Detecting Financial Fraud Using Rule-Based Techniques

By

Saiful Islam

17MCSE018P

Department of Computer Science and Engineering

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

This dissertation is submitted for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

February, 2024

CERTIFICATION

The thesis titled *Detecting Financial Fraud Using Rule-Based Techniques*
Submitted by **Saiful Islam**
Roll No **17MCSE018P** Session **2017-2018**
has been accepted as satisfactory in partial fulfillment of the requirement for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING** on **15/01/2024**.

BOARD OF EXAMINERS

1. _____ Supervisor
   Dr. Md. Mokammel Haque
   Professor, Dept. of Computer Science and Engineering
   Chittagong University of Engineering & Technology (CUET)
   Chittagong- 4349, Bangladesh

2. _____ Member
   Dr. Abu Hasnat Mohammad Ashfak Habib
   Professor & Head
   Dept. of Computer Science and Engineering
   Chittagong University of Engineering & Technology (CUET)
   Chittagong- 4349, Bangladesh

3. _____ Member
   Dr. Muhammad Ibrahim Khan
   Professor, Dept. of Computer Science and Engineering
   Chittagong University of Engineering & Technology (CUET)
   Chittagong- 4349, Bangladesh

4. _____ Member
   Dr. Mohammed Moshiul Hoque
   Professor, Dept. of Computer Science and Engineering
   Chittagong University of Engineering & Technology (CUET)
   Chittagong- 4349, Bangladesh

5. _____ External

Prof. Dr. Mohammad Osiur Rahman
Professor, Dept. of Computer Science and Engineering
UNIVERSITY OF CHITTAGONG (CU)
Chittagong-4331, Bangladesh

Declaration

The undersigned hereby certifies that no portion of this thesis has ever been submitted for consideration for any degree or diploma at any other location.

Saiful Islam
Student ID: 17MCSE018P
Session:2017-2018
Department of Computer Science and Engineering
Chittagong University of Engineering and Technology

I dedicate this thesis to my **father, mother, and all my family members**.

Acknowledgements

First and foremost, all praises to Allah, the Lord of the worlds. The completion of this thesis would have been impossible without the permission and blessings of the Almighty Allah.

I extend my sincere gratitude to my supervisor, Prof. Dr. Md. Mokammel Haque, for his invaluable guidance, patience, and understanding throughout the duration of this thesis. His insightful suggestions, meaningful discussions, and constant encouragement have played a pivotal role in my exploration of the field of machine learning. Working with him has been an honor and a pleasure. I also want to express my appreciation to all the teachers in the Department of Computer Science and Engineering for their friendly interactions and support, with special thanks to Prof. Dr. Md. Mokammel Haque, whose effective suggestions were instrumental in completing this thesis.

My heartfelt thanks go to Prof. Dr. Asaduzzaman, Prof. Dr. Mohammad Moshiul Hoque, Prof. Dr. Muhammad Ibrahim Khan, and Prof. Dr. Abu Hasnat Mohammad Ashfak Habib for their comprehensive feedback on my thesis.

I am deeply grateful to my wife for her unwavering love, courage, support, and understanding. I extend my cordial respect and sincere appreciation to my teachers, friends, and colleagues for their unforgettable cooperation, constructive suggestions, sacrifices, and warm support throughout the entire process.

Lastly, I would like to acknowledge and thank the Department of Computer Science and Engineering (CSE) at CUET for providing me with numerous opportunities to present my work.

Abstract

Financial fraud is a growing problem that poses a significant threat to the banking industry, the government sector, and the public. In response, financial institutions must continuously improve their fraud detection systems. While preventive and security measures are put in place to mitigate financial fraud, criminals persistently adjust and develop new methods to circumvent fraud prevention systems. This dynamic adaptation poses challenges for quantitative techniques and predictive models. To address the challenge of unbalanced financial datasets, this study aims to develop rules to detect fraud transactions and improve accuracy using Anomaly Reduction Boundary Based Oversampling (ARBBO) method. The performance of the proposed model is evaluated using various metrics such as accuracy, precision, recall, f1-score, confusion matrix, and ROC values. The proposed model is compared to several existing machine learning models such as Random Forest (RF), Decision Tree (DT), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN), Naive Bayes (NB), and Logistic Regression (LR) using one benchmark dataset. The experimental results demonstrate that the classifiers performed better with the resampled data, and the suggested Rule-Based model with ARBBO in financial fraud detection outperformed then other algorithms by achieving an accuracy and precision of 0.998 and 0.998, respectively.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

F	complex function
F	complex function
F	complex function

Greek Symbols

$\imath$	unit imaginary number $\sqrt{-1}$
$\imath$	unit imaginary number $\sqrt{-1}$
$\imath$	unit imaginary number $\sqrt{-1}$
π	$\simeq 3.14\ldots$
π	$\simeq 3.14\ldots$
π	$\simeq 3.14\ldots$

Superscripts

j	superscript index
j	superscript index
j	superscript index

Subscripts

0	subscript index
0	subscript index

0 subscript index

Other Symbols

$\oint_\gamma$ integration around a curve γ

$\oint_\gamma$ integration around a curve γ

$\oint_\gamma$ integration around a curve γ

Acronyms / Abbreviations

CIF Cauchy's Integral Formula

CIF Cauchy's Integral Formula

CIF Cauchy's Integral Formula

Chapter 1

Introduction

1.1 Background

Financial fraud remains a significant concern within the business community. Despite technological advancements, there is a noticeable increase in fraud cases [1]. Over the past decade, the surge in e-commerce has led to a substantial uptick in the utilization of credit cards. This heightened reliance on credit cards has, in turn, resulted in a consistent rise in fraudulent transactions [2], posing a significant challenge to the financial industry. According to a recent report, the financial impact of credit card fraud reached 27.85 billion dollars in 2018, marking a 16.2% increase from the 23.97 billion dollars lost in 2017. Projections suggest that this figure could escalate to 35 billion dollars by 2023 [3]. Figure 1.1 describes the amount of credit card fraud from 2013 to 2027. The implementation of effective fraud monitoring and prevention measures stands as a viable solution to mitigate these losses. As e-commerce has flourished in recent decades, more people have adopted online transactions, contributing to the prevalence of card payments. Unfortunately, this increased reliance on card payments has created favorable conditions for the rise in fraudulent activities. Fraud, as defined by the Oxford Dictionary [4], involves wrongful or criminal deception resulting in financial or personal gain. Fraud detection involves identifying unusual cardholder behaviors compared to their previous card usage profiles.

