showing time taken by algorithms for training phase on multivariate datasets

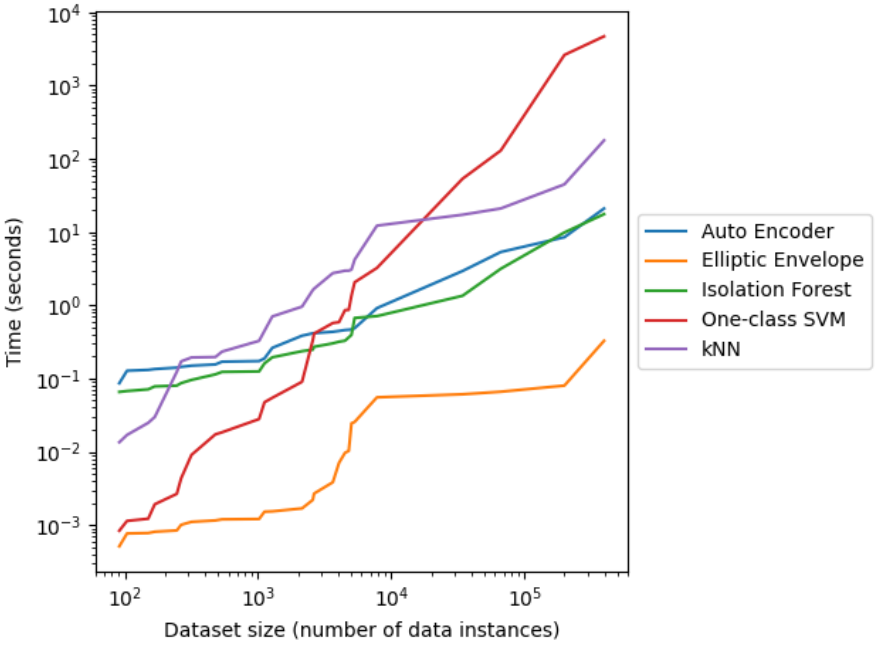


Figure 6.12: Showing time taken by algorithms for test phase on multivariate datasets

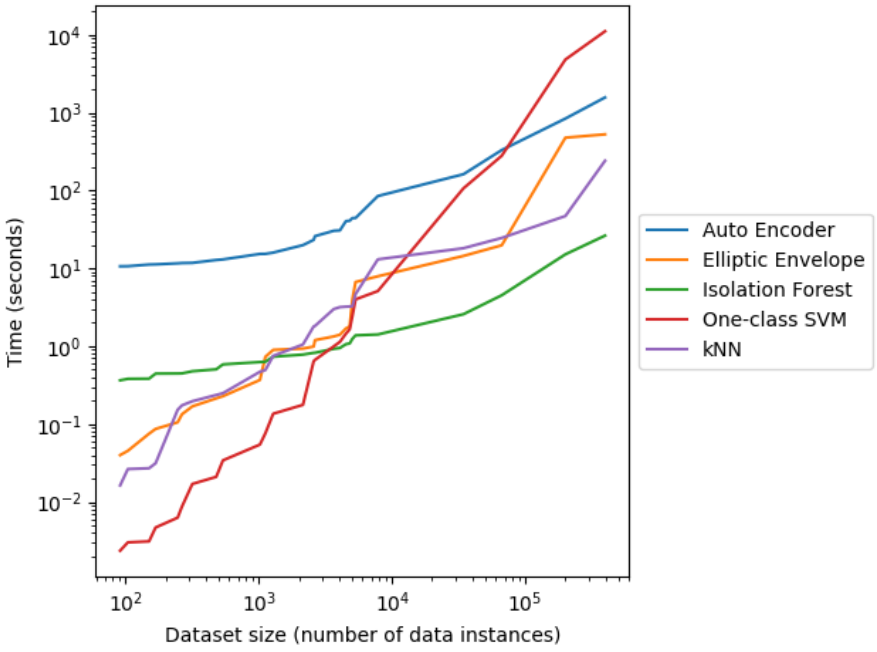


Figure 6.13: Showing time taken by algorithms for training and test phase (combined) on multivariate datasets

Figures 6.11 and 6.12 show computational complexity of algorithms in training and test phase on multivariate datasets. It is observed that One-class SVM computational complexity increase enormously with the increase in dataset size. The computational complexity of the elliptic envelope was lowest in the test phase. However, it is much higher in the training phase. KNN performed better in the training phase than test phase. Autoencoder is computationally expensive but less expensive than One-class SVM.

It can be observed and evident from figures 6.10, 6.11, 6.12 and 6.13 that the Isolation forest algorithm has better performance in terms of computational complexity on multivariate datasets. However, for univariate datasets it is second best after elliptic envelope. This analysis also concludes that using deep learning is not always more expensive than traditional machine learning algorithms and it depends underlying approach used by algorithm for finding anomaly. The figures above show that the performance of autoencoder in comparison to other traditional machine learning algorithms.

6.4 Evaluation of OADB based on defined requirements

In this section, we are evaluating proposed Open Anomaly Detection Benchmark (OADB) for the requirements mentioned in Section 5.2.

- **R1:** OADB has the capability to compare accuracy performance. OADB save accuracy results of algorithms in results folder.
- **R2:** OADB provides capability for measuring computational complexity of algorithms. It also provides information about time taken taken during

training and test phase.

- **R3:** OADB allows integration of new algorithm and dataset. Integration of new algorithm and dataset is two step process. This process is described in Section 5.8.
- **R4:** OADB provides functionality to visualize the performance of algorithms as heat map. One can do in depth analysis by clicking on cells in heat map and even visualize the performance of each algorithm against each dataset. Accuracy can be evaluated using F1, precision, recall, average precision score metrics. Figure. 6.4, 6.14 and 6.15 show visualization is divided into 4 stages heat map, bar chart showing average score for sub folders of a data repository, bar chart showing scores on dataset files in sub folders and detailed view of algorithm performance on selected dataset.

