Using Anomaly Detection to investigate suspicious activity in sports betting

Matteo Farrugia

Supervisor: Prof. Matthew Montebello

Co-supervisor: Dr. Vincent Vella

June, 2024

Submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.

L-Università ta' Malta
Faculty of Information & Communication Technology

Acknowledgements

I would like to thank my family and friends for their continuous support as I completed this dissertation. In particular, I would like to thank my parents for instilling a desire to learn from an early age and for their encouragement even when times were tough. I would also like to thank my close friends, who were always ready to provide a listening ear and support me throughout the journey. Furthermore, I would also like to thank my fellow leaders within my Scout Group, who continuously provided motivation, support and encouragement from start to finish. Moreover, I would like to thank my girlfriend who unwaveringly stood by my side from the very first moment of this dissertation, always providing her unconditional love, support and encouragement, especially when it was needed the most.
I would also like to express my gratitude towards my employers and my colleagues as without their continuous effort to provide support, this dissertation would not have been possible. Finally, I would also like to thank both my supervisors, Professor Matthew Montebello and Dr. Vincent Vella for their ongoing and continuous patience and support throughout the dissertation from the very beginning. Without their contribution, this dissertation would surely not have been as fruitful as it was. In the case of Professor Montebello, I would also like to show my appreciation for all the guidance that he provided even before this dissertation, from the start of my journey as an undergraduate and all the way through my postgraduate studies.

Abstract

The past few years have seen surges in popularity across the sports betting industry, with several countries removing laws banning sports betting, while betting operators continue to develop their online platforms, enabling customers to place bets from the comfort of their own homes.
The research performed involved the application of several different Artificial Intelligence techniques to sports betting data to detect and identify suspicious bets within the dataset, including Anomaly Detection. These suspicious or anomalous bets could represent customers who are particularly desirable for betting operators, customers who are undesirable for betting operators, or customers who are using certain knowledge to attempt to gain an illicit advantage against the betting operator. Suspicious betting patterns could also indicate potential criminal activity, such as fraud, money laundering and match-fixing.
The first of two experiments involved the development of an Anomaly Detector that served as a binary classifier, indicating only if a given bet is anomalous or not. Once the Anomaly Detector was implemented, several enhancements were applied to improve the performance of this detector by tackling some of the main challenges known to be present within Anomaly Detection problems. These challenges include the Class Imbalance Problem and Concept Drift. The former is caused since anomalies, by definition, are rarely observed within a dataset. Meanwhile, the latter is caused by changes in the underlying data distribution as time passes.
Meanwhile, the second experiment focused on expanding the Anomaly Detector into a multi-class Customer Classifier. This classifier would be capable of differentiating between the various types of anomalies present within the dataset. During the experiment, three different classifiers were compared to establish which would best perform the task at hand. Once the best-performing algorithm was identified, similar improvements to those applied to the Anomaly Detector performing binary classification were applied to the multi-class classifier to address the same two aforementioned problems.
From the results obtained, the enhancements that were applied to the Anomaly Detector successfully improved its performance. Furthermore, a Random Forest proved to be the best multi-class classifier, and once enhanced, it proved to be successful when compared to literature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few years, the sports betting industry has undergone a considerable increase in popularity, with the industry as a whole experiencing a period of significant financial gains with every passing year [1, 2]. Such a surge in popularity and usage has resulted in the global market value of the sports betting industry ballooning to a value of approximately 243 billion US dollars [1]. Several reasons for these increases exist, with one example being the lifting of bans that were in place in countries such as the United States and Canada, which had previously outright prohibited sports betting [1]. The amendments made in the legal framework of such countries have opened up a whole new market for operators to access and enabled the industry to thrive [1]. Another easily identifiable underlying cause for the observed increases originates from a change in mindset of betting operators, namely the transition from a physical venue to mostly online platforms [1, 2]. This brought about a huge increase in availability [1, 2], which proved critical during the lockdown caused by the COVID-19 pandemic, as it allowed people to place bets from the comfort and safety of their own homes.

However, for as long as people have been competing to earn some form of reward, there have been those who sought to place themselves at an advantage compared to their competitors. Such is this desire to win, that some individuals are willing to resort to illicit lengths to strengthen their position compared to their rivals [3]. This extends even to modern sports, with numerous cases across several different sports having been exposed in the past few years [3]. In 2016, Russian athletes were found to have participated in a state-sponsored doping scheme, with Russian authorities also attempting to conceal the scandal and prevent any of its athletes from testing positive during drug tests [4]. A large number of Russian athletes were then prohibited from competing in a wide range of sporting events, including over 100 athletes who were deemed ineligible to compete at the Rio 2016 Olympic Games [4]. Similar logic can also be extended to games that are

based around gambling, with players having developed a range of different techniques with the objective of placing themselves in a more advantageous position compared to the betting operator or other competing players. One of the more well-known techniques is that of card counting, which is commonly applied to games such as Blackjack [5]. Although this practice is not strictly illegal, as opposed to the doping example mentioned earlier, its use can significantly empower those who have the ability to use it with a massive advantage over the dealer [5]. It can therefore be concluded that betting operators must keep a watchful eye for customers who seek to obtain an edge over others, particularly by utilising certain knowledge that other players may not have at their disposal [6, 7, 8].

Since there is an ever-increasing amount of betting activity taking place through online means, a much larger volume of data is being generated and gathered about players and their behaviours by major betting operators. With certain players being ready to resort to illicit methods of gaining an advantage, it is this very same data that is the most prominent tool at the disposal of gambling operators when it comes to identifying those players who are maliciously making use of their own knowledge. Normally, such players who have sufficient additional knowledge to make a difference represent only a very small portion of the clients that a gambling operator has. Owing to this, the patterns that can be identified within their specific data will not be consistent with the very vast majority of the company's customers. Such an inconsistency implies that the data belonging to such customers can be defined as anomalous data [9]. Therefore, these anomalies within sports betting data can be used to determine whether a customer's betting patterns can be said to be fraudulent or malicious. Such scenarios can represent simply an attempt for one to turn a profit from a personal financial perspective, but they can also be a part of more serious and severe criminal behaviour, including, but not limited to, financial crime, money laundering, fraud and match-fixing [10].

As mentioned before, with increasing amounts of betting activity taking place online, the major betting operators have been generating significant quantities of data about their players. Furthermore, efforts that are made by players to gain an advantage over others through illicit means are also found within this data. This data is therefore the most prominent means at the disposal of gambling operators to catch those players using their knowledge for malicious purposes. Since such players ordinarily represent a small minority of an operator's customer base, their data will contain patterns that are inconsistent with those of ordinary customers. Hence it can be said that the data of these customers can be defined as anomalous [9]. Such anomalies within sports betting data can be used to identify both fraudulent or other malicious betting scenarios which may represent not only an attempt at personal financial gain but may also be associated with illegal activities, including financial crimes, money laundering, fraud or match-fixing [10].

1.1 Motivation

When it comes to the literature, a good number of works have been found that look at Anomaly Detection methods applied to Sports Betting. Some of these works focus on using Anomaly Detection to identify potential illegal activities, such as match-fixing [11, 12]. Other authors opt to take a different direction and choose to investigate the detection and identification of problem gamblers: people who have developed, or are in the process of developing, a gambling dependency or addiction [13, 14]. Other authors take yet another approach and attempt to predict or detect the presence of fraud within a sports betting data set [15, 16]. Finally, there are other authors whose research focuses on the field of sports betting with the aim of successfully predicting the outcome of sporting events, with the most popular being football matches [17, 18]. Although several works have been identified, all of which pertain to the implementation of Anomaly Detection within sports betting [10, 11, 12, 19], very few of the works found dealt with the detection of anomalies within sports betting from the perspective of a betting operator, regardless of whether the Anomaly Detection takes place at a bet level or at a customer level. It was both the existence of this niche as well as the fact that the author is employed in the sports betting sector that ultimately motivated the author of this work to select this area as the focus of this dissertation.

By definition, an anomaly is an element observable within a data set that is not consistent with the vast majority of the data within the data set [9, 10]. Within numerous industry applications, including the field of sports betting, these anomalies are buried within massive volumes of data, which are often too large to be properly monitored by humans alone since thousands of new data elements are generated per day. Within sports betting, amongst the many thousands of bets placed on any given day, only a handful of those will actually have been placed by sharp customers, undesirable customers or beneficial customers. Artificial Intelligence can be of great help in examining the bets that have been placed on any given day and flag any bets that human operators may want to take a closer look at. Such a task can be accomplished by a number of potential different algorithms that specialise in Anomaly Detection, as can be seen in Chapter 3 of this dissertation.

The further process of differentiation between a customer being sharp or undesirable is a further challenge that can be undertaken. It is also a rather difficult task, even for human operators, as this process often involves the recognition of more subtle patterns to determine whether or not a customer is using additional knowledge in an attempt to gain an additional edge to increase his winnings and potentially defraud the betting operator. This leads to the further application of a different set of Artificial Intelligence

techniques: Classification. Since there are 3 different types of anomalous customers (the exceptionally good, the undesirable and the sharp), this enables the use of multi-class classification algorithms to differentiate between the three at a transactional level.

However, considering the ever-changing dynamic that exists within industries such as sports betting, it is essential that the context can also be taken into account. An example of recent events that massively changed the context of sports betting was the COVID-19 pandemic. At the start of the pandemic, with multiple countries worldwide going into lockdown, sports betting almost ground to a halt, due to the mass cancellation of almost all sporting events across the globe [20]. However, the industry adapted, as during this period, as well as the time in which the world started to gradually return to normality, the industry saw a surge in betting on esports [20]. Even today, while traditional sports betting remains significantly more popular, esports betting has a higher engagement now than in pre-COVID times [20]. When it comes to traditional sports, there are also cases where operators need to take context into account, this time for high-traffic situations, such as during the FIFA World Cup [21]. During this period, since it is one of the world's most widely followed competitions, it is all-hands-on-deck for sports betting operators, as it tends to be the busiest time for operators, working around the clock and dealing with a massive volume of bets [21]. This is further reinforced by data collected for this dissertation, which consists of a full year, including the 2022 FIFA World Cup. Sure enough, a significant increase in the volume of bets was noticed during the months in which the competition took place. It can therefore be said that models that were trained on a certain date will eventually become out of date. Moreover, it follows that the next step in this process is to figure out when the model is no longer effective and requires retraining on new, better data, to ensure that the results provided by the aforementioned algorithms are based on data that is up to date.

Events such as these cause fundamental changes to the underlying distribution of data, in this case, to the volume and nature of the bets that are placed with operators [20, 21]. These changes in the data based on external factors over time can also be referred to as Concept Drift [22, 23]. It can therefore be inferred that the context in which a bet is placed is critical to determine whether or not it is anomalous, and that should a bet be considered anomalous at one point in time, it may not necessarily be so at a different point in time, particularly if the context has changed significantly.

1.2 Aim and Objectives

The primary aim of this dissertation is to make use of Artificial Intelligence techniques to investigate and identify customers who have the potential to be undesirable or sharp from the perspective of a betting operator. It is desirable for betting operators to catch both types of customers. This is because such individuals do not only have the potential to cause damage to the company and its profits, but they could also be affiliated with illicit activities, with some examples being fraud, money laundering and match-fixing. This aim can be further subdivided and achieved through the following objectives.

- **Objective 1 - Detection of Suspicious Bets** - This objective focuses on the research, investigation and implementation of a suitable Anomaly Detection Technique in order to identify any bets that may have the potential of being affiliated with malicious activities. Additionally, this objective also seeks to enhance the Anomaly Detector to maximise its performance.
- **Objective 2 - Enhancing Anomaly Detectors for Customer Classification** - This second objective seeks to expand on the first objective by identifying and comparing several multi-class classification algorithms, in order to classify the person who has placed a specific bet as being a beneficial customer, an undesirable customer or a sharp customer.
- **Objective 3 - Concept Drift Detection** - This final objective seeks to enhance the algorithms developed for both previous objectives by attempting to detect Drift present within the data. This is done so as to catch changes in the data set that could cause data that is anomalous at one time to be non-anomalous at a different point in time and vice versa.

1.3 Proposed Solution

Throughout the research that was performed, although several works were found that addressed Anomaly Detection [11, 24, 25, 26, 27] and sports betting [6, 12, 15, 16, 28, 29], insufficient papers were found which dealt with Anomaly Detection in sports betting from the perspective of the betting operator. The overall aim of this dissertation is to develop a solution capable of detecting and identifying customers who are undesirable to the operator or those who can be said to be sharp customers. In the case of the data that was provided for this dissertation, for a customer to be undesirable to the betting operator,

they would be defined as a customer who, should they be left alone without taking action, would cause losses of a non-negligible nature to the betting operator. Meanwhile, customers who are defined as sharp are those customers who make use of their own personal knowledge, either about the sport taking place or about the betting industry as a whole (such as only ever placing bets with the operator when it provides the best odds when compared to other operators).

Based on this, the solution that is being proposed in order to achieve this aim, based on the literature reviewed, can be split into two main experiments. The first such experiment would be to develop an Anomaly Detector that is capable of identifying bets within the data set that are anomalous. Such an Anomaly Detector would classify bets in a binary manner, as either anomalous or not. Once this is in place, the next stage of this experiment would be to enhance this Anomaly Detector to maximise its performance, with at least one such enhancement focusing on addressing Concept Drift.

The second experiment being proposed is to expand this Anomaly Detector from a binary classifier to a multi-class classifier. This would convert the Anomaly Detector developed in the first experiment into a classifier that predicts the type of customer who placed any given bet. Such a classification process would also serve to differentiate between the different types of anomalies present within the data. Furthermore, just as the original Anomaly Detector was enhanced so as to maximise its performance, this multi-class Customer Classifier will also be enhanced to improve its performance as much as possible, with at least one enhancement also attempting to mitigate the effects of Concept Drift.

1.4 Contributions

The main contribution that this dissertation makes to the literature available is to introduce a fresh perspective on the application of Anomaly Detection to the field of sports betting. By doing so, it serves to occupy the niche that was observed during the research process, to address the specified challenges within sports betting. Hence, this dissertation also serves to contribute this perspective toward readily available literature addressing problems found within sports betting using an analytical approach. Moreover, it serves to assimilate Anomaly Detection within sports betting to that of Fraud Detection, hence bringing the two closer together and opening a new realm of possibilities for future researchers who wish to attempt a similar use case. Such an assimilation serves to enhance both domains by both expanding the applications of Anomaly Detection and reinforcing the available approaches to perform fraud detection. Finally, it serves to lay

the groundwork which prospective authors within the sports betting industry can build upon. By detailing the approach taken, the selections made and the results obtained, this work also serves as a reference point for future authors who wish to advance the use of Artificial Intelligence, both within academia as well as among research and development teams within the sports betting industry.

1.5 Document Structure

An outline of the rest of this dissertation is as follows. Chapters 2 and 3 delve into some underlying concepts and examine literature that is present both regarding the algorithms used and the context of the dissertation. Chapter 4 discusses the approach taken, while Chapter 5 documents and examines the results from the implementation, evaluates and analyses them. Finally, Chapter 6 summarises the conclusions that have been reached from the investigation.

Chapter 2

Background

2.1 Overview

The following chapter discusses a number of the underlying concepts which form the backbone of this dissertation, along with the relevance of said concepts to the research performed. The first subsection will provide an overview of sports betting, followed by an overview of anomaly detection, classification and Drift concepts that are utilised as a part of the implementation stage of this dissertation.

2.2 Sports Betting

The underlying principle behind sports betting can be described as a game between the player and the betting operator. The player attempts to make a prediction on a minimum of one market or aspect of the game. This can be on the result of the match, the exact score, how many goals are scored in the game, whether or not both teams will score, and many more. The player also stakes an amount of money against a probability-based odd offered by the operator. If the player wins, they earn back the stake, plus a payout based on the odds offered on that market. On the other hand, if the player gets it wrong, the operator gets to keep the stake.

The earliest known instance of sports betting was by the ancient Greeks, where betting would take place over the victors of competitions in the Olympic Games [30, 31]. Shortly after that, in Roman times, betting was commonplace in the Colosseum and other arenas on sports such as chariot racing and gladiatorial combat [31]. As time went by, people continued to gamble for leisure, as was the case in medieval England where a cock-fighting industry sprouted and such events were specifically organised at betting

offices [30]. More recently, towards the beginning of the 20th century, there were two developments that further boosted the popularity of sports betting on a global scale [30]. The first was the rise of football to become one of the world's most popular sports, and the second was the invention of the television, enabling sports to be broadcast live all across the world [30]. These two concurrent developments enabled sports betting to start occurring on a global scale, with customers able to bet on games happening thousands of kilometres away [30]. The most recent major milestone in sports betting was the surge in availability and popularity of the Internet. The introduction of online sports betting took the sports betting industry to the next level, further enabling a major boost in the expansion and development of the sports gambling sector [30, 32].

When looking at this particular solution to the problem under investigation, the data was sourced from a betting operator, whose sports betting platform, also known as a sportsbook, operates globally through an online application and a website, as is common practice throughout the sports betting industry [30, 32]. The data set provided contained bets that were placed across a very wide range of sports, from popular, well-known sports such as football, basketball and ice hockey, to rising sports such as padel and darts [33, 34], and even esports competitions.

The number of different markets offered are then dependent on the popularity of the sport. Less popular sports would have a more basic group of markets, such as Match Winner markets, which represent a bet on the outcome of the match (win, loss or draw) [35]. Rising sports, such as darts, would then have markets allowing customers to bet on the winner of a specific tournament or league, known as an Outright bet. More popular sports will then have a myriad of other possible markets for bets to be placed on, such as the number of goals or points scored, either in total or by a specific team or even by a particular player. These are known as Over/Under markets and also extend to other different events, such as the number of fouls committed or the number of set pieces or dead-ball scenarios that occurred. Alternatively, such bets can also be restricted to a specific time period within a match, such as the first or second half of a match. Customers are also able to combine multiple different bets into one betslip, to potentially earn a larger sum of money but only if all the combined bets are successful.

Also offered by the operator on certain markets is the ability to cash out on a bet[36]. Most customers will wait for the event that they placed a bet on to finish and should their bet have been won, they would receive their prize, or payout, while if their bet would have been lost, the operator gets to keep the money that the customer had staked. However, on certain markets, should customers be in a losing position and wish to cut their losses, operators allow the customer to do so and pay out an amount of money that is lower than the amount that the customer had originally staked. Alternatively, customers could be in

a winning position and accept a payout that is lower than they would have received had they waited until the outcome of the market has been settled since would not want to risk the game turning around and costing them all their potential winnings [36].

2.3 Anomalies

Within the context of Artificial Intelligence and the data that it requires, anomalies can be defined as rarely observed patterns within a data set that appear significantly inconsistent with the vast majority of the data [9, 10, 37]. The process of detection and identification of these anomalies is fundamentally rendered challenging for two reasons: their scarcity within copious amounts of data, and the high dimensionality that is present within most data sets [9, 37]. Sure enough, when tackling problems using algorithms involving Artificial Intelligence, traditional machine learning models are frequently unsuccessful at identifying anomalous data and taking it into account during training [9, 37].

Despite this, it is critical that said anomalies are found because they may have several different potential ramifications if left uncaught, depending on the context of the data set. For example, anomalies found in medical imaging data could reveal that a patient has a cancerous tumour [24, 25], while anomalies found in biometric data could indicate that there has been an unauthorised attempt to gain entry to a protected site without the correct security clearance [24, 9, 25]. Alternatively, it could also be a malicious attempt to access a device, system or platform that one is not meant to have access to [9, 10, 24]. Finally, anomalies in financial or transactional data at a bank could be an indicator of fraudulent transactions or the potential presence of other criminal financial activity, such as money laundering or tax evasion [9, 24, 25].

When looking at the data that was utilised as part of this dissertation, anomalies within the data set are represented by three different types of customers. Furthermore, since the data is in the form of bets, then anomalous bets are those bets that are placed by customers that fall under these types of customers.

The first type of anomalous customer is the undesirable customer. These customers are customers who have been identified as presenting betting patterns that are not healthy for the betting operator. Examples of such customers include customers who undergo a sudden and abrupt increase in activity or customers who only place bets on markets where a given operator is presenting better odds than its competitors. Alternatively, it could also represent a customer who attempts to conceal instances of the latter by combining such bets with others having a very high probability of being won.

The second type of anomalous customer is the sharp customer. These customers are

customers who have some form of knowledge that enables them to gain an advantage over the operator. These customers tend to focus on a sport or a league where their knowledge extends to, especially where this sport or league is not one that is particularly well-known or where the sport is relatively new to the industry, such as esports. Alternatively, these customers could have knowledge as a proficient gambler and could monitor an operator's platforms ready to pounce if they see an opportunity that they know is favourable. They could also be customers who have a very fast television feed or are in the stadium during a match and attempt to place bets on markets that they know will be won before the operator closes them, particularly events that they would be aware of before the bookmakers' feed catches up.

Both of the above classifications represent negative types of customers. Both of these behaviors could represent malicious activities, which may simply be to make money off the betting operators for one's own financial gain, but could also represent criminal acts such as money laundering, fraud or match-fixing. However, there exists a third anomalous classification that is more beneficial to betting operators. These customers are known to the betting operators as exceptionally good customers. Such customers would bet relatively large sums of money for fun, while at the same time being able to do so in a financially sustainable manner without developing a gambling problem or addiction. These players would also have mixed results, typically in favour of the betting operator.

2.4 Customer Classification

In Artificial Intelligence, there exist two types of problems, depending on the state of the desired output [38]. These are classification problems, where the output desired is categorical, and regression problems, in which the desired output is a continuous value [38]. Over the years, several different approaches and algorithms have been devised to tackle such problems. Some examples include Naive Bayesian classification, Artificial Neural Networks, Support Vector Machines, Logistic Regression and Random Forests [39, 40]. One thing that all of the above have in common is that they are most commonly applied to differentiate between two different classes, also known as binary classification [39, 40].

However, certain situations will require that the algorithms involved are able to classify data instances into more than two potential classes, a process known as multi-class classification [40, 38]. With some of these algorithms being suited for binary classification, some adaptations may be required for them to operate efficiently for multi-class classification. One such example would be the use of multiple binary classifiers in such a way that each binary classifier attempts to classify whether or not a given data element is

a member of one of the potential classifications, which is a binary classification [40, 41]. Several such classifiers, one per potential classification, can then be combined such that, due to the nature of these one-vs-all systems, only one such binary classifier should return a positive result, which is also the result of the multi-class classifier [40, 41]. Another potential approach is the use of probability where, instead of outputting a boolean value representing whether or not the data element is a member of the classification, the classifier instead outputs the probability of such membership [38]. These classifiers are then combined, with one for each class, and the result of the overall classifier is the class whose individual classifier returned the highest probability of membership [38].

Since within the data set being used for this dissertation, the anomalies are individual customers, then the type of customer is the dependent variable. Based on the solution that was proposed in Section 1.3, the second experiment that is to be performed will focus almost entirely on expanding a binary classifier into a multi-class classifier. Furthermore, the use of generating the probability rather than the classification could also be one of the enhancements within the proposed solution.

2.5 Concept Drift

One of the biggest issues when dealing with data that spans a considerable period of time is that the manner in which said data is distributed may change as time passes [22]. Such a concept, known as Drift or Concept Drift, particularly comes into play when the data under consideration is continuously generated in real-time, or when the modelling that is taking place attempts to model certain behaviours which, by human nature, are susceptible to change over time [22]. Since these changes can cause Artificial Intelligence algorithms to degrade in performance, it is therefore critical that instances of Drift are caught and acted upon [22, 42].

Different types of Concept Drift exist, with some referring to the changes that actually occur, namely Real and Virtual Concept Drift, while others refer to the pace of the change that takes place, such as Abrupt or Gradual Drift [22, 42]. The key difference between the two is between the type of underlying changes that take place. In the case of Virtual Concept Drift, it is a change in the distribution of training data that varies as time goes by without changing the decision boundaries between different classes within the data [22, 42, 43]. Meanwhile, for Real Concept Drift, it is the underlying concept that changes with time, this time including shifts in the decision boundaries between different classes [22, 42, 43]. Meanwhile, the main types of Drift that take place in terms of the pace of the change include Sudden (or Abrupt), Gradual, Incremental or Recurring [22]. As the name

suggests, Sudden Drift is when the old concept shifts completely to the new concept at one point in time, while in Gradual Drift the new concept first shows up as an occasional outlier within the old concept [22]. These occasional outliers increase in frequency until the new concept has become the main concept [22]. A real-world example of this would be the changing of several market behaviours due to inflation [22]. Meanwhile, in Incremental Drift, both concepts exist and there is a gradual shift between the old and the new concept [22]. The main difference between Gradual and Incremental drift is in the manner in which the transition takes place [22]. In Gradual Drift, the transition takes place in a continuous manner, while in Incremental Drift, the change takes place at specific intervals, such as monthly or quarterly [22]. Finally, Recurrent Drift implies that there are cycles where the concept rotates between being in place and not [22]. A real-world example of Recurrent Drift would be seasonal sale patterns [22].

In the case of sports betting, the effects of Concept Drift are often felt. The data for this dissertation consists of millions of bets, one after the other, for a period of a whole year. Throughout a year, fluctuations take place as seasons start and finish and different competitions enter different stages. For example, the summer months generally have a lower amount of traffic as they represent the off-season. An exception would be the presence of a major tournament, such as the FIFA World Cup, which has the opposite effect and causes a massive surge in traffic [21]. Such real-world events impact the underlying distribution of data, causing bets that would be suspicious during the regular season to not necessarily be anomalous at a different time of the year. The volume of data present in the data set therefore presents a level of detail that would make the data set vulnerable to the effects of Concept Drift. Hence, addressing this problem was included as one of the objectives of this dissertation.

2.6 Conclusion

The above chapter detailed some of the underlying concepts that form the backbone of the dissertation as a whole. In the following section, these concepts and several algorithms that can be used to implement them are discussed in further detail, along with an exploration of the numerous different works that have been made using such techniques as part of their own implementations.

Chapter 3

Literature Review

3.1 Overview

The following chapter outlines and details the research that served as a foundation for this dissertation. The first section provides an outline of the application of Artificial Intelligence to the field of sports betting, providing the underlying context. With that in place, the next part details several works which perform similar tasks to this dissertation. The subsequent sections delve into the research performed into individual algorithms and metrics that could be utilised in order to address the problems outlined in Section 1.2. Some of the areas that are explored within this chapter include Anomaly Detection, Decision Trees and Random Forests and Autoencoder Neural Networks. Further sections proceed to explore multi-class classification, as well as Support Vector Machines and Random Forest classifiers that are capable of supporting multiple potential classes. Finally, the chapter also observes how the problem of Concept Drift has been addressed by other authors in several papers where it has been encountered.

3.2 Artificial Intelligence and Sports Betting

3.2.1 Overview

The application of Artificial Intelligence to the sports betting industry has already undergone numerous investigations as several authors have explored the combination through their publications. Some examples of investigations that have been found by the author of this dissertation include investigations pertaining to Sports Prediction and others that attempt to catch potential cases of match-fixing.

3.2.2 Sports Prediction

A first example of a paper which investigated the topic of Sports Prediction is the work of Horvat and Job [44]. Their paper took an exploratory approach and looked at a number of different algorithms at all stages of the process [44]. This does not only refer to the actual algorithms themselves but also to different methods of performing the preprocessing stages, such as Feature Selection, as well as different potential evaluation metrics [44]. They investigated a multitude of previous works and took note of what those works used at each stage [44].

During the preprocessing stage, a significant portion of the papers that Horvat and Job analysed make use of Feature Selection [44]. What differed was the approach taken, with several different techniques used to reduce the dimensionality of the data sets under consideration. These included Principal Component Analysis and the use of metrics such as Mutual Information, prediction accuracy, Pearson Correlation and Information Gain [44]. Other papers analysed by Horvat and Job put additional measures in their feature selection, such as selecting only the most relevant features, dividing the data by quartiles or considering feature information gain at a team level rather than at a database level [44].

When looking at the actual Artificial Intelligence Algorithms that were used in the papers analysed in the works of Horvat and Job [44]. From these papers, the most commonly used algorithms to predict the result of sporting matches were Neural Networks, Support Vector Machines, Decision Trees (including Random Forests), Logistic Regression and Bayesian Classifiers [44]. Other algorithms that were used to a lesser degree included Linear Regression, K-Nearest Neighbours, Fuzzy Logic and Genetic Programming [44]. The two then proceeded to break down the accuracy of the different works in a number of ways, splitting them by sport, algorithm used, comprehensiveness of the data set used and the date of publication of the works concerned [44].

A second paper that was consulted by the author of this dissertation is a work by Fianho et al. [45]. This work reviewed a number of different Artificial Intelligence techniques with the objective of laying the groundwork for future research into football result prediction [45]. Several different techniques were studied, including Bayesian and Logistic Regression, Neural Networks, Support Vector Machines and Fuzzy Logic [45]. Using the knowledge from their research, Fianho et al. then proceeded to propose their own model for future authors to consider. Specifically, this is a model which predicts a one-hot vector consisting of two zeroes and a one, where the location of the one within the vector represents whether a team won, lost or drew a match [45]. Furthermore, they also proposed using more than one of the aforementioned algorithms to be able to compare the results and select the most accurate [45].

A third set of authors who ventured into the sports prediction field are Geng and Hu [29]. The work of these authors differs from others seen previously in that, while the previous papers focused on analysing more traditional machine learning techniques such as Support Vector Machines and Logistic Regression, Geng and Hu explored and implemented a system using a different paradigm: Genetic Algorithms [29]. Their approach also attempted to predict the ranking of teams in the end-of-season playoffs rather than each individual match [29]. They also only consider a single sporting competition - the American National Basketball Association (NBA) [29].