Alerts are triggered when target transactions show a probability exceeding the fraud classification threshold based on these differences. Fraudulent transactions typically occur through unauthorized access to card information, such as credit card numbers [5], email addresses, phone numbers [6], and more, leading to monetary theft. According to the Federal Trade Commission [7], credit card fraud cases numbered 459,297, with identity theft cases increasing by 44.6% from 271,927 in 2019 to 393,207 in 2020. Figure 1.2 funds lost as a result of contact fraud and data breaches in the business sector in 2018. It is evident that the

Fig. 1.1 Card Fraud in worldwide from 2013 to 2027.

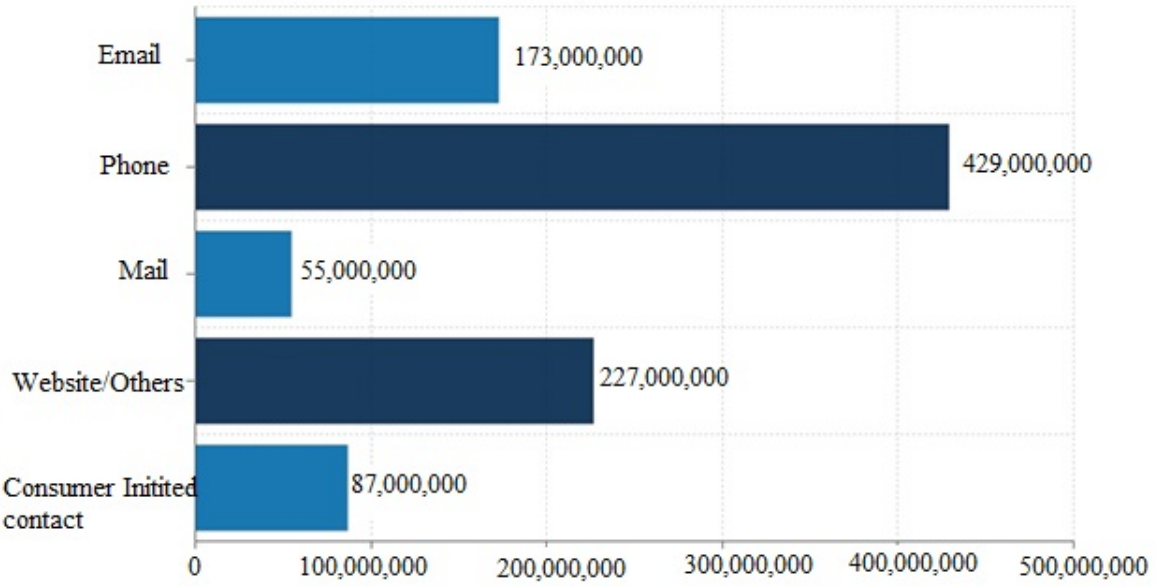

Fig. 1.2 Money lost due to fraud by method of contact.

majority of credit card fraud and identity theft were conducted over the phone or via websites (Figure 1.2). Therefore, the work of this dissertation is strongly motivated by the growing necessity to discover an appropriate solution that removes these cybercrimes. Identity theft crimes represent a different kind of threat. Credit card fraud was the most frequent identity

theft, according to a 2018 research that was updated in 2020 [7] and made public by the Shift Credit Card Processing website. According to Figure 1.3, it was responsible for 29% of all identity theft reports in 2018. Additionally, as shown in Figures 1.4, the study disclosed the identity theft fraud reports in the US in 2018. The number of credit card fraud incidents has risen in the past year and is expected to rise. Because the stolen money has the potential to severely harm these financial systems and stifle company operations, it has an impact on both consumers and enterprises. Unauthorized financial transactions might even push businesses into bankruptcy and make it impossible for them to get regular payments.