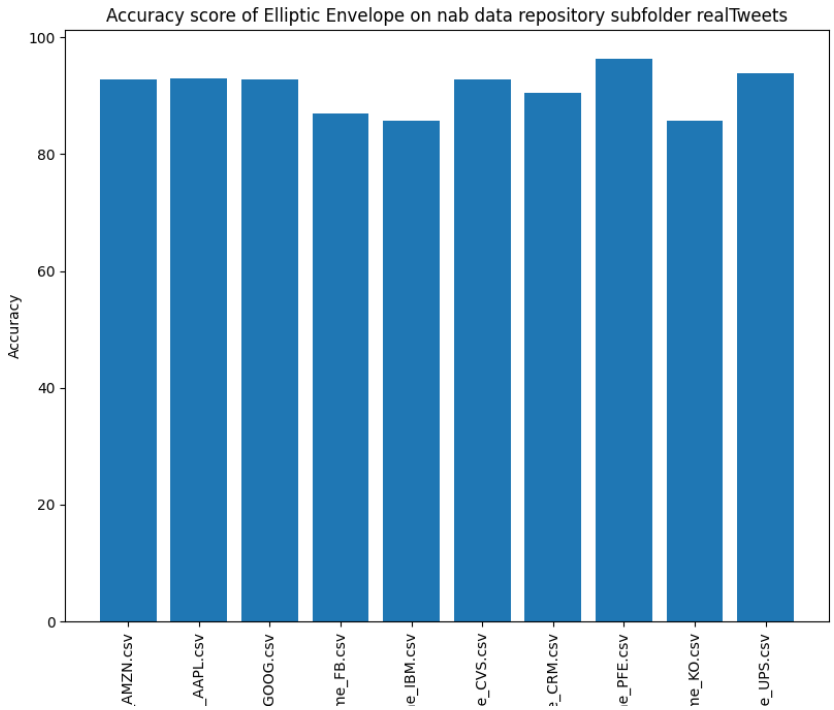


Figure 6.14: Example of OADB visualization on data folder level

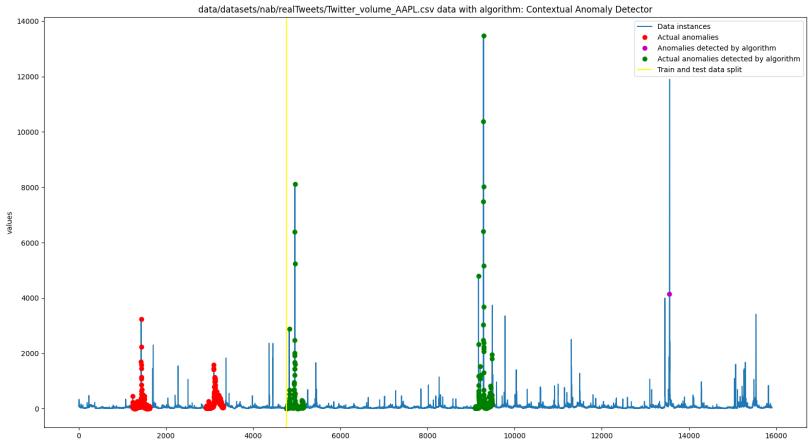


Figure 6.15: Example of OADB visualization of an algorithm accuracy performance against single dataset file

- **R5:** Process for expansion of benchmark with new algorithms and datasets is explained in Section 5.8. It also has detailed documentation about its working documentation. Documentation and source code of benchmark will be available in a public repository on Github.
- **R6:** OADB can be configured to work in certain way both by passing parameters for running algorithms and visualizing results. In running phase, one can provide parameters to adjust the training and test data split size. Similarly deciding to update existing results or not. In visualising phase, one can decide to evaluate algorithms accuracy on desired metric by passing accuracy measure metric parameter.
- **R7:** Most of datasets included in OADB are collected from real world e.g. Yahoo, NAB datasets.
- **R8:** OADB has datasets of wide variety with different sizes, source and characteristics. It is evident from analysis done in Section 6.2.
- **R9:** Analysis of trade offs between training data size and accuracy can be done by running benchmark with multiple training test split sizes. There is room for improvement in OADB to better fulfil this requirement.
- **R10:** OADB can be configured to be used for point and aggregate anomalies by passing "use_windows". If set to true, OADB marks aggregate anomaly detected even only on data instance in anomalous group is detected. OADB can be improved to address the need for contextual anomaly evaluation.
- **R11:** Autoencoder is only deep learning algorithm integrated in OADB. This allows comparison of traditional machine learning and deep learning. More deep learning algorithms should be added in OADB to get correct evaluation of deep learning algorithms for anomaly detection.

Figure. 6.3 provide summary about how many requirements are fulfilled by OADB.

Benchmark	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	Total score
OADB	✓	✓	✓	✓	✓	✓	✓	✓	*	*	*	9.5

Table 6.3: Evaluation of Open Anomaly Detection Benchmark (OADB) based on defined requirements

✓ represent benchmark fulfill the requirement and assigned 1 score.
✗ represent benchmark does not fulfill the requirement and assigned 0 score.
* represent benchmark partially fulfill the requirement and assigned 0.5 score.

It is evident from analysis that OADB has performed well according to our defined requirements. OADB has good visualization capability to provide overview of algorithms performance in terms of accuracy and computational complexity. OADB allows integration of both traditional machine learning and deep learning algorithms while existing benchmarks focus on only one e.g. NAB. OADB has univariate and multivariate datasets and has some algorithms working with only univariate datasets. It allows the integration of a maximum number of algorithms used for anomaly detection in OADB.

OADB is available as part of DataBench Toolbox. DataBench Toolbox has been created as part of DataBench project to provide a better way to search and select benchmarks according to needs of a user and problem domain. DataBench project aimed to make benchmarking process easier [23].