In their process, the algorithm had a set of possibilities at any given time and updated this space by randomly selecting a sample of these and finding the two fittest parents to generate two new possible outcomes by performing a subtree swap at a random point within the two parent possibilities [29]. These then replaced the least fit possibilities within the set of possible playoff brackets [29]. This fitness was judged by a weighting system designed to prioritise the later rounds of the tournament and more heavily penalise predictions which were further away from the actual result [29].

3.2.3 Maximisation of Winnings

While some researchers focused purely on the result of a match, a separate subset of authors investigated sports prediction with a different aim. Specifically, this objective was to game the system and work out exactly how to best exploit the sports betting system using Artificial Intelligence to maximise their winnings. Based on the author's research, this is exemplified in two main papers, namely the work produced by Damian Borycki [28] and the work by Hubáček et al. [17], with both focusing on finding an optimal betting strategy.

Borycki's work used a k-Nearest Neighbours algorithm combined with a Genetic Algorithm for optimisation [28]. His objective was not to predict the winning team, but to maximise the profits from a bet [28]. In the data collected, there are 134 different parameters, so that the algorithm could take into account more information than just the teams taking part and potentially some history, such as the players taking part, the formation used and the disposition of the team as a whole [28]. He then made use of the k-Nearest Neighbours algorithm to utilise the principle of using similar previous bets which were won to calculate the probability that any given bet will be won [28]. This was then followed by the use of the Genetic Algorithm to improve performance by updating and optimising the data used as part of the k-Nearest Neighbours algorithm [28]. The Genetic Algorithm used a similar process to that used by Geng and Hu, except that it was applied to the games rather than to a tournament bracket. It also used a fitness function

that was more focused on maximising profitability rather than one focused on accuracy [28].

Meanwhile, Hubáček et al. took a slightly different approach, where their data focused on a team's form throughout the current season [17]. Some elements of this data were at a player level, while others were at a team level. Also included were some features describing the bet itself, such as the odds [17]. They then applied two different Artificial Intelligence algorithms to separately attempt to predict the outcome of individual matches [17]. The techniques selected by these authors were Logistic Regression, an Artificial Neural Network and a Convolutional Neural Network [17]. These methods were used to obtain a value between 0 and 1 representing the probability of victory on a given bet, and this was then combined with a confidence threshold and a number of other heuristics to gradually converge on an optimal player strategy [17].

3.3 Anomaly Detection

3.3.1 Overview

Within the context of applying Artificial Intelligence to Sports Betting, the application that the author of this dissertation opted to investigate is that of Anomaly Detection. The process of Anomaly Detection can be defined as "the problem of identifying patterns/observations, which appear to be inconsistent with the population of normal data" [25]. Through the research performed by the author, it is evident that multiple concepts, methods and techniques in one's arsenal can be used when tackling a problem which falls under the Anomaly Detection umbrella. Within this section, Section 3.3.2 illustrates several works that investigated such problems and made use of Anomaly Detection in their efforts, implementations and research. Meanwhile, the main challenge that was encountered by most researchers who attempted to make use of Anomaly Detection algorithms, namely that of Class Imbalance, is then detailed in Section 3.3.3.

3.3.2 Single Class Classifiers

One potential approach to tackling Anomaly Detection problems was found to be used through two works by Fatemifar et al. [25, 46]. In both works, a configuration of single-class classifiers was used, combining the result of each classifier using a weighting averaging system to obtain one final result which stated whether or not the data was anomalous [25, 46]. Their earlier work applied Anomaly Detection to cybersecurity, more specifically to the detection of facial spoofing attacks [46]. This work made use of three different

single-class algorithms, before combining the result through weighted averaging, namely single-class variations of a Support Vector Machine, a Mahalanobis Distance based detector and a Gaussian Mixture Model [46]. The system as a whole was then refined using a Genetic Algorithm, which used the Half Total Error Rate of the internal combined classifier as a fitness function [46].

Meanwhile, the later work of Fatemifar et al. took this concept as a starting point and elevated it further, providing it with a more general form, so that it could be used across various disciplines [25]. In this framework, they took a set of single-class classifiers, normalised their output and combined the result of each classifier through weighted averaging [25]. This time, instead of a Genetic Algorithm, Fatemifar et al. made use of Particle Swarm Optimisation to refine their system of classifiers [25]. Anomalies were then identified by setting a threshold based on a conventional metric, which was selected to be the Equal Error Rate by Fatemifar et al. [25]. It should be noted that their implementation was designed in such a way that the system would be trained entirely using non-anomalous data, with the exception of the setting of the Error Rate Threshold which, by nature, required anomalous examples in order to be calculated [25]. This operated using the principle of identifying instances of their data as non-anomalous, classifying the remainder as anomalous by elimination [25]. This justified the decision to include exclusively non-anomalous data in the training data, as the inclusion of anomalous data would only have served to throw off the classifiers [25].

Finally, a work that made use of such classifiers within the Sports Betting sector was that of Kim et al. [11]. The focus of their study was to detect potential cases of match-fixing using data from the matches themselves as well as sports betting data, particularly the odds of a number of markets. In their implementation, the authors made use of five classifiers arranged in such a way that each individual classifier made its own prediction about whether or not a given data element was anomalous. The classifiers were a Support Vector Machine, a k-Nearest Neighbours classifier, a Random Forest classifier, a Logistic Regression classifier and an ensemble classifier which made use of all four previous classifiers. The result given by each of the five classifiers was then used to take a vote between them and if a majority classified a match as anomalous, then the overall system declared it to be anomalous and therefore worth looking into as there may have been a case of match-fixing occurring in that match.

3.3.3 The Class Imbalance Problem

One of the most prominent issues faced by researchers when attempting to make use of Anomaly Detection is that of Class Imbalance [25, 47]. Class Imbalance can be defined

as the case when there is one potential classification that contains substantially fewer instances than the rest [47]. In certain cases, this divide can be such that the minority class makes up 5% or less of the overall data set [47] and in some cases, even 1% or less. In the case of Anomaly Detection, by nature, anomalies do not frequently occur and will make up the minority class in such problems [47]. This subsection details a number of papers which attempt to tackle this problem as part of their work investigating Anomaly Detection.

One of the most popular techniques that has been observed in the literature is known as SMOTE [48, 26]. This technique, whose name stands for Synthetic Minority Over-sampling Technique, is a method of oversampling which seeks to increase the size of the minority classifications to rebalance the data set [48]. It does so in a manner which does not generate duplicates of already present elements from the minority classifications, but rather seeks to generate new synthetic examples [48]. This is done by combining features from the k nearest samples to create a new distinct sample [48]. This process is then performed iteratively, taking new samples every time to generate more distinct minority samples until the different classes are more balanced [48].

Another technique that has been used to address the problem of class imbalance is the ADASYN technique [49, 26]. Also known as Adaptive Synthetic Sampling for Imbalanced Learning, this technique also generates new minority data samples [49]. However, while SMOTE generates an arbitrary number of new data samples, ADASYN does so in a manner that adapts the decision boundary so as to include samples which are more difficult to classify [49]. The algorithm starts out by calculating the degree of imbalance and the number of new elements that would be required to address this imbalance [49]. It then finds, for each member of the minority class, the proportion of that element's k nearest neighbours that belong to the majority class and normalises this value [49]. Next, the number of synthetic examples required is found and then for each member of the minority class, these examples are generated based on one of its k nearest neighbours (randomly selected), a difference vector and a random number between 0 and 1 [49].

Another study that attempts to address the issue of Class Imbalance is the work of Ahsan et al., which made use of Oversampling to address the issue of Class Imbalance. The three techniques under the author's observation were SMOTE, ADASYN and CGAN [26]. For CGAN, this method was based upon two neural networks, one known as a generator and the other known as the discriminator [26]. Each network had 4 hidden layers and was optimised using Stochastic Gradient Descent over 30,000 epochs with a learning rate of 0.0001 [26]. Through these two networks, over a million records were generated to help solve the class imbalance, with the output and the original training data being fed to an adversarial environment reinforcement learning classifier which then performed the

standard classification process [26].

Another method under consideration was seen as part of the work of Wang et al., who proposed a variation of the Support Vector Machine that catered for the Class Imbalance problem [47]. This type of machine learning algorithm operates by attempting to find a boundary to separate the given data elements into their distinct classes [47, 27]. The way in which Wang et al. went about this was by assigning weights to the different features that made up the aforementioned boundary, or hyperplane [47]. These weights were then adjusted such that both the accuracy of anomalous samples and non-anomalous samples were maximised by maximising the result of their Geometric Mean, which incorporates both the True Positive Rate and the True Negative Rate [47]. Once the authors developed their own method to minimise Class Imbalance, they then compared it against some more established methods, such as the application of SMOTE when sampling the data [47]. This same formula was also used by Fatemafir et al. as part of their work to develop a generic framework for Anomaly Detection [25], which was further elaborated on earlier, in Section 3.3.2.

Finally, Mirkhani also made use of an imbalanced data set as part of an investigation using sports betting data [50]. While this work made use of similar data to the data set provided for this dissertation, the focus of Mirkhani was to develop Artificial Intelligence algorithms that were capable of identifying customers who had opted to self-exclude themselves from a betting operator's platform [50]. Such self-exclusion essentially locks a player out of their account, serving as a safety feature that allows a player to help prevent themselves from problem gambling or developing a gambling addiction [50]. Despite the fact that Mirkhani's data set is at a customer level and the one being used for this dissertation is at a bet level, since they are applied to a similar use case, they can be said to be comparable [50]. In the case of Mirkhani's data set, 3,562 customers had self-excluded out of a total of 148,684, which amounts to approximately 2.40% [50].

3.4 Studies with Similar Objectives

Having discussed Anomaly Detection in the previous section, the following subsection details several academic works whose authors opted to address problems that fall under the Anomaly Detection umbrella, while also having similar objectives to this dissertation. The first work under consideration is that of Kim et al., whose research investigated the use of Deep Learning to perform Anomaly Detection within Sports Betting [10]. Specifically, these authors attempted to use such techniques so as to detect and identify cases of potential match-fixing [10]. Keeping this end goal in mind, Kim et al. selected the K-

League as their main competition to focus on [10]. This league represents the top tier of the South Korean football pyramid and the data itself consisted of historical betting odds from this competition [10]. Furthermore, this data, which was organised into a knowledge graph data structure, compiled odds originating from multiple bookmakers [10]. It was also monitored for shifting in these odds as the outcomes became more clear-cut. The data was also labelled using expert knowledge to indicate whether or not the match in question could be a part of a match-fixing case [10]. With match-fixing cases being an absolute rarity, these could therefore also be defined as anomalous or as an attempt to commit fraud against the bookmaker [10]. A ResNet model was then trained to detect and identify these anomalous matches and converge on an identifying set of features, proceeding to pass them to a Deep Neural Network which then performed the process of classification, catching anomalous, fraudulent matches [10].

Meanwhile, Kim et al. also had a second further work that also attempted to tackle fraud detection from the perspective of sports betting [11]. More specifically, they sought to prevent illegal gambling pertaining to match-fixing by athletes for financial gain [11]. The main difference between the approach that Kim et al. took in their earlier work to this one was that while in the earlier study [10] they focused on using Deep Learning techniques, in the more recent paper [11], the algorithms used were more focused on traditional machine learning models [10, 11]. In this second work, Kim et al. built their own data set using match betting data from several different betting companies obtained from a dedicated API. This API, in turn, made use of automatic web scraping to mine data from numerous websites to obtain detailed information at player, team, match and league levels for a number of different sports [11]. Once this data was loaded from the API, several different preprocessing methods were used so as to bring the data into a usable, structured representation, such as normalisation, standardisation and aggregation [11]. Once the data was in the desired form, Kim et al. made use of five different algorithms, all of which are based on traditional machine learning techniques [11]. These were Support Vector Machines, Random Forests, Logistic Regression and k-Nearest Neighbours, as well as an ensemble model, which integrates all four algorithms and aggregates their parameters [11]. Once all five algorithms were properly trained, their results for each element were pooled and a vote was taken among all five algorithms, with a minimum of three algorithms classifying a data element as anomalous required for the system as a whole to declare it as an anomaly [11].

Finally, another work which performed a similar task to this dissertation is that of Tsykunov [51]. Through his paper, Tsykunov investigates the suitability of a number of machine learning models to detect and identify problem gamblers within a data set consisting of the betting history of customers from a betting operator [51]. To accomplish

their goal, some of the different algorithms that Tsykunov tested include Decision Trees, Random Forests, Support Vector Machines, AdaBoost, k-Nearest Neighbours and Logistic Regression [51]. Through the metrics that were then used by Tsykonov when they compared the results of the different algorithms, the Random Forest was found to be the best under almost all of the evaluation metrics [51].

3.5 Logistic Regression

3.5.1 Overview

Anomaly Detection can be said to be a binary classification, where any given data element is either anomalous or not. The following section, as well as subsequent sections, detail the various classifiers that were considered for use as part of this dissertation and discuss a number of works that used such classification techniques, starting from that of Logistic Regression. This algorithm works by attempting to assign a weighting to each potential feature in the data set and then find the optimal set of weight-feature pairs to obtain an equation that represents the data set as accurately as possible [52]. This makes it a versatile technique with a variety of applications, including education [52], the Internet of Things [53] and healthcare [54].

3.5.2 Applying Logistic Regression to Anomaly Detection

In the work by Abbasi et al., the authors made use of Logistic Regression to detect intruders through Anomaly Detection in the Internet of Things [53]. They made use of the N-Balot data set, which contains over 110 different features pertaining to network traffic between a number of different devices [53]. A logistic regression classifier was then applied to the data, and the most important features were identified as those features having the highest weighting [53]. Next, the most important features of the non-anomalous class were found by sorting the features in descending order of importance, and then applying them one by one to get the True Positive Rate (TPR) and the False Positive rate (FPR). The difference between the two was then mapped to an ROC Curve, repeating the process until the result of subtracting the False Positive Rate from the True Positive Rate is close to 1 and the addition of further features does not significantly change the result [53]. The authors also ran an Artificial Neural Network with the same objective in parallel so as to compare the performance of this technique [53].

Meanwhile, Al-Haija et al. made use of the logistic regression algorithm within the context of cybersecurity [55]. Specifically, their objective was to use logistic regression to

detect anomalies which represented port scanning attacks [55]. In this case, the anomalies were scattered within a data set of network traffic between devices on a Local Area Network within a laboratory environment [55]. Once collected, the preprocessing stage consisted of a shuffling of the data set elements, along with the application of k-fold cross-validation while splitting each fold into a training set and a test set to be passed to several machine learning algorithms to undergo Anomaly Detection [55]. The algorithms were all treated independently and Logistic Regression was one of several algorithms that were used to attempt to distinguish the attacks from the regular traffic on the network [55]. The other algorithms used included Decision Trees, Discriminant Classifiers and a Naive Bayesian Classifier [55]. Once these were all trained and tested, the results of each were compared to one another and it was the Logistic Regression Model that was found to be the most effective of all the algorithms attempted throughout the paper, having the highest accuracy (99.4%) and the shortest overhead time (0.454 nanoseconds) [55].

3.5.3 Logistic Regression for Multi-Class Classification

While Anomaly Detection is traditionally a binary classification, different categories of anomalies may exist within the same data set. In the experience of the author of this dissertation, when applied to the case of Sports Betting, an anomalous bet could be of three types, an exceptionally good customer, an undesirable customer or a sharp customer. Within the literature, different authors were found to look at Logistic Regression with the view of expanding it to perform multi-class classification rather than just a binary classification. One such set of authors is Wang et al., who investigated multi-class logistic regression applied to the task of Feature Selection to develop an enhanced preprocessing process [56]. Their proposition involved modifying traditional logistic regression and adapting it to their multi-class problem [56]. Their function started on the basis of the creation of a matrix of coefficients such that it met a number of conditions [56]. These conditions were then converted into either probabilistic ones or into ones that can be used to control this weighting matrix [56]. Once these were set up, the authors also designed a function that would describe the probability that a data element was indeed a member of a given class [56]. In order to implement the above steps, the authors made use of a probabilistic multi-class Logistic Regression algorithm, maximising the probability that class membership was correctly assigned [56]. The final step taken was that normalisation was applied to the algorithm and it was then compared to a number of other potential algorithms that could have been used for the same purpose [56].

Another work, namely that of Purwandari et al., also investigated at how logistic regression could be used to perform multi-class classification [57]. More specifically, their

work looked at the manner in which various multi-class classification algorithms performed at predicting weather patterns using data sourced from a Twitter data set of Indonesian tweets mentioning different types of weather [57]. However, the nature of using Twitter as a data set necessitated a rather long preprocessing stage, as it had a very high volume of noise [57]. This stage included several subtasks, such as case-folding, stripping the data of URLs, usernames and other links, ensuring only alphabetic characters are used, Stemming, Tokenisation and stop-word removal [57]. Once the preprocessing had taken place, the authors attempted to classify the tweets into one of several different weather categories, including sunny, cloudy, raining, moderate rain and storm [57]. The algorithms utilised included Support Vector Machines, Multinomial Naive Bayes and Logistic Regression [57]. Based on the results obtained, the authors established that the best of the three algorithms attempted was the Support Vector Machine [57].

3.6 Support Vector Machines

3.6.1 Overview

A second type of classifier that was encountered in the literature pertaining to Anomaly Detection was the Support Vector Machine. Support Vector Machines are a form of supervised machine learning algorithms used to perform classification on labelled data [27]. This algorithm treats data as a mapping across a multi-dimensional space, with a number of dimensions equal to the number of features in the data [27]. The principle under which this classifier operates is that it attempts to find the optimal separation or hyperplane to classify the data into its different classes [27, 47]. This makes Support Vector Machines ideal for binary classifications, owing to their innate approach that splits a hyperdimensional space in two [58]. However, there do exist variations of the Support Vector Machine algorithms that function as single class classifiers, that is a yes or no classifier depending on whether a data element is a member of a specific classification [59, 60] and for multi-class classification, which classifies into three or more potential classifications [61]. Simple examples of how a boundary can be drawn to differentiate between two classes can be seen in Figure 3.1 where the example on the left represents a linear split, while the example on the right represents a more complex split.

3.6.2 Anomaly Detection Using Support Vector Machines

One example of the use of Support Vector Machines was the work of Ma et al., where the authors of this work used Support Vector Machines to detect anomalies in network

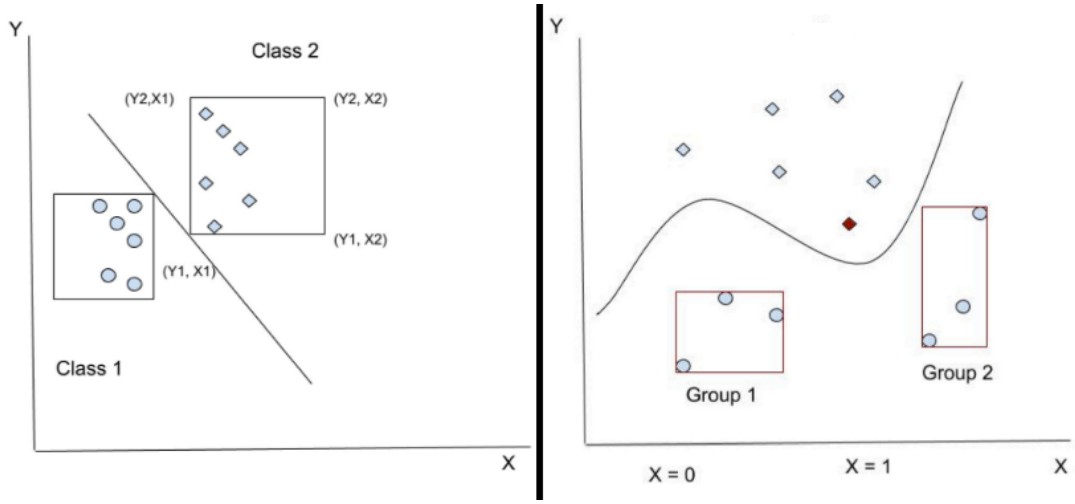

Figure 3.1: Visualisation of Support Vector Machines [60]

traffic [58]. In their publication, Ma et al. first transformed their data into vector form, as this made it easier to be mapped to the aforementioned multi-dimensional space [58]. They performed this process using a combination of k-gram modelling and mean pooling operations [58]. To do this, URLs from the authors' database were first analysed and a vocabulary of anomalous URLs or URL segments was generated [58]. Next, a series of rules and the generated vocabulary mapped the URLs into weighted vectors, which weer in turn set to a fixed length through k-gram sliding windows [58]. The second stage of the process was to train the Support Vector Machine that the authors would continue to use [58]. Once this was complete, the remaining phase of the project was to optimise the hyperparameters of the Support Vector Machines in order to obtain the best possible results through an iterative single-dimension approach, converging closer to the optimal results with each iteration [58].

Another work which implemented this technique was the work produced by Yang et al., who dealt with the use of Support Vector Machines to perform Anomaly Detection in the field of the Internet of Things [59]. Their data was sourced from numerous general-purpose devices, such as laptop computers, as well as more specialist appliances including smart home devices, including smart fridges and smart dishwashers [59]. Since their objective was to increase the detection rate of both conventional anomalies as well as to identify the presence of new behaviour, the authors made use of data from two different scenarios: one where the data was labelled, like in lab-controlled conditions, and another where the data was unlabelled, as though it were being collected directly from the end user [59]. When it came to the technical difference between the two scenarios, it boiled down to two main algorithm hyperparameters, namely the kernel bandwidth,

which represented the distance between points within the training data, and the number of Gaussian Mixture Model components used by the specific variation of Support Vector Machines that the authors selected [59].

Finally, Barbado et al. attempted to extract rules from the Anomaly Detection process performed using a Support Vector Machine so as to create an explainable version of this algorithm [60]. The preprocessing step involved the scaling of continuous-value features and the encoding of categorical features [60]. Once complete, they applied a Support Vector Machine to the data, trained it and validated it, using a single-class variation of the technique [60]. Once the Support Vector Machine was fully trained, the next step was to start extracting rules [60]. Had all the features represented categorical variables, it would have been the simplest case, as the rule for anomalous data points is the set of unique combinations of feature values that make up each individual data point [60]. On the other hand, if there was a mix of both categorical and continuous-value variables, then the set of rules was the hypercube of each possible combination of the categorical features. [60] A hypercube is obtained by using intuition from clustering algorithms, such that a hypercube contains a group of points of the same cluster (in this case, points having the same combination of values in their categorical variables), based around the centroid [60]. Any points within a given hypercube that had a different combination of values in their categorical variables compared to the rest of the hypercube could be classified as anomalous [60]. The number of clusters or hypercubes was then iteratively increased until there were no anomalous values in any hypercube, hence meaning that a fully representative set of rules that encompassed the entire data set had been converged upon [60].

3.6.3 Multi-Class Support Vector Machines

Meanwhile, other authors made use of Support Vector Machines to tackle multi-class classification, despite the Support Vector Machine being inherently designed for binary classification, such as in the work of Yi and Zhang [62]. In their work, they proposed a method of using multiple binary Support Vector machines in such a configuration that each was performing a one-vs-all classification [62]. Each Support Vector Machine in turn determined whether or not the given data element was a member of a specific class or not [62]. Moreover, since any classifier attempts to minimise the error generated, they also proposed two methods in which the reliability of their classifiers could be increased. This method also operated based on the assumption that the overall error may not necessarily be known [62].

In their first method, known as Static Reliability, the pair of authors proposed that the

training set error could be used to estimate the overall classifier's error rate [62]. Delving deeper into their formulae shows that in this method, the reliability measure would be the same for any given sample [62]. In other words, the reliability metric would be static across all possible examples [62]. The other method proposed is Dynamic Reliability, where the key concept changed into an attempt to test the reliability of the classifier over a set of similar samples to the one under consideration [62]. It was referred to as dynamic because, in this case, the sample became relevant because it and its k nearest neighbours which belong to the same class would have an effect on the value of the reliability metric [62]. Hence, this method took context into account at the expense of requiring recalculation for every sample that was taken under consideration [62]. With these concepts established, Yi and Zhang then proceeded to apply them to a number of different data sets, which spanned a variety of potential applications, including Image Segmentation, Wine Recognition, Iris Plant Classification and Optical Character Recognition [62].

Meanwhile, Lalabadi et al. also made use of Support Vector Machines, but applied them specifically to the field of aquaculture [63]. More specifically, these authors chose to apply this Artificial Intelligence technique with the objective of assessing the freshness of seafood [63]. They used a data set comprised of images of several different fish with varying degrees of freshness and the main factors that they looked at were the colour of the eyes and the gills of the fish, as these were established to be key indicators of how fresh a fish is [63]. Once features have been extracted from these images, the authors then applied two AI methods to perform this classification, one being Support Vector Machines and the other being Artificial Neural Networks [63]. In both cases, this was a multi-class classification problem as the objective of the classifier was to determine how many days old the fish was, with options of the fish being 1, 3, 5, 7 or 9 days old [63]. Although in this case, the Support Vector Machine was outperformed by the Neural Network, it was still able to correctly determine the freshness of the fish with an accuracy of 68% from the eye colour and of 84% from the gill colour [63].

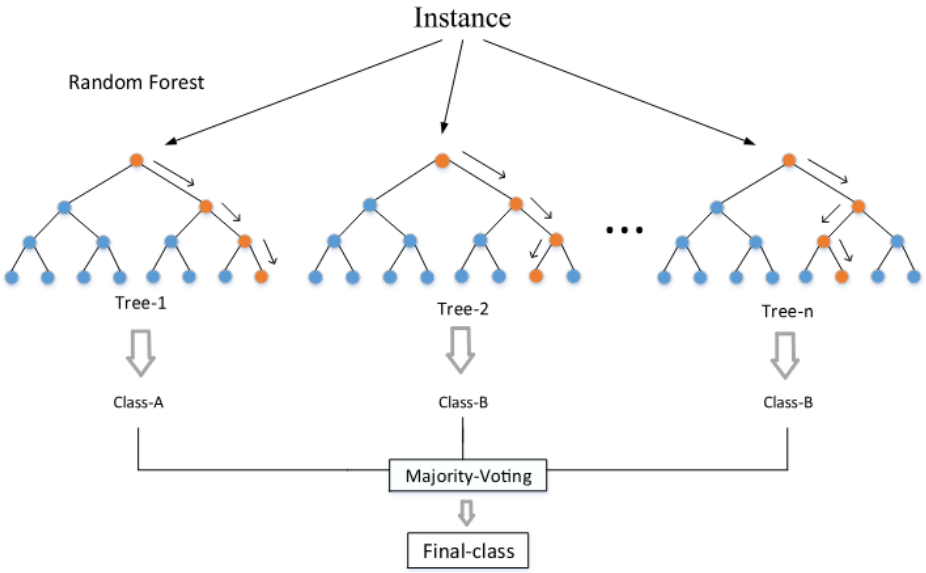

Figure 3.2: Visualisation of Random Forest Classifier [67]

3.7 Random Forest Classifier

3.7.1 Overview

A third classification algorithm that features heavily in the literature concerning Anomaly Detection is the Random Forest Classifier. First developed in 2001 by Leo Breiman, the Random Forest Classifier is a type of ensemble classifier that aggregates a number of different Decision Tree configurations [64, 65]. It is a combination of two main techniques, namely Decision Trees and bootstrap aggregation, the latter of which is also known as bagging and prevents the results of the algorithm from suffering due to the effects of overfitting [66]. The manner in which such an algorithm works is that first several potential decision trees are generated from the space of possible decision trees that could be used to describe the original data [66]. These trees are generated by randomly selecting a number of samples from the original data and then further randomly selecting various features from that sample, splitting the tree in such a manner as to maximise Information Gain based on these selected features [66]. Once these trees are fully formed, the aggregation may take place either by a vote in the case of the random forest being applied to a classification problem, or by averaging all the outputs in the case of the forest being applied to a regression problem [66, 67]. This entire process can be observed in Figure 3.2.

3.7.2 Random Forest for Detection of Anomalies and Fraud

In terms of uses of the Random Forest in literature relevant to this dissertation, the most common application of Random Forests within the field of Anomaly Detection is in the process of Fraud Detection, as can be seen in the works of Lin and Jian, Aftab et al. and of Shah and Sharma [66, 68, 69]. In the paper produced by Lin and Jian, the authors made use of both an Autoencoder as well as a Random Forest algorithm with the aim of detecting fraudulent credit card activity [66]. Their approach first passed the data through the Autoencoder, followed by the application of a probabilistic Random Forest so as to obtain the probability of any given data element being anomalous [66]. The main advantage of such a Random Forest is that the threshold is not fixed and can be tweaked accordingly to maximise the number of anomalies that are successfully caught [66].

Meanwhile, in the work of Aftab et al., before performing any form of classification, they implemented the SMOTE oversampling technique to rebalance the data set since fraudulent transactions can be considered as anomalies and therefore, represent a very small minority of transactions [68]. However, their focus was on determining the optimal supervised machine learning algorithm from a range of traditional options, namely, Support Vector Machines, Random Forest, Decision Trees and Logistic Regression [68]. It should be noted that as part of the results of this paper, the authors came to the conclusion that out of the four, Random Forests were the best-performing algorithm under every evaluation metric investigated [68]. Such evaluation metrics included Accuracy, F1 Score, Precision, Recall, Area Under the Curve and the Matthews Correlation Coefficient [68].

Furthermore, the work of Shah and Sharma focused on the use of Decision Trees and Random Forests as part of their investigation into credit card fraud [69]. Their approach made use of oversampling and undersampling techniques during the preprocessing stage to get around the aforementioned imbalance present within the data [69]. However, despite their efforts, once the data was applied to the two different classification algorithms for training and testing, the authors' results were inconclusive as in every set of results produced by the authors, their classifiers always suffered from overfitting [69].