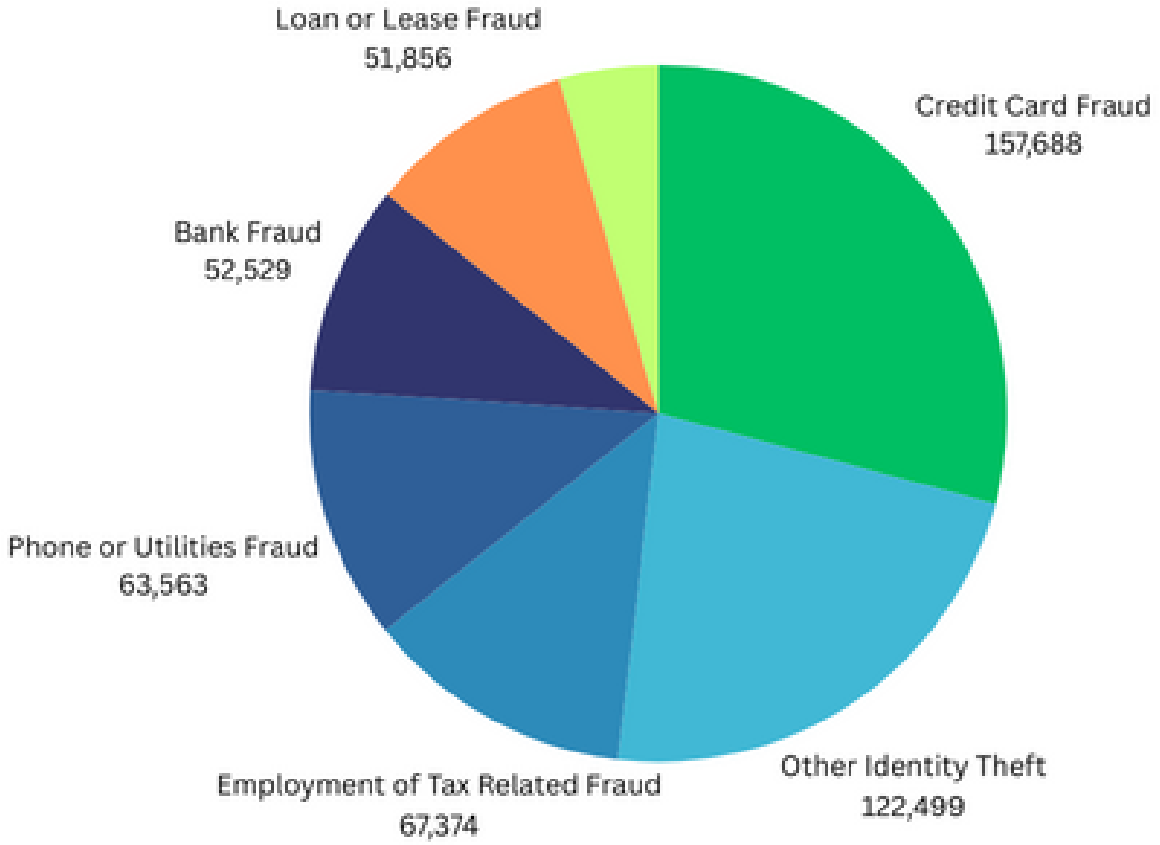

Fig. 1.3 Reports on identity theft fraud.

Credit card fraud can be categorized into two main types: application fraud [8] and behavior fraud [9]. Application fraud involves deceptive credit card applications, where a fraudster initiates a new credit card process using false identity details, and the issuer unwittingly accepts the request. On the other hand, behavior fraud occurs after a credit card has been legitimately issued and refers to transactions involving fraudulent behavior. The detection of credit card fraud poses a significant challenge for both credit card users and financial organizations. The ability to identify even a small number of fraudulent transactions

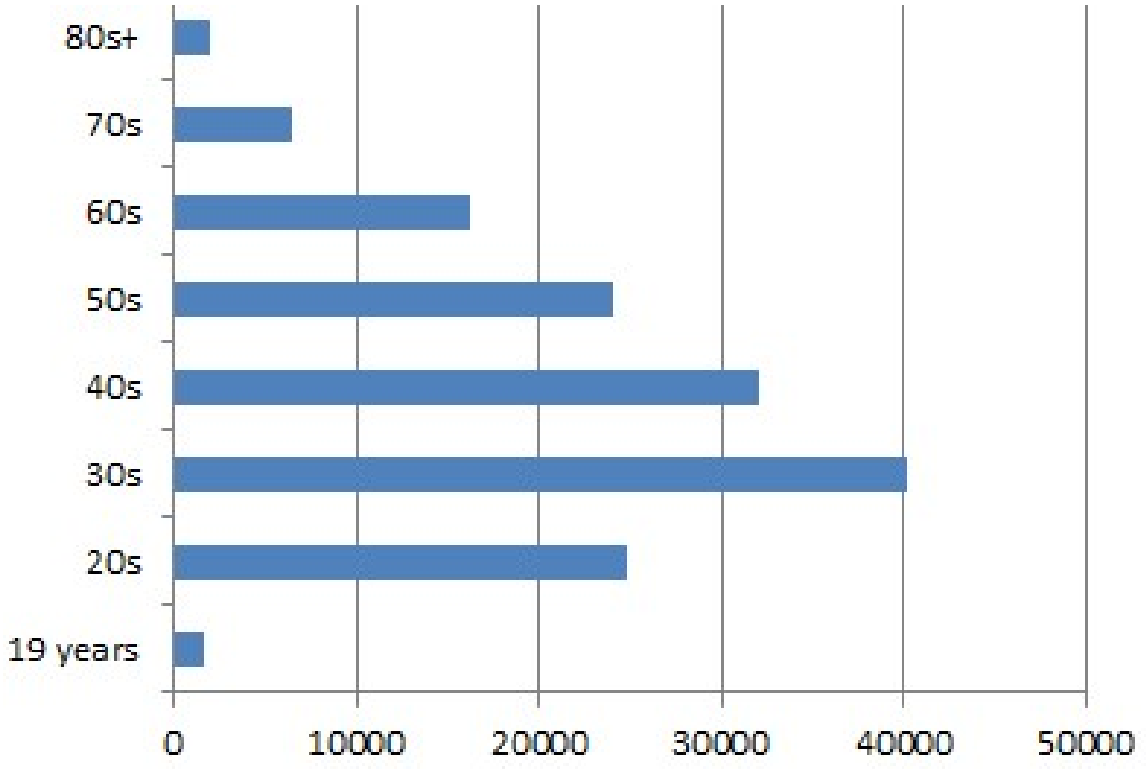

Fig. 1.4 Instances of identity theft in the United States.

is crucial, as it can safeguard substantial amounts of money. Consequently, credit card fraud has emerged as a significant concern for researchers seeking effective solutions in fraud detection and prevention. Credit card fraud (CCF) constitutes a form of identity theft where an unauthorized individual, distinct from the card owner, engages in illicit transactions using stolen credit card or account information. Fraud may occur when a credit card is lost, stolen, or counterfeited. Additionally, the prevalence of card-not-present fraud, involving the unauthorized use of credit card numbers in online transactions, has grown due to the surge in online shopping. The expansion of e-banking and various online payment platforms has led to a notable increase in fraud, including Credit Card Fraud (CCF), resulting in annual losses amounting to billions of dollars. In the current era of digital payments, detecting CCF has become a paramount objective. For business owners, the shift toward a cashless culture is undeniable. Traditional payment methods are becoming obsolete, hindering business expansion. Customers increasingly prioritize debit and credit card payments, emphasizing the need for businesses to adapt their systems to accommodate diverse payment methods. It is anticipated that this trend will intensify in the coming years [10].