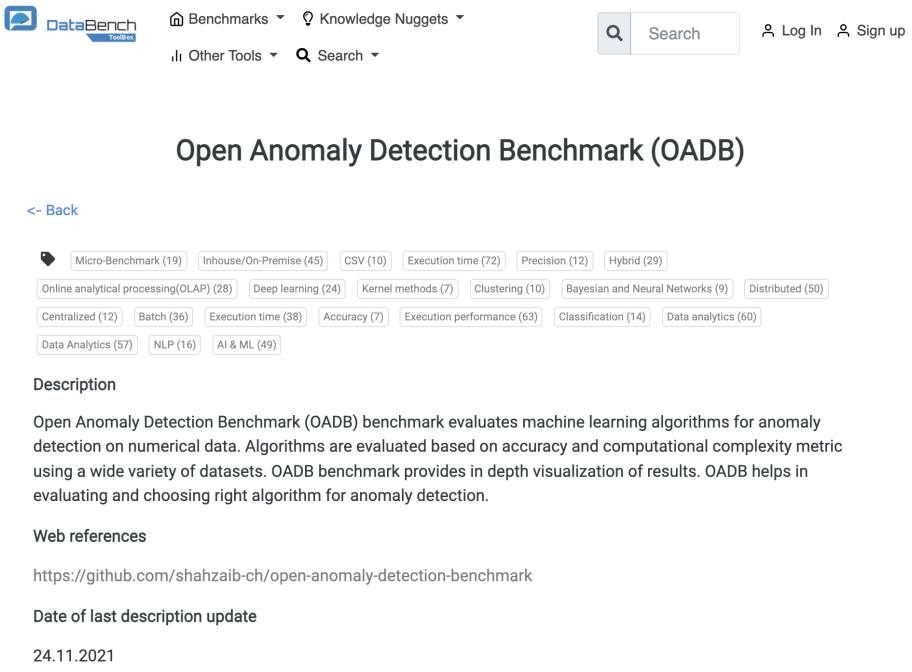


Figure 6.16: OADB as part of DataBench toolbox

Part IV

Summary and outlook

Chapter 7

Conclusion and future work

This chapter provides a summary of benchmarking experiments done in the previous chapter and conclusions drawn. As every scientific research faces some limitations, we also face a fair share of limitations. We have mentioned the limitations and defined the scope of the thesis in this chapter. Afterward, we have pointed out a few directions for future research.

7.1 Contributions summary

We have provided a comprehensive analysis of existing benchmarks focused on evaluating and comparing machine learning algorithms for anomaly detection in numerical data. These benchmarks are evaluated regarding a particular set of essential requirements relevant to practitioners and their shortcomings are outlined. We have summarized challenges in anomaly detection and developing a good anomaly detection benchmark. Various benchmarks perform well in some aspects of benchmarking and have shortcomings in others.

A new benchmark, Open Anomaly Detection Benchmark (OADB), is proposed and developed to fill the gaps of the existing benchmarks. This benchmark focuses on providing a wide variety of datasets for evaluating machine learning algorithms for anomaly detection. The proposed Open Anomaly Detection Benchmark (OADB) also focuses on transparency in results and provides the ability to view results in depth.

We have identified popular machine learning algorithms used for anomaly detection in numerical data. Also, integrated these algorithms in OADB and evaluated them. Our interpretation of results is provided along with directions for future work.

7.2 Scope and limitations

Anomaly detection can be divided into different branches based on datatype e.g. video, image, text, numerical. In this study, we have focused on univariate and multivariate numerical data for the evaluation of algorithms.

There has been a limited focus on tuning machine learning algorithms to perform better on each dataset. Our focus has been to evaluate algorithms on a wide variety of datasets without assuming anything about the dataset.

Another limitation is, fewer benchmarking experiments using OADB are done. It is due to high computational power and the longer time required to complete each experiment.

Only one deep learning algorithm, Autoencoder, is integrated into OADB for evaluation. More deep learning algorithms should be integrated to better understand the tradeoffs between traditional machine learning and deep learning algorithms. Many more traditional machine learning algorithms are available that can be integrated into OADB.

7.3 Conclusions

We have drawn following conclusions from this study to answer our research questions as described in Section 1.2.

Conclusion for Q1: We have identified essential requirements for a good anomaly detection benchmark, described in Section 5.2. In-depth visualization of results, extensibility, and a wide variety of datasets are the most important requirements among all.

Conclusion for Q2: In terms of accuracy, there is no single machine learning algorithm that can perform best on all datasets. One needs to have a closer look at datasets and consider algorithm performance on datasets relevant to them. Some algorithms perform better on specific datasets and not so well on other datasets.

Windowed Gaussian algorithm on univariate datasets and Isolation Forest on multivariate datasets have the highest precision and f1 score on average. Clustering Based Local Outlier Factor has the relatively highest recall score among the algorithms integrated into OADB. In contrast, One-class SVM has best recall score on multivariate datasets. These are results based on anomaly threshold optimized by the algorithm based on contamination in the dataset. Elliptic envelope has the highest average precision score, followed by Windowed Gaussian. Isolation forest has the highest average precision score on multivariate datasets. Considering all the important metrics for accuracy, we conclude that Windowed Gaussian on univariate datasets and Isolation forest on multivariate datasets perform better for anomaly detection than other machine learning algorithms integrated into the OADB.

In terms of computational complexity, Elliptic envelope performs better on univariate datasets but with an increasing number of features it performance degrade. Isolation forest followed by elliptic envelope has low computational complexity on univariate datasets with increasing size of dataset. Isolation forest also scales much better than other algorithms in OADB, with an increase in both the number of features and data instances in the dataset.

Conclusion for Q3: Computational complexity of algorithms generally increase with size of dataset. However, this increase is not same for all algorithms. KNN CAD highest computational complexity and it is due to compute intensive operation such as matrix calculations. After it, One-class SVM and KNN are highly sensitive to dataset size.

7.4 Future work

Our work done in this thesis can be expanded in various dimensions. One dimension could be integrating more datasets in OADB with various characteristics e.g. datasets with contextual anomalies. Then, dividing datasets into different categories based on anomaly types and evaluating algorithms for each category of datasets.

Another dimension is to add more deep learning algorithms in OADB to compare them with traditional algorithms and evaluate their performance. It will help find the conditions when it is more favorable to deep learning or traditional machine learning.

One direction is to investigate the following questions. At what size of the dataset does deep learning start to outperform traditional machine learning algorithms? What is the impact of an increasing size of training data on the accuracy of an algorithm for anomaly detection? It will be beneficial for researchers in decision-making for the choice of algorithm.

Bibliography

[1] *A smoother ride and a more detailed Map thanks to AI*. Google. URL: https://blog.google/products/maps/google-maps-101-ai-power-new-features-io-2021/.

[2] Charu Aggarwal. *Data Classification*. Chap. Instance-based Learning: A Survey. DOI: https://doi.org/10.1201/b17320.

[3] Charu Aggarwal, Alexander Hinneburg and Daniel Keim. 'On the Surprising Behavior of Distance Metric in High-Dimensional Space'. In: *International Conference on Database Theory* (1st Feb. 2002).