Another rather different work that makes use of Random Forests is the work of Kopp et al. [70]. The authors in this paper also ventured into the realm of Explainable Artificial Intelligence, having the objective of creating a Random Forest algorithm that not only performs Anomaly Detection, but is also capable of explaining why any given element of the data set is an anomaly [70]. This explanation was intended to be in the form of rules that would be human-understandable [70]. The approach taken for this, in addition to the standard Random Forest implementation, was that once a list of anomalies was

obtained, they were first clustered to group together any similar anomalies, minimising the total number of rules that were required to cover all the anomalies detected [70]. Clusters of similar anomalies were then grouped together into one forest [70]. The rules were then extracted using one of two processes. The first, for minimal explanation, was designed to obtain the rules for a single element by following the path of the decision trees within the forest by continuously splitting the node in which the anomalous element was found and applying a relevant threshold accordingly [70]. The second, for maximal explanation, was designed to contain the set of rules which describe all the ways in which the anomalies could be defined as anomalous [70]. This process involved the random selection of two training sets, one of which contained the anomaly and another which contained no anomalies [70]. Using each training set, it then formulated a tree and stored the rule and threshold associated with each feature in the set of rules, while removing that feature from the two training sets [70]. The process continued until the training set containing the anomalous sample behaved entirely as expected, particularly the remaining features of anomalous samples [70]. The authors then proceeded to apply this process to a number of different applications [70]. This included Optical Character Recognition of handwritten numbers and network security [70].

3.8 Autoencoders

A final potential algorithm which is used in the literature to tackle problems that fall under Anomaly Detection through the use of Deep Learning is an Autoencoder, such as the one that was used by Min et al. [19]. In their work, Min et al. chose to attempt to detect anomalies within a betting context, more specifically that of horse racing [19]. They do so through the combination of Autoencoders and Long Short-Term Memory [19]. The way that such Autoencoders work is by compressing the data that is passed to them through a series of encoders and then attempting to restore it to its original dimensionality by using a series of decoders [19]. This architecture behind such a process can be visualised through Figure 3.3 (diagram obtained from [66]).

Min et al. proposed an implementation where such an Autoencoder was created and given a Long Short-Term Memory architecture [19]. Once created, data was then provided to the Long Short-Term Memory Autoencoder (LSTM-AE) for compression and restoration. What was then in place was a metric that measured the error present in the restored data, such as the Mean Squared Error [19]. Metrics such as this one, or similar, would serve as a score for how anomalous the data was [19]. Had this score exceeded a particular threshold for any element of a given data set, that element would have been classified

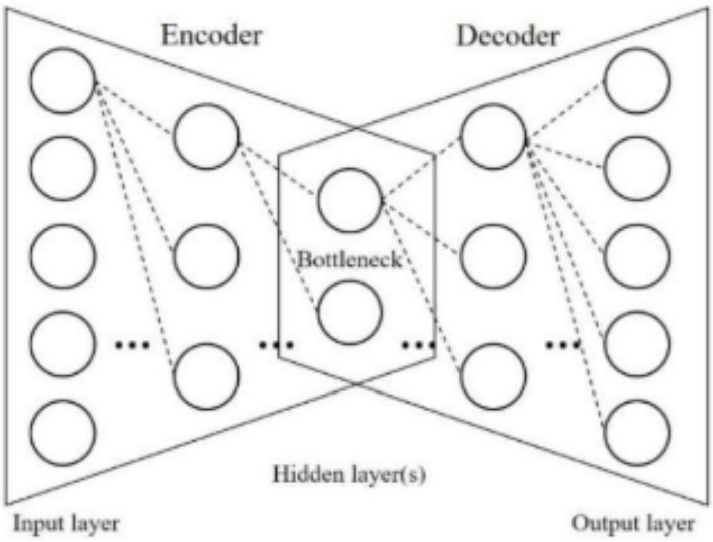

Figure 3.3: Visualisation of Random Forest Classifier

as anomalous [19]. Since this method took place in a situation where a comparison of the restored data versus the original data occured, it was necessary that the models were trained using non-anomalous data, so that the Autoencoder could restore data in the way that it should be if it were non-anomalous [19]. Had the comparison returned a high error, it would have been a sign that the corresponding data element in the original data set was not as it was expected to be, as the reconstructed element would be as expected [19]. Therefore, it could be deduced that that element is anomalous [19].

Another work that performed Anomaly Detection using Autoencoders is the work of Lin and Jiang [66]. In this work, an Autoencoder was incorporated along with a Random Forest as part of an Anomaly Detection process [66]. In such a system, the role of the Autoencoder was to obtain a set of features, or expected features, that were used to train the Random Forests further down the line [66]. It was also applied to validation data later on, before being passed to the Random Forest [66]. In this way, the Autoencoder was made use of as a sort of preprocessor [66].

A final work that made use of Autoencoders is that of Ahmad et al. who utilised an Autoencoder to perform Anomaly Detection in quality control [71]. More specifically, the Autoencoders proposed by these authors attempted to detect faults or defects in rotating machines [71]. This work also made use of Long Short-Term Memory, and this combination made use of a system of weightings that were converged upon using a Stochastic Gradient Descent algorithm along with an Adam optimiser [71]. Once data was passed through the Autoencoder, it was assigned an anomaly score such that any score below a

certain threshold could be considered non-anomalous [71]. Another use of the Autoencoder as part of this work was for feature extraction, where features extracted by the encoder part of the Autoencoder were compared to manually extracted features so as to feed these features to an Isolation Forest algorithm and determine how the whole system could have been improved [71].

3.9 Evaluation Techniques

3.9.1 Common Metrics

Upon being fully trained, all Artificial Intelligence algorithms, techniques and models must be properly tested. This includes an evaluation and analysis of the results, which are most often in the form of one or more representative metrics, of which several different types can be found within the literature. Said metrics enable the visual representation of the performance of these algorithms. Certain metrics were found to be often used when dealing with classification problems, such as Anomaly Detection, where classifiers must determine whether or not a given data element is anomalous [10]. The following are the most common metrics that were encountered by the author of this dissertation as part of the research performed.

Furthermore, when considering the data set that underwent analysis for this dissertation and the scope of Anomaly Detection, when applying the following equations, a bet that was classified as a True Positive represented an anomalous bet that was correctly identified as anomalous, while a True Negative classification represented a non-anomalous bet that was correctly identified as non-anomalous. Meanwhile, a False Positive classification indicated that a non-anomalous bet was wrongly classified as anomalous, while a False Negative classification indicated that an anomalous bet was not caught as anomalous.

- **Accuracy**: Percentage of correct classifications [10, 72, 73, 19, 74]

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalClassifications} \tag{3.1}$$

- **Precision**: Proportion of correctly classified positive classifications [10, 72, 19, 74]

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{3.2}$$

- **Recall**: rate of True Positive classifications out of all existing positive data instances [10, 72, 19, 74]

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3.3}$$

- **False Positive Rate**: represents the percentage of False Positive classifications out of all classifications made. [72]

$$FalsePositiveRate = \frac{FalsePositives}{TotalClassifications} \tag{3.4}$$

- **F1 Score**: combines precision and recall and can be mathematically defined as the harmonic mean of the two metrics [72, 19]

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.5}$$

- **Sensitivity**: proportion of anomalies detected out of the total anomalies present [47]

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3.6}$$

- **Specificity**: proportion of non-anomalous data entries recorded as such out of the total number of non-anomalous records [47]

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{3.7}$$

- **Geometric Mean**: a measure that takes into account both the accuracy of a classifier on anomalous data only and the accuracy of the classifier on non-anomalous data only [47, 25]. Note that in the following equations, Geometric Mean is being shortened to G-Mean.

$$G - Mean = \sqrt{\frac{TruePositives}{(TruePositives + FalseNegatives)} * \frac{TrueNegatives}{TrueNegatives + FalsePositives}} \tag{3.8}$$

$$G - Mean = \sqrt{Sensitivity * Specificity} \tag{3.9}$$

- **Confusion Matrix**: a visualization in matrix form of how instances of all different classifications were classified, where the position within the matrix represents both the classification that was made by the classifier and its true classification. [72, 73]

3.9.2 Evaluating on a Normal Distribution

Alternatively, another method of evaluating Anomaly Detection can be found in the paper of Deutscher et al., in which the resulting values of the process were combined into one value, which was then mapped onto a normal distribution [75]. This means that the resulting set of values would have a mean of 0 and a standard deviation of 1, where a value of 0 for a given data element would mean that it was perfectly non-anomalous and the further away from 0 the value was, the more anomalous it could be considered to be [75]. The nature of these compiled values meant that they could then undergo statistical testing through a Hypothesis Test to definitively prove whether or not there was a significant deviation from the mean [75]. Should this actually have been the case, a statistical difference would have been definitively proven, and therefore there was a very high probability that the data element under consideration was anomalous [75].

3.9.3 Selecting the Optimal Threshold

One thing that is crucial when performing a binary classification is to establish how likely it has to be for a given data element to be a member of a given class for it to be classified as such [76]. In the case of Anomaly Detection, this principle applies to the likelihood of a given element being anomalous. For this reason, the correct probability threshold must be set and any element with a higher probability of being anomalous than that threshold is classified as an anomaly.

Further to this, the work of Bahnsen et al. explored two different methods of converging on the optimal value for this threshold within the context of Fraud Detection [76]. The first such method was a calibration of the base probabilities, since the data set on which an algorithm was trained would have had a different optimal threshold value to that of the data that it was actually attempting to classify [76]. This could have been due to the use of undersampling or oversampling algorithms to correct data for Class Imbalance [76]. The method in which this was handled was through a formula which used a sample population to estimate the probability threshold of the actual population [76]. However, this relied heavily on the assumption that the selected estimation population had a rate of positive to negative classifications which was the same as the actual population [76].

The second method analysed by Bahnsen et al. made use of the ROC Curve [76]. In this case, an uncalibrated ROC Curve had be calibrated such that it would be in a convex form [76]. To do so, Bahnsen et al. made use of a Convex Hull algorithm to ensure that it was properly calibrated [76]. Once the convex ROC Curve was formed, new calibrated probabilities could be extracted by first grouping the points on the probability axis and then the slope of the calibrated curve used to determine the probabilities [76]. To compare the two, a cost function was used, which extracted a value based on a matrix alongside more traditional metrics such as Precision, Recall and F1-score[76].

3.10 Concept Drift

3.10.1 Overview

Finally, further to the Class Imbalance Problem identified earlier in Section 3.3.3, a second major problem that was encountered in the literature when researching Anomaly Detection was that of Concept Drift, which is the change in the context of data over time. When tackling any problem which requires a constant stream of data, it is crucial to take the context into account [23, 77]. This is especially the case when applied to financial, transactional or sports betting data, where transactions that could be suspicious or anomalous at one point in time may not necessarily remain so a couple of weeks or months down the line. This phenomenon is known as Concept Drift [74, 23, 77].

3.10.2 Concept Drift Detector

The first work that investigated Concept Drift was that of Lu et al. [74], which provided a generic Concept Drift Detection framework as well as outlined several different techniques that could be used to develop a Concept Drift Detector [74]. They broke the detection process down into four distinct phases, where the first is the Data Retrieval Stage [74]. This stage was used to either load in the entire data set or extract it from a whole stream, depending on the type of data source [74]. The second phase was the Data Modelling stage, which was where the preprocessing occurred, particularly Feature Extraction and Dimensionality Reduction, so as to extract the features with the most impact on Concept Drift [74]. The third stage was the Test Statistics Calculation phase, in which at least one metric that measured dissimilarity was formulated and applied to the data [74]. This could also have been applied after some form of clustering had been implemented and the dissimilarity measured on a cluster level [74]. Finally, the fourth stage was the Hypothesis Testing Stage, where the metrics obtained in the previous section were

used in a Hypothesis Test to determine whether there was a statistically significant difference, proving that there had indeed been a shift in the data [74]. Once this process was detailed, the authors of the paper outlined several algorithms that could be used. Some of these included Drift Detection Method (DDM), which works by detecting significant changes in the online error rate of the data under consideration, the Statistical Test of Equal Proportion Detector, which compares the two most recent time samples and tests whether or not the difference between the two deviates from the normal distribution and finally, the Adaptive Windowing algorithm, which bases its detection on the sample means of the two individual windows, whose sizes do not need to be pre-declared [74]. Other algorithms mentioned included Relativised Discrepancy, the Information Theoretic Approach, Statistical Change Detection, Competence Modelling, Just-in-Time Classifiers and Linear Four-Rate detectors [74].

Considering the ever-changing dynamic that exists within industries such as sports betting, it is essential that the context can also be taken into account. Real-world events could have a huge impact on the data observed by betting operators. Events that cause sports to stop would automatically decrease the betting activity on those sports, the major competitions generally cause surges in popularity and betting activity [20, 21]. These real-world events greatly impact the underlying distribution of data and hence, bring about Concept Drift [22, 23]. When analysing the sports betting industry, operators deal with potentially hundreds of thousands of bets a day, with such volumes providing ample opportunity for Concept Drift to be observed [22, 23]. Since the dataset used for this dissertation consists of data comprising a whole year, which includes one of the biggest sporting events in the world, namely the FIFA World Cup, this principle applies, and Concept Drift is a genuine concern, particularly considering the volume of data within the data set.

3.10.3 Incorporating Concept Drift into Anomaly Detection

Meanwhile, a second work that was reviewed was that of Ahmad et al. [77]. However, these authors made reference to Concept Drift as part of their work which applied Anomaly Detection to streaming data [77]. In their paper, they detailed the properties of an ideal Anomaly Detection algorithm within their context of continuous data streams [77]. These properties included Online Prediction, which meant that every individual data element had to be classified as anomalous or not before the next one arrived and Continuous Learning, which meant that the algorithm had to operate without having to store the entire stream [77]. Furthermore, the algorithm had to be unsupervised and free from manual intervention, while also being adaptable to dynamic environments and the effects

of Concept Drift [77]. Finally, anomalies also had to be detected as early and quickly as possible, whilst minimizing the number of false positives and false negatives that were made during classification [77]. The authors then proceeded to try to implement an algorithm that satisfied all of the above, starting from the concept of High Temporal Memory (HTM) [77]. To do so, they started by feeding the data through an encoder, followed by a pooling process [77]. This produced a vector which was passed through a network of neurons that were HTM-enabled, which in turn were used by the network to generate a sparse vector that attempted to predict whether the element was anomalous or not [77]. Based on this output, a prediction error score was calculated which, thanks to the HTM used, could adapt to changes in the underlying data distribution [77]. Finally, to prevent a high rate of false positives or false negatives, the prediction error was again modelled into a distribution, serving as an indirect metric [77]. A second metric, referred to as Anomaly Likelihood was then used to officially classify an element as anomalous or not, and this metric and its threshold were both based on the model's historical prediction distribution [77].

3.11 Key Findings and Selection of Algorithms

The objectives of this dissertation were satisfied through two experiments. The first of these experiments implemented an Anomaly Detector and compared its performance to a number of other results based on similar implementations in the literature. Such a detector will be developed with the scope of detecting bets that were placed by suspicious individuals or other bets which are suspicious. This experiment will also implement a number of additional enhancements to the Anomaly Detector so as to improve and maximise its performance. The main algorithm that was identified to perform this task, based on the literature encountered is a Random Forest, as this was found to be the most common traditional machine learning technique applied to Anomaly Detection in Sports Betting, and was found to be a more successful Anomaly Detector than other such algorithms [10, 11, 68], Also identified were Autoencoders, which were the most commonly encountered Deep Learning technique in the literature [66, 19, 71]. However, based on the volume of data and the training speed of the two algorithms in question, the Random Forest was selected for use in this experiment. Meanwhile, three further enhancements were identified for application to the Random Forest as part of this experiment. The first of these is the SMOTE oversampling technique, which was selected to attempt to mitigate the Class Imbalance Problem [47, 48, 26]. The second is the use of a number of different probability thresholds to determine the optimal probability threshold required

for a given bet to be classified as anomalous [76]. The final enhancement that was identified pertained to the mitigation of the problem presented by Concept Drift. Based on the literature encountered, the Adaptive Windowing technique has been selected [74].

The second experiment is dedicated to the extension of the Anomaly Detector to cater for multi-class classification, enabling the identification of suspicious individuals from the bets placed, and also the type of customer that placed a given bet. This is useful since an anomalous customer can be either exceptionally good, undesirable or particularly sharp. Through the literature reviewed, three potential algorithms were identified that had the potential to perform this task. The selection was based on the most commonly found methods for performing multi-class classification, namely Random Forest, Support Vector Machines and Logistic Regression [53, 56, 57, 62]. Once implemented, the three will be compared and the best-performing of the three will be identified. Once identified, the SMOTE and Concept Drift Detection enhancements that would have been applied in the first experiment will both be incorporated into the multi-class classifier and the performance of both implementations will be compared to one another and to the literature.

3.12 Conclusion

The above chapter detailed the algorithms that have been investigated for potential use in this dissertation, as well as several works that have made use of these same techniques. It also served to identify which algorithms were actually selected by the author of this dissertation to be implemented as part of the experiments performed. In the following chapter, these algorithms will be put to use, with said chapter detailing the methodology used in order to implement the required code to perform experiments in line with the objectives specified in Section 1.2.

Chapter 4

Methodology

4.1 Overview

The following chapter details the various algorithms, techniques and configurations that were made use of by the author of this dissertation with the aim of satisfying the objectives laid out in Section 1.2. The code for the implementation developed for this dissertation was done entirely in Python, using version 3.11.8. This section will first give an overview of the data set used and of the overall process. Subsequent sections of the chapter will detail each different experiment used to satisfy the objective of the dissertation, starting from the algorithms that were implemented to identify anomalous bets. Following this will be the algorithms that were used to attempt to classify the type of customer. Both experiments made use of Concept Drift Detection at different points in the overall process. Through Table 4.1, the mapping between the experiments performed and the objectives of this dissertation are visualised and outlined. Meanwhile, Figure 4.1 shows the developed implementation from a higher level, while subsequent sections go through the individual experiments and a deeper level.

4.2 The Data Set

4.2.1 Overview

The data set that is made use of as part of this dissertation consists of a sample of bets from a sports betting operator. The bets within the data set were placed over the course of a year and contained within it are a relatively small number of bets placed by customers who, as described earlier in Section 1.1, can be said to be anomalous as they are exceptionally good, undesirable or sharp.

Experiments			
Number	Objective	Algorithms Selected	Description
1	1, 3	Random Forest SMOTE Probability Thresholds Adaptive Windowing	Implement Anomaly Detector using Random Forest. Enhance the performance of the Anomaly Detector using SMOTE, Thresholds and Adaptive Windowing
2	2, 3	Random Forest Support Vector Machine Logistic Regression Adaptive Windowing SMOTE	Enhance the Anomaly Detector from Experiment 1, experimenting with three different classifiers to find the best one. Apply SMOTE and Adaptive Windowing to the best one to maximise performance.

Table 4.1: Experiments performed by objective

4.2.2 Fields Present

Several fields within this data pertain specifically to the individual bets under consideration, such as the stake placed on a given bet, whether the bet is on a single outcome occurring (known as a single bet) or if it is a bet on multiple different outcomes occurring (known as a combination or accumulator bet) and what odds were offered by the bookmaker for the bet as a whole. The data also takes a deeper view by providing the stake and odds for the individual selections making up any given bet. It also contains the outcome of the bet and its individual selections, specifying whether the bet was Won or Lost, or if it remained unsettled at the time of extraction. Accompanying this bet-level data is also whether or not the bet was part of a Bonus campaign, whether or not the customer opted to cash out the bet ahead of time and the amount that was actually paid out to the customer. It should be noted that financial information, namely the amounts staked and the amounts paid out have been multiplied by a factor to protect the data owner's proprietary financial figures, while still maintaining proportionality of the figures preserving the information that they would provide to this dissertation.

Along with this data, there was also data at an event level, including the sport, competition and fixture upon which the bet was placed as well as on which market the bet was placed (for example, the number of goals), the selection and line of that market, including both numerical values (such as Over 2.5 goals scored) and any teams or players concerned (such as the Home team to win or a specific player to score at least 1 goal). Temporal data was also included, such as the date when the bet was placed and the start time of any given match. Further to this, owing to the sheer volume of data, the data was split up

into CSV files according to when the bets were placed, with each file containing all bets within the data set that were placed in the space of a given week. A clear summary of the data fields can be found in Table 4.2.

Finally, the dependent variable within this entire data set is the type, or classification, of Customer present. This information serves as the labelling within the Anomaly Detection component of this dissertation, where the two majority classes represent non-anomalous data and the remaining minority classes represent anomalous bets. This variable also serves as labelling for the multi-class classification component of the dissertation, whose objective is to predict the classification of customer who places a given bet, enabling a distinction between the type of anomalies present within the data set.

Field	**Type of Data**	**Cleaning Applicable**
Customer	Identifier	Convert to smaller value identifier
Country	Integer	N/A - Already numerically categorical
Type of Customer	Dependent Variable, Integer	N/A - Already numerically categorical
Betslip	Identifier	Convert to smaller value identifier
Bet	Identifier	Convert to smaller value identifier
Bet Type	Boolean (Single/Accumulator)	Convert to True/False
No. of Selections	Integer	N/A - Already numerical
DatePlaced	Date	Convert to number of seconds
BetStake	Decimal	N/A - Already numerical, not true values
Odds	Decimal	N/A - Already numerical
BetSettled	Text Value	Convert to categorical integer
Payout	Decimal	N/A - Already numerical, not true values
Cashout	Text Value	Convert to categorical integer
Bet Selection	Identifier	Convert to smaller value identifier
Sport	Text Value	Convert to categorical integer
Competition	Text Value	Convert to categorical integer
Match	Text Value	Convert to categorical integer
SelectionStake	Decimal	N/A - Already numerical, not true values
SelectionOdds	Decimal	N/A - Already numerical
SelectionSettlement	Text Value	Convert to categorical integer
Market	Text Value	Convert to categorical integer
Selection	Decimal	N/A - Already numerical
Deadline	Date	Convert to number of seconds
Phase	Boolean (Prematch/Live)	Convert to True/False
Bonus	Text Value	Convert to categorical integer

Table 4.2: Different data types and the cleaning they require

Number of Bets				
Month	**Total**	**Non-Anomalous**	**Anomalous**	**Anomalous Percent**
1	29,027,504	28,894,898	132,606	0.46%
2	27,797,505	27,669,792	127,713	0.46%
3	38,127,520	37,939,277	188,243	0.49%
4	36,466,688	36,290,318	176,370	0.48%
5	28,643,425	28,424,017	219,408	0.77%
6	26,716,400	26,508,882	207,518	0.78%
7	26,831,025	26,683,779	147,246	0.55%
8	30,467,522	30,290,863	176,659	0.58%
9	34,644,290	34,470,644	173,646	0.50%
10	32,279,561	32,119,967	159,594	0.49%
11	22,819,050	22,706,175	112,875	0.49%
12	20,302,625	20,212,590	90,035	0.44%
OVERALL	**354,123,115**	**352,211,202**	**1,911,913**	**0.54%**

Table 4.3: Monthly split of bets throughout the data set

4.2.3 Data Observations

In terms of looking at the data itself, the data consisting of approximately one file per week for a 12-month period. A total of 354,123,115 total bets were included in the data set, representing an average of just under 30 million bets per month. Of this total, 1,911,913 bets were placed by anomalous customers throughout the course of the year. This therefore means that, within the data set approximately 0.54% of all bets were placed by anomalous customers. A visual breakdown by month is visualised in Table 4.3, while the full breakdown by week can be seen in Appendix A.

Two notable observations that can be seen within this table include a spike in the number of bets present in Month 3 and Month 4 as well as the sudden drop in the number of bets in Months 11 and 12. Based on a knowledge of the time when this dataset was taken, the spike observed in Months 3 and 4 represents the months in which the FIFA World Cup 2022 took place, while the drop witnessed in Months 11 and 12 represents the months of early summer, when most European football leagues are in between seasons.

A further breakdown of the 354 million bets can be seen below, in Table 4.4. From this, it can be noted that the vast majority (98.43%) of bets in the data set were placed by regular or new customers, rendering the remaining customers as the minority classes. Of these, the undesirable and sharp classifications, which together make up approximately 0.54% of the overall data set, can be said to be anomalous with the potential to constitute suspicious activity.

Split of Customers into different types		
Type of Customer	**Number of Customers**	**Proportion**
Normal	275,823,798	77.89%
New	72,692,503	20.53%
Exceptionally Good	3,694,901	1.04%
Undesirable	1,398,259	0.39%
Sharp	513,654	0.15%

Table 4.4: Split of the dataset according to the type of customer who placed the bet

It can also be noted that throughout this dissertation, processing is performed at a weekly level, while results are presented on a monthly level. Since the data within the data set for each month was split into four files, by week, this generated results for each week. However, since the implementations described in Sections 4.4.2.1 and 4.5.2.1 operated in monthly cycles, the weekly results for each metric that was used were aggregated into a monthly average, to get the overall performance of the algorithms for the entire month. Such an approach was taken in order to enable retraining to take place periodically while also leaving a sufficient amount of time before retraining so as to get a representative result of the performance of the Anomaly Detector and multi-class classifiers that were developed. Furthermore, considering the volume of data present within the data set and the computation time that would have been required, it was determined that the most feasible interval between when the algorithm would be retained was one month. Therefore, since one full training and test cycle would be performed over the course of one month, it was determined that the results would be presented in a monthly aggregation.

4.2.4 Data Cleaning and Preprocessing

With the data being in a mostly raw form, some aspects of the data were in a textual data type. These fields were therefore required to be converted to a numerical format. This was easiest for categorical variables, such as the league, competition, fixture, market of the bet and more. Meanwhile, more continuous values, such as dates, were converted into the amount of time elapsed from the first date that would have been eligible to be a part of that file, namely the start of the week. Other data, such as the type of bet (single or accumulator) or whether or not a bonus was applied was set to a boolean value. Other fields, such as the anonymised customer identifier and the bet identifier, consisted of a very large number, with the customer identifier containing up to 10 digits and the bet identifiers containing up to 17 digits. These were remapped into a smaller numerical value

so as to ease up on the storage and memory requirements of the system. Furthermore, when required, the data was split into a training and test set such that the ratio of the split was that the first week of the month would have 70% of the data in a given data file allocated to the training set and the remaining 30% to the validation set while the remaining three weeks were used for testing. It should be noted that this approach was taken only for those algorithms where retraining was done in a periodical manner, as the introduction of Concept Drift Detection allowed the retraining of the data set to be done dynamically. In this case, the split between training and validation took place in the same ratios when required.

Once all data was converted into a numerical form, minimal preprocessing was required. Dimensionality Reduction was performed since a number of fields were removed from the data that was to be passed to the main algorithms of the solution, as these had no bearing on whether a bet was anomalous or not. The prime examples of these were identifiers, which did not affect whether or not a bet was anomalous. Normalisation was also performed, so as to bring certain fields within a range of 0 to 1, where 0 represented the lowest value used and 1 represented the highest.

4.3 Overall Solution

With the data loaded and cleaned, the first experiment was to perform Anomaly Detection to identify anomalous bets. The algorithm selected to perform this process was a Random Forest, based on the literature encountered in Chapter 3. This stage of the project focuses on the implementation of this Anomaly Detector and the assessment of its performance. Once this was completed, a number of potential enhancements to the Anomaly Detector were applied to the detector and the effectiveness of each was then evaluated. These enhancements included SMOTE, an oversampling technique used to mitigate the effects of the Class Imbalance Problem. Next, a separate technique performed was to assess what was the optimal probability threshold for the Anomaly Detector. The final enhancement was the introduction of a Concept Drift Detector that would enable the retraining of the Anomaly Detector as required, based on when changes in the underlying data were detected. Given the findings of the literature, particularly those described in Section 3.10.2, the main method that was selected to perform this Concept Drift Detection was the Adaptive Windowing method.

On the other hand, the second experiment performed focused on expanding on the Anomaly Detector that was implemented in the first experiment and converting the algorithm from an Anomaly Detector, whose approach is that of a binary classifier determin-

ing whether or not a given bet is anomalous, into a multi-class Anomaly Classifier. Such a classifier was tasked with identifying the type of customer who placed a given bet. This allowed for anomalous bets to be caught, but also included the reason behind why a given bet was anomalous, specifically whether the bet was placed by an undesirable customer, a sharp customer, or a particularly good customer. Based on the literature, multiple potential classifiers were identified and their performance was then analysed to select the best-performing of the three classifiers to move forward with. Once a classifier was selected, the SMOTE and Concept Drift Detectors were implemented into this classifier so as to attempt to maximise its performance by mitigating the Class Imbalance Problem and Concept Drift respectively. In view of the reviewed literature, the three algorithms that were selected to perform this task were Logistic Regression, Support Vector Machines or another Random Forest, while the Concept Drift Detection was also performed using Adaptive Windowing.

The following sections will put forward in more detail the process that was followed for each of the experiments described, together with some figures to better visualise the developed implementation. Further to the latter point, in Figure 4.1, a high-level overview of the way the system is intended to work once all the objectives have been fully implemented can be observed.