Consequently, it is imperative for financial institutions to prioritize the implementation of

an automated fraud detection system. In supervised Credit Card Fraud (CCF) detection, the objective is to develop a machine learning (ML) model using available transactional credit card payment data. This model is designed to effectively differentiate between fraudulent and nonfraudulent transactions, enabling the system to make informed decisions about the legitimacy of incoming transactions. To counteract card fraud, significant resources and funding have been dedicated to developing a robust fraud-detection system aimed at preventing financial losses. Various machine learning algorithms have been employed to analyze extensive data, encompassing classical methods such as logistic regression [11], support vector machine [12], decision trees [13], hidden Markov models [14], and state-of-the-art techniques like gradient boosting tree [15] and deep learning [16]. Notably, gradient boosting tree and deep learning, specifically CatBoost and Deep Neural Network (DNN), stand out as highly promising solutions, known for their exceptional performance in fraud detection. CatBoost, with its ability to incorporate user transaction history, is particularly advantageous, making it the preferred choice for handling both new users and those with a transaction history simultaneously processing transactions, CatBoost excels in leveraging historical customer data, whereas DNN is utilized for identifying fraud in transactions involving unknown users. Addressing challenges related to data categorization, the support vector machine (SVM) stands out as a supervised machine learning (ML) technique widely utilized in diverse domains, including image recognition [17], credit rating [18], and public safety [19].

SVM is adept at handling both linear and nonlinear binary classification problems, determining a hyperplane that effectively separates input data into the support vector, offering superiority over other classifiers. While neural networks were initially employed for identifying credit card theft [18], deep learning (DL), a subset of ML, has taken center stage with a focus on DL approaches. In recent years, deep learning methods have garnered significant attention, showcasing promising outcomes across applications like computer vision, natural language processing, and voice. However, the application of deep neural networks in Credit Card Fraud (CCF) identification has been relatively understudied [17]. Notably, it employs various deep learning algorithms for CCF detection. In the contemporary landscape, a majority of transactions occur online, involving credit cards and various payment systems, providing convenience for both companies and consumers. Amidst this digital transaction environment, banks play a crucial role in ensuring the legality and non-fraudulent nature of all transactions. Detecting fraud proves to be a challenging task, given the persistent efforts of fraudsters to make illegitimate transactions appear genuine [20].

To address this challenge, banks find themselves compelled to recruit skilled software engineers and fraud detection experts, in addition to investing in specialized software, resulting in significant financial expenditures. Traditional approaches to fraud detection have relied on

expert systems [21], which are techniques designed to solve problems and provide answers within specific contexts. However, a drawback of expert systems lies in their increasing maintenance costs as they become more specialized [21]. The advent of artificial intelligence (AI) holds the potential to revolutionize the financial services industry. Within the realm of AI, machine learning (ML) encompasses models for prediction and pattern recognition, requiring minimal human intervention. In the financial services sector, the application of ML methods holds the promise of enhancing outcomes for both businesses and consumers, serving as a potent tool in the fight against credit fraud. Currently, extensive efforts [20][22][23][24] are underway to develop machine learning (ML) models for combating credit fraud. Employing ML for predictive modeling holds the potential to enhance efficiency, reduce costs, improve quality, and elevate customer satisfaction [25]. However, a significant challenge and potential obstacle in these models lies in their lack of transparency in decision-making processes.

These models often function as black boxes, revealing only their inputs and outputs, making it challenging to comprehend their internal workings. Consequently, understanding their properties becomes intricate, and certain risks may go undetected. This complexity poses a substantial barrier to the integration of ML in existing credit fraud detection (CFD) systems [26][27]. The opacity of these black-box models carries implications for financial regulators, who must consider the opportunities for improved compliance and safety facilitated by ML. Simultaneously, they need to be mindful of the potential ways in which ML could be employed to subvert the objectives of existing regulations. For instance, the United States prohibits discrimination based on various categories, such as race, sex, and marital status. Furthermore, a lending algorithm could be deemed in violation of this prohibition even if it doesn't explicitly use any of the prohibited categories but relies on data highly correlated with these protected categories. The lack of transparency could present an even more challenging issue in the European Union, where the General Data Protection Regulation adopted in 2016, set to take effect in 2018, grants citizens the right to receive explanations for decisions solely based on automated processing [28]. Despite its limitations, machine learning (ML) holds potential applications in various areas within financial services. The inherent opacity of ML systems, however, imposes significant constraints on their use in crafting regulations [28].

For instance, the U.S. prohibits discrimination based on categories such as race, sex, and marital status [23], posing challenges in leveraging ML for regulatory purposes. The consequences of employing black-box models include potential biases in results and the inherent difficulty in understanding the rationale followed by algorithms to reach specific conclusions [24][29]. There's a risk that the data used to train ML models may not be representative in fraud operations [28], leading to the potential endorsement of erroneous decisions. Some

organizations advocate for additional guidance on interpreting existing regulations, emphasizing the need to break down barriers through enhancing the interpretability of these models. Since regulatory authorities are composed of humans, explanations must be comprehensible to humans. Similarly, decision models should be easily understandable, allowing scrutiny of the attributes essential for generating intelligible explanations [30]. Machine learning is primarily utilized in fraud detection to enhance organizations and financial institutions' capabilities in identifying fraudulent transactions. However, fraud detection can present challenges for machine learning due to various reasons. Such as:

- The data exhibits significant imbalance, with a very small number of fraudulent transactions.
- The data undergoes continuous evolution over time.
- Privacy concerns limit the availability of real-world datasets.

However, because of the class imbalance in the datasets, credit card fraud detection is still difficult to learn [31]. Nonetheless, credit card fraud detection poses a learning challenge primarily attributed to class imbalance within datasets [32]. While other issues contribute to hindrances in credit card fraud detection, class imbalance stands out as the most crucial challenge [33]. This imbalance is a prevalent concern in various real-world machine learning (ML) applications, where datasets exhibit uneven class distributions, such as one class (the majority) having significantly more samples than the other class (the minority). Credit card transaction datasets commonly face imbalances, as legitimate transactions far outnumber fraudulent ones [34]. Traditional ML algorithms generally perform optimally with balanced data, and the skewed class distribution in credit card datasets can result in biased performance toward the majority class. This bias occurs because these algorithms prioritize error rates rather than considering class distribution [35].