[4] Anant Agrawal et al. 'Disease Prediction Using Machine Learning'. In: *Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT)* ID 3167431 (23rd Apr. 2018). DOI: 10.2139/ssrn.3167431.

[5] Subutai Ahmad et al. 'Unsupervised real-time anomaly detection for streaming data'. In: *Neurocomputing*. Online Real-Time Learning Strategies for Data Streams 262 (1st Nov. 2017). ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.04.070.

[6] Stamatios-Aggelos N. Alexandropoulos, Sotiris B. Kotsiantis and Michael N. Vrahatis. 'Data preprocessing in predictive data mining'. In: *The Knowledge Engineering Review* 34 (2019). ISSN: 0269-8889, 1469-8005. DOI: 10.1017/S026988891800036X.

[7] Redhwan Al-amri et al. 'A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data'. In: *Applied Sciences* 11 (Jan. 2021). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/app11125320.

[8] G. Anand and Rambabu Kodali. 'Benchmarking the benchmarking models'. In: *Benchmarking: An International Journal* 15 (30th May 2008). ISSN: 1463-5771. DOI: 10.1108/14635770810876593.

[9] Archana Anandakrishnan et al. 'Anomaly Detection in Finance: Editors' Introduction'. In: *Proceedings of the KDD 2017* (2017).

[10] *Anodot | Anomaly Detection for Business Monitoring*. Anodot. URL: https://www.anodot.com.

[11] *Anomaly Detection Toolkit (ADTK) — ADTK 0.6.2 documentation*. URL: https://adtk.readthedocs.io/en/stable/index.html.

[12] Stephen D. Bay et al. *The UCI KDD Archive*. ACM SIGKDD Explorations Newsletter. 1999. URL: http://kdd.ics.uci.edu/.

[13] Nitin Bhatia and Vandana. 'Survey of Nearest Neighbor Techniques'. In: *IJCSIS vol. 8, No. 2, 2010* (1st July 2010). arXiv: 1007.0085.

[14] Bob Ferrell and Steven Santuro. *NASA Shuttle Valve Data*. 2005. URL: http://www.cs.fit.edu/~pkc/nasa/data/.

[15] Mohammad Braei and Dr-Ing Sebastian Wagner. *Anomaly detection in univariate time-series: a survey on state-of-the-art*. 2020.

[16] Evgeny Burnaev and Vladislav Ishimtsev. *Conformalized density- and distance-based anomaly detection in time-series data*. 16th Aug. 2016. arXiv: 1608.04585.

[17] Guilherme O. Campos et al. 'On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study'. In: *Data Mining and Knowledge Discovery* 30.4 (July 2016), pp. 891–927. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-015-0444-8.

[18] Raghavendra Chalapathy and Sanjay Chawla. *Deep Learning for Anomaly Detection: A Survey*. 23rd Jan. 2019. arXiv: 1901.03407.

[19] Raghavendra Chalapathy, Aditya Krishna Menon and Sanjay Chawla. *Anomaly Detection using One-Class Neural Networks*. 10th Jan. 2019. arXiv: 1802.06360.

[20] Varun Chandola, Arindam Banerjee and Vipin Kumar. 'Anomaly Detection: A Survey'. In: *ACM Comput. Surv.* 41 (1st July 2009). DOI: 10.1145/1541880.1541882.

[21] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper*. 2019. URL: https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf.

[22] 'Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress'. In: (26th Aug. 2021). arXiv: 2009.13807. URL: http://arxiv.org/abs/2009.13807.

[23] *DataBench – Big Data Benchmarking project*. URL: https://www.databench.eu/.

[24] *DeepBench*. 26th Sept. 2021. URL: https://github.com/baidu-research/DeepBench.

[25] Youcef Djenouri et al. 'A Survey on Urban Traffic Anomalies Detection Algorithms'. In: *IEEE Access* 7 (2019), pp. 12192–12205. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2893124. URL: https://ieeexplore.ieee.org/document/8612931/.

[26] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019. URL: https://archive.ics.uci.edu/ml.

[27] *EGADS Java Library*. original-date: 2015-05-06T17:47:52Z. 9th Aug. 2021. URL: https://github.com/yahoo/egads.

[28] Salvador García, Julián Luengo and Francisco Herrera. *Data Preprocessing in Data Mining*. Vol. 72. Intelligent Systems Reference Library. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-10246-7 978-3-319-10247-4. DOI: 10.1007/978-3-319-10247-4. URL: http://link.springer.com/10.1007/978-3-319-10247-4.

[29] Markus Goldstein. *Anomaly detection in large datasets*. URL: http://www.gbv.de/dms/tib-ub-hannover/787899437.pdf.

[30] Laura Graesser and Wah Loon Keng. *1. Introduction to Reinforcement Learning - Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. ISBN: 9780135172490. 2019. URL: https://learning.oreilly.com/library/view/foundations-of-deep/9780135172490/ch01.xhtml.

[31] Johannes Grohmann et al. 'A Taxonomy of Techniques for SLO Failure Prediction in Software Systems'. In: *Computers* 9.1 (11th Feb. 2020), p. 10. ISSN: 2073-431X. DOI: 10.3390/computers9010010.

[32] Gavin Hackeling. *Mastering Machine Learning with scikit-learn, Chapter 2*. Second Edition. ISBN: 978-1-78829-987-9. URL: https://learning.oreilly.com/library/view/mastering-machine-learning/9781788299879/.

[33] Fouzi Harrou et al. 'An unsupervised monitoring procedure for detecting anomalies in photovoltaic systems using a one-class Support Vector Machine'. In: *Solar Energy* 179 (1st Feb. 2019), pp. 48–58. ISSN: 0038-092X.

[34] H. M. Hashemian and Wendell C. Bean. *State-of-the-Art Predictive Maintenance Techniques*. 2010.

[35] David Heckerman. 'A Tutorial on Learning With Bayesian Networks'. In: *arXiv:2002.00269 [cs, stat]* (8th Mar. 2021). arXiv: 2002.00269. URL: http://arxiv.org/abs/2002.00269.

[36] Chaudhry Rehan Ikram. 'A benchmark for evaluating Deep Learning based Image Analytics'. In: (2019). Accepted: 2019-08-26T23:46:34Z. URL: https://www.duo.uio.no/handle/10852/69588.

[37] Vincent Jacob et al. *Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series*.