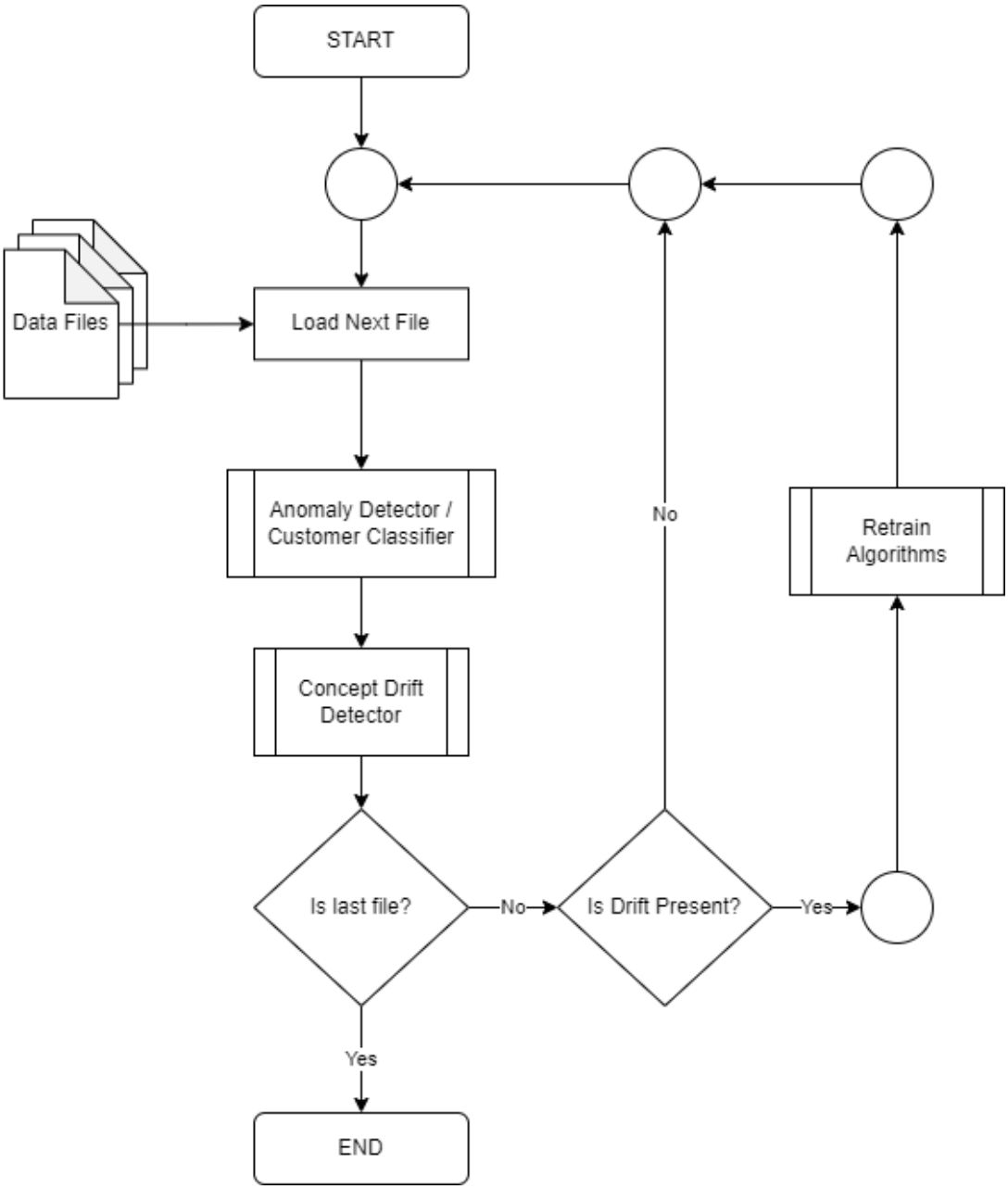

Figure 4.1: High-Level Overview of the overall system

4.4 Experiment 1: Detection of Suspicious Bets

4.4.1 Overview

This section of the implementation focuses on addressing the first objective of this dissertation which, in turn, pertains to the implementation of a suitable Anomaly Detection algorithm to identify anomalous bets, since these bets may also represent bets placed with malicious intentions that may include illicit or criminal activity. It also details the enhancements that take place to improve the performance of said Anomaly Detector, including SMOTE, Probability Thresholding and a Concept Drift Detector. The latter also partly addresses the third objective, enabling the Random Forest to be retrained when required, as opposed to periodically.

4.4.2 Approach Taken

4.4.2.1 Anomaly Detector

The main approach that was selected for this phase of the project was a Random Forest. It was selected owing to the fact that a significant amount of cited works that investigated Anomaly Detection using traditional machine learning techniques made use of Random Forests [66, 68, 69], including even studies that found them to be the best performing techniques [68].

As part of the implementation, when each file of data was cleaned, the cleaned version of the data was saved to be passed to this algorithm. These files were loaded and passed to the Random Forest in chronological order. The algorithm was retrained in a periodic fashion, with the first file of any given month being split into a training and test set, the former of which was used to train, or retrain, the algorithm. Once trained, the algorithm was then tested against the test set of that week and the data from the remainder of the month.

In terms of the configurations, the Random Forest algorithm itself was implemented using the Scikit-learn Python library [78]. In this configuration, two parameters required tuning. These were the proportion of the split on the files that formed the first week of the month as well as the number of estimators that formed the Random Forest. Based on the literature encountered, the training and test sets were split such that 70% of the data in the file was assigned to the training set and 30% to the test set. Furthermore, for the number of estimators, it was found that a higher number of estimators increased the accuracy of the classifier, but also contributed to a larger amount of time required to run the algorithm. Since the sheer volume of data also prolonged the algorithm runtime, a

balance between the number of estimators and the computation time was required. This balance was struck by selecting a value of 100. Figure 4.2 demonstrates how the process described above was set up.

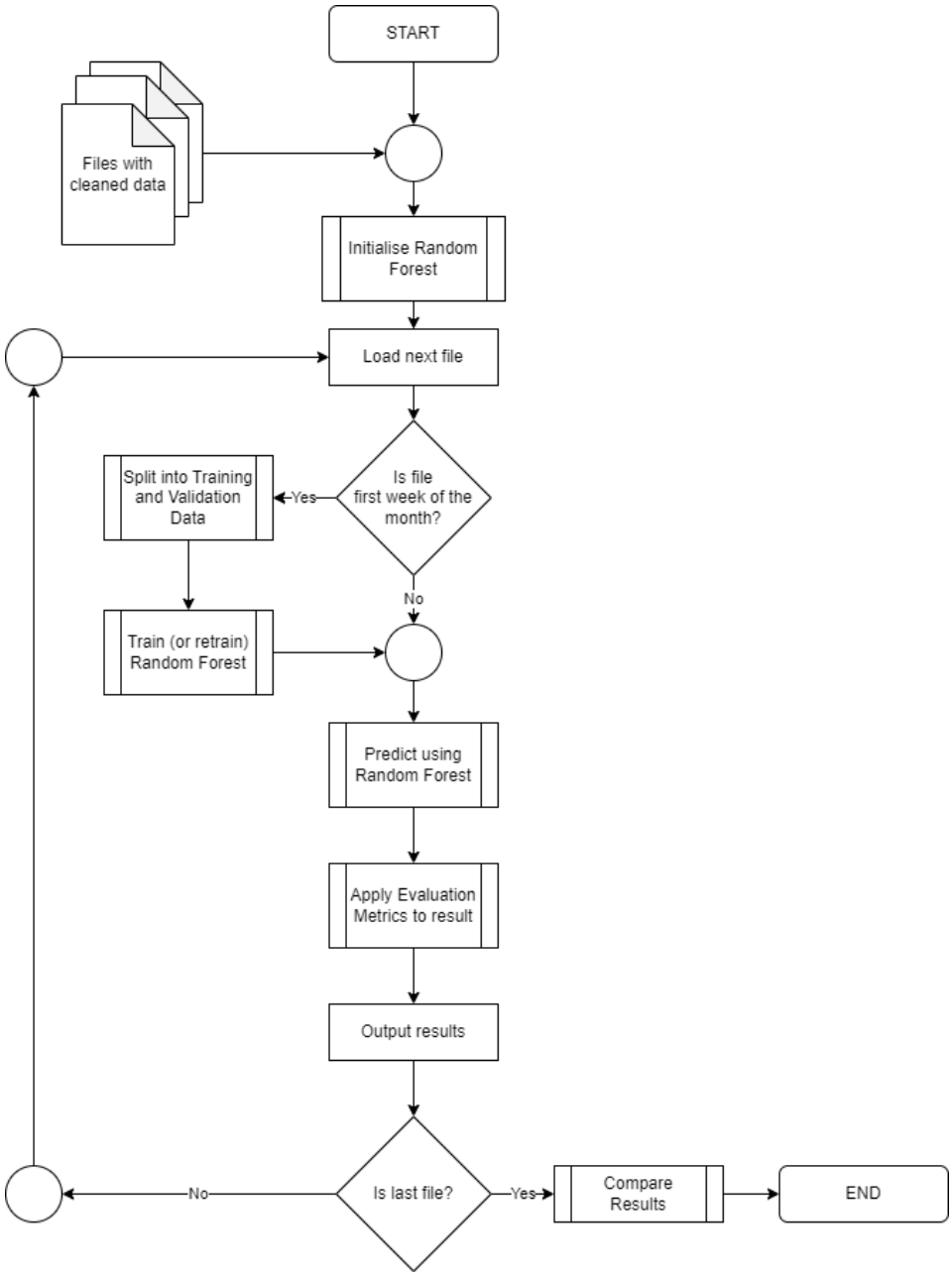

Figure 4.2: Overview of the initial Anomaly Detector

4.4.2.2 Enhancement of the Anomaly Detector

As stated earlier, three selected enhancements were used to improve the performance of the Anomaly Detector that was developed in Section 4.4.2.1. The first of these was the SMOTE oversampling technique [48], whose goal was to reduce the impact of the Class Imbalance Problem described in Section 3.3.3, which is a common issue encountered within Anomaly Detection problems. The second technique was to converge on a more optimal probability-based threshold for the Anomaly Detector, rather than the default value of 0.5 used by the code implemented through the Scikit-learn library [78].

The SMOTE technique makes use of a different Python library within the implementation, namely the Imbalanced-learn library [79]. The appropriate command from this library was inserted directly after the data was split into training and test sets, enabling the Random Forest to be trained on data that had been oversampled, hence mitigating the impact of Class Imbalance. Meanwhile, to implement Probability Thresholding, the line modified was that which was responsible for initiating the process of predicting whether a bet is anomalous or not. This modification caused the Random Forest to return the probability of a bet being anomalous as well as the probability that a bet is non-anomalous, rather than just a single value representing the prediction. From there, several different thresholds were attempted, starting from 0.5 (the default from the Scikit-learn library), increasing in increments of 0.05 up until a maximum value of 0.95. The results were then compared to one another to determine which value presented the best results. This value was then selected as the optimal threshold.

As stated in Section 2.5, Concept Drift refers to changes that take place in the underlying distribution of the data under consideration over time [22]. As outlined through Section 1.1, when applied to the field of sports betting, such changes could be caused by real-world events, such as impact of the COVID-19 pandemic or major international tournaments, such as the FIFA World Cup. Such events may cause changes in the patterns concealed within the data, both favourably or detrimentally [20, 21].

Based on the literature that was encountered in Section 2.5, it was decided that the Concept Drift Detector that would be implemented would use the Adaptive Windowing technique [74] owing to the fact that it does not require the size of the input to be predeclared. To implement this technique within the already developed algorithms, the River Python library was used [80]. Within the context of expanding, enhancing and improving the algorithms developed in Section 4.4.2.1, the Concept Drift Detectors were used to better establish when to retrain the algorithm. Rather than periodically retraining the Anomaly Detector, the Concept Drift Detector would establish when the underlying data distribution shifted and the model was retrained dynamically and as required, rather than

at regular intervals. Once Concept Drift Detection had been incorporated and accounted for, the same evaluation metrics were used to evaluate performance so as to maintain a like-for-like comparison. The enhancements that were made to the Anomaly Detector during this stage of the implementation can be viewed in Figure 4.3, where the components in white represent components of the original Anomaly Detector and components in blue represent the enhancements made to the algorithm.

4.5 Experiment 2: Extending the Anomaly Detector to a Multi-Class Customer Classifier

4.5.1 Overview

This section of the implementation focuses on addressing the second objective of this dissertation which refers to the expansion of the Anomaly Detector developed in Section 4.4.2.1 such that it becomes a multi-class classifier. Once this modification was applied to the entire dataset, the algorithm was also able to identify the type of customer who placed any given bet and, by extension, enabled the algorithm to differentiate between the different types of anomalous bets. The three algorithms whose performances were compared to one another in this experiment were a Random Forest Classifier, a Support Vector Machine and a Logistic Regression Classifier. Once the best performing of these three was identified, it was further developed through some enhancements which were added on so as to maximise the performance of this algorithm. More specifically, these were two of the enhancements used in the first experiment of this dissertation: SMOTE and Concept Drift Detection. Hence, this experiment also satisfied the third objective.

4.5.2 Approach Taken

4.5.2.1 Selected Algorithms

For this second experiment, three algorithms were selected to identify which of them produced the best performance for the task at hand. Based on the literature encountered in Sections 3.5.3, 3.6.3 and 3.7, the Random Forest Classifier, Support Vector Machine and Logistic Regression Classifier were among the most common techniques cited to perform the task of multi-class classification.

In view of this, three algorithms were selected to be used to perform multi-class classifications to identify the type of customer that placed a given bet, namely Logistic Regression, Support Vector Machines and another Random Forest. In terms of coding these

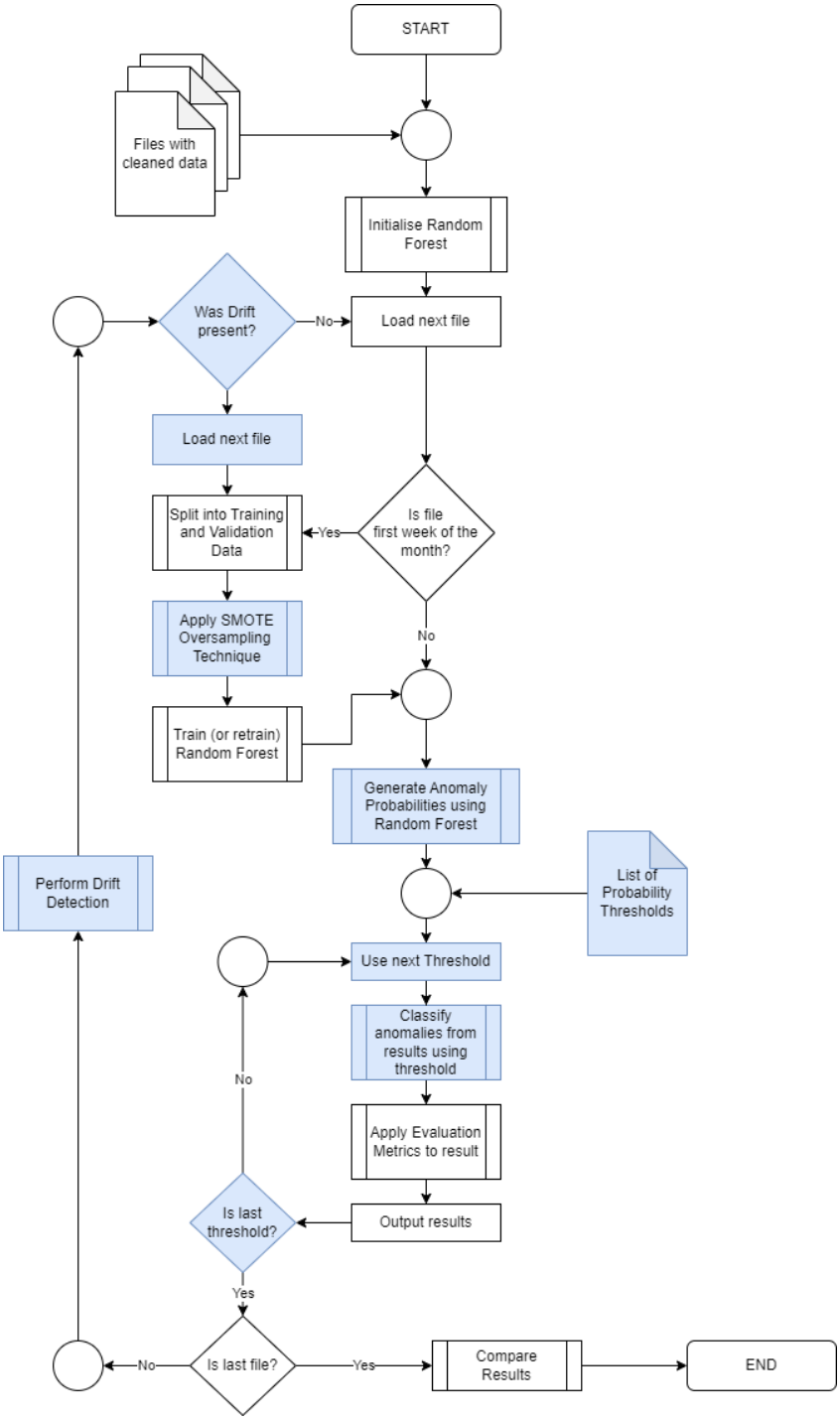

Figure 4.3: Enhanced Anomaly Detector using Random Forest - components in blue represent changes from Figure 4.2

algorithms, all three were implemented on the entire database by making use of the Scikit-learn Python library [78]. In terms of implementation, when the data was loaded from the files, should the data file under consideration belong to the first week of the month, the algorithms were trained, or retrained, in a periodical fashion. Once the models were trained, all three multi-class classifiers attempted to make their predictions on which type of customer placed the bets under consideration. This process was then repeated for all data files in chronological order until all files were iterated through. Once results were available for all three classifiers, their performance was compared to determine which of the three was the best performing. These were also compared to the original Anomaly Detector, so as to compare the results generated by a multi-class classifier as opposed to those generated by a binary classifier.

Figure 4.4 demonstrates the enhancements and additions that were made to the algorithm in Section 4.4.2.1 to achieve the scope of this experiment using the implementation detailed in this section. It should be noted that elements of the diagram with a white colour represent the components of the algorithm present in Section 4.4.2.1, while components in red represent the improvements made to the algorithm in this section.

Furthermore, just as some enhancements were applicable to the original Anomaly Detector, enhancements were also applicable to the Anomaly Classifiers to maximise their performance. Before these enhancements were made, the best performing of the three classifiers was identified, based on the results that all three classifiers generated. Once identified, two enhancements were made to this classifier, namely SMOTE and Concept Drift Detection. As mentioned earlier, SMOTE was applied immediately after splitting into training and test sets so as to address the Class Imbalance problem by oversampling and training using this oversampled data. Furthermore, Concept Drift remains an issue of concern, so another enhancement that was made to the Anomaly Classfier was implemented, just as was done in Section 4.4.2.2, where a Concept Drift Detector was introduced. This detector was then used to determine when retraining would occur, instead of retraining taking place at regular intervals. Once these enhancements were in place, a final implementation was developed and its performance was compared to the performance of the Anomaly Classifiers before applying these enhancements, as well as to other similar works found in the literature. Figure 4.5 details the final version of the algorithm, including Concept Drift Detection. Within this diagram, components in white represent those that originate from the original Anomaly Detector, while components that represent the enhancements made to the algorithm enabling it to become a multi-class classifier are visualised in red and further enhancements made to maximise the performance of the system are visualised in blue.

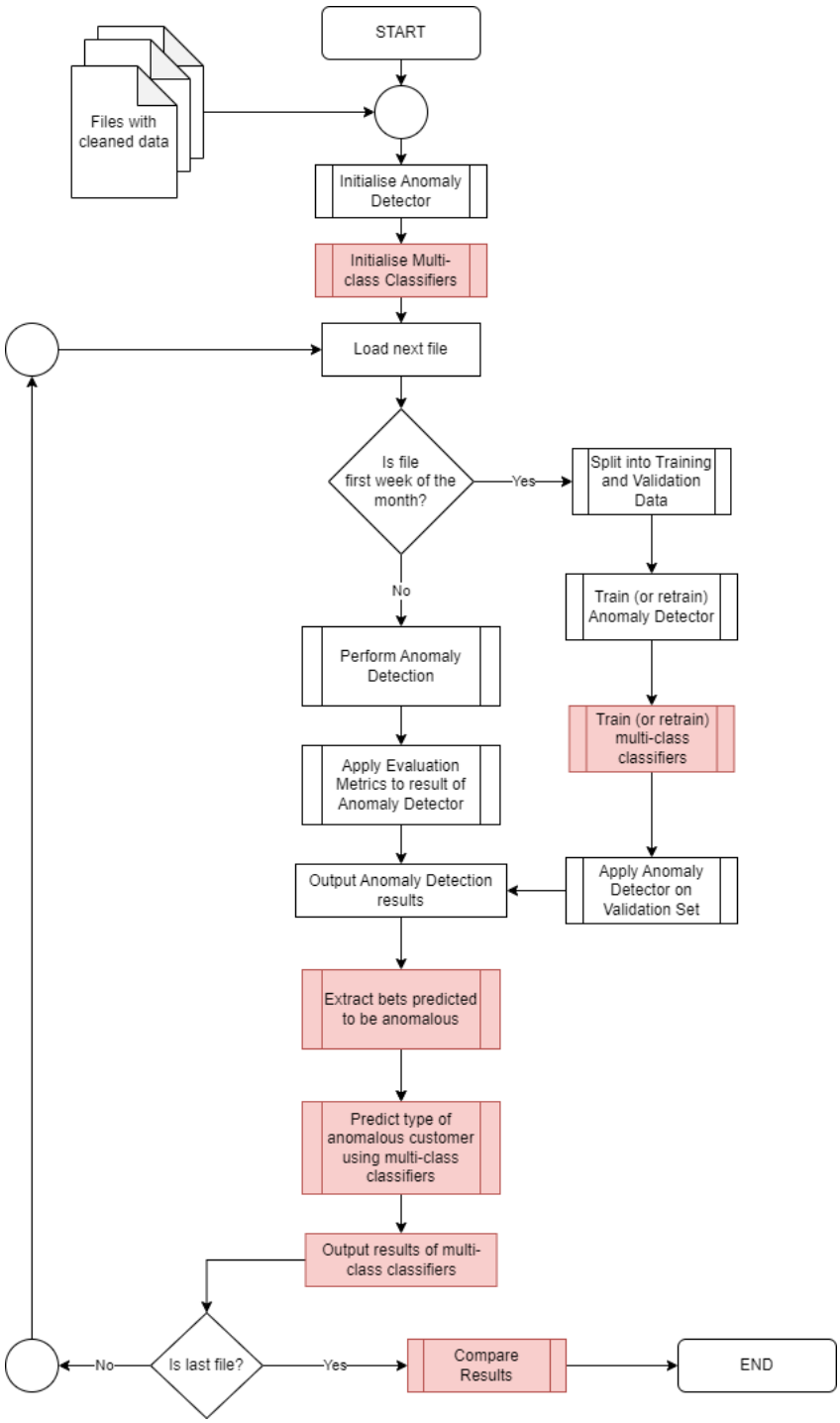

Figure 4.4: Implementation enhancing Anomaly Detector using multi-class classifiers for comparison - components in red represent changes made in this section from Figure 4.2

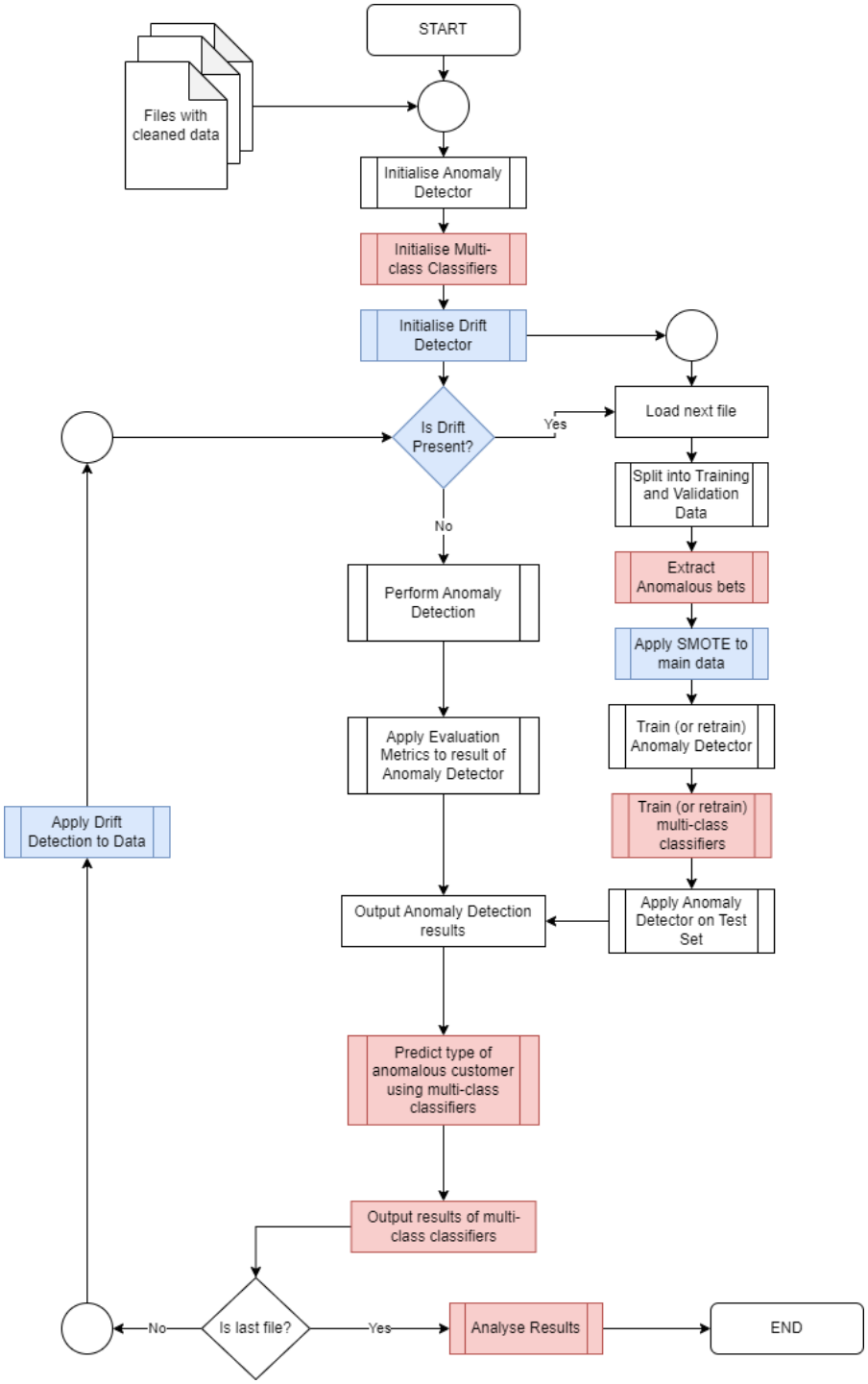

Figure 4.5: Final developed system including Anomaly Detector, Customer Classifiers and Concept Drift - Components in Blue indicate enhancements while components in Red indicate the expansion made to include multi-class classification

4.6 Evaluation, Benchmarking, Libraries and Parameters

4.6.1 Evaluation and Benchmarking

Across all the different experiments that were performed, it is the same five metrics that were utilised in order to measure the performance of the implemented algorithms. Three of these metrics were Precision, Recall and Specificity, with the other two, namely F1 Score and Geometric Mean, being metrics that were calculated based on two of the initial three metrics. For the first experiment performed, Random Forests were applied to Anomaly Detection. The decision to make use of these metrics has its source in the literature, with several papers taking on a similar task having used these metrics [25, 69, 72, 81, 71, 26]. Of these five metrics, Precision and Specificity were given less importance as they require True Negative samples included in their calculations [25, 69, 72, 81, 71, 26]. Furthermore, owing to the implementation of Anomaly Detectors using similar algorithms in these works, they were also to be used as a benchmark for the implemented algorithms. Following this, enhancements were made to the aforementioned algorithm using SMOTE, different thresholds and Concept Drift Detectors to attempt to improve the performance. Since it is the same algorithm being evaluated, the same metrics were used and the same papers could be used as a benchmark.

The second experiment involved the application of three classification algorithms, namely a Random Forest, a Support Vector Machine and a Logistic Regression classifier, which expanded the Anomaly Detector implemented from a binary classifier into a multi-class classifier, capable of providing more information than the initial detector. In terms of metrics, the same metrics were identified owing to their use within the literature [11, 51, 58, 57]. Since these three algorithms served a purpose of classification, different papers were used for benchmarking, particularly that of Kim et al. and that of Tsykunov, which contain applications of the algorithms implemented in this dissertation used within similar contexts [11, 51]. Another reason why different papers were used to evaluate this experiment was to ensure that a like-for-like comparison took place between the algorithms being developed and those being used as a benchmark during evaluation. It should also be noted here that, in order to cater for the extension from binary classification into multi-class classification, the metrics make use of a weighted average adaptation. Such mechanisms for Precision, Recall and Specificity operate by calculating a one-vs-all version of the traditional metrics for each individual classification, and then performing a weighted average dependant on the number of data elements in each classification [82]. Hence, not only is the fact that multiple classes are being evaluated accounted for, but also the fact that these classes are imbalanced, as the weighted average method takes

into account the uneven distribution of data between the different classes [82].

Finally, both experiments applied Concept Drift Detection to enhance the algorithms developed during said experiments. Since this enhancement only improves the previous algorithms by switching them from a system of periodical retraining to a more dynamic one, comparisons were made of the same algorithms before and after the Concept Drift Detectors were applied. For this reason, the same metrics were used for evaluation and the same papers were used for benchmarking.

4.6.2 Libraries and Parameters

Throughout the implementation of the project, numerous Python libraries were utilised in order to construct the various algorithms put together and generate results accordingly. A list of these libraries and packages can be seen in Table 4.5.

Library	**Version**	**Used for**
Python	3.8.11	Implementation of the entire solution
Pandas	2.2.0	Initial loading of data and handling during preprocessing
Random	N/A	Used during preprocessing
Scikit-learn	1.4.0	Main library used to create Random Forests, Support Vector Machines and Logistic Regression
Imbalanced-learn	0.12.2	Library used to invoke SMOTE and the Specificity Metrics
River	0.21.0	Used to incorporate Concept Drift
DateTime	5.4	Used during preprocessing
NumPy	1.26.4	Used to obtain descriptive statistics
Pickle-secure	0.99.9	Used to store and load preprocessed data to reduce computation time

Table 4.5: Python libraries used in the implementation stage

Meanwhile, the ensuing table (Table 4.6) details the different parameters that were tested for each algorithm. It should be noted that, owing to the volume of data present, runtime presented itself as a major issue, with certain runs requiring hours, and sometimes even days, to complete. Therefore, when selecting the parameters to be used, a balance had to be struck between maximising the performance of the algorithm and maintaining a feasible runtime.