Consequently, more misclassifications tend to happen for minority class examples compared to majority class samples [36]. Typically, addressing the issue of unbalanced datasets involves a two-fold approach, focusing on both data and algorithmic aspects, and sometimes their combination [37]. On the data front, scholars commonly employ resampling techniques, involving operations like copying, synthesizing, and deleting original samples to adjust sample numbers and mitigate the impact of imbalanced datasets. Resampling techniques are categorized into oversampling for minority class samples and undersampling for majority class samples. In oversampling, the primary concept is to augment the number of minority class samples for achieving class balance. Common methods involve replicating samples and generating new samples. Random Oversampling (ROS) involves randomly replicating original samples to expand the number of minority class samples, but it may introduce noise samples, affecting dataset quality [38]. Generating new samples entails deriving them

from one or more original samples, indirectly reflecting minority class features. The classic oversampling method is the SMOTE algorithm [39], which selects the line connecting two original samples to determine a point on the line as the new sample. However, SMOTE still faces challenges, such as generating noise samples and susceptibility to the distribution of original samples, potentially causing the new samples to deviate from the actual distribution. Subsequent scholars have made improvements to SMOTE in terms of noise reduction and generation algorithms, introducing methods like Borderline-SMOTE [40], Adasyn [41], LR-SMOTE [42], and others.

Undersampling achieves class balance by reducing the number of majority class samples, employing techniques such as undersampling based on clustering algorithms and Edited Nearest Neighbor (ENN) [43]. In practice, most imbalanced datasets result from too few samples in the minority class, making oversampling a focal point in this field [44].

1.2 Conventional machine learning models

In many scientific fields, machine learning is essential, and its applications are used on a regular basis. Examples of its applications include spam email filtering, weather forecasting, product recommendations, medical diagnosis, facial recognition, fraud detection, and more. The study of learning, or the challenge of gaining information via experience, is known as machine learning (ML). Usually, this process entails observing a phenomenon and formulating a hypothesis about it in order to make predictions or, more generally, behave logically. We can describe machine learning (ML) as the process of extracting information from data because, for computers, data provides the experience or phenomenon to learn. Data mining, pattern recognition, and statistics are all strongly related to machine learning [45]. Simultaneously, it becomes a branch of computer science that focuses specifically on the algorithmic aspect of knowledge extraction. In conclusion, machine learning (ML) focuses on developing algorithms that can recognize patterns in data without human intervention. Several popular classification methods are described in this section, such as K-Nearest Neighbor, Support Vector Machine, Random Forest, Naive Bayes, Multi Layer Perceptron, and Logistic Rigression.

1.2.1 K-Nearest Neighbour

Instead of learning to generalize, an instance-based learning algorithm produces predictions by comparing a new instance with the examples present in the training dataset. W. Daelemans and associates [46]. As a result, instance-based algorithms spend more time in prediction

and require less time for training. One of the instance-based or memory-based learning algorithms is K-Nearest Neighbor (KNN) in Figure 1.5. Comparing the characteristics of groups of instances is the core idea behind KNN. The instances that belong to the same class label and have similar properties are neighbours. A new unclassified instance's class label can be found by locating its closest neighbors. The class label with the greatest occurrences among those neighbors will be allocated to the new instance, where K is the number of neighbors to take into account. The KNN algorithm is comprised of 4 stages:

1. Select a suitable value for the parameter K: Select a suitable K value: The KNN model's performance is significantly influenced by the value of K. A different K could lead to a different label assignment because it suggests a different number of neighbors to take into account. Finding the optimal K that neither overfits nor underfits the model is crucial. Cross-validation using distinct K choices on the training set is one way to find a solution. Next, for each K, calculate the folds' average accuracy and F1-score. Finally, by contrasting the outcomes of various K, we are able to determine the ideal K.

2. Compute the distances between the unclassified instance and the training instances: Determine the distances between the training instances and the unclassified instance: The primary KNN criterion is distance. There are three widely used techniques, such as:

- **Euclidean Distance:** The square root of the sum of the squared distances between two data points is used by the L2 norm to calculate the distance using equation 1.1.

$$d_{Euclidean}(a,b) = \sqrt{\sum_{i=1}^{j}(a_j - b_j)^2} \tag{1.1}$$

  where b_j stands for the j_{th} property of the instances of a and b, and a = a_j.
- **Manhattan Distance:** The distance is determined by adding together the absolute differences in distance between two data points using equation 1.2.

$$d_{Manhattan}(a,b) = \sum_{i=1}^{j}|a_j - b_j| \tag{1.2}$$

  where b_j stands for the j_{th} property of the instances of a and b, and a = a_j.
- **1-Cosine Distance:** It takes the cosine difference between two vectors, a and b, then subtracts it from 1 using equation 1.3.

$$dist(a,b) = 1 - cos\theta \tag{1.3}$$

where the angle between two vectors is represented by θ.

- **Hamming Distance:** Unlike the previous two, it indicates whether two things belong in the same category or not.

3. Identify the K nearest neighbors of the unclassified instance: Locate the unclassified instance's K closest neighbors: In step 1, an optimized K value is calculated. The technique can then be used to anticipate unclassified instances by locating the K nearest neighbors after the K and a distance option was selected in step 2.

4. Tally the count and assign values accordingly: Count and assign: K neighbors of the unclassified instance are selected, as seen in Figure 3.1. Next, KNN counts how many neighbors each class label has. In the event when K=3 and label B has the greatest occurrences, the instance will be assigned to class B. Voting is another name for this.

KNN is an easy-to-implement, straightforward algorithm. When appropriate K and distance algorithms are chosen, KNN can perform well; however, it can also perform poorly if the dataset has a large number of outliers or noise. Furthermore, because KNN does not have a process for learning the data distribution, Larose and Larose et al. [47] point out that the prediction technique with KNN is computationally expensive.