[38] D. Jakhar and I. Kaur. 'Artificial intelligence, machine learning and deep learning: definitions and differences'. In: *Clinical and Experimental Dermatology* 45.1 (2020), pp. 131–132. ISSN: 1365-2230. DOI: 10.1111/ced.14029.

[39] Andreas Kaplan and Michael Haenlein. 'Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence'. In: *Business Horizons* 62.1 (1st Jan. 2019), pp. 15–25. ISSN: 0007-6813. DOI: 10.1016/j.bushor.2018.08.004.

[40] Md Karbir, Abdur Onik and Tanvir Samad. 'A Network Intrusion Detection Framework based on Bayesian Network using Wrapper Approach'. In: *International Journal of Computer Applications* 166 (1st Apr. 2017), pp. 975–8887. DOI: 10.5120/ijca2017913992.

[41] Iurii D. Katser and Vyacheslav O. Kozitsin. *Skoltech Anomaly Benchmark (SKAB)*. Type: dataset. DOI: 10.34740/KAGGLE/DSV/1693952.

[42] Jóakim von Kistowski et al. 'How to Build a Benchmark'. In: ICPE 2015 - Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. 1st Feb. 2015. DOI: 10.1145/2668930.2688819.

[43] Rocco Langone, Alfredo Cuzzocrea and Nikolaos Skantzos. 'Interpretable Anomaly Prediction: Predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools'. In: *Data & Knowledge Engineering* 130 (Nov. 2020). ISSN: 0169023X. DOI: 10.1016/j.datak.2020.101850.

[44] Alexander Lavin and Subutai Ahmad. *Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark*. 12th Oct. 2015. DOI: 10.1109/ICMLA.2015.141.

[45] Yang Li et al. 'Network anomaly detection based on TCM-KNN algorithm'. In: *Proceedings of the 2nd ACM symposium on Information, computer and communications security*. ASIACCS '07. New York, NY, USA: Association for Computing Machinery, 20th Mar. 2007, pp. 13–19. ISBN: 978-1-59593-574-8. DOI: 10.1145/1229285.1229292.

[46] Konstantinos G. Liakos et al. *Machine Learning in Agriculture: A Review*. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute. Aug. 2018. DOI: 10.3390/s18082674. URL: https://www.mdpi.com/1424-8220/18/8/2674.

[47] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. 'Isolation Forest'. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008 Eighth IEEE International Conference on Data Mining. ISSN: 2374-8486. Dec. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[48] Wei Lu and Issa Traore. 'Unsupervised anomaly detection using an evolutionary extension of k-means algorithm'. In: *International Journal of Information and Computer Security* 2 (1st Jan. 2008). Publisher: Inderscience Publishers, pp. 107–139. ISSN: 1744-1765. DOI: 10.1504/IJICS.2008.018513.

[49] Yafeng Lu et al. *The State-of-the-Art in Predictive Visual Analytics*. 2017.

[50] Luis Martí et al. 'Anomaly Detection Based on Sensor Data in Petroleum Industry Applications'. In: *Sensors* 15.2 (Feb. 2015). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, pp. 2774–2797. DOI: 10.3390/s150202774.

[51] Peter Mattson et al. 'MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance'. In: *IEEE Micro* 40.2 (Mar. 2020). Conference Name: IEEE Micro, pp. 8–16. ISSN: 1937-4143. DOI: 10.1109/MM.2020.2974843.

[52] Kishan G. Mehrotra, Chilukuri K. Mohan and HuaMing Huang. *Anomaly Detection Principles and Algorithms*. Terrorism, Security, and Computation. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-67524-4. DOI: 10.1007/978-3-319-67526-8.

[53] Mehryar Mohri, Afshin Rostamizadeh and Ameet Talwalkar. *Foundations of Machine Learning, second edition*. MIT Press, 25th Dec. 2018. 505 pp. ISBN: 978-0-262-35136-2.

[54] Avashlin Moodley. 'Language Identification With Decision Trees: Identification Of Individual Words In The South African Languages'. PhD thesis. 31st Jan. 2016. DOI: 10.13140/RG.2.2.25539.81445.

[55] Mohsin Munir et al. 'A Comparative Analysis of Traditional and Deep Learning-Based Anomaly Detection Methods for Streaming Data'. In: 16th Dec. 2019. DOI: 10.1109/ICMLA.2019.00105.

[56] Mohsin Munir et al. 'DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series'. In: *IEEE Access* PP (19th Dec. 2018), pp. 1–1. DOI: 10.1109/ACCESS.2018.2886457.

[57] Arundhati Navada et al. 'Overview of use of decision tree algorithms in machine learning'. In: *2011 IEEE Control and System Graduate Research Colloquium*. 2011 IEEE Control and System Graduate Research Colloquium. June 2011, pp. 37–42. DOI: 10.1109/ICSGRC.2011.5991826.

[58] Md Asri Ngadi and Salima Benqdara. *Machine Learning Techniques for Anomaly Detection: An Overview*. 2013.

[59] *Nsl-kdd data set for network-based intrusion detection systems*. 2009. URL: https://www.unb.ca/cic/datasets/nsl.html.

[60] Randal S. Olson et al. 'PMLB: a large benchmark suite for machine learning evaluation and comparison'. In: *BioData Mining* 10.1 (11th Dec. 2017), p. 36. ISSN: 1756-0381. DOI: 10.1186/s13040-017-0154-4.

[61] Guansong Pang, Chunhua Shen and Anton van den Hengel. *Deep Anomaly Detection with Deviation Networks*. 19th Nov. 2019. arXiv: 1911.08623.

[62] Guansong Pang et al. 'Deep Learning for Anomaly Detection: A Review'. In: *ACM Computing Surveys* 54.2 (Apr. 2021), pp. 1–38. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3439950. arXiv: 2007.02500.

[63] Karishma Pawar and Vahida Attar. 'Deep learning approaches for video-based anomalous activity detection'. In: *World Wide Web* 22 (1st Mar. 2019). DOI: 10.1007/s11280-018-0582-1.

[64] Shebuti Rayana. *{ODDS} Library*. 2016. URL: http://odds.cs.stonybrook.edu.

[65] *S5 - A Labeled Anomaly Detection Dataset, version 1.0*. URL: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s.