Experiment	Algorithm	Parameters	Values Tested	Selected
1	Random Forest	n_estimator	25, 50, 75, 100	100
	SMOTE	sampling strategy	0.1, 0.3, 0.5	0.3
		k_neighbours	3, 5, 7, 9	5
		random_state	100	100
	Thresholding	Probability Threshold	0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95	See Section 4.3
	Adaptive Windowing	N/A	N/A	N/A
2	Random Forest	n_estimator	25, 50, 75, 100	100
	Support Vector Machine	kernel	linear, rbf	linear
	Logistic Regression	N/A	N/A	N/A
	Adaptive Windowing	N/A	N/A	N/A
	SMOTE	k_neighbours	3, 5, 7, 9	5
		random_state	100	100

Table 4.6: List of Algorithm Parameters attempted and selected

4.7 Link to the Code for the Implementation

The implementation of the methods described in this section was performed through a Python Notebook (Jupyter notebook). This notebook, containing both code and results for the methods detailed in this section, can be found at the following Google Drive link.

https://drive.google.com/drive/folders/1xiqQl07k2YzBMKGMpjJtitCLT9R_OEO?uspsharing

4.8 Conclusion

The above chapter detailed the approach that was taken to implement the algorithms identified earlier into two main experiments so as to satisfy the objectives defined in Section 1.2. In the following section, the results of these algorithms will be presented and the performance indicated through these metrics will be evaluated.

Chapter 5

Results and Evaluation

5.1 Overview

Having implemented the process detailed in the previous chapter, the following chapter details the results obtained by these algorithms and compares the performance of each algorithm to that of algorithms in similar works. Within the chapter, sections are split according to the experiments performed in Chapter 4, with Section 5.2 focusing on the first experiment and Section 5.3 focusing on the second experiment, with Section 5.4 summarising the outcome.

5.2 Experiment 1: Detection of Anomalous Bets

5.2.1 Initial Random Forest Anomaly Detector

The scope of the Anomaly Detector developed for this objective was to detect suspicious bets throughout the given data set. The algorithm that was selected to accomplish this task was a Random Forest Classifier. This algorithm was selected as it was observed in multiple sources to perform the task of Anomaly Detection. Furthermore, as mentioned in Section 4.4.2.1, five metrics were used in order to evaluate the performance of the Anomaly Detector developed and compare how it performed in relation to other similar Anomaly Detectors in the literature. These metrics were Precision, Recall, F1 Score, Specificity and Geometric Mean. For the reasons specified in Section 4.2.3, while the data files were split and the experiments performed in a weekly manner, the results presented in the following tables throughout the chapter are visualised by month, taking an average of the results obtained for each week in that month. The full weekly split of results can be

found in Appendix B (see Appendix B), while the monthly result for the Anomaly Detector can be seen in Table 5.1.

Random Forest					
Month	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.3134	0.1354	0.1888	0.9986	0.3269
2	0.3306	0.2022	0.2476	0.9979	0.4005
3	0.3067	0.2364	0.2549	0.9951	0.4593
4	0.3353	0.2590	0.2602	0.9925	0.4788
5	0.3131	0.2034	0.2381	0.9927	0.4141
6	0.3062	0.2039	0.2342	0.9929	0.4107
7	0.3450	0.1862	0.2321	0.9975	0.3790
8	0.3150	0.1589	0.2066	0.9985	0.3122
9	0.3396	0.2436	0.2754	0.9964	0.4582
10	0.3387	0.2331	0.2670	0.9964	0.4513
11	0.2566	0.1704	0.1957	0.9939	0.3629
12	0.2982	0.2072	0.2204	0.9896	0.4315
Average	**0.3182**	**0.2030**	**0.2364**	**0.9957**	**0.4049**

Table 5.1: Monthly Result of Random Forest Anomaly Detector

In terms of performance, several works made use of Random Forests as an Anomaly Detector within a sports betting context. While none of the encountered literature matches the perspective of this dissertation exactly (Anomaly Detection in Sports Betting from the perspective of the betting operator), several implementations of other authors were found to be close enough to enable proper comparison with the results generated. For example, some of the papers concerned deal with fraud detection on a transactional level, while another created a generic framework for Anomaly Detection. Since attempts to gain an advantage against a betting operator can be construed as an attempt to defraud the betting operator, fraud detection papers were deemed comparable to the process of detecting anomalous customers within a list of betting transactions, as the scope of these projects is also to detect anomalous transactions within a data set of transactional records. Table 5.2 outlines the initial results and indicates how these compare to other works. It should be noted that numbers marked with an * were calculated based on the confusion matrices included as part of other authors' results.

From Table 5.2, it can be seen that, in their initial forms, the developed Anomaly Detector lags some way behind those seen in works by other authors. However, the implementation developed maintains a Recall, F1 Score and Geometric Mean that come close to, or exceed, those seen in the work by Shah and Sharma [69]. It can therefore be said that the Random Forest implementation does successfully exceed the performance

Random Forest	Precision	Recall	F1 Score	Specificity	Geometric Mean
Implementation Used	0.3182	0.2030	0.2364	0.9957	0.4049
Shah and Sharma	0.9969	0.1506	0.2616	*0.9997**	*0.3880**
Elmrabit et al.	0.8816	0.8850	0.8487	-	-
Fatemifar et al.	-	-	-	-	0.8369
Aftab et al.	0.8780	0.8181	0.8471	-	-

Table 5.2: Comparison of Random Forest Anomaly Detector to others found in literature

of one of the benchmark papers in two out of the three key metrics, and gets close to the performance of the same paper in a third metric. Despite this, when comparing the results of the developed Anomaly Detector to the other works encountered, improvements are required so as to reach the standards set by other authors [69, 72, 25, 68]. Therefore, the decision to explore a number of enhancements to improve performance can be adequately justified.

One potential reason for the performance of the developed implementation to be some way off those achieved by other researchers could be the imbalance present within the data set. As can be seen in Table 4.3, only approximately 0.54% of all bets placed are anomalous. Such an imbalance in the training data could result in the results produced by the Anomaly Detector being heavily skewed towards the non-anomalous data samples. Hence, it was decided that out of the three enhancements that were identified, the first of them to be applied would be the SMOTE oversampling technique, as this technique is used specifically to counter the effect of the Class Imbalance Problem.

5.2.2 Enhancements to the Initial Anomaly Detector

Throughout the rest of this experiment, three potential enhancements were implemented to improve the results of the Anomaly Detector that were observed in Section 5.2.1. Through the literature reviewed, the established methods that were attempted were the use of an oversampling technique, namely SMOTE, the use of different probability thresholds required to classify a bet as anomalous and the implementation of a Concept Drift Detector. This section can therefore be split into three sets of results, one for each technique.

5.2.2.1 Oversampling Technique - SMOTE

In this experiment, the SMOTE oversampling technique was applied to the training data prior to training in an attempt to better the performance of the Random Forest Anomaly

Detector. As specified in Section 4.6, the same metrics were then applied to measure the performance of the Anomaly Detector after the incorporation of SMOTE. The scores of these metrics were then compared to the results of the Random Forest Anomaly Detector without SMOTE, to determine whether or not the application of SMOTE improved the performance of the implementation. Finally, the results were compared again to the papers used as a benchmark in Section 5.2.1, as well as to an additional paper by Ahsan et al. which also compares the use of a Random Forest in Anomaly Detection before and after the application of SMOTE [26]. Table 5.3 details the results obtained by the Anomaly Detector incorporating the SMOTE technique, taking an average for the whole month. (See Appendix B for the full weekly result)

Random Forest incl. SMOTE					
Month	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.2539	0.1679	0.1964	0.9965	0.3743
2	0.3277	0.2142	0.2569	0.9982	0.4215
3	0.2638	0.2606	0.2431	0.9916	0.4836
4	0.3172	0.2550	0.2619	0.9937	0.4738
5	0.2798	0.2366	0.2429	0.9876	0.4545
6	0.2248	0.2791	0.2150	0.9764	0.5031
7	0.2645	0.2024	0.2260	0.9964	0.3956
8	0.2634	0.1848	0.2145	0.9963	0.3589
9	0.2822	0.2061	0.2362	0.9969	0.3953
10	0.2883	0.2332	0.2509	0.9957	0.4498
11	0.2289	0.1760	0.1958	0.9951	0.3628
12	0.2433	0.2143	0.2138	0.9936	0.4373
Average	**0.2698**	**0.2192**	**0.2295**	**0.9932**	**0.4259**

Table 5.3: Results of Random Forest Anomaly Detector including SMOTE

When comparing these results with those of the Anomaly Detector without SMOTE, the original Random Forest (without SMOTE) was found to outperform the Random Forest including SMOTE in 3 of the 5 metrics, namely Precision, Specificity and F1 Score. However, in Section 4.6, both Precision and Specificity were said to have less importance than the other three metrics, based on the author's findings in the available literature. Therefore, when looking only at the three metrics with the higher importance, namely Recall, F1 Score and Geometric Mean, the Random Forest with SMOTE then outperforms the original in two metrics out of three.

Looking at how the two implementations (both with SMOTE and without) performed compared to the benchmark studies that were identified, similar conclusions can be made to those that were made in Section 5.2.1. Performance in the key metrics approaches or

Random Forest	Precision	Recall	F1 Score	Specificity	G-Mean
Base Implementation	0.3182	0.2030	0.2364	0.9957	0.4049
Base Implem. + SMOTE	0.2698	0.2192	0.2295	*0.9932*	*0.4259*
Shah and Sharma	0.9969	0.1506	0.2616	*0.9997*	*0.3880*
Elmrabit et al.	0.8816	0.8850	0.8487	-	-
Fatemifar et al.	-	-	-	-	*0.8369*
Aftab et al.	0.8780	0.8181	0.8471	-	-

Table 5.4: Comparison of Random Forest Anomaly Detector both with and without SMOTE to others in literature

exceeds those obtained by one other work found in literature, but requires improvement to be on par with the other works. However, when comparing to the work of Ahsan et al., it should be noted that although the implemented solutions register lower performances than Ahsan et al. registered when applying SMOTE to their Random Forest, all metrics recorded a decrease in performance in Ahsan et al.'s work, while in the developed solution for this dissertation, the Recall score improves after the application of SMOTE [26]. Furthermore, the decreases seen in the developed solution are smaller decreases in magnitude than those seen in Ahsan et al's results, as can be seen in Table 5.5 [26]. When considering that the application of SMOTE did not improve the results of the base implementation of the Anomaly Detector, it was noted that, despite the different parameters that were used during the implementation (seen in Table 4.6), there could have been another configuration of parameters that would have provided a more successful result. Furthermore, it was determined that another problem could also be affecting the performance of the Anomaly Detector. Furthermore, when also taking into account the sheer number of bets present within the data set, as well as the fact that time passing is also a factor that is considered by the data set, the inclusion of Concept Drift Detection as an enhancement to the Anomaly Detector was further justified.

Random Forest	Precision	Recall	F1 Score	Specificity	G-Mean
Base Implementation	0.3182	0.2030	0.2364	0.9957	0.4049
Base Implem. + SMOTE	0.2698	0.2192	0.2295	0.9932	0.4259
Ahsan et al. - No SMOTE	0.9347	0.9359	0.9144	-	-
Ahsan et al. - SMOTE	0.9245	0.9234	0.8921	-	-
Improvement - Implementation	- 0.0484	0.0162	- 0.0069	- 0.0025	*0.0210*
Improvement - Ahsan et al.	- 0.0102	- 0.0125	- 0.0223	-	-

Table 5.5: Comparison of the differences between Random Forest anomaly detectors before and after SMOTE applications

5.2.2.2 Probability Thresholding

When observing the results that were found by varying the probability threshold required to classify a given data element as anomalous, different trends were observed for different metrics. The trends belonging to these metrics are visible in Table 5.6 and can be split into three types.

The Specificity metric was the first, where it starts at a value of 0.9950 when any element with a likelihood of 0.5 or greater is classified as anomalous. It then continuously increases as the probability threshold increases. When the minimum required probability to classify an element as anomalous reaches 0.8, the Specificity metric reaches a value of 1, which is the maximum value possible for this metric. It then remains so even as the threshold continues to increase.

The second pattern was observed on the Precision metric. Similarly to Specificity, as the threshold increases from 0.5, the larger the result given by the Precision metric. However, in this case, the Precision reaches its peak when the minimum probability required to classify a bet as anomalous is 0.75. Then, as the required probability continues to increase, the Precision drops sharply and significantly.

The third and final pattern concerns the remaining three metrics, namely Recall, F1 Score and Geometric mean, which represent all three of the prioritised metrics mentioned in Section 4.6. Unlike the previous two metrics, all three of these metrics follow the same trend. They are all at their highest when the minimum probability required for a bet to be classified as anomalous is 0.5. As this threshold increases, they continually get lower and lower, with the algorithm continuing to get worse as the minimum threshold hones in on only the most likely examples.

Averages by Threshold					
Threshold	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
0.50	0.3109	0.2034	0.2342	0.9950	0.4072
0.55	0.3286	0.1648	0.2124	0.9976	0.3462
0.60	0.3450	0.1353	0.1859	0.9989	0.2897
0.65	0.3602	0.1134	0.1609	0.9995	0.2395
0.70	0.3790	0.0977	0.1417	0.9998	0.1988
0.75	0.3894	0.0849	0.1262	0.9999	0.1655
0.80	0.3668	0.0741	0.1133	1.0000	0.1419
0.85	0.3134	0.0634	0.1003	1.0000	0.1260
0.90	0.2472	0.0520	0.0853	1.0000	0.1133
0.95	0.2487	0.0392	0.0672	1.0000	0.0981

Table 5.6: Results of running Random Forest algorithm with different thresholds

Since it was established that the three priority metrics were Recall, F1 Score and Geometric Mean, it could be established that the best-performing probability threshold was 0.5. Furthermore, not only do the results for this threshold maximise the three priority metrics, but it is the most comparable to the results obtained by the implementations in Section 5.3. By extension, this version is also comparable to works found in literature, which are observable in Tables 5.2 and 5.4, while several of those with higher thresholds but worse performance are not.

5.2.2.3 Concept Drift Detector

The third enhancement that was implemented attempted to address the problem of Concept Drift. With the underlying distribution of data being subject to change as time goes by, as is the case for sports betting data, the optimal solution, as established by the literature, is the retraining of the algorithms concerned [74, 77]. Prior to this enhancement being applied, the Anomaly Detector was retrained periodically using the data from the first week of any given month. By implementing a Concept Drift Detector, the performance of the developed solution was assessed with the Anomaly Detector being trained dynamically and when necessary. Based on observations during the runtime, this resulted in the model retraining itself approximately once every two weeks, depending on the events taking place at that time. The results of the Anomaly Detector including a Concept Drift Detector for dynamic retraining can be seen in Table 5.7.

Anomaly Detector using Concept Drift - By Month					
Month	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.7608	0.5210	0.6181	0.9993	0.7205
2	0.8019	0.5578	0.6579	0.9994	0.7466
3	0.8241	0.5830	0.6828	0.9994	0.7633
4	0.7690	0.5663	0.6522	0.9992	0.7521
5	0.7677	0.5897	0.6668	0.9988	0.7668
6	0.7858	0.5872	0.6720	0.9989	0.7658
7	0.7955	0.5531	0.6524	0.9992	0.7432
8	0.8333	0.5970	0.6955	0.9993	0.7723
9	0.8165	0.5680	0.6698	0.9994	0.7531
10	0.7968	0.5638	0.6601	0.9993	0.7505
11	0.7587	0.5229	0.6189	0.9992	0.7228
12	0.7597	0.5105	0.6102	0.9993	0.7140
Average	**0.7892**	**0.5600**	**0.6547**	**0.9992**	**0.7476**

Table 5.7: Results of Anomaly Detector using Concept Drift

As can be observed, the overall performance of the Anomaly Detector including the

Concept Drift Detector significantly exceeds the performance of the Anomaly Detector without it. Through the results observed, it was found that the dynamically retrained algorithm performed better under all five performance metrics used.

Furthermore, comparisons were also made between the implemented Anomaly Detector both with and without the Concept Drift Detector, alongside the same similar Anomaly Detectors implemented by other authors in the literature that were used in Section 5.2.1. This was done to establish whether or not the inclusion of Concept Drift Detection in the retraining process provides an additional boost in performance compared to detectors that were trained periodically. Furthermore, these comparisons also served to establish how the performance of the developed Anomaly Detector including the Concept Drift Detection compared to the detectors and classifiers developed by other authors in their own works. This comparison is illustrated in Table 5.8.

Anomaly Detector	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
Base Implementation	0.3182	0.2030	0.2364	0.9957	0.4049
Base Implementation + SMOTE	0.2698	0.2192	0.2295	0.9932	0.4259
Base Implem. + SMOTE + Drift Detection	0.7892	0.5600	0.6547	0.9992	0.7476
Shah and Sharma	0.9969	0.1506	0.2616	*0.9997*	*0.3880*
Elmrabit et al.	0.8816	0.8850	0.8487	-	-
Fatemifar et al.	-	-	-	-	0.8369
Aftab et al.	0.8780	0.8181	0.8471	-	-

Table 5.8: Comparison between Anomaly Detector implementations with and without Drift Detection and other Anomaly Detectors from literature

As has already been stated, the performance of the Anomaly Detector including the Concept Drift Detector significantly outperforms the detector which is retrained periodically. Comparing the performance of the Anomaly Detector including Concept Drift Detection to the other works in the literature, the position of the developed implementation is in a far better position than it was before Concept Drift Detection was applied. When looking at the papers selected for benchmarking, the values of the Recall, F1 Score and Geometric Mean now significantly exceed the performance of the work by Shah and Sharma, while the Precision and Specificity values have become much closer. Looking at the other works, while performance has become much closer to that achieved by other authors, the results of the developed implementation remain short of the performance achieved by said other authors. Also noted through these comparisons was that the introduction of SMOTE, while producing some improvement, did not make a significant difference to the performance of the Anomaly Detector.

The improved performance of the algorithm using Concept Drift Detection provided a potential explanation for this. Namely, this observation was that within the data used for this dissertation, the volume is such that the impact of Concept Drift presents a much greater effect on the model during training than the problem of Class Imbalance. It should also be noted that anomalous bets can be further subdivided into multiple categories. The improvement observed in the results, combined with the fact that there are multiple types of anomalies present within the data set both served to further justify the expansion of the Anomaly Detector into a multi-class classifier to attempt to indicate the type of customer present, as this may also serve to improve the results that have been observed thus far.

5.3 Experiment 2: Expanding the Anomaly Detector into a Multi-Class Customer Classifier

5.3.1 Expanding into Multi-Class Classifier

Within the data set, there are different possible reasons for a bet to be anomalous. An anomaly could indicate an undesirable customer, a sharp customer, or a customer who is particularly favourable. Therefore, this is what was addressed during the second experiment. This experiment used three different machine learning algorithms and applied them over the entire dataset in order to augment the developed Anomaly Detector, indirectly also determining the type of customer that placed such an anomalous bet. These three techniques were a Random Forest Classifier, a Support Vector Machine and a Logistic Regression classifier.

5.3.1.1 Logistic Regression Classifier

The first attempted classifier was the Logistic Regression Classifier, whose results can be seen in Table 5.9. Although traditionally a binary classification algorithm, the Scikit-learn library enabled such a classifier to be applied to a scenario such as this one, where there are multiple possible classifications for each data element, such as this one and where anomalies can be of three different types [78].

These results, when compared to the performance of the Anomaly Detector itself, already prove to provide better results in two key metrics, namely the Recall and F1 Score. Furthermore, the Precision is also seen to perform significantly better, while the Geometric Mean remains approximately in line with the Anomaly Detector. This much-improved performance comes at the cost of the Specificity metric; however, this metric was established to not be a priority based on the literature encountered in earlier sections of

Logistic Regression - By Month					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6343	0.7964	0.7062	0.2036	0.4026
2	0.6120	0.7823	0.6867	0.2177	0.4126
3	0.6260	0.7912	0.6990	0.2088	0.4064
4	0.5438	0.7350	0.6241	0.2650	0.4362
5	0.5045	0.6835	0.5773	0.3165	0.4610
6	0.6238	0.7877	0.6962	0.2123	0.4087
7	0.6466	0.8012	0.7157	0.1988	0.3991
8	0.6465	0.7960	0.7134	0.2040	0.4029
9	0.6556	0.8012	0.7209	0.1988	0.3991
10	0.6388	0.7961	0.7088	0.2039	0.4029
11	0.6585	0.7935	0.7193	0.2065	0.4047
12	0.6541	0.8088	0.7233	0.1912	0.3933
Average	**0.6204**	**0.7811**	**0.6909**	**0.2189**	**0.4108**

Table 5.9: Results of the Logistic Regression Classifier when categorising the bets by the type of customer

this dissertation. Even so, the results of this classifier represent the highest values for Precision, Recall and F1 score that have been generated by the classifier so far.

Logistic Regression Classifier	Precision	Recall	F1 Score	Specificity	G-Mean
Implementation	0.6204	0.7811	0.6909	0.2189	0.4108
Kim et al.	0.3636	0.8570	0.5016	0.8170	0.8368
Purwandari et al.	0.9080	0.9020	0.8940	-	-
Tsykunov	0.6700	0.6700	0.4800	-	-

Table 5.10: Comparison of the Logistic Regression classifiers to other similar classifiers in literature

When looking at how this classifier compares to other similar classifiers found in the literature (see Table 5.10), three similar classifiers have been encountered. Looking at the key metrics, of the four classifiers, the developed implementation ranks as the second-best performing when taking into account F1 Score and third-best in terms of Recall. Although the third-best in Recall, the developed Logistic Regression was quite close to the better performing Recall in the better performing papers, which had not been the case with the original detector. However, two of the four papers do not present a Specificity or Geometric Mean result. The developed algorithm also outperforms one of the works considered in the Precision metric, while being within touching distance of another. It can therefore be said that the algorithm developed performs well when compared to those developed by other authors.

5.3.1.2 Support Vector Machines

The second algorithm attempted was the Support Vector Machine. As mentioned in Section 3.6, while this algorithm is also typically used for binary classification, there are approaches that can adapt it for multi-class classification. According to the Scikit-learn documentation, the library does allow for the application of the implementation to multi-class scenarios and actively caters for such cases in its underlying code [78]. Table 5.11 shows the results of the attempt to classify the different types of customers present within the data through the bets that they place, including anomalous types.

Support Vector Machine - By Month					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6729	0.5720	0.6118	0.4400	0.4882
2	0.6202	0.7823	0.6919	0.2177	0.4126
3	0.6260	0.7912	0.6989	0.2088	0.4065
4	0.5438	0.7349	0.6241	0.2651	0.4362
5	0.5677	0.6020	0.5839	0.4041	0.4896
6	0.6208	0.7878	0.6944	0.2122	0.4086
7	0.7116	0.0645	0.1173	0.9395	0.2425
8	0.6339	0.7961	0.7058	0.2039	0.4028
9	0.6555	0.8011	0.7208	0.1989	0.3992
10	0.6680	0.6561	0.6616	0.3366	0.4689
11	0.6295	0.7933	0.7020	0.2067	0.4047
12	0.1320	0.1759	0.1018	0.8241	0.3807
Average	**0.5902**	**0.6298**	**0.5762**	**0.3715**	**0.4117**

Table 5.11: Results of the Support Vector Machine when categorising the bets by the type of customer

When comparing the performance of this classifier to the performance of the Anomaly Detector, it operates with better performance than the original detector in the Precision, Recall and F1 Score metrics, while being on par with the detector in terms of the Geometric mean and lagging behind with the Specificity metrics.

Meanwhile, a comparison of the performance of the Support Vector Machine to the Logistic Regression classifier reveals that the Logistic Regression Classifier also outperforms the Support Vector Machine in three out of five metrics including two of the prioritised three, namely Precision, Recall and F1 Score. Meanwhile, the Support Vector Machine outperformed the Logistic Regression Classifier using the Specificity metric and marginally so when considering the Geometric Mean. It can therefore be said that the Logistic Regression Classifier outperforms the Support Vector Machine when trying to classify the type of customer that has placed a specific bet. More detailed results can be

Support Vector Machines	Precision	Recall	F1 Score	Specificity	G-Mean
Implementation	0.5902	0.6298	0.5762	0.3715	0.4117
Kim et al.	0.1667	0.8570	0.2791	0.7840	0.8197
Purwandari et al.	0.9140	0.9300	0.9219	-	-
Tsykunov	0.6500	0.2900	0.4000	-	-

Table 5.12: Comparison of the Support Vector Machine to other similar classifiers in literature

found in Table 5.15

Furthermore, when using a benchmark based on other similar classifiers in the literature, three main similar works were identified. Taking the key metrics of the developed Support Vector Machine, the implementation developed by the author of this dissertation performs better than a number of other Support Vector Machines found in the literature. Out of the papers listed in Table 5.12, the developed implementation ranks third using Recall and Precision, as well as second under F1 score. However, when comparing the performance of Logistic Regression classifers and Support Vector Machines by the same papers, it was noticed that, from an overall perspective, Logistic Regression outperformed Support Vector Machines for the identification of different types of anomalies.

5.3.1.3 Multi-Class Random Forest Classifier

The third and final algorithm implemented was the Random Forest Classifier - a multi-class variation of the algorithm used during the first experiment performed. The performance of this final algorithm applied to the separation and classification of the different types of customers placing the bets, separating the different types of anomalies, can be seen in Table 5.13.

Looking at the results of the Random Forest Classifier compared to other sources in the literature, similar results are seen. However, Purwandari et al., while making use of both Support Vector Machines and Logistic Regression in their work, do not make use of Random Forests [57]. This leaves two papers that may serve as a benchmark for the Random Forest Classifier, namely those of Kim et al. and of Tsykunov. As can be seen in Table 5.14, the developed Random Forest Classifier outperforms Kim et al. in Precision, Recall and F1 Score. Meanwhile, it also gets very close to the performance achieved by Tsykunov using all three metrics included by Tsykunov in his work.

Moreover, when comparing the performance of the Random Forest Classifier to the performance of the other two algorithms implemented, it was found that the Random Forest Classifier exceeded the performance of both other classifiers in all five metrics used. Looking at all three classifiers, according to the Precision, Recall and F1 Score, the

Multi-Class Random Forest Classifier - By Month					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.7335	0.7953	0.7629	0.3028	0.4865
2	0.7479	0.7979	0.7717	0.4022	0.5582
3	0.7553	0.7968	0.7752	0.4148	0.5687
4	0.7097	0.7267	0.7181	0.5062	0.6045
5	0.6980	0.7180	0.7078	0.5391	0.6199
6	0.7493	0.7696	0.7593	0.4501	0.5853
7	0.7612	0.7986	0.7792	0.4114	0.5665
8	0.7548	0.7954	0.7744	0.4205	0.5721
9	0.7660	0.8071	0.7858	0.4069	0.5656
10	0.7658	0.8129	0.7884	0.3926	0.5560
11	0.7605	0.8144	0.7861	0.3846	0.5487
12	0.7725	0.8212	0.7958	0.3775	0.5476
Average	**0.7479**	**0.7878**	**0.7671**	**0.4174**	**0.5650**

Table 5.13: Results of the Random Forest Classifier when categorising the bets by the type of customer

Random Forest Classifier	**Precision**	**Recall**	**F1 Score**	**Specificity**	**G-Mean**
Implementation	0.7479	0.7878	0.7671	0.4174	0.565
Kim et al.	0.3333	0.4290	0.3751	0.937	0.6340
Tsykunov	0.7700	0.7900	0.7800	-	-

Table 5.14: Comparison of the Random Forest Classifier to other similar classifiers in literature

Random Forest classifier achieved the highest scores, followed by the Logistic Regression Classifier. Meanwhile, for the remaining two metrics, the Random Forest Classifier still performs the best, but this time the next-best classifier is the Support Vector Machine. However, in all cases, the Random Forest Classifier produced the best results when attempting to identify the type of customer who placed a given bet.

5.3.1.4 Comparison of the Multi-Class Classifiers

The above comparisons between the different metrics for each multi-class classifier are summarised in Table 5.15. For reasons specified above, and owing to the fact that it performed the most similarly to the implementations from other works in literature with better performances, the Random Forest Classifier was determined to be the best performing implementation of the three attempted. Being the most successful implementation, the Random Forest Classifier was selected as the Anomaly Classifier to be further enhanced in the final part of this experiment. Furthermore, since it was the addition of both SMOTE

and a Concept Drift Detector that significantly improved the results of the Anomaly Detector, the decision to make use of these two techniques to maximise the performance of the multi-class Anomaly Classifier is, once again, further justified.

Classifier	Precision	Recall	F1 Score	Specificity	G-Mean
Random Forest	0.7479	0.7878	0.7671	0.4174	0.5650
Logistic Regression	0.6204	0.7811	0.6909	0.2189	0.4108
Support Vector Machine	0.5902	0.6298	0.5762	0.3715	0.4117

Table 5.15: Overview of the average results of each of the different implemented classifiers

5.3.2 Enhancement of Multi-Class Classifiers using SMOTE and Concept Drift Detection

The final part of this experiment investigated the application of SMOTE and Concept Drift to the multi-class classifier. This was done for two main reasons. The first was to attempt to maximise the performance of the multi-class classifier by mitigating the two biggest problems known to exist within similar data sets: Concept Drift and Class Imbalance. Meanwhile, the second reason is to enable an equivalent comparison between the performance achieved by a multi-class classifier, relative to a binary classifier such as the Anomaly Detector in the first experiment. Since the Random Forest Classifier was found to have the best results, it was this classifier that was selected to be enhanced. Once again, during the runtime, it was observed that the classifier was retrained, on average, once every two weeks, according to the fixtures and sports taking place during the time of year represented by the data file being passed to said classifier. The results of the application of Concept Drift Detection and SMOTE to the multi-class Random Forest Classifier can be seen in Table 5.16

Through the results of this experiment, it was found that the multi-class Anomaly Classifier performed better than the Anomaly Detector in four out of five metrics, including all three of the prioritised ones. Specifically, this superior performance was observed under Precision, Recall, F1 Score and Geometric Mean. Furthermore, as noted in Section 5.3.1, a better performance in these metrics came at the cost of Specificity. While this was observed to be an almost perfect result in the Anomaly Detector, this was reduced to a lower but still respectable 0.6922 in the enhanced multi-class classifier.