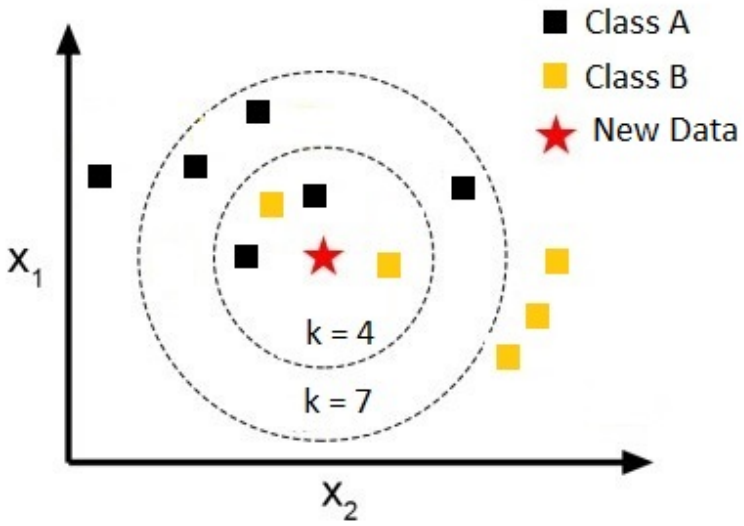

Fig. 1.5 An example of categorizing a new instance to class A or B using KNN.

1.2.2 Support Vector Machine

Support Vector Machines (SVM) is another popular algorithm used by many researchers. H.Zhang et al.[48] introduced that the objective of the SVM is distinctly classifing instances

by calculating an n-dimensional hyperplane. The hyperplane separates the data points into two groups, where each group belongs to one class label. Figure 1.6 shows an example of using a two-dimensional hyperplane to separate data points. The left one represents the learning process of an SVM model. It maximizes the distances between the hyperplane and each side of data points. As R. Gandhi et al.[49] explained, to train an SVM model is to find the optimal hyperplane which maximizes the margin between the data points and hyperplane. The process of hyperplane optimisation follows the equation 1.4:

$$min_w \lambda ||w||^2 + \sum_{i=1}^{n} c(x, y, f(x)) \tag{1.4}$$

where **w** is a weighted vector and λ is an activation function. Equation 1.5 illustrates the two scenarios of the hinge loss that SVM uses, which is represented by the value $c(x, y, f(x))$. The cost is zero if the true value and the forecasted value match. If not, $1 - y \cdot f(x)$ will be the outcome and the cost. By using gradient descent to update its weights, SVM seeks to minimize loss with cost function 1.4 and loss 1.5.

$$c(x, y, f(x)) = \begin{cases} 1 - f(x) = f(z)^2 & \text{if } y \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

According to research by Duan et al. [50], SVM has less computational complexity than KNN. The reason for this is that KNN is more sensitive to dataset volume than SVM is. The number of chosen support vectors, which is often small, determines how computationally complex SVM is. In addition, SVM has the benefit of being able to handle datasets with both low and high dimensions. Stated differently, it can be applied to datasets that are either linearly separable or linearly inseparable. High-dimensional datasets can be used to train and generalize SVM models. Furthermore, SVM has access to a variety of kernel functions, including Sigmoid, Gaussian, and linear ones. SVM operates on binary datasets by design. However, techniques like the "one-against-rest" strategy by Bishop et al.[45] can be used to classify many classes.

1.2.3 Random Forest

Random Forest (RF) is a widely-used approach for categorization. E. Kremic and A. Subasi [51] note its popularity due to a simple learning procedure and quick learning curve. According to A. Liaw et al. [52], RF is an ensemble of decision trees, selecting data randomly for inducing trees (refer to Figure 1.7). RF functions as an aggregate classifier, combining multiple decision tree classifiers. The key idea is to train the trees sufficiently so that each

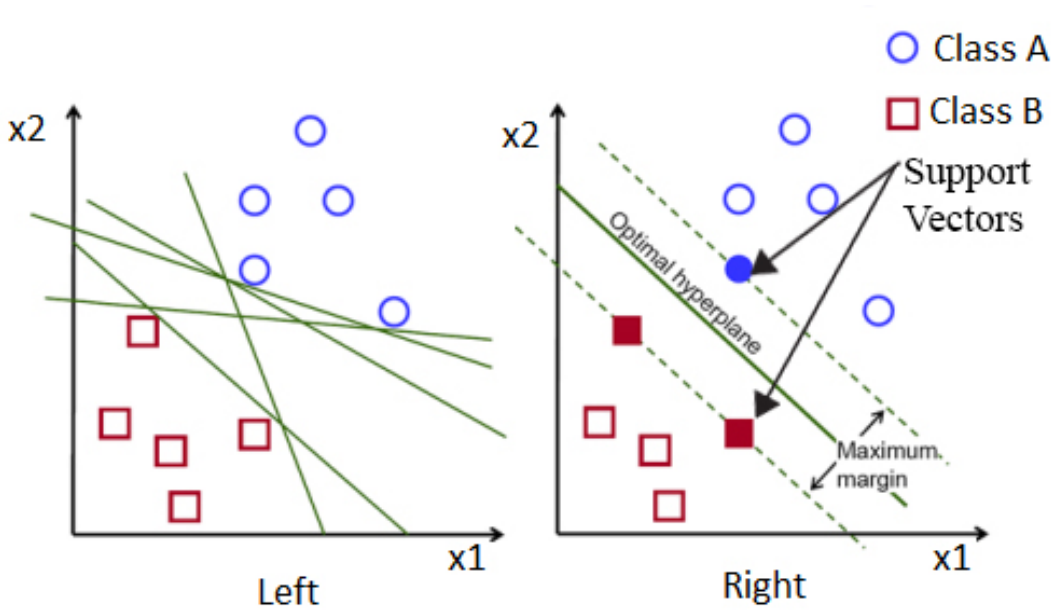

Fig. 1.6 An illustration of how to use SVM to locate the hyperplane that divides the red square and the blue circle.

contributes to the model's structure. In RF classification, each tree in the forest chooses a subset of data examples and provides a classification decision. The final decision results from aggregating all outputs through a majority vote. This model, which relies on a dataset with a consistent distribution across the trees [53], is versatile and applicable to both classification and regression problems.