[66] Edin Sabic et al. 'Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data'. In: *AI & SOCIETY* 36 (1st Mar. 2021). DOI: 10.1007/s00146-020-00985-1.

[67] Sachin Salunkhe et al. 'Prediction of life of piercing punches using artificial neural network and adaptive neuro fuzzy inference systems'. In: *International Journal of Materials Engineering Innovation* 10 (7th Feb. 2019), pp. 20–33. DOI: 10.1504/IJMATEI.2019.10019116.

[68] A. L. Samuel. 'Some Studies in Machine Learning Using the Game of Checkers'. In: *IBM Journal of Research and Development* 3.3 (July 1959). Conference Name: IBM Journal of Research and Development, pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210.

[69] Hassan Sarmadi and Abbas Karamodin. 'A novel anomaly detection method based on adaptive Mahalanobis-squared distance and one-class kNN rule for structural health monitoring under environmental effects'. In: *Mechanical Systems and Signal Processing* 140 (9th June 2020). DOI: 10.1016/j.ymssp.2019.106495.

[70] Yutaka Sasaki. 'The truth of the F-measure'. In: (), p. 5.

[71] *scikit-learn: machine learning in Python — scikit-learn 1.0 documentation*. URL: https://scikit-learn.org/stable/.

[72] *Seasonal Hybrid ESD (S-H-ESD) AnomalyDetection R package*. original-date: 2014-12-09T17:46:24Z. 9th Aug. 2021. URL: https://github.com/twitter/AnomalyDetection.

[73] Elham Serkani et al. 'Hybrid Anomaly Detection Using Decision Tree and Support Vector Machine'. In: *International Journal of Electrical and Computer Engineering* 12.6 (1st May 2018), pp. 431–436. URL: https://publications.waset.org/10009167/hybrid-anomaly-detection-using-decision-tree-and-support-vector-machine.

[74] Hyun Joon Shin, Dong-Hwan Eom and Sung-Shick Kim. 'One-class support vector machines—an application in machine fault detection and classification'. In: *Computers & Industrial Engineering* 48.2 (1st Mar. 2005), pp. 395–408. ISSN: 0360-8352. DOI: 10.1016/j.cie.2005.01.009.

[75] K. Sindhu Meena and S. Suriya. 'A Survey on Supervised and Unsupervised Learning Techniques'. In: *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*. Ed. by L. Ashok Kumar, L. S. Jayashree and R. Manimegalai. Cham: Springer International Publishing, 2020, pp. 627–644. ISBN: 978-3-030-24051-6. DOI: 10.1007/978-3-030-24051-6_58.

[76] Ida Solheim and Ketil Stølen. *Technology research explained*. 1st Mar. 2007.

[77] Ming-Yang Su. 'Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers'. In: *Expert Systems with Applications* 38.4 (1st Apr. 2011), pp. 3492–3498. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2010.08.137.

[78] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*.

[79] Mahbod Tavallaee et al. 'A detailed analysis of the KDD CUP 99 data set'. In: *IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA* 2 (1st July 2009). DOI: 10.1109/CISDA.2009.5356528.

[80] Kai Ming Ting. 'Confusion Matrix'. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 209–209. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_157.

[81] Aditya Varshney. *Support Vector Machines and the Kernel Trick*. The Startup. 24th Nov. 2020. URL: https://medium.com/swlh/support-vector-machines-and-the-kernel-trick-f946991ebc76.

[82] Pei Wang. 'On Defining Artificial Intelligence'. In: *Journal of Artificial General Intelligence* 10.2 (1st Jan. 2019), pp. 1–37. ISSN: 1946-0163. DOI: 10.2478/jagi-2019-0002.

[83] Weng-Keen Wong et al. 'Bayesian Network Anomaly Pattern Detection for Disease Outbreaks'. In: (), p. 8.

[84] Yang Xin et al. 'Machine Learning and Deep Learning Methods for Cybersecurity'. In: *IEEE Access* 6 (2018). Conference Name: IEEE Access, pp. 35365–35381. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2836950.

[85] Chunyong Yin et al. 'An Improved K-Means Using in Anomaly Detection'. In: *2015 First International Conference on Computational Intelligence Theory, Systems and Applications (CCITSA)*. 2015 First International Conference on Computational Intelligence Theory, Systems and Applications (CCITSA). Dec. 2015, pp. 129–132. DOI: 10.1109/CCITSA.2015.11.

[86] Chunhui Yuan and Haitao Yang. 'Research on K-Value Selection Method of K-Means Clustering Algorithm'. In: *J* 2 (18th June 2019), pp. 226–235. DOI: 10.3390/j2020016.

[87] Rui Zhang et al. *Network Anomaly Detection Using One Class Support Vector Machine*.

[88] Yue Zhao, Zain Nasrullah and Zheng Li. 'PyOD: A Python Toolbox for Scalable Outlier Detection'. In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[89] Zhi-Hua Zhou. 'Ensemble Learning'. In: *Machine Learning*. Ed. by Zhi-Hua Zhou. Springer Singapore, 2021, pp. 181–210. ISBN: 978-981-15-1967-3. DOI: 10.1007/978-981-15-1967-3_8.

Appendix A

Benchmark source code

We have made Open Anomaly Detection Benchmark code with documentation available to everyone at Github. Other people working with anomaly detection algorithms can utilize this and hopefully contribute to make it better in the future. The repository can be found on following link.