Furthermore, comparisons were not only made between the implemented Anomaly Detector and Anomaly Classifier, both with and without the Concept Drift Detector, but also to other classifiers in the literature that were used in earlier sections. This was done

Anomaly Classifier using Concept Drift - By Month					
Month	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.8829	0.889	0.886	0.6352	0.7492
2	0.8991	0.9009	0.9	0.707	0.7981
3	0.901	0.9028	0.9019	0.6937	0.7914
4	0.8664	0.8686	0.8674	0.7172	0.7886
5	0.8236	0.8268	0.8252	0.7253	0.7737
6	0.8863	0.8899	0.8881	0.6708	0.7726
7	0.9053	0.9074	0.9063	0.6916	0.7921
8	0.8999	0.9023	0.9011	0.6866	0.787
9	0.9041	0.9066	0.9053	0.6913	0.7916
10	0.901	0.9039	0.9024	0.7003	0.7956
11	0.9011	0.9038	0.9024	0.7047	0.798
12	0.9047	0.9076	0.9062	0.6826	0.7871
Average	**0.8896**	**0.8925**	**0.891**	**0.6922**	**0.7854**

Table 5.16: Results of Anomaly Classifier using Concept Drift

to establish whether or not the inclusion of Concept Drift Detection in the retraining process provided an additional boost in performance compared to detectors and classifiers that were trained periodically. Furthermore, this also served to establish whether the inclusion of Concept Drift Detection provided the developed classifier with a better performance compared to the classifiers implemented by other authors in their own works. This comparison can be seen in Table 5.17.

Anomaly Classifier	**Precision**	**Recall**	**F1 Score**	**Specificity**	**G-Mean**
Base Implementation	0.7479	0.7878	0.7671	0.4174	0.565
Base Implementation + SMOTE + Drift Detection	0.8896	0.8925	0.8910	0.6922	0.7854
Kim et al.	0.3333	0.429	0.3751	*0.937*	*0.6340*
Tsykunov	0.77	0.79	0.7800	-	-

Table 5.17: Comparison between Anomaly Classifier implementation with and without Drift Detection and other Anomaly Classifiers from literature

Once again, the application of Concept Drift in the retraining process has noticeably improved the performance compared to the implementation which retrained the classifiers periodically. It should also be noted that when compared to works by other authors who performed a similar task, the implementation that made use of Concept Drift Detection noticeably exceeded the performance of both benchmarking papers in almost all metrics, except for Specificity, which was not one of the prioritised metrics.

5.4 Conclusion

Throughout this chapter, the various results that were obtained across all the experiments performed as part of this dissertation were presented, discussed and evaluated. The first part of the chapter investigated the performance of an Anomaly Detector at identifying anomalous bets. Next, three different enhancements were applied to this Anomaly Detector so as to improve its performance. From these, it was established that the algorithm had mixed results when the SMOTE oversampling technique was applied to the data prior to training. Furthermore, better results were obtained when data elements were classified as anomalous if the likelihood of the bet being anomalous was greater than 0.5. The third enhancement investigated whether the application of a Concept Drift Detector to retrain the Anomaly Detector when necessary induced a better performance than when it was being retrained periodically. Based on the results obtained in this section and compared to the literature, it was confirmed that retraining the algorithm when Concept Drift was detected did provide better results than periodical retraining. The second experiment took three different classification algorithms which can be adapted for multi-class classification and attempted to establish which would be the most suitable to be applied to identify the various types of anomalies by categorising the different types of customers present within the entire data set. From this, it was established that a multi-class Random Forest Classifier returned the best performance. Finally, the experiment also investigated whether the application of SMOTE in conjunction with a Concept Drift Detector to the multi-class Random Forest Classifier also improved its performance, as it did for the Anomaly Detector. Based on the results obtained in this section and compared to the literature, it was confirmed that the inclusion of these enhancements provided the best performance of all the implementations developed for this dissertation. Moreover, it was also observed that the results of this final enhanced multi-class classifier also outperformed the benchmark papers that were identified from the literature.

Chapter 6

Conclusion

This dissertation investigated numerous potential Artificial Intelligence algorithms and techniques to develop a system capable of identifying anomalous customers from the perspective of a betting operator, where such customers have the potential to be exceptionally good, undesirable or sharp with their knowledge. This process was done incrementally where each step in the process made its own contribution to the developed solution such that the complete implementation is assembled at the end of the final step. The entire process was then split into a total of three objectives which were used to design and perform two experiments.

The first objective required the implementation of an Anomaly Detector capable of determining whether or not a given bet could be said to be anomalous. This objective was satisfied through the first experiment performed, which implemented such an Anomaly Detector using a Random Forest Classifier. The results of the implemented Anomaly Detector were then compared to other Anomaly Detectors from the literature, where results indicated that improvements were required for the implementation to be on par with those from the literature. Further to this, the same first experiment also attempted to implement a number of enhancements to improve the performance of the Anomaly Detector. The first of these was the application of an oversampling technique so as to attempt to address the problem of Class Imbalance known to exist within Anomaly Detection problems (see Section 3.3.3). In this case, the SMOTE oversampling technique was selected and the results from the implementation of this technique showed mixed results, but also showed some small improvements in key metrics, and was therefore included going forward. The second enhancement was to attempt to modify the probability threshold such that bets that would be classified as anomalous required the Anomaly Detector to return a greater likelihood of those bets being anomalous for them to be classified as such. Based on the results generated, the threshold producing the best results was one of 0.5,

which was the same threshold used by the base implementation of the Anomaly Detector. Hence, this part of the experiment failed to provide superior results. The third and final enhancement applied during this experiment was the implementation of a Concept Drift Detector so as to retrain the algorithm dynamically when required, as opposed to periodically. More details about the use of a Concept Drift Detector can be found later in this Section. The results of the implementation using such a detector however, provided the best results for the Anomaly Detector.

The second objective was then focused on extending the Anomaly Detector that was developed into a multi-class classifier capable of separating bets according to the type of customer who placed the bet. By extension, this would also separate anomalous bets into their different types and hence was referred to as an Anomaly Classifier or a Customer Classifier. This objective was satisfied through the second experiment performed, where three different classification algorithms were identified from the literature to be capable of performing such a task. Each of these was then implemented over the entire dataset and the results were compared to one another and to the results of similar classifiers found in the literature. The three selected classifiers were a Logistic Regression Classifier, a Support Vector Machine and a Random Forest Classifier. Based on the results provided by each, it was observed that, out of the three classifiers, the Random Forest Classifier visibly exceeded the performance of the other two algorithms. It was therefore selected to be enhanced further so as to maximise the performance of the Anomaly Classifier as a whole. Since the best results for the Anomaly Detector were observed when applying both SMOTE and a Concept Drift Detector, the same two enhancements were implemented to the Random Forest Classifier in this experiment and these also produced very positive results.

The third and final objective was to enhance both the Anomaly Detector from the first objective and the Anomaly Classifier from the second objective in such a way as to mitigate the effects of another key problem encountered when dealing with Anomaly Detection problems, namely that of Concept Drift. This objective was satisfied during both experiments as each experiment separately incorporated a Concept Drift Detector to enable dynamic retraining when required, as opposed to the periodic retraining that took place during both experiments before such a detector was implemented. After implementing the Concept Drift Detectors, both the Anomaly Detector and the best performing Anomaly Classifier were retrained when their respective Concept Drift Detectors indicated that there had been a shift in the underlying data distribution. Based on findings in the literature, the Adaptive Windowing technique was selected to implement the Concept Drift Detectors in both experiments. Once results were generated, it was very clear that when the Anomaly Detector and Anomaly Classifier were retrained as required

when indicated by the Concept Drift Detectors, the two algorithms returned significantly better results, with the results of the Anomaly Detector, which previously fell well short, now being closer to those observed in the literature, while the results of the Anomaly Detector also outperformed the papers that were used as a benchmark in four out of the five metrics used for evaluation. Owing to this superior performance, it could therefore be said that following the results of the algorithms including a Concept Drift Detector, a final implementation was developed which successfully addressed the initial problem being tackled by this dissertation as well as satisfied all the objectives declared in Section 1.2.

6.1 Critique and Limitations

One of the most prominent limitations that presented itself was the issue of computational resources and time. Considering the massive volume of data that was present within the dataset, the algorithms developed required a considerable amount of time to execute. While the volume of data means that a certain amount of runtime will always be necessary, further techniques to reduce the dimensionality of the dataset, such as Principle Component Analysis (PCA) could be incorporated so as to reduce the runtime required by further decreasing the number of dimensions within each bet.

Another limitation was also identified for similar reasons, concerning the parameters of the machine learning algorithms used as part of the process. While the main parameter used and experimented with for the Random Forest was the number of estimators, several other parameters can be observed within the official documentation [78]. Owing to the fact that, due to the massive volume of data, the runtime was already very high, the remaining parameters that could be shifted away from their default values were not modified. These values had the potential to improve the result of the classifiers, however, it would have come at the cost of requiring additional computation time. When considering that some algorithms required up to a week in order to generate a full set of results, the decision was taken to keep the runtime down, preventing the algorithms' time complexities from becoming infeasible to implement.

Similarly, the fact that the application of SMOTE did not noticeably improve the results of the Anomaly Detector could also have been an issue with the parameters. However, despite the fact that the parameters attempted as part of this dissertation did not work effectively, there could be other potential configurations that would best suit the given dataset. Owing to the volume of data and the proportion of anomalies, the fact that performance was not improved may also be attributed to the parameters, which may not

have been optimal for the given data set.

6.2 Future Work

Throughout the process of writing this dissertation, a number of ideas for future work came to mind. The first such potential idea would be to address one of the limitations stated in Section 6.1. More specifically, this potential improvement would be to experiment with one or more potential preprocessing techniques to reduce the size of the data by incorporating further dimensionality reduction techniques, for example, Principle Component Analysis, which would reduce the total computation time required for a full set of results to be generated. Furthermore, another possible future work addresses the other critique mentioned in Section 6.1. To further improve the performance, more parameters can be experimented with and the results further analysed. However, this must go hand in hand with the aforementioned dimensionality reduction techniques, as additional increases in the time required could potentially render the algorithm infeasible.

In addition, another aspect that could be further explored through future work concerns the parameters that are used during the application of SMOTE to the data set before the Anomaly Detector undergoes training. Such an investigation could focus on the fine-tuning of these hyperparameters to better align with the volume and imbalance presented by a similar data set. The SMOTE oversampling technique could also be applied to a number of different Anomaly Detectors, so as to determine which algorithm produces the most pronounced improvements when SMOTE is applied to it.

Another potential expansion of the work done is to develop another classifier whose performance could be compared to the Anomaly Detectors and Customer Classifiers implemented as part of this dissertation. This classifier, however, would be one that makes use of Neural Networks and Deep Learning, to compare how Deep Learning techniques compare to the traditional machine learning techniques that were selected for this dissertation following a review of the literature.

Furthermore, a variety of enhancement techniques could also be attempted to determine how these would perform compared to the ones that were implemented. One such example of this would be to implement other oversampling techniques, such as ADASYN or CGAN, as well as a couple of undersampling techniques such that only those that perform the best can be selected to enhance the developed algorithms. It would also include some contrasting approaches. As seen in Section 5.2.2.1, mixed results were obtained using SMOTE. However, a different technique or approach could have a greater, more positive effect on the results of the Anomaly Detectors and Customer Classifiers.

A further concept that could merit investigation would be to modify what the results of the developed solution represent. In the case of this dissertation, the results of the system consisted of several evaluation metrics used to provide a measure of the overall performance of the algorithm. The proposed concept for future work would be to supplement the evaluation metrics by also presenting a list of bets. This list would consist of bets that were judged to be anomalous by the Anomaly Detector or that were classified as one of the anomalous customer types by the Customer Classifier.

Another potential extension of this work that could be considered for future research would be to specialise such Anomaly Detectors and Anomaly classifiers. Under this use case, one would be searching for a specific type of anomalous transaction, such as specifically for problem gamblers or for cases of money laundering or match-fixing.

Based on the author's experience, another form of undesirable behavior that could be present within sports betting data is the presence of bots, which repeatedly place the same bet with minimal stake to try to flow under the radar. Such bets that would be placed by these bots would have a very high degree of similarity, which could enable them to be caught. A further enhancement to the Anomaly Detector to catch further suspicious bets other than through anomalies present would be to incorporate one or more out of several existing similarity metrics in such a way that if several consecutive bets by the same customer were found to be nearly identical to one another, this customer could be flagged for further attention.

Finally, another concept would be the configuration of such detectors in such a way that, by default, new customers would have their first few bets analysed by the Customer Classifier and the system would recommend that human employees of the betting operator take a closer look at suspicious customers as they are starting out, pre-empting the behaviour of suspicious customers and allowing the operator to take action before these customers pose a threat at making illicit gains. The system could also be adapted to perform its operations in real-time. However, this would also require the ability to perform the retraining process in the background so that, if the system is operating in real-time 24 hours a day, 7 days a week, there would not be a blackout period where the layer of monitoring that such a system would provide is not present.

References

[1] J. Torrance, M. O'Hanrahan, J. Carroll, and P. Newall, "The structural characteristics of online sports betting: a scoping review of current product features and utility patents as indicators of potential future developments," *Addiction Research & Theory*, pp. 1–15, 2023.

[2] E. A. Killick and M. D. Griffiths, "Sports betting advertising: A systematic review of content analysis studies," *International Journal of Mental Health and Addiction*, pp. 1–27, 2022.

[3] F. Zimniuch, *Crooked: A history of cheating in sports*. Taylor Trade Publications, 2009.

[4] A. Duval, "The russian doping scandal at the court of arbitration for sport: lessons for the world anti-doping system," *The International Sports Law Journal*, vol. 16, no. 3, pp. 177–197, 2017.

[5] M. Vidámi, L. Szilágyi, and D. Iclanzan, "Real valued card counting strategies for the game of blackjack," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 63–73.

[6] R. Etuk, T. Xu, B. Abarbanel, M. N. Potenza, and S. W. Kraus, "Sports betting around the world: A systematic review," *Journal of Behavioral Addictions*, vol. 11, no. 3, pp. 689–715, 2022.

[7] B. J. Riley, S. Harris, T. Nye, Z. Javidi-Hosseinabad, and M. Baigent, "Graded exposure therapy for online mobile smartphone sports betting addiction: a case series report," *Journal of Gambling Studies*, vol. 37, no. 4, pp. 1263–1275, 2021.

[8] W. Andreff, "Different types of manipulation in sport," in *The Palgrave handbook on the economics of manipulation in sport*. Springer, 2018, pp. 13–35.

[9] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long, "Deep learning for anomaly detection," in *Proceedings of the 13th international conference on web search and data mining*, 2020, pp. 894–896.

[10] C. Kim, J.-H. Park, D. Kim, and J.-Y. Lee, "Detectability of sports betting anomalies using deep learning-based resnet: Utilization of k-league data in south korea," *Annals of Applied Sport Science*.

[11] C. Kim, J.-H. Park, and J.-Y. Lee, "Ai-based betting anomaly detection system to ensure fairness in sports and prevent illegal gambling," *Scientific Reports*, vol. 14, no. 1, p. 6470, 2024.

[12] O. Chertov and I. Zhuk, "Detection of match-fixing in football matches using a conformal anomaly detector." in *IT&I*, 2023, pp. 205–219.

[13] F. Peres, E. Fallacara, L. Manzoni, M. Castelli, A. Popovič, M. Rodrigues, and P. Estevens, "Time series clustering of online gambling activities for addicted users' detection," *Applied Sciences*, vol. 11, no. 5, p. 2397, 2021.

[14] P. Selvaraju, "Predicting future problem gamblers using machine learning algorithms," 2022.

[15] M. de Sousa Tedim, "Predicting fraud behaviour in online betting," Master's thesis, Universidade NOVA de Lisboa (Portugal), 2018.

[16] D. Forrest and I. G. McHale, "An evaluation of sportradar's fraud detection system," *University of Liverpool & University of Salford, Manchester*, 2015.

[17] O. Hubáček, G. Šourek, and F. Železnỳ, "Exploiting sports-betting market using machine learning," *International Journal of Forecasting*, vol. 35, no. 2, pp. 783–796, 2019.

[18] J. Stübinger, B. Mangold, and J. Knoll, "Machine learning in football betting: Prediction of match results based on player characteristics," *Applied Sciences*, vol. 10, no. 1, p. 46, 2019.

[19] M. Min, J. J. Lee, H. Park, and K. Lee, "Detecting anomalous transactions via an iot based application: A machine learning approach for horse racing betting," *Sensors*, vol. 21, no. 6, p. 2039, 2021.

[20] A. Yüce, S. Gökce Yüce, H. Katırcı, V. Aydoğdu, W. Chiu, and M. D. Griffiths, "The effect of the covid-19 pandemic on sports betting tipsters as professional bettors: A qualitative interview study," *Sustainability*, vol. 15, no. 9, p. 7729, 2023.

[21] D. Y. Liu, W. C. Tsai, P. L. Liu, and C. Y. Fang, "Determinants of sales revenue in innovation diffusion effects of taiwan sports lottery during the fifa world cup 2018," *International Journal of Research in Business and Social Science (2147-4478)*, vol. 10, no. 4, pp. 43–58, 2021.

[22] S. Agrahari and A. K. Singh, "Concept drift detection in data stream mining: A literature review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9523–9540, 2022.

[23] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[24] T. Barbariol, F. D. Chiara, D. Marcato, and G. A. Susto, "A review of tree-based approaches for anomaly detection," *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*, pp. 149–185, 2022.

[25] S. Fatemifar, M. Awais, A. Akbari, and J. Kittler, "Developing a generic framework for anomaly detection," *Pattern Recognition*, vol. 124, p. 108500, 2022.

[26] R. Ahsan, W. Shi, X. Ma, and W. Lee Croft, "A comparative analysis of cgan-based oversampling for anomaly detection," *IET Cyber-Physical Systems: Theory & Applications*, vol. 7, no. 1, pp. 40–50, 2022.

[27] S. Eltanbouly, M. Bashendy, N. AlNaimi, Z. Chkirbene, and A. Erbad, "Machine learning techniques for network anomaly detection: A survey," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. IEEE, 2020, pp. 156–162.

[28] D. Borycki, "A strategy in sports betting with the nearest neighbours search and genetic algorithms," *Annales Universitatis Mariae Curie-Skłodowska. Sectio AI, Informatica*, vol. 11, no. 1, 2011.

[29] S. Geng and T. Hu, "Sports games modeling and prediction using genetic programming," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–6.

[30] M. Kargün, Y. S. Ağaoğlu, and M. Kaya, "The economics of sports betting," *European Journal of Physical Education and Sport Science*, 2020.

[31] V. Matheson, "An overview of the economics of sports gambling and an introduction to the symposium," *Eastern Economic Journal*, vol. 47, pp. 1–8, 2021.

[32] K. Sproston, C. Hanley, K. Brook, N. Hing, and S. Gainsbury, "Marketing of sports betting and racing," 2015.

[33] T. Kasiga and T. Bro, "Padel an increasing cause of sport-related eye injuries in sweden," *Acta Ophthalmologica*, vol. 102, no. 1, pp. 74–79, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/aos.15685

[34] D. Goller, "Analysing a built-in advantage in asymmetric darts contests using causal machine learning," *Annals of Operations Research*, vol. 325, no. 1, pp. 649–679, 2023.

[35] D. Turchetto, "Sports betting: a new asset class to bet on." Universita' degli studi di Padova, 2022.

[36] E.-L. Sinclair, L. Clark, M. Wohl, M. Keough, and H. Kim, "Cash outs during in-play sports betting: Who, why, and what it reveals," *Addictive Behaviors*, vol. 154, p. 108008, 2024.

[37] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–37, 2020.

[38] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[39] M. Zohair, R. Chandra, S. Tiwari, and S. Agarwal, "A model fusion approach for severity prediction of diabetes with respect to binary and multiclass classification," *International Journal of Information Technology*, vol. 16, no. 3, pp. 1955–1965, 2024.

[40] S. Kang, "Using binary classifiers for one-class classification," *Expert Systems with Applications*, vol. 187, p. 115920, 2022.

[41] Z. Dong, C. Xu, J. Xu, B. Zou, J. Zeng, and Y. Y. Tang, "Generalization capacity of multi-class svm based on markovian resampling," *Pattern Recognition*, vol. 142, p. 109720, 2023.

[42] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 77–94, 2022.

[43] A. Pesaranghader, H. L. Viktor, and E. Paquet, "Mcdiarmid drift detection methods for evolving data streams," in *2018 International joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–9.

[44] T. Horvat and J. Job, "The use of machine learning in sport outcome prediction: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 5, p. e1380, 2020.

[45] G. Fialho, A. Manhães, and J. P. Teixeira, "Predicting sports results with artificial intelligence–a proposal framework for soccer games," *Procedia Computer Science*, vol. 164, pp. 131–136, 2019.

[46] S. Fatemifar, M. Awais, S. R. Arashloo, and J. Kittler, "Combining multiple one-class classifiers for anomaly based face spoofing attack detection," in *2019 International Conference on Biometrics (ICB)*. IEEE, 2019, pp. 1–7.

[47] G. Wang, J. Yang, and R. Li, "Imbalanced svm-based anomaly detection algorithm for imbalanced training datasets," *Etri Journal*, vol. 39, no. 5, pp. 621–631, 2017.

[48] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[49] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 2008, pp. 1322–1328.

[50] S. R. Mirkhani, "Self-excluder classifier." Birbeck College, University of London, 2019.

[51] N. Tsykunov, "Early detection of online gambling addiction." Tallinn University of Technology, 2020.

[52] L. Niu, "A review of the application of logistic regression in educational research: Common issues, implications, and suggestions," *Educational Review*, vol. 72, no. 1, pp. 41–67, 2020.

[53] F. Abbasi, M. Naderan, and S. E. Alavi, "Anomaly detection in internet of things using feature selection and classification based on logistic regression and artificial neural network on n-baiot dataset," in *2021 5th International Conference on Internet of Things and Applications (IoT)*. IEEE, 2021, pp. 1–7.

[54] F. George, S. M. R. Chowdhury, and S. Gulati, "Application of logistic regression on heart disease data and a] review of some standardization methods," *International Journal of Psychological Studies*, vol. 11, no. 5, pp. 1–1, 2022.

[55] Q. A. Al-Haija, E. Saleh, and M. Alnabhan, "Detecting port scan attacks using logistic regression," in *2021 4th International symposium on advanced electrical and communication technologies (ISAECT)*. IEEE, 2021, pp. 1–5.

[56] J. Wang, H. Wang, F. Nie, and X. Li, "Feature selection with multi-class logistic regression," *Neurocomputing*, vol. 543, p. 126268, 2023.

[57] K. Purwandari, J. W. Sigalingging, T. W. Cenggoro, and B. Pardamean, "Multi-class weather forecasting from twitter using machine learning aprroaches," *Procedia Computer Science*, vol. 179, pp. 47–54, 2021.

[58] Q. Ma, C. Sun, B. Cui, and X. Jin, "A novel model for anomaly detection in network traffic based on kernel support vector machine," *Computers & Security*, vol. 104, p. 102215, 2021.

[59] K. Yang, S. Kpotufe, and N. Feamster, "An efficient one-class svm for anomaly detection in the internet of things," *arXiv preprint arXiv:2104.11146*, 2021.

[60] A. Barbado, Ó. Corcho, and R. Benjamins, "Rule extraction in unsupervised anomaly detection for model explainability: Application to oneclass svm," *Expert Systems with Applications*, vol. 189, p. 116100, 2022.

[61] F. Zhu, W. Zhang, X. Chen, X. Gao, and N. Ye, "Large margin distribution multi-class supervised novelty detection," *Expert Systems with Applications*, vol. 224, p. 119937, 2023.

[62] Y. Liu and Y. Zheng, "One-against-all multi-class svm classification using reliability measures," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, pp. 849–854 vol. 2.

[63] H. M. Lalabadi, M. Sadeghi, and S. A. Mireei, "Fish freshness categorization from eyes and gills color features using multi-class artificial neural network and support vector machines," *Aquacultural Engineering*, vol. 90, p. 102076, 2020.

[64] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[65] K. Fawagreh, M. M. Gaber, and E. Elyan, "Random forests: from early developments to recent advancements," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 602–609, 2014.

[66] T.-H. Lin and J.-R. Jiang, "Anomaly detection with autoencoder and random forest," in *2020 International Computer Symposium (ICS)*. IEEE, 2020, pp. 96–99.

[67] G.-F. Fan, L.-Z. Zhang, M. Yu, W.-C. Hong, and S.-Q. Dong, "Applications of random forest in multivariable response surface for short-term load forecasting," *International Journal of Electrical Power & Energy Systems*, vol. 139, p. 108073, 2022.

[68] A. Aftab, I. Shahzad, M. Anwar, A. Sajid, and N. Anwar, "Fraud detection of credit cards using supervised machine learning," *Pakistan Journal of Emerging Science and Technologies (PJEST*, vol. 4, no. 3, 2023.

[69] D. Shah and L. K. Sharma, "Credit card fraud detection using decision tree and random forest," in *ITM Web of Conferences*, vol. 53. EDP Sciences, 2023, p. 02012.

[70] M. Kopp, T. Pevnỳ, and M. Holeňa, "Anomaly explanation with random forests," *Expert Systems with Applications*, vol. 149, p. 113187, 2020.

[71] S. Ahmad, K. Styp-Rekowski, S. Nedelkoski, and O. Kao, "Autoencoder-based condition monitoring and anomaly detection method for rotating machines," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 4093–4102.

[72] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *2020 international conference on cyber security and protection of digital services (cyber security)*. IEEE, 2020, pp. 1–8.

[73] C. Alexandre and J. Balsa, "Client profiling for an anti-money laundering system," *arXiv preprint arXiv:1510.00878*, 2015.

[74] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[75] C. Deutscher, E. Dimant, and B. R. Humphreys, "Match fixing and sports betting in football: Empirical evidence from the german bundesliga," *Available at SSRN 2910662*, 2017.

[76] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Improving credit card fraud detection with calibrated probabilities," in *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 2014, pp. 677–685.

[77] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[78] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[79] [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[80] T. F. of Online Machine Learning, "Adwin." [Online]. Available: https://riverml.xyz/dev/api/drift/ADWIN/

[81] U. Yokkampon, A. Mowshowitz, S. Chumkamon, and E. Hayashi, "Robust unsupervised anomaly detection with variational autoencoder in multivariate time series data," *IEEE Access*, vol. 10, pp. 57 835–57 849, 2022.