1.2.4 Naive Bayes

The Naïve Bayes method was first presented by G.John and P.Langley in 1995 [54]. This model is an example of a probabilistic classifier. This model suggests that it is capable of simultaneously obtaining predictions for several classes. The Bayes Theorem is the foundation of this model. The probabilistic classifiers in Naïve Bayes allow this model to predict more than one class. Conditional probability is the foundation for the decision. Instead of using a single method, this approach makes use of a collection of algorithms, although they are all based on the same idea. According to this concept, every variable contributes to the outcome in a distinct and equal way. Additionally, this model has a distinct advantage over other models because it needs very little training data [55]. The Naïve Bayes classifier selects the decision with the highest probability based on the Bayes theorem [56]. Based on known values and probabilities, Bayesian probability is computed. The following

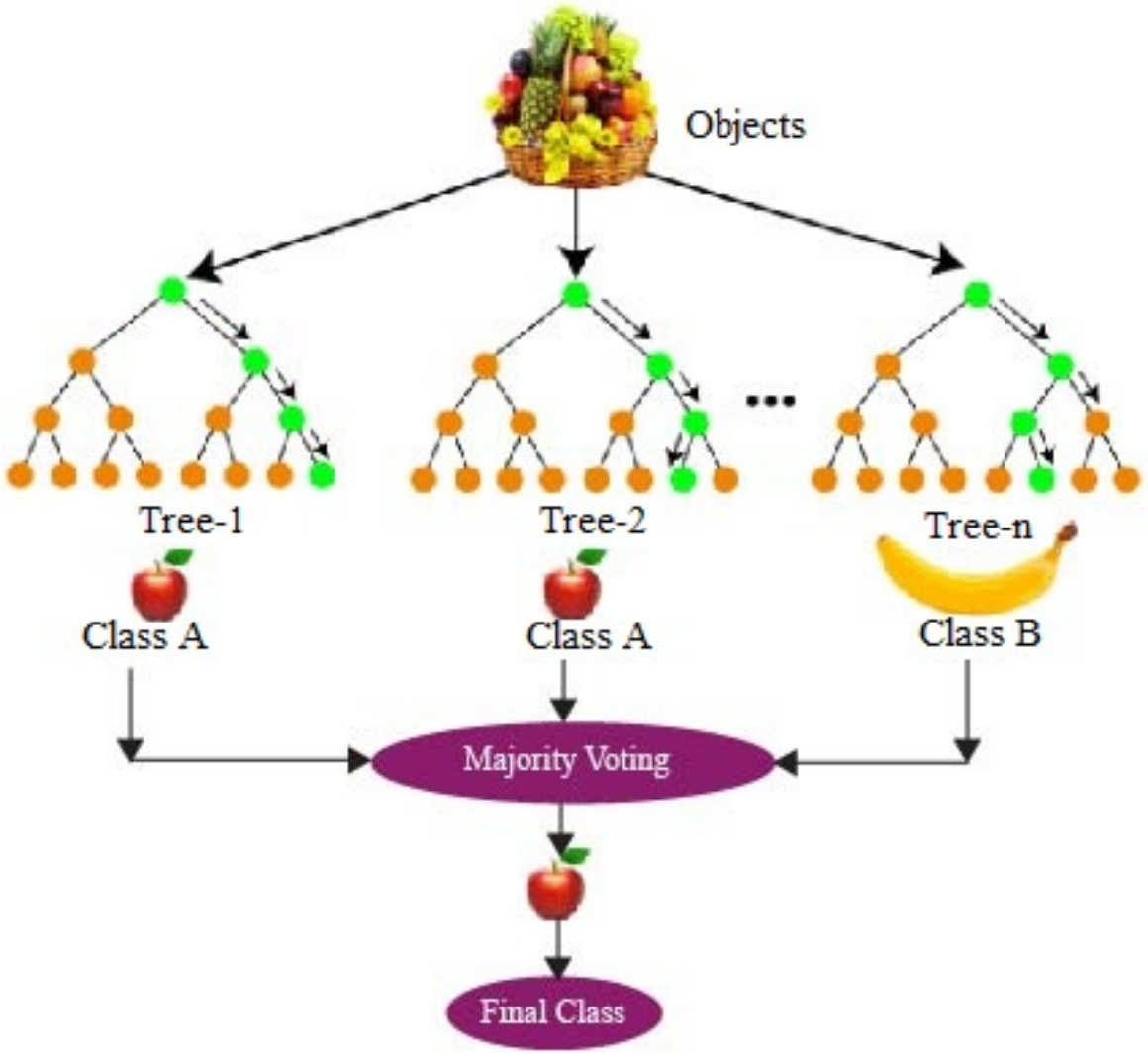

Fig. 1.7 An illustration of a Random Forest classifier.

formula represents the supervised machine learning algorithm Naïve Bayes 1.6.

$$P(A/B) = \frac{P(A \cap B)}{P(B)} \tag{1.6}$$

The posterior likelihood P (A|B), or the likelihood of result (A) under certain conditions (B), can be found using the Bayes theorem. Without any knowledge of specific conditions, the Bayes Theorem computes the later probability by relating it to the preceding likelihood of the outcome via the probability ratio P (B|A) = P (B). The Naïve Bayes theorem is predicated on the idea that every element affects the result separately and is, thus, naïve.

1.2.5 Logistic Rigression

A functional technique called logistic regression (LR) [57] forecasts a binary response probability depending on one or more factors and it is represented in 1.7 and 1.8. Data mining tasks using additional statistical models, such as discriminant analysis, regression analysis, multiple-logistic regression, and other analyses, are included in the LRn model. When it comes to credit card fraud, the LR model is quite helpful since it can forecast certain outcomes based on the presence or absence of characteristic values using a collection of variables called predictor variables. For every independent variable in the model, odds ratios can be calculated using LR coefficients. Compared to characteristic analysis, it is applicable to a wider variety of study scenarios. LR is a statistical technique applied to situations involving binary classification. Based on one or more predictor variables, it models the likelihood that an instance belongs to a specific class. LR is used for classification, not regression, despite its name. Predicted values between 0 and 1, which represent probabilities, are mapped using the logistic function, also called the sigmoid function.

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}} \tag{1.7}$$

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} \tag{1.8}$$

Where:

$P(Y=1|X)$ is the probability that the dependent variable Y is 1 given the values of predictor variables X.