https://github.com/shahzaib-ch/open-anomaly-detection-benchmark
Following is the structure of OADB with description of vital folders and files.

```
Open Anomaly Detection Benchmark
├── README.md (Documentation for benchmark)
├── analysis (Contains code for analysis of datasets)
│   └── data_folders_analysis.py
├── core (Contains code for tuning algorithms on datasets)
│   └── core.py
├── data (Contains datasets and code for organising them)
│   ├── README.md
│   ├── dataset_collector.py
│   └── datasets
│       ├── nab
│       ├── odd
│       ├── ucr
│       └── yahoo
└── detector (Contains implementation of ML algorithms)
    ├── angle_based_outlier_detector.py
    ├── auto_encoder.py
    ├── base_detector.py
    ├── bayes_change_point.py
    ├── clustering_based_local_outlier_factor.py
    ├── contextual_anomaly_detector.py
    ├── detector_aggregator.py
    ├── elliptic_envelope.py
    ├── expose.py
    ├── generalized_esd_test.py
    ├── isolation_forest.py
    ├── k_nearest_neighbors.py
    ├── knn_cad.py
    └── local_outlier_factor.py
```

```
    ├── one_class_support_vector_machine.py
    ├── principal_component_analysis.py
    ├── relative_entropy.py
    └── windowed_gaussian.py
├── helper (Contains generic helper functions to be used in the project)
    ├── ResultWindowLabeler.py
    ├── common_methods.py
    ├── dataset_viewer.py
    └── labels_helper.py
├── main.py (Entry point for running benchmark)
├── play.py (A python file to run code fragments when desired)
├── preprocessor (Contains code to pre process datasets before use)
├── preprocessor.py
├── requirements.txt (Contains list of python packages required to
    run this project)
├── result (Folder to save result of benchmarking experiment)
├── result_accuracy_visualizer.py (Entry point for visualising results)
└── visualizer (Contains code for visualizing results of benchmark)
    ├── heatmap_helper.py
    ├── result_accuracy_visualizer.py
    ├── result_collector.py
    ├── result_data_keys.py
    └── result_metric_calculators.py
```

Appendix B

Detailed benchmarking results

We have presented and analyzed important benchmarking results generated by OADB. However, full detailed results not shown. Following are detailed computational complexity results.