[82] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

Appendix A

Number of Anomalous and Non-Anomalous Bets split by Week

Number of Bets				
Week	**Total**	**Non-Anomalous**	**Anomalous**	**Anomalous Percent**
1	6,137,833	6,108,237	29,596	0.48%
2	6,304,500	6,273,771	30,729	0.49%
3	7,785,476	7,752,984	32,492	0.42%
4	8,799,695	8,759,906	39,789	0.45%
5	8,352,174	8,313,481	38,693	0.46%
6	8,659,704	8,619,467	40,237	0.46%
7	7,976,132	7,940,063	36,069	0.45%
8	2,809,495	2,796,781	12,714	0.45%
9	8,954,706	8,916,033	38,673	0.43%
10	8,546,130	8,504,466	41,664	0.49%
11	9,773,987	9,727,172	46,815	0.48%
12	10,852,697	10,791,606	61,091	0.56%
13	8,847,303	8,805,262	42,041	0.48%
14	7,531,981	7,496,574	35,407	0.47%
15	7,919,287	7,880,950	38,337	0.48%
16	12,168,117	12,107,532	60,585	0.50%

Table A.1: Full weekly split of anomalous and non-anomalous bets - Part 1

Number of Bets				
Week	**Total**	**Non-Anomalous**	**Anomalous**	**Anomalous Percent**
17	8,860,162	8,810,886	49,276	0.56%
18	6,038,307	6,001,627	36,680	0.61%
19	6,425,563	6,366,448	59,115	0.92%
20	7,319,393	7,245,056	74,337	1.02%
21	6,095,167	6,052,334	42,833	0.70%
22	5,475,422	5,409,322	66,100	1.21%
23	7,222,177	7,170,822	51,355	0.71%
24	7,923,634	7,876,404	47,230	0.60%
25	6,335,477	6,298,021	37,456	0.59%
26	6,329,880	6,293,496	36,384	0.57%
27	8,630,994	8,584,971	46,023	0.53%
28	5,534,674	5,507,291	27,383	0.49%
29	5,654,270	5,624,272	29,998	0.53%
30	7,701,674	7,655,725	45,949	0.60%
31	8,473,451	8,424,795	48,656	0.57%
32	8,638,127	8,586,071	52,056	0.60%
33	8,102,759	8,059,898	42,861	0.53%
34	7,373,268	7,333,984	39,284	0.53%
35	9,863,549	9,814,552	48,997	0.50%
36	9,304,714	9,262,210	42,504	0.46%
37	8,478,258	8,439,943	38,315	0.45%
38	7,394,365	7,356,616	37,749	0.51%
39	7,923,776	7,885,001	38,775	0.49%
40	8,483,162	8,438,407	44,755	0.53%
41	6,165,270	6,136,405	28,865	0.47%
42	4,897,461	4,869,502	27,959	0.57%
43	6,118,119	6,086,445	31,674	0.52%
44	5,638,200	5,613,823	24,377	0.43%
45	3,865,416	3,848,999	16,417	0.42%
46	4,553,578	4,531,686	21,892	0.48%
47	5,514,944	5,492,583	22,361	0.41%
48	6,368,687	6,339,322	29,365	0.46%

Table A.2: Full weekly split of anomalous and non-anomalous bets - Part 2

Appendix B

Full Weekly Results of Anomaly Detectors

B.1 Random Forest Anomaly Detector

Random Forest - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.8859	0.3808	0.5326	0.9998	0.6170
2	0.1348	0.0693	0.0916	0.9978	0.2631
3	0.1027	0.0459	0.0635	0.9983	0.2142
4	0.1302	0.0457	0.0676	0.9986	0.2135
5	0.9369	0.5526	0.6952	0.9998	0.7433
6	0.1226	0.1072	0.1144	0.9964	0.3268
7	0.1275	0.1065	0.1161	0.9967	0.3258
8	0.1354	0.0425	0.0647	0.9988	0.2060
9	0.9302	0.5316	0.6766	0.9998	0.7290
10	0.1004	0.1648	0.1248	0.9928	0.4045
11	0.0864	0.1286	0.1033	0.9935	0.3574
12	0.1096	0.1205	0.1148	0.9945	0.3462
13	0.9484	0.5777	0.7180	0.9999	0.7600
14	0.0599	0.2074	0.0930	0.9846	0.4519
15	0.0508	0.1440	0.0751	0.9869	0.3770
16	0.2821	0.1067	0.1548	0.9986	0.3264

Table B.1: Full Weekly Results of initial Random Forest Anomaly Detector - Part 1

Random Forest - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
17	0.9246	0.5033	0.6518	0.9998	0.7094
18	0.0717	0.0623	0.0667	0.9951	0.2490
19	0.0450	0.1038	0.0628	0.9795	0.3189
20	0.2110	0.1441	0.1713	0.9965	0.3790
21	0.9278	0.5292	0.6740	0.9997	0.7274
22	0.0465	0.1272	0.0681	0.9791	0.3529
23	0.1453	0.0894	0.1107	0.9962	0.2984
24	0.1053	0.0699	0.0841	0.9964	0.2640
25	0.9358	0.5356	0.6812	0.9998	0.7317
26	0.0763	0.0709	0.0735	0.9950	0.2656
27	0.2732	0.0474	0.0807	0.9993	0.2176
28	0.0946	0.0910	0.0927	0.9957	0.3010
29	0.9547	0.5491	0.6972	0.9999	0.7410
30	0.1723	0.0328	0.0552	0.9991	0.1811
31	0.0495	0.0302	0.0375	0.9966	0.1736
32	0.0835	0.0235	0.0366	0.9984	0.1531
33	0.9434	0.5907	0.7265	0.9998	0.7685
34	0.1850	0.0913	0.1223	0.9979	0.3019
35	0.0869	0.1499	0.1101	0.9923	0.3857
36	0.1431	0.1426	0.1429	0.9955	0.3768
37	0.9399	0.5487	0.6928	0.9998	0.7407
38	0.1850	0.0913	0.1223	0.9979	0.3019
39	0.0869	0.1499	0.1101	0.9923	0.3857
40	0.1431	0.1426	0.1429	0.9955	0.3768
41	0.9082	0.4927	0.6388	0.9998	0.7018
42	0.0424	0.0612	0.0501	0.9921	0.2464
43	0.0491	0.0669	0.0566	0.9933	0.2577
44	0.0267	0.0608	0.0371	0.9904	0.2455
45	0.9190	0.4481	0.6025	0.9998	0.6694
46	0.1269	0.1542	0.1393	0.9949	0.3917
47	0.0155	0.1288	0.0276	0.9666	0.3529
48	0.1316	0.0976	0.1121	0.9970	0.3120
Average	**0.3165**	**0.2033**	**0.2351**	**0.9952**	**0.4071**

Table B.2: Full Weekly Result of initial Random Forest Anomaly Detector - Part 2

B.2 Random Forest Anomaly Detector including SMOTE

SMOTE - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6666	0.4373	0.5282	0.9989	0.6610
2	0.1476	0.0819	0.1053	0.9977	0.2858
3	0.0339	0.0752	0.0467	0.9910	0.2731
4	0.1674	0.077	0.1055	0.9983	0.2772
5	0.8028	0.5583	0.6586	0.9994	0.7469
6	0.1603	0.1042	0.1263	0.9975	0.3224
7	0.1624	0.1238	0.1405	0.9971	0.3514
8	0.1853	0.0704	0.1020	0.9986	0.2651
9	0.7981	0.5618	0.6594	0.9994	0.7493
10	0.0849	0.2181	0.1222	0.9885	0.4644
11	0.0361	0.1303	0.0565	0.9832	0.3580
12	0.1362	0.1321	0.1341	0.9953	0.3626
13	0.8003	0.5898	0.6791	0.9993	0.7677
14	0.0690	0.1695	0.0981	0.9892	0.4095
15	0.0431	0.1139	0.0626	0.9877	0.3354
16	0.3565	0.1466	0.2078	0.9987	0.3827
17	0.7172	0.5516	0.6236	0.9988	0.7422
18	0.0188	0.1104	0.0321	0.9647	0.3263
19	0.0947	0.1192	0.1056	0.9894	0.3435
20	0.2883	0.1652	0.2101	0.9974	0.4060
21	0.7640	0.5613	0.6471	0.9988	0.7487
22	0.0342	0.2329	0.0596	0.9474	0.4697
23	0.0470	0.1689	0.0736	0.9755	0.4060
24	0.0538	0.1531	0.0797	0.9839	0.3881

Table B.3: Full Weekly Result of Random Forest Anomaly Detector including SMOTE - Part 1

SMOTE - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.7984	0.5736	0.6676	0.9991	0.7570
26	0.0854	0.0726	0.0785	0.9955	0.2688
27	0.0927	0.0465	0.0620	0.9976	0.2155
28	0.0813	0.1170	0.0959	0.9934	0.3409
29	0.8278	0.5892	0.6884	0.9994	0.7674
30	0.0981	0.0459	0.0626	0.9975	0.2140
31	0.0341	0.0540	0.0418	0.9912	0.2312
32	0.0936	0.0499	0.0651	0.9971	0.2230
33	0.8304	0.5991	0.6960	0.9994	0.7737
34	0.0733	0.0908	0.0812	0.9939	0.3004
35	0.0536	0.0404	0.0461	0.9964	0.2007
36	0.1714	0.0940	0.1214	0.9979	0.3063
37	0.8072	0.5630	0.6634	0.9994	0.7501
38	0.1732	0.1073	0.1325	0.9974	0.3271
39	0.0801	0.1351	0.1005	0.9924	0.3661
40	0.0926	0.1274	0.1072	0.9934	0.3557
41	0.7631	0.5270	0.6235	0.9992	0.7257
42	0.0551	0.0653	0.0598	0.9936	0.2547
43	0.0703	0.0589	0.0641	0.9959	0.2423
44	0.0269	0.0527	0.0356	0.9917	0.2286
45	0.7575	0.4823	0.5894	0.9993	0.6943
46	0.0824	0.1386	0.1033	0.9925	0.3709
47	0.0373	0.1189	0.0568	0.9875	0.3427
48	0.0959	0.1172	0.1055	0.9949	0.3414
Average	**0.2698**	**0.2192**	**0.2294**	**0.9931**	**0.4259**

Table B.4: Full Weekly Results of Random Forest Anomaly Detector including SMOTE - Part 2

B.3 Anomaly Detectors using Random Forest and different Probability Thresholds

Probability Threshold - 0.5 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.8806	0.3842	0.5349	0.9997	0.6197
2	0.1538	0.0692	0.0954	0.9981	0.2627
3	0.0789	0.0526	0.0631	0.9974	0.2290
4	0.1798	0.0621	0.0923	0.9987	0.2490
5	0.9359	0.5555	0.6972	0.9998	0.7453
6	0.1061	0.1269	0.1155	0.9950	0.3553
7	0.1375	0.1240	0.1304	0.9965	0.3516
8	0.0994	0.0651	0.0787	0.9973	0.2549
9	0.9248	0.5458	0.6864	0.9998	0.7387
10	0.0852	0.1976	0.1190	0.9896	0.4422
11	0.0840	0.1394	0.1049	0.9927	0.3720
12	0.1007	0.1115	0.1058	0.9944	0.3330
13	0.9504	0.5911	0.7289	0.9999	0.7688
14	0.0753	0.1647	0.1034	0.9905	0.4039
15	0.0546	0.1078	0.0725	0.9909	0.3269
16	0.2180	0.0942	0.1315	0.9983	0.3066
17	0.9186	0.5112	0.6569	0.9997	0.7149
18	0.1096	0.0599	0.0775	0.9970	0.2444
19	0.0346	0.0893	0.0499	0.9769	0.2954
20	0.1338	0.1263	0.1299	0.9947	0.3544
21	0.9211	0.5274	0.6707	0.9997	0.7261
22	0.0465	0.1637	0.0724	0.9731	0.3992
23	0.1808	0.1112	0.1377	0.9964	0.3328
24	0.1252	0.0911	0.1055	0.9962	0.3013

Table B.5: FUll weekly results for Random Forest anomaly detector with threshold 0.5 - Part 1

Probability Threshold - 0.5 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9304	0.5418	0.6848	0.9998	0.7360
26	0.0825	0.0854	0.0839	0.9945	0.2915
27	0.2713	0.0519	0.0871	0.9993	0.2277
28	0.0917	0.0788	0.0848	0.9961	0.2802
29	0.9458	0.5725	0.7133	0.9998	0.7566
30	0.1529	0.0316	0.0524	0.9989	0.1778
31	0.0387	0.0322	0.0352	0.9954	0.1790
32	0.0962	0.0312	0.0472	0.9982	0.1766
33	0.9469	0.5954	0.7311	0.9998	0.7716
34	0.0653	0.1209	0.0848	0.9907	0.3461
35	0.1372	0.0588	0.0823	0.9982	0.2422
36	0.1076	0.0815	0.0927	0.9969	0.2850
37	0.9316	0.5474	0.6896	0.9998	0.7398
38	0.1852	0.0877	0.1190	0.9980	0.2958
39	0.0918	0.1285	0.1071	0.9938	0.3573
40	0.1532	0.1283	0.1396	0.9962	0.3575
41	0.9042	0.5076	0.6502	0.9997	0.7124
42	0.0422	0.0768	0.0545	0.9900	0.2757
43	0.0614	0.0851	0.0713	0.9932	0.2907
44	0.0288	0.0711	0.0410	0.9896	0.2653
45	0.9158	0.4504	0.6039	0.9998	0.6711
46	0.0950	0.1338	0.1111	0.9938	0.3647
47	0.0140	0.1109	0.0249	0.9683	0.3276
48	0.0964	0.0831	0.0893	0.9964	0.2878
Average	**0.3109**	**0.2034**	**0.2342**	**0.9950**	**0.4072**

Table B.6: Full weekly results for Random Forest anomaly detector with threshold 0.5 - Part 2

Probability Threshold - 0.55 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9036	0.3527	0.5073	0.9998	0.5938
2	0.1740	0.0444	0.0707	0.9990	0.2105
3	0.0783	0.0311	0.0445	0.9985	0.1762
4	0.1962	0.0411	0.068	0.9992	0.2026
5	0.9487	0.5208	0.6725	0.9999	0.7216
6	0.1324	0.0704	0.0919	0.9978	0.2650
7	0.1736	0.0811	0.1105	0.9982	0.2845
8	0.1025	0.0193	0.0326	0.9992	0.1390
9	0.9376	0.5043	0.6558	0.9999	0.7101
10	0.1042	0.1417	0.1201	0.9940	0.3753
11	0.0950	0.0789	0.0862	0.9964	0.2804
12	0.1210	0.0655	0.0850	0.9973	0.2557
13	0.9608	0.5516	0.7008	0.9999	0.7426
14	0.0853	0.1164	0.0985	0.9941	0.3402
15	0.0553	0.0617	0.0583	0.9949	0.2478
16	0.2481	0.0520	0.0860	0.9992	0.228
17	0.9334	0.4783	0.6325	0.9998	0.6915
18	0.1490	0.0272	0.0459	0.9991	0.1647
19	0.0526	0.0391	0.0449	0.9935	0.1971
20	0.2069	0.0774	0.1127	0.9981	0.2780
21	0.9366	0.4912	0.6445	0.9998	0.7008
22	0.0493	0.1073	0.0675	0.9835	0.3248
23	0.2199	0.0692	0.1053	0.9982	0.2629
24	0.1596	0.0568	0.0838	0.9982	0.2381

Table B.7: Full weekly results for Random Forest anomaly detector with threshold 0.55 - Part 1

Probability Threshold - 0.55 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9469	0.5092	0.6623	0.9998	0.7136
26	0.0885	0.0529	0.0662	0.9969	0.2296
27	0.2865	0.0272	0.0497	0.9996	0.1650
28	0.0942	0.0420	0.0581	0.9980	0.2048
29	0.9577	0.5353	0.6867	0.9999	0.7316
30	0.1422	0.0140	0.0255	0.9995	0.1183
31	0.0415	0.0136	0.0205	0.9982	0.1164
32	0.0923	0.0142	0.0246	0.9992	0.1192
33	0.9621	0.5586	0.7068	0.9999	0.7474
34	0.0746	0.0719	0.0732	0.9952	0.2676
35	0.1742	0.0299	0.0510	0.9993	0.1729
36	0.1464	0.0423	0.0657	0.9989	0.2056
37	0.9444	0.5110	0.6632	0.9999	0.7148
38	0.2153	0.0572	0.0904	0.9989	0.2391
39	0.1059	0.0812	0.0919	0.9966	0.2845
40	0.2013	0.0872	0.1217	0.9982	0.2951
41	0.9208	0.4673	0.6199	0.9998	0.6835
42	0.0611	0.0387	0.0473	0.9966	0.1963
43	0.0580	0.0415	0.0484	0.9965	0.2035
44	0.0386	0.0349	0.0367	0.9962	0.1865
45	0.9352	0.4106	0.5707	0.9999	0.6408
46	0.1141	0.0789	0.0933	0.9970	0.2805
47	0.0183	0.0679	0.0289	0.9852	0.2586
48	0.1283	0.0439	0.0654	0.9986	0.2094
Average	**0.3286**	**0.1648**	**0.2124**	**0.9976**	**0.3462**

Table B.8: Full weekly results for Random Forest anomaly detector with threshold 0.55 - Part 2

Probability Threshold - 0.6 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9257	0.3251	0.4812	0.9999	0.5702
2	0.2008	0.0285	0.0499	0.9994	0.1687
3	0.0762	0.0197	0.0313	0.9990	0.1404
4	0.2134	0.0279	0.0494	0.9995	0.1671
5	0.9615	0.4784	0.6389	0.9999	0.6916
6	0.1599	0.0312	0.0522	0.9992	0.1765
7	0.2043	0.0479	0.0776	0.9992	0.2188
8	0.1172	0.0052	0.0099	0.9998	0.0720
9	0.9472	0.4659	0.6246	0.9999	0.6825
10	0.1157	0.0878	0.0998	0.9967	0.2958
11	0.1029	0.0369	0.0543	0.9985	0.1920
12	0.1357	0.0312	0.0508	0.9989	0.1767
13	0.9697	0.5165	0.6740	0.9999	0.7187
14	0.0833	0.0772	0.0801	0.9960	0.2772
15	0.0536	0.0304	0.0388	0.9974	0.1741
16	0.2467	0.0228	0.0418	0.9997	0.1511
17	0.9453	0.4449	0.6051	0.9999	0.6670
18	0.2066	0.0106	0.0201	0.9998	0.1028
19	0.0757	0.0141	0.0238	0.9984	0.1188
20	0.2684	0.0412	0.0714	0.9993	0.2028
21	0.9516	0.4555	0.6161	0.9998	0.6749
22	0.0531	0.0665	0.0590	0.9905	0.2566
23	0.2733	0.0389	0.0681	0.9993	0.1971
24	0.2038	0.0323	0.0557	0.9992	0.1796

Table B.9: Full weekly results for Random Forest anomaly detector with threshold 0.6 - Part 1

Probability Threshold - 0.6 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9595	0.4717	0.6325	0.9999	0.6868
26	0.0844	0.0271	0.0410	0.9983	0.1645
27	0.2893	0.0114	0.0220	0.9998	0.1069
28	0.1001	0.0188	0.0317	0.9992	0.1372
29	0.9671	0.5021	0.6610	0.9999	0.7086
30	0.1305	0.0054	0.0103	0.9998	0.0733
31	0.0489	0.0054	0.0097	0.9994	0.0734
32	0.0892	0.0060	0.0112	0.9996	0.0774
33	0.9714	0.5186	0.6762	0.9999	0.7201
34	0.0773	0.0370	0.0501	0.9976	0.1922
35	0.2191	0.0137	0.0257	0.9998	0.1168
36	0.1822	0.0173	0.0317	0.9996	0.1317
37	0.9550	0.4727	0.6324	0.9999	0.6875
38	0.2432	0.0338	0.0594	0.9995	0.1838
39	0.1249	0.0449	0.0661	0.9985	0.2118
40	0.2618	0.0513	0.0857	0.9992	0.2263
41	0.9399	0.4250	0.5853	0.9999	0.6519
42	0.0775	0.0168	0.0276	0.9989	0.1294
43	0.0593	0.0189	0.0286	0.9984	0.1373
44	0.0551	0.0155	0.0242	0.9988	0.1245
45	0.9486	0.3637	0.5258	0.9999	0.6031
46	0.1274	0.0363	0.0565	0.9988	0.1904
47	0.0218	0.0301	0.0253	0.9945	0.1731
48	0.1341	0.0154	0.0277	0.9995	0.1242
Average	**0.3450**	**0.1353**	**0.1859**	**0.9989**	**0.2897**

Table B.10: Full weekly results for Random Forest anomaly detector with threshold 0.6 - Part 2

Probability Threshold - 0.65 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9398	0.2922	0.4458	0.9999	0.5405
2	0.2315	0.0183	0.0338	0.9997	0.1351
3	0.0690	0.0114	0.0196	0.9994	0.1068
4	0.2443	0.0204	0.0376	0.9997	0.1427
5	0.9703	0.4374	0.6029	0.9999	0.6613
6	0.1880	0.0111	0.0210	0.9998	0.1054
7	0.2102	0.0217	0.0393	0.9996	0.1472
8	0.1795	0.0017	0.0033	1.0000	0.0406
9	0.9574	0.4237	0.5874	0.9999	0.6509
10	0.1275	0.0455	0.0670	0.9985	0.2130
11	0.1097	0.0135	0.0240	0.9995	0.1161
12	0.1354	0.0104	0.0193	0.9996	0.1020
13	0.9747	0.4789	0.6422	0.9999	0.6920
14	0.0781	0.0460	0.0579	0.9974	0.2141
15	0.0491	0.0126	0.0200	0.9988	0.1121
16	0.2271	0.0074	0.0144	0.9999	0.0861
17	0.9533	0.4086	0.5720	0.9999	0.6392
18	0.2457	0.0031	0.0062	0.9999	0.0560
19	0.1107	0.0039	0.0075	0.9997	0.0625
20	0.2991	0.0144	0.0274	0.9998	0.1198
21	0.9626	0.4191	0.5839	0.9999	0.6473
22	0.0566	0.0351	0.0433	0.9953	0.1869
23	0.3396	0.0169	0.0323	0.9998	0.1301
24	0.2395	0.0153	0.0287	0.9997	0.1235

Table B.11: Full weekly results for Random Forest anomaly detector with threshold 0.65 - Part 1

Probability Threshold - 0.65 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9702	0.4364	0.6021	0.9999	0.6606
26	0.0727	0.0104	0.0182	0.9992	0.1020
27	0.2914	0.0038	0.0075	1.0000	0.0618
28	0.0981	0.0057	0.0108	0.9997	0.0755
29	0.9730	0.4603	0.6250	0.9999	0.6784
30	0.1248	0.0018	0.0036	0.9999	0.0430
31	0.0717	0.0024	0.0046	0.9998	0.0488
32	0.1044	0.0027	0.0052	0.9999	0.0515
33	0.9791	0.4814	0.6455	0.9999	0.6938
34	0.0742	0.0153	0.0253	0.9990	0.1235
35	0.2786	0.0050	0.0099	0.9999	0.0709
36	0.2216	0.0047	0.0092	0.9999	0.0684
37	0.9630	0.4341	0.5984	0.9999	0.6588
38	0.2590	0.0163	0.0306	0.9998	0.1275
39	0.1378	0.0207	0.0359	0.9994	0.1437
40	0.3288	0.0250	0.0465	0.9997	0.1581
41	0.9534	0.3839	0.5474	0.9999	0.6195
42	0.0924	0.0060	0.0113	0.9997	0.0775
43	0.0625	0.0075	0.0135	0.9994	0.0868
44	0.0834	0.0057	0.0107	0.9997	0.0755
45	0.9565	0.3213	0.4810	0.9999	0.5668
46	0.1283	0.0114	0.0210	0.9996	0.1068
47	0.0270	0.0095	0.0141	0.9986	0.0975
48	0.1409	0.0043	0.0083	0.9999	0.0655
Average	**0.3602**	**0.1134**	**0.1609**	**0.9995**	**0.2395**

Table B.12: Full weekly results for Random Forest anomaly detector with threshold 0.65 - Part 2

Probability Threshold - 0.7 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9522	0.2609	0.4096	0.9999	0.5108
2	0.3015	0.0118	0.0227	0.9999	0.1087
3	0.0786	0.0071	0.0131	0.9996	0.0845
4	0.2945	0.0120	0.0231	0.9999	0.1096
5	0.9738	0.3957	0.5628	1.0000	0.6291
6	0.1997	0.0038	0.0074	0.9999	0.0613
7	0.1772	0.0063	0.0123	0.9999	0.0797
8	0.1429	0.0003	0.0006	1.0000	0.0177
9	0.9649	0.3836	0.5489	0.9999	0.6193
10	0.1547	0.0213	0.0374	0.9994	0.1459
11	0.1173	0.0033	0.0065	0.9999	0.0577
12	0.1084	0.0021	0.0042	0.9999	0.0463
13	0.9800	0.4366	0.6041	1.0000	0.6607
14	0.0713	0.0220	0.0336	0.9986	0.1482
15	0.0542	0.0042	0.0077	0.9996	0.0646
16	0.1874	0.0016	0.0031	1.0000	0.0396
17	0.9610	0.3746	0.5391	0.9999	0.6120
18	0.2857	0.0006	0.0012	1.0000	0.0245
19	0.1667	0.0005	0.0009	1.0000	0.0214
20	0.3791	0.0025	0.0049	1.0000	0.0498
21	0.9691	0.3824	0.5484	0.9999	0.6183
22	0.0579	0.0147	0.0234	0.9981	0.1211
23	0.4322	0.0060	0.0118	0.9999	0.0772
24	0.3182	0.0065	0.0128	0.9999	0.0808

Table B.13: Full weekly results for Random Forest anomaly detector with threshold 0.7 - Part 1

Probability Threshold - 0.7 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9775	0.4003	0.5680	0.9999	0.6327
26	0.0555	0.0026	0.0049	0.9997	0.0508
27	0.2692	0.0008	0.0015	1.0000	0.0276
28	0.0746	0.0010	0.0019	0.9999	0.0314
29	0.9800	0.4187	0.5868	1.0000	0.6471
30	0.0741	0.0003	0.0007	1.0000	0.0187
31	0.1326	0.0012	0.0023	1.0000	0.0342
32	0.0899	0.0007	0.0015	1.0000	0.0274
33	0.9856	0.4432	0.6114	1.0000	0.6657
34	0.0769	0.0047	0.0089	0.9997	0.0686
35	0.3492	0.0013	0.0027	1.0000	0.0367
36	0.2857	0.0010	0.0021	1.0000	0.0322
37	0.9743	0.3948	0.5619	1.0000	0.6283
38	0.2864	0.0061	0.0120	0.9999	0.0784
39	0.1614	0.0076	0.0145	0.9998	0.0871
40	0.4675	0.0100	0.0195	0.9999	0.0998
41	0.9619	0.3434	0.5062	0.9999	0.5860
42	0.1019	0.0021	0.0041	0.9999	0.0459
43	0.0755	0.0028	0.0054	0.9998	0.0527
44	0.1115	0.0014	0.0027	1.0000	0.0368
45	0.9666	0.2764	0.4299	1.0000	0.5257
46	0.1439	0.0037	0.0071	0.9999	0.0604
47	0.0338	0.0021	0.0040	0.9998	0.0463
48	0.2260	0.0011	0.0022	1.0000	0.0335
Average	**0.3790**	**0.0977**	**0.1417**	**0.9998**	**0.1988**

Table B.14: Full weekly results for Random Forest anomaly detector with threshold 0.7 - Part 2

Probability Threshold - 0.75 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9607	0.2328	0.3748	1.0000	0.4825
2	0.3320	0.0055	0.0108	0.9999	0.0739
3	0.0846	0.0031	0.0060	0.9999	0.0557
4	0.3054	0.0039	0.0078	1.0000	0.0628
5	0.9782	0.3505	0.5161	1.0000	0.5920
6	0.2500	0.0014	0.0028	1.0000	0.0373
7	0.1250	0.0011	0.0022	1.0000	0.0333
8	0.1429	0.0000	0.0000	1.0000	0.0000
9	0.9716	0.3397	0.5034	1.0000	0.5828
10	0.1484	0.0056	0.0107	0.9998	0.0746
11	0.066	0.0004	0.0008	1.0000	0.0207
12	0.0714	0.0003	0.0005	1.0000	0.0162
13	0.9854	0.3959	0.5648	1.0000	0.6292
14	0.0687	0.0085	0.0152	0.9995	0.0923
15	0.0731	0.0007	0.0014	1.0000	0.0270
16	0.1735	0.0003	0.0006	1.0000	0.0168
17	0.9665	0.3366	0.4994	0.9999	0.5802
18	0.0000	0.0000	0.0000	1.0000	0.0000
19	0.1429	0.0000	0.0000	1.0000	0.0000
20	0.4800	0.0003	0.0005	1.0000	0.016
21	0.9757	0.3419	0.5064	0.9999	0.5847
22	0.0494	0.0034	0.0064	0.9995	0.0585
23	0.5175	0.0011	0.0023	1.0000	0.0339
24	0.4247	0.0013	0.0026	1.0000	0.0362

Table B.15: Full weekly results for Random Forest anomaly detector with threshold 0.75 - Part 1

Probability Threshold - 0.75 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9847	0.3630	0.5305	1.0000	0.6025
26	0.0523	0.0004	0.0008	1.0000	0.0203
27	0.2857	0.0002	0.0003	1.0000	0.0132
28	0.0208	0.0000	0.0000	1.0000	0.0000
29	0.9836	0.3780	0.5462	1.0000	0.6148
30	0.0588	0.0000	0.0000	1.0000	0.0000
31	0.0833	0.0001	0.0002	1.0000	0.0101
32	0.1077	0.0003	0.0005	1.0000	0.0164
33	0.9902	0.4004	0.5703	1.0000	0.6328
34	0.0862	0.0007	0.0014	1.0000	0.0267
35	0.5455	0.0002	0.0005	1.0000	0.0156
36	0.3043	0.0002	0.0003	1.0000	0.0128
37	0.9771	0.3554	0.5212	1.0000	0.5961
38	0.4074	0.0015	0.0029	1.0000	0.0382
39	0.2101	0.0020	0.0040	1.0000	0.0451
40	0.5510	0.0018	0.0036	1.0000	0.0425
41	0.9707	0.2982	0.4562	1.0000	0.5460
42	0.1094	0.0005	0.0010	1.0000	0.0224
43	0.0453	0.0003	0.0007	1.0000	0.0186
44	0.1463	0.0002	0.0005	1.0000	0.0157
45	0.9756	0.2352	0.3790	1.0000	0.4849
46	0.1691	0.0011	0.0021	1.0000	0.0324
47	0.0474	0.0004	0.0008	1.0000	0.0201
48	0.2857	0.0001	0.0003	1.0000	0.0117
Average	**0.3894**	**0.0849**	**0.1262**	**0.9999**	**0.1655**

Table B.16: Full weekly results for Random Forest anomaly detector with threshold 0.75 - Part 2

Probability Threshold - 0.8 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9680	0.2000	0.3315	1.0000	0.4472
2	0.3176	0.0015	0.0030	1.0000	0.0391
3	0.1467	0.0008	0.0017	1.0000	0.0288
4	0.3220	0.0010	0.0019	1.0000	0.0309
5	0.9828	0.3044	0.4648	1.0000	0.5517
6	0.3654	0.0005	0.0009	1.0000	0.0217
7	0.1333	0.0002	0.0004	1.0000	0.0149
8	0.0000	0.0000	0.0000	1.0000	0.0000
9	0.9755	0.2969	0.4552	1.0000	0.5448
10	0.1457	0.0007	0.0014	1.0000	0.0264
11	0.0189	0.0000	0.0000	1.0000	0.0000
12	0.0000	0.0000	0.0000	1.0000	0.0000
13	0.9884	0.3521	0.5193	1.0000	0.5934
14	0.0579	0.0018	0.0035	0.9999	0.0425
15	0.0000	0.0000	0.0000	1.0000	0.0000
16	0.1000	0.0000	0.0000	1.0000	0.0000
17	0.9749	0.3013	0.4603	1.0000	0.5489
18	0.0000	0.0000	0.0000	1.0000	0.0000
19	0.0000	0.0000	0.0000	1.0000	0.0000
20	0.0000	0.0000	0.0000	1.0000	0.0000
21	0.9813	0.3033	0.4633	1.0000	0.5507
22	0.0654	0.0006	0.0012	0.9999	0.025
23	0.4286	0.0000	0.0000	1.0000	0.0000
24	0.4615	0.0001	0.0003	1.0000	0.0113