$\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients of the model.

$X_1, X_2, \ldots, X_n$ are the predictor variables.

e is the base of the natural logarithm.

1.2.6 Decision Tree

Quinlan [48] invented the Decision Tree (DT) technique, which can handle consecutive data. A table of tree appearances composed of internal, root, and leaf nodes is called a decision tree. Similar to Figure 1.8, the trained system that generates a set of conditions at each level informs the conclusion made by the decision tree. The decision tree is based on data mining methods that use the breadth-first or depth-first greedy approaches to recursively divide a dataset of records [58] [13]. There are lines connecting each node and leaf. The Decision Tree classification method may assign a single leaf node or several branch nodes to each

node. An illustration of Decision Tree construction and how this model makes a decision depending on the variables used is shown in Figure 1.8. By breaking difficult problems down into simpler ones and building a Decision Tree based on the knowledge gained by data mining, the Decision Tree solves problems. The foundation of the Decision Tree model is the extremely accurate and small-scale construction of a tree.

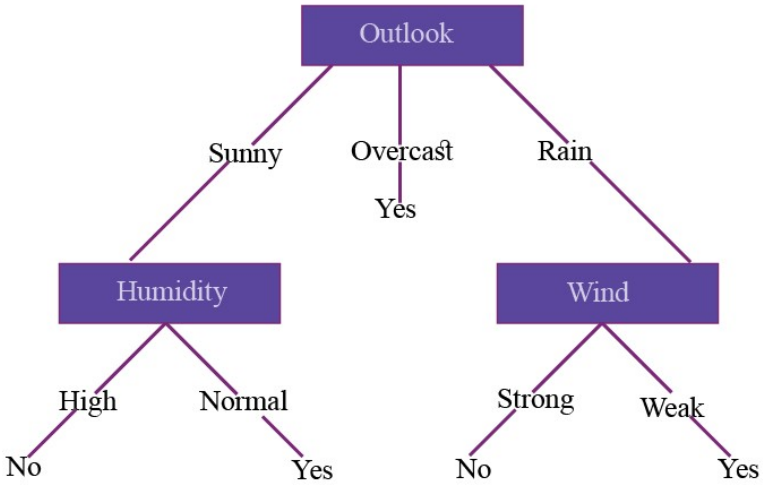

Fig. 1.8 An illustration of a Decision Tree classifier.

1.3 Financial Fraud

Financial fraud, an insidious practice involving deceptive activities to illicitly obtain money or assets [59], has burgeoned in recent years due to the rapid evolution of technology and the pervasive digitization of society. The nefarious exploitation of these technological advances by fraudsters has led to an alarming increase in the frequency and sophistication of financial fraud cases. The spectrum of financial fraud encompasses various categories, with bank fraud, corporate fraud, and insurance fraud being among the most prevalent [60]. In the realm of bank fraud, the unauthorized initiation of transactions within an individual's bank account is a recurrent occurrence. Corporate fraud often materializes in the form of financial statement fraud or manipulative schemes involving securities and commodities, while insurance fraud spans diverse domains such as car and health care insurance [60].

Within the realm of bank fraud, several subtypes stand out, including credit card fraud, money laundering, and mortgage fraud [61]. Credit card fraud, in particular, has garnered significant attention in recent investigative efforts aimed at enhancing financial fraud detection.

The machinations involved in credit card fraud often exploit vulnerabilities in online transactions and payment systems, necessitating continuous advancements in detection mechanisms and security protocols. The specialized focus on credit card fraud underscores the urgency of understanding its nuances and developing robust countermeasures. Subsequent sections will delve into the intricate details of credit card fraud, shedding light on the methodologies employed by fraudsters, emerging trends, and the technological innovations crucial for effective detection and prevention. By unraveling the intricacies of specific types of financial fraud, researchers and practitioners can stay one step ahead of the perpetrators, fortifying the financial landscape against deceptive practices. Credit card fraud occurs when an unauthorized individual exploits someone else's credit card information to make unauthorized purchases, presenting a pervasive challenge on a global scale. Perpetrators typically gain access to the cardholder's physical credit card information, and in response, payment card and credit card issuers are actively implementing measures to safeguard customers from such fraudulent activities [62].

This type of fraud can manifest in various forms, including unauthorized online transactions or the misuse of another person's card for in-store shopping [63]. Given the prevalence of credit card fraud, both cardholders and issuers share concerns about potential fraudulent activities. Cardholders, understandably, worry about the security of their financial information, while issuers face the responsibility of compensating customers for losses incurred in many instances. The collaborative efforts of payment card and credit card issuers, coupled with advancements in security measures, aim to mitigate the risks associated with credit card fraud and enhance the overall safety of electronic transactions. Credit card issuers employ rule-based methods and various strategies for fraud detection. One approach involves utilizing highly advanced fraud detection software, which scrutinizes transactions and, based on historical data, determines their legitimacy. Another method is to identify patterns in credit card holders' behavior. If a transaction deviates from the usual pattern, the credit card company initiates an investigation to ascertain its validity [64]. For instance, individuals often use their credit cards to pay for lunch at a restaurant or make purchases at stores while outside their city, transactions that deviate from their normal spending patterns. In such cases, banks may request validation of these credit card transactions to verify if the cardholder indeed made these purchases at specific locations. This adherence to rule-based regulations is a standard practice for banks in certain situations. However, there are instances where these transactions are false positives. Customers might confirm that they did use their credit card at a particular restaurant or store for the first time. In such situations, customers are asked to verify and confirm the accuracy of the transactions. If any transactions are identified

as invalid or fraudulent, it is crucial for the cardholder to promptly inform the credit card issuer, request the account be closed, and obtain a new credit card.

1.4 Types of Financial Fraud

It is possible to identify many different kinds of fraud [65], such as application fraud, theft/counterfeit fraud, computer intrusion, credit card fraud, and telecommunication fraud.

1.4.1 Credit Card Fraud

There are two types of credit card fraud.

- **Online Credit Card Fraud**: Online credit card fraud is a sophisticated form of financial crime where criminals exploit the digital landscape