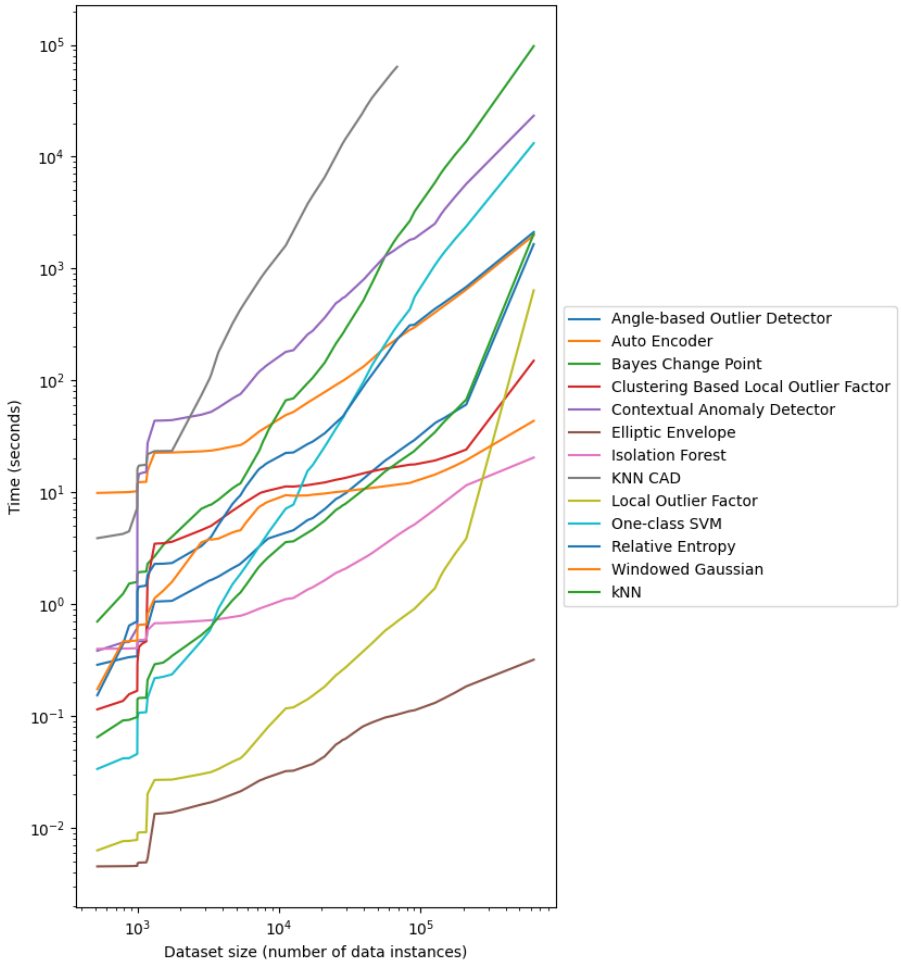


Figure B.1: Time taken by algorithms for training phase on univariate datasets

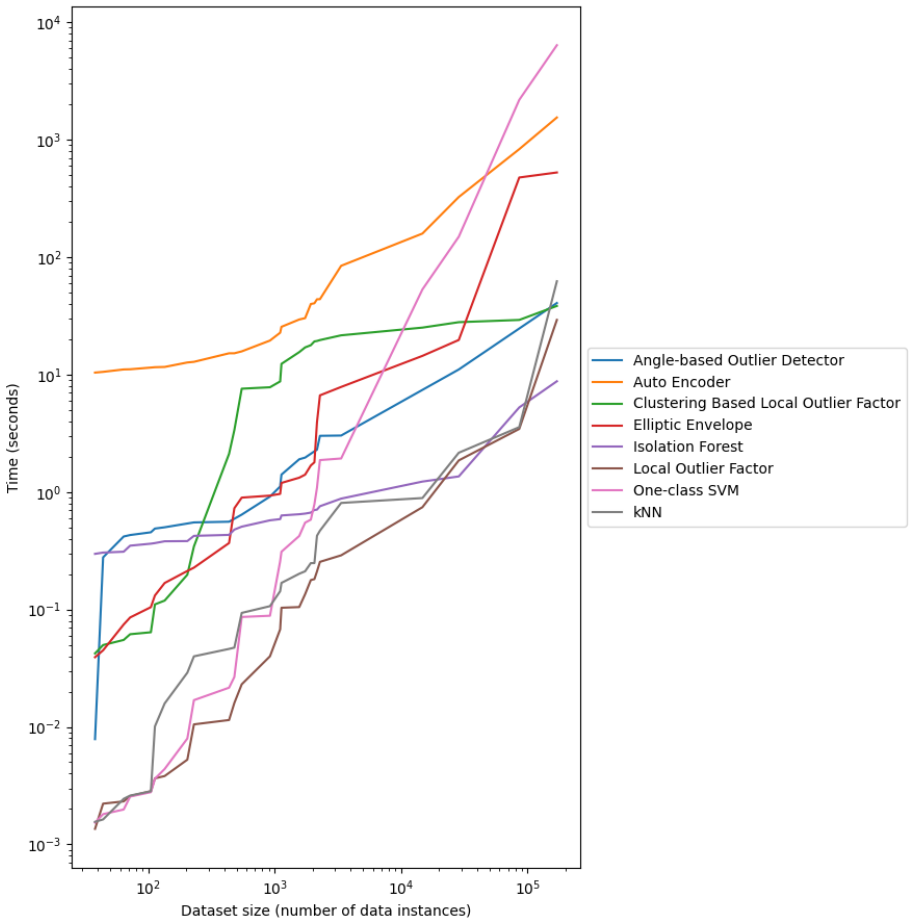


Figure B.2: Time taken by algorithms for training phase on multivariate datasets

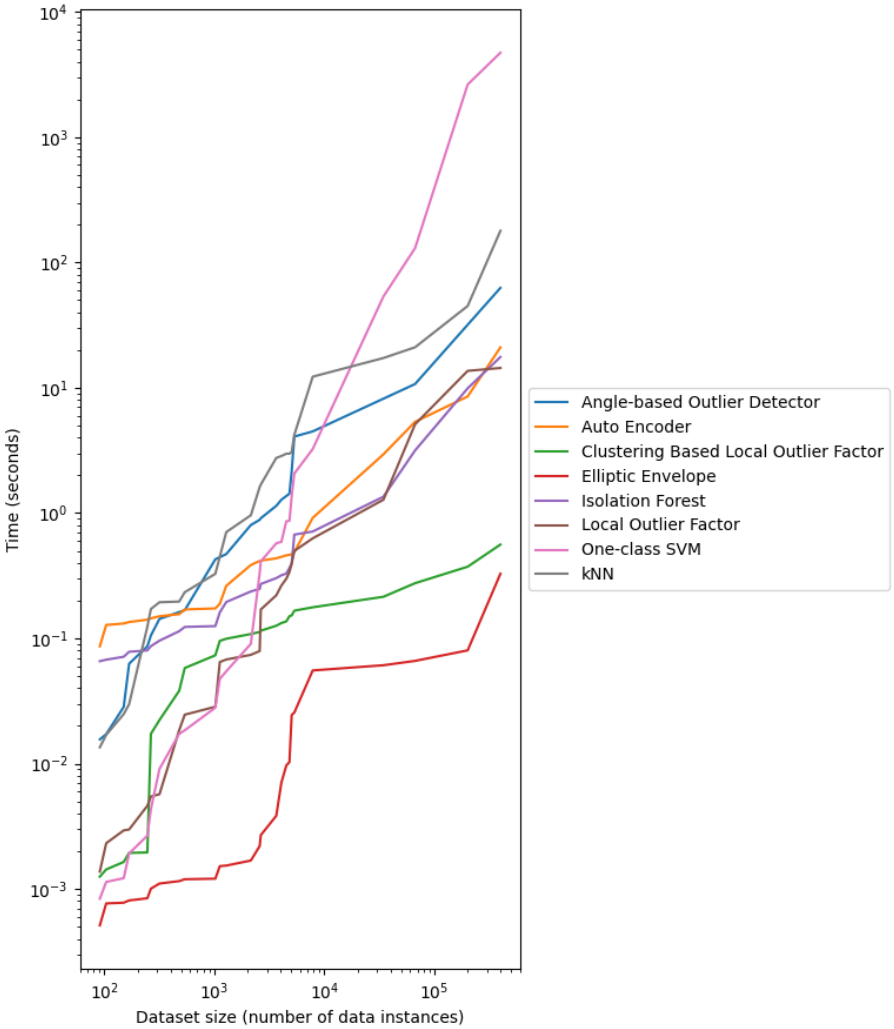


Figure B.3: Time taken by algorithms for test phase on multivariate datasets

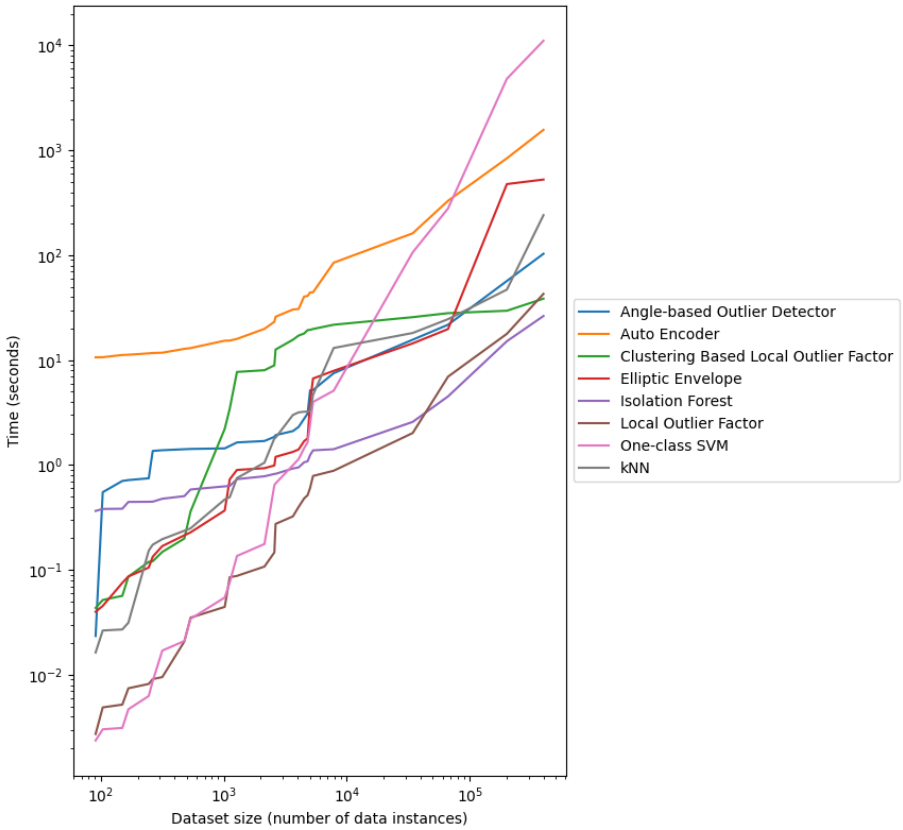


Figure B.4: Time taken by algorithms for training phase on multivariate datasets