Table B.17: Full weekly results for Random Forest anomaly detector with threshold 0.8 - Part 1

Probability Threshold - 0.8 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9898	0.3293	0.4941	1.0000	0.5738
26	0.0882	0.0001	0.0002	1.0000	0.0100
27	0.3333	0.0000	0.0000	1.0000	0.0000
28	0.0000	0.0000	0.0000	1.0000	0.0000
29	0.9873	0.3364	0.5018	1.0000	0.5800
30	0.0000	0.0000	0.0000	1.0000	0.0000
31	0.0000	0.0000	0.0000	1.0000	0.0000
32	0.0000	0.0000	0.0000	1.0000	0.0000
33	0.9935	0.3558	0.5240	1.0000	0.5965
34	0.1111	0.0000	0.0000	1.0000	0.0000
35	0.5000	0.0000	0.0000	1.0000	0.0000
36	0.4000	0.0000	0.0000	1.0000	0.0000
37	0.9817	0.3177	0.4801	1.0000	0.5637
38	0.2500	0.0000	0.0000	1.0000	0.0000
39	0.1667	0.0002	0.0004	1.0000	0.0134
40	0.8333	0.0002	0.0004	1.0000	0.0149
41	0.9730	0.2571	0.4068	1.0000	0.5071
42	0.1429	0.0001	0.0002	1.0000	0.0100
43	0.0488	0.0001	0.0002	1.0000	0.0100
44	0.2000	0.0000	0.0000	1.0000	0.0000
45	0.9794	0.1933	0.3229	1.0000	0.4397
46	0.1923	0.0002	0.0005	1.0000	0.0151
47	0.0000	0.0000	0.0000	1.0000	0.0000
48	0.0000	0.0000	0.0000	1.0000	0.0000
Average	**0.3668**	**0.0741**	**0.1133**	**1.0000**	**0.1419**

Table B.18: Full weekly results for Random Forest anomaly detector with threshold 0.8 - Part 2

Probability Threshold - 0.85 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9761	0.1696	0.289	1.0000	0.4118
2	0.6250	0.0005	0.0010	1.0000	0.0221
3	0.0909	0.0000	0.0000	1.0000	0.0000
4	0.3333	0.0000	0.0000	1.0000	0.0000
5	0.9873	0.2602	0.4119	1.0000	0.5101
6	0.3000	0.0000	0.0000	1.0000	0.0000
7	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000
9	0.9807	0.2531	0.4023	1.0000	0.5031
10	0.2000	0.0000	0.0000	1.0000	0.0000
11	0.0000	0.0000	0.0000	1.0000	0.0000
12	0.0000	0.0000	0.0000	1.0000	0.0000
13	0.9911	0.3019	0.4628	1.0000	0.5494
14	0.1714	0.0002	0.0003	1.0000	0.0130
15	0.0000	0.0000	0.0000	1.0000	0.0000
16	0.0000	0.0000	0.0000	1.0000	0.0000
17	0.9771	0.2599	0.4106	1.0000	0.5098
18	0.0000	0.0000	0.0000	1.0000	0.0000
19	0.0000	0.0000	0.0000	1.0000	0.0000
20	0.0000	0.0000	0.0000	1.0000	0.0000
21	0.9882	0.2616	0.4137	1.0000	0.5115
22	0.0000	0.0000	0.0000	1.0000	0.0000
23	0.0000	0.0000	0.0000	1.0000	0.0000
24	0.0000	0.0000	0.0000	1.0000	0.0000

Table B.19: Full weekly results for Random Forest anomaly detector with threshold 0.85 - Part 1

Probability Threshold - 0.85 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9933	0.2898	0.4487	1.0000	0.5383
26	0.0000	0.0000	0.0000	1.0000	0.0000
27	0.0000	0.0000	0.0000	1.0000	0.0000
28	0.0000	0.0000	0.0000	1.0000	0.0000
29	0.9886	0.2894	0.4477	1.0000	0.5379
30	0.0000	0.0000	0.0000	1.0000	0.0000
31	0.0000	0.0000	0.0000	1.0000	0.0000
32	0.0000	0.0000	0.0000	1.0000	0.0000
33	0.9962	0.3069	0.4692	1.0000	0.5540
34	0.0000	0.0000	0.0000	1.0000	0.0000
35	0.0000	0.0000	0.0000	1.0000	0.0000
36	0.0000	0.0000	0.0000	1.0000	0.0000
37	0.9840	0.2777	0.4331	1.0000	0.5269
38	0.0000	0.0000	0.0000	1.0000	0.0000
39	0.5000	0.0000	0.0000	1.0000	0.0000
40	0.0000	0.0000	0.0000	1.0000	0.0000
41	0.9782	0.2173	0.3556	1.0000	0.4661
42	0.0000	0.0000	0.0000	1.0000	0.0000
43	0.0000	0.0000	0.0000	1.0000	0.0000
44	0.0000	0.0000	0.0000	1.0000	0.0000
45	0.9794	0.1545	0.2670	1.0000	0.3931
46	1.0000	0.0000	0.0000	1.0000	0.0000
47	0.0000	0.0000	0.0000	1.0000	0.0000
48	0.0000	0.0000	0.0000	1.0000	0.0000
Average	**0.3134**	**0.0634**	**0.1003**	**1.0000**	**0.1260**

Table B.20: Full weekly results for Random Forest anomaly detector with threshold 0.85 - Part 2

Probability Threshold - 0.9 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9831	0.1306	0.2306	1.0000	0.3614
2	0.0000	0.0000	0.0000	1.0000	0.0000
3	0.0000	0.0000	0.0000	1.0000	0.0000
4	0.0000	0.0000	0.0000	1.0000	0.0000
5	0.9910	0.2086	0.3447	1.0000	0.4568
6	0.0000	0.0000	0.0000	1.0000	0.0000
7	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000
9	0.9847	0.2045	0.3387	1.0000	0.4522
10	0.0000	0.0000	0.0000	1.0000	0.0000
11	0.0000	0.0000	0.0000	1.0000	0.0000
12	0.0000	0.0000	0.0000	1.0000	0.0000
13	0.9949	0.2468	0.3955	1.0000	0.4968
14	0.0000	0.0000	0.0000	1.0000	0.0000
15	0.0000	0.0000	0.0000	1.0000	0.0000
16	0.0000	0.0000	0.0000	1.0000	0.0000
17	0.9825	0.2122	0.3490	1.0000	0.4606
18	0.0000	0.0000	0.0000	1.0000	0.0000
19	0.0000	0.0000	0.0000	1.0000	0.0000
20	0.0000	0.0000	0.0000	1.0000	0.0000
21	0.9925	0.2172	0.3565	1.0000	0.4661
22	0.0000	0.0000	0.0000	1.0000	0.0000
23	0.0000	0.0000	0.0000	1.0000	0.0000
24	0.0000	0.0000	0.0000	1.0000	0.0000

Table B.21: Full weekly results for Random Forest anomaly detector with threshold 0.9 - Part 1

Probability Threshold - 0.9 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9956	0.2422	0.3896	1.0000	0.4921
26	0.0000	0.0000	0.0000	1.0000	0.0000
27	0.0000	0.0000	0.0000	1.0000	0.0000
28	0.0000	0.0000	0.0000	1.0000	0.0000
29	0.9941	0.2436	0.3913	1.0000	0.4935
30	0.0000	0.0000	0.0000	1.0000	0.0000
31	0.0000	0.0000	0.0000	1.0000	0.0000
32	0.0000	0.0000	0.0000	1.0000	0.0000
33	0.9979	0.2655	0.4195	1.0000	0.5153
34	0.0000	0.0000	0.0000	1.0000	0.0000
35	0.0000	0.0000	0.0000	1.0000	0.0000
36	0.0000	0.0000	0.0000	1.0000	0.0000
37	0.9874	0.2322	0.3760	1.0000	0.4819
38	0.0000	0.0000	0.0000	1.0000	0.0000
39	0.0000	0.0000	0.0000	1.0000	0.0000
40	0.0000	0.0000	0.0000	1.0000	0.0000
41	0.9845	0.1756	0.2980	1.0000	0.4190
42	0.0000	0.0000	0.0000	1.0000	0.0000
43	0.0000	0.0000	0.0000	1.0000	0.0000
44	0.0000	0.0000	0.0000	1.0000	0.0000
45	0.9794	0.1160	0.2074	1.0000	0.3405
46	0.0000	0.0000	0.0000	1.0000	0.0000
47	0.0000	0.0000	0.0000	1.0000	0.0000
48	0.0000	0.0000	0.0000	1.0000	0.0000
Average	**0.2472**	**0.0520**	**0.0853**	**1.0000**	**0.1133**

Table B.22: Full weekly results for Random Forest anomaly detector with threshold 0.9 - Part 2

Probability Threshold - 0.95 - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.9903	0.0918	0.1679	1.0000	0.3029
2	0.0000	0.0000	0.0000	1.0000	0.0000
3	0.0000	0.0000	0.0000	1.0000	0.0000
4	0.0000	0.0000	0.0000	1.0000	0.0000
5	0.9929	0.1564	0.2703	1.0000	0.3955
6	0.0000	0.0000	0.0000	1.0000	0.0000
7	0.0000	0.0000	0.0000	1.0000	0.0000
8	0.0000	0.0000	0.0000	1.0000	0.0000
9	0.9927	0.1519	0.2635	1.0000	0.3897
10	0.0000	0.0000	0.0000	1.0000	0.0000
11	0.0000	0.0000	0.0000	1.0000	0.0000
12	0.0000	0.0000	0.0000	1.0000	0.0000
13	0.9983	0.1859	0.3135	1.0000	0.4312
14	0.0000	0.0000	0.0000	1.0000	0.0000
15	0.0000	0.0000	0.0000	1.0000	0.0000
16	0.0000	0.0000	0.0000	1.0000	0.0000
17	0.9878	0.1556	0.2689	1.0000	0.3945
18	0.0000	0.0000	0.0000	1.0000	0.0000
19	0.0000	0.0000	0.0000	1.0000	0.0000
20	0.0000	0.0000	0.0000	1.0000	0.0000
21	0.9961	0.1618	0.2784	1.0000	0.4023
22	0.0000	0.0000	0.0000	1.0000	0.0000
23	0.0000	0.0000	0.0000	1.0000	0.0000
24	0.0000	0.0000	0.0000	1.0000	0.0000

Table B.23: Full weekly results for Random Forest anomaly detector with threshold 0.95 - Part 1

Probability Threshold - 0.95 - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9986	0.1861	0.3137	1.0000	0.4314
26	0.0000	0.0000	0.0000	1.0000	0.0000
27	0.0000	0.0000	0.0000	1.0000	0.0000
28	0.0000	0.0000	0.0000	1.0000	0.0000
29	0.9965	0.1870	0.3149	1.0000	0.4325
30	0.0000	0.0000	0.0000	1.0000	0.0000
31	0.0000	0.0000	0.0000	1.0000	0.0000
32	0.0000	0.0000	0.0000	1.0000	0.0000
33	0.9986	0.2207	0.3616	1.0000	0.4698
34	0.0000	0.0000	0.0000	1.0000	0.0000
35	0.0000	0.0000	0.0000	1.0000	0.0000
36	0.0000	0.0000	0.0000	1.0000	0.0000
37	0.9944	0.1851	0.3121	1.0000	0.4302
38	0.0000	0.0000	0.0000	1.0000	0.0000
39	0.0000	0.0000	0.0000	1.0000	0.0000
40	0.0000	0.0000	0.0000	1.0000	0.0000
41	0.9945	0.1245	0.2214	1.0000	0.3529
42	0.0000	0.0000	0.0000	1.0000	0.0000
43	0.0000	0.0000	0.0000	1.0000	0.0000
44	0.0000	0.0000	0.0000	1.0000	0.0000
45	0.9947	0.0760	0.1411	1.0000	0.2756
46	0.0000	0.0000	0.0000	1.0000	0.0000
47	0.0000	0.0000	0.0000	1.0000	0.0000
48	0.0000	0.0000	0.0000	1.0000	0.0000
Average	**0.2487**	**0.0392**	**0.0672**	**1.0000**	**0.0981**

Table B.24: Full weekly results for Random Forest anomaly detector with threshold 0.95 - Part 2

B.4 Anomaly Detectors using Concept Drift

Anomaly Detector using Concept Drift - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6766	0.4322	0.5275	0.9990	0.6571
2	0.8117	0.5755	0.6735	0.9993	0.7584
3	0.7839	0.5200	0.6253	0.9994	0.7209
4	0.7710	0.5561	0.6461	0.9993	0.7454
5	0.8124	0.5556	0.6599	0.9994	0.7452
6	0.8001	0.5499	0.6518	0.9994	0.7413
7	0.8080	0.5710	0.6691	0.9994	0.7554
8	0.7872	0.5545	0.6507	0.9993	0.7444
9	0.8040	0.5740	0.6698	0.9994	0.7574
10	0.8303	0.5829	0.6849	0.9994	0.7633
11	0.8122	0.5719	0.6712	0.9994	0.7560
12	0.8497	0.6030	0.7054	0.9994	0.7763
13	0.8007	0.5815	0.6737	0.9993	0.7623
14	0.8038	0.5866	0.6782	0.9993	0.7656
15	0.7679	0.5561	0.6451	0.9992	0.7454
16	0.7037	0.5408	0.6116	0.9989	0.7350
17	0.7148	0.5586	0.6271	0.9988	0.7469
18	0.7450	0.5508	0.6333	0.9988	0.7417
19	0.8430	0.6791	0.7522	0.9988	0.8236
20	0.7679	0.5703	0.6545	0.9989	0.7548
21	0.7657	0.5686	0.6526	0.9988	0.7536
22	0.7862	0.6024	0.6822	0.9987	0.7757
23	0.8001	0.6128	0.6940	0.9989	0.7824
24	0.7911	0.5651	0.6593	0.9991	0.7514

Table B.25: Full Results of Anomaly Detector including Concept Drift Detection for retraining - Part 1

Anomaly Detector using Concept Drift - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.7935	0.5611	0.6574	0.9991	0.7487
26	0.7638	0.5131	0.6139	0.9991	0.7160
27	0.8039	0.5567	0.6578	0.9993	0.7458
28	0.8208	0.5813	0.6806	0.9994	0.7622
29	0.8395	0.5936	0.6955	0.9994	0.7702
30	0.8355	0.5963	0.6959	0.9993	0.7719
31	0.8257	0.5734	0.6768	0.9993	0.7570
32	0.8326	0.6248	0.7139	0.9992	0.7901
33	0.8434	0.6013	0.7021	0.9994	0.7752
34	0.7915	0.5676	0.6611	0.9992	0.7531
35	0.8452	0.5841	0.6908	0.9995	0.7641
36	0.7860	0.5188	0.6250	0.9994	0.7200
37	0.7998	0.5647	0.6620	0.9994	0.7512
38	0.7987	0.5361	0.6416	0.9993	0.7319
39	0.7854	0.5637	0.6563	0.9993	0.7505
40	0.8032	0.5905	0.6806	0.9992	0.7682
41	0.7604	0.5304	0.6249	0.9992	0.7280
42	0.7356	0.5052	0.5990	0.9990	0.7104
43	0.7490	0.5430	0.6296	0.9991	0.7365
44	0.7899	0.5131	0.6221	0.9994	0.7161
45	0.7677	0.4827	0.5927	0.9994	0.6945
46	0.7802	0.5438	0.6409	0.9993	0.7372
47	0.7435	0.4766	0.5809	0.9993	0.6902
48	0.7475	0.5390	0.6264	0.9992	0.7339
Average	**0.7892**	**0.5600**	**0.6547**	**0.9992**	**0.7476**

Table B.26: Full Result of Anomaly Detector including Concept Drift Detection for retraining - Part 2

B.5 Anomaly Classifiers using different classification algorithms

Random Forest Classifier - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.8123	0.8308	0.8215	0.4366	0.6022
2	0.7031	0.7840	0.7414	0.2355	0.4297
3	0.7162	0.7689	0.7416	0.3206	0.4965
4	0.7025	0.7976	0.7470	0.2185	0.4175
5	0.9061	0.9079	0.9070	0.7170	0.8068
6	0.6995	0.7524	0.7250	0.3245	0.4941
7	0.6984	0.7651	0.7303	0.2925	0.4731
8	0.6876	0.7661	0.7248	0.2746	0.4587
9	0.9042	0.9058	0.9050	0.7026	0.7977
10	0.7026	0.7578	0.7291	0.3176	0.4905
11	0.7105	0.7597	0.7343	0.3290	0.5000
12	0.7037	0.7639	0.7326	0.3100	0.4866
13	0.8975	0.8988	0.8982	0.6808	0.7822
14	0.7096	0.7329	0.7211	0.3860	0.5319
15	0.6487	0.6727	0.6604	0.4562	0.5539
16	0.5832	0.6022	0.5926	0.5019	0.5498
17	0.7907	0.7927	0.7917	0.7490	0.7706
18	0.6255	0.6587	0.6417	0.4703	0.5566
19	0.6709	0.6960	0.6832	0.5011	0.5906
20	0.7049	0.7247	0.7147	0.4359	0.5620
21	0.8803	0.8840	0.8822	0.6936	0.7830
22	0.7177	0.7398	0.7286	0.3999	0.5439
23	0.6962	0.7294	0.7124	0.3566	0.5100
24	0.7030	0.7253	0.7140	0.3505	0.5042

Table B.27: Full weekly results of the multi-class anomaly classifier using a Random Forest Classifier - Part 1

Random Forest Classifier - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9056	0.9078	0.9067	0.6885	0.7906
26	0.7154	0.7793	0.7460	0.2830	0.4696
27	0.7157	0.7416	0.7284	0.3754	0.5276
28	0.7080	0.7655	0.7356	0.2989	0.4784
29	0.9109	0.9126	0.9118	0.7147	0.8076
30	0.7142	0.7713	0.7417	0.3050	0.4850
31	0.7080	0.7489	0.7279	0.3509	0.5127
32	0.6863	0.7490	0.7163	0.3115	0.4830
33	0.9104	0.9123	0.9114	0.7054	0.8022
34	0.7142	0.7780	0.7447	0.2767	0.4640
35	0.7204	0.7695	0.7441	0.3427	0.5135
36	0.7189	0.7686	0.7429	0.3029	0.4825
37	0.9044	0.9070	0.9057	0.7002	0.7969
38	0.7254	0.7775	0.7506	0.3141	0.4942
39	0.7132	0.7752	0.7429	0.2970	0.4798
40	0.7203	0.7918	0.7544	0.2591	0.4529
41	0.9148	0.9169	0.9158	0.7226	0.8140
42	0.7078	0.7742	0.7395	0.2896	0.4735
43	0.7077	0.7757	0.7401	0.2818	0.4675
44	0.7117	0.7906	0.7491	0.2445	0.4397
45	0.9029	0.9059	0.9044	0.6777	0.7835
46	0.7336	0.7970	0.7640	0.2743	0.4676
47	0.7207	0.7866	0.7522	0.2760	0.4659
48	0.7326	0.7952	0.7626	0.2819	0.4735
Average	**0.7479**	**0.7878**	**0.7671**	**0.4174**	**0.5650**

Table B.28: Full weekly results of the multi-class anomaly classifier using a Random Forest Classifier - Part 2

Support Vector Machine - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6706	0.7210	0.6949	0.2825	0.4513
2	0.665	0.6233	0.6435	0.3968	0.4973
3	0.6764	0.5296	0.5941	0.4922	0.5105
4	0.6794	0.4143	0.5148	0.5884	0.4937
5	0.6325	0.7891	0.7022	0.2109	0.4079
6	0.6207	0.7825	0.6922	0.2175	0.4126
7	0.6158	0.7847	0.6901	0.2153	0.4110
8	0.6118	0.7728	0.6829	0.2272	0.4190
9	0.6269	0.7918	0.6997	0.2082	0.4060
10	0.6235	0.7896	0.6968	0.2104	0.4076
11	0.6275	0.7922	0.7003	0.2078	0.4058
12	0.6260	0.7912	0.6989	0.2088	0.4065
13	0.6216	0.7884	0.6951	0.2116	0.4084
14	0.6207	0.7878	0.6943	0.2122	0.4088
15	0.5212	0.7219	0.6054	0.2781	0.4480
16	0.4116	0.6415	0.5014	0.3585	0.4795
17	0.5283	0.5160	0.5221	0.5044	0.5102
18	0.5434	0.6060	0.5730	0.4149	0.5014
19	0.5646	0.6309	0.5959	0.3434	0.4655
20	0.6347	0.6550	0.6447	0.3540	0.4815
21	0.5948	0.7712	0.6716	0.2288	0.4200
22	0.6161	0.7849	0.6903	0.2151	0.4109
23	0.6286	0.7929	0.7012	0.2071	0.4053
24	0.6437	0.8023	0.7143	0.1977	0.3982

Table B.29: Full weekly results of the multi-class anomaly classifier using a Support Vector Machine - Part 1

Support Vector Machine - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.7323	0.0667	0.1223	0.9376	0.2501
26	0.6913	0.0622	0.1142	0.9410	0.2420
27	0.7019	0.0928	0.1639	0.9117	0.2909
28	0.7209	0.0361	0.0688	0.9675	0.1869
29	0.6394	0.7997	0.7106	0.2003	0.4003
30	0.6467	0.8042	0.7169	0.1958	0.3968
31	0.6323	0.7952	0.7045	0.2048	0.4036
32	0.6169	0.7855	0.6911	0.2145	0.4105
33	0.6446	0.8029	0.7151	0.1971	0.3978
34	0.6937	0.7993	0.7428	0.2007	0.4005
35	0.6366	0.7978	0.7081	0.2022	0.4016
36	0.6470	0.8044	0.7172	0.1956	0.3967
37	0.6732	0.6142	0.6423	0.3785	0.4822
38	0.6639	0.6903	0.6768	0.2963	0.4522
39	0.6579	0.6640	0.6610	0.3214	0.4620
40	0.6772	0.6557	0.6663	0.3504	0.4793
41	0.6464	0.8040	0.7166	0.1960	0.3970
42	0.6093	0.7806	0.6844	0.2194	0.4138
43	0.6212	0.7882	0.6948	0.2118	0.4086
44	0.6411	0.8006	0.7120	0.1994	0.3995
45	0.0317	0.1782	0.0539	0.8218	0.3827
46	0.0311	0.1764	0.0529	0.8236	0.3812
47	0.0308	0.1754	0.0524	0.8246	0.3803
48	0.4345	0.1734	0.2479	0.8266	0.3786
Average	**0.5902**	**0.6298**	**0.5762**	**0.3715**	**0.4117**

Table B.30: Full weekly results of the multi-class anomaly classifier using Support Vector Machine - Part 2

Logistic Regression - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.6325	0.7953	0.7046	0.2047	0.4035
2	0.6216	0.7884	0.6952	0.2116	0.4084
3	0.6360	0.7975	0.7076	0.2025	0.4019
4	0.6472	0.8045	0.7174	0.1955	0.3966
5	0.6227	0.7891	0.6961	0.2109	0.4079
6	0.6122	0.7825	0.6870	0.2175	0.4126
7	0.6158	0.7847	0.6901	0.2153	0.4110
8	0.5972	0.7728	0.6737	0.2272	0.4190
9	0.6271	0.7919	0.6999	0.2081	0.4060
10	0.6235	0.7896	0.6968	0.2104	0.4076
11	0.6275	0.7922	0.7003	0.2078	0.4058
12	0.6260	0.7912	0.6989	0.2088	0.4065
13	0.6218	0.7885	0.6953	0.2115	0.4084
14	0.6207	0.7878	0.6943	0.2122	0.4088
15	0.5212	0.7219	0.6054	0.2781	0.4480
16	0.4116	0.6415	0.5014	0.3585	0.4795
17	0.3833	0.6191	0.4735	0.3809	0.4856
18	0.4116	0.6416	0.5015	0.3584	0.4795
19	0.6401	0.7100	0.6733	0.2900	0.4538
20	0.5829	0.7635	0.6611	0.2365	0.4250
21	0.5943	0.7709	0.6711	0.2291	0.4203
22	0.6161	0.7849	0.6903	0.2151	0.4109
23	0.6412	0.7928	0.7090	0.2072	0.4053
24	0.6437	0.8023	0.7143	0.1977	0.3982

Table B.31: Full weekly results of the multi-class anomaly detector using a Logistic Regression Classifier - Part 1

Logistic Regression - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.6451	0.8032	0.7155	0.1968	0.3976
26	0.6401	0.8001	0.7112	0.1999	0.3999
27	0.6575	0.7992	0.7214	0.2008	0.4006
28	0.6439	0.8024	0.7145	0.1976	0.3982
29	0.6389	0.7993	0.7101	0.2007	0.4005
30	0.6467	0.8042	0.7169	0.1958	0.3968
31	0.6323	0.7952	0.7045	0.2048	0.4036
32	0.6679	0.7854	0.7219	0.2146	0.4105
33	0.6452	0.8033	0.7156	0.1967	0.3975
34	0.6937	0.7993	0.7428	0.2007	0.4005
35	0.6366	0.7978	0.7081	0.2022	0.4016
36	0.6470	0.8044	0.7172	0.1956	0.3967
37	0.6376	0.7985	0.7090	0.2015	0.4011
38	0.6310	0.7943	0.7033	0.2057	0.4042
39	0.6243	0.7901	0.6975	0.2099	0.4072
40	0.6625	0.8013	0.7253	0.1987	0.3990
41	0.6471	0.8044	0.7172	0.1956	0.3966
42	0.6093	0.7806	0.6844	0.2194	0.4138
43	0.6870	0.7882	0.7341	0.2118	0.4086
44	0.6907	0.8007	0.7416	0.1994	0.3995
45	0.6534	0.8083	0.7226	0.1917	0.3936
46	0.6547	0.8091	0.7238	0.1909	0.3930
47	0.6544	0.8089	0.7235	0.1911	0.3931
48	0.6541	0.8088	0.7233	0.1912	0.3933
Average	**0.6204**	**0.7811**	**0.6909**	**0.2189**	**0.4108**

Table B.32: Full weekly results of the multi-class anomaly classifier using a Logistic Regression Classifier - Part 2

B.6 Anomaly Classifiers using Concept Drift Detectors

Anomaly Classifier using Concept Drift - Part 1					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
1	0.8120	0.8307	0.8212	0.4355	0.6015
2	0.9107	0.9124	0.9116	0.7342	0.8185
3	0.8985	0.9007	0.8996	0.6717	0.7778
4	0.9104	0.9123	0.9114	0.6993	0.7988
5	0.9057	0.9074	0.9066	0.7153	0.8057
6	0.8989	0.9006	0.8997	0.7036	0.7960
7	0.8971	0.8992	0.8982	0.6970	0.7917
8	0.8945	0.8964	0.8955	0.7120	0.7989
9	0.9043	0.9059	0.9051	0.7032	0.7981
10	0.8967	0.8985	0.8976	0.6802	0.7818
11	0.8996	0.9018	0.9007	0.6910	0.7894
12	0.9034	0.9050	0.9042	0.7005	0.7962
13	0.8944	0.8958	0.8951	0.6701	0.7748
14	0.9033	0.9053	0.9043	0.7142	0.8041
15	0.8648	0.8675	0.8661	0.7409	0.8017
16	0.8029	0.8056	0.8042	0.7435	0.7739
17	0.7913	0.7933	0.7923	0.7498	0.7712
18	0.7957	0.7981	0.7969	0.7448	0.7710
19	0.8409	0.8448	0.8428	0.7322	0.7864
20	0.8665	0.8709	0.8687	0.6742	0.7663
21	0.8794	0.8831	0.8812	0.6914	0.7814
22	0.8810	0.8853	0.8831	0.6654	0.7675
23	0.8916	0.8950	0.8933	0.6756	0.7776
24	0.8930	0.8961	0.8946	0.6508	0.7637

Table B.33: Full results of Anomaly Classifier using Concept Drift Detection for retraining - Part 1

Anomaly Classifier using Concept Drift - Part 2					
Week	**Precision**	**Recall**	**F1 Score**	**Specificity**	**Geometric Mean**
25	0.9076	0.9096	0.9086	0.6966	0.7960
26	0.8991	0.9015	0.9003	0.6697	0.7770
27	0.9083	0.9104	0.9094	0.7108	0.8044
28	0.9061	0.9080	0.9070	0.6892	0.7911
29	0.9091	0.9110	0.9100	0.7100	0.8042
30	0.9073	0.9094	0.9083	0.6933	0.7940
31	0.8919	0.8947	0.8933	0.6609	0.7689
32	0.8913	0.8940	0.8926	0.6822	0.7809
33	0.9113	0.9131	0.9122	0.7079	0.8040
34	0.9022	0.9049	0.9036	0.6918	0.7912
35	0.8981	0.9009	0.8995	0.6784	0.7818
36	0.9047	0.9073	0.9060	0.6869	0.7895
37	0.9068	0.9092	0.9080	0.7090	0.8029
38	0.8956	0.8988	0.8972	0.6861	0.7852
39	0.8930	0.8965	0.8947	0.6944	0.7890
40	0.9085	0.9109	0.9097	0.7115	0.8051
41	0.9147	0.9168	0.9157	0.7219	0.8135
42	0.8858	0.8895	0.8876	0.6937	0.7855
43	0.8973	0.9001	0.8987	0.7026	0.7952
44	0.9064	0.9087	0.9076	0.7006	0.7979
45	0.9037	0.9066	0.9051	0.6803	0.7853
46	0.9052	0.9082	0.9067	0.6870	0.7899
47	0.9054	0.9082	0.9068	0.6828	0.7874
48	0.9046	0.9075	0.9060	0.6802	0.7857
Average	**0.8896**	**0.8924**	**0.8910**	**0.6922**	**0.7854**

Table B.34: Full Results of Anomaly Classifier using Concept Drift Detection for retraining - Part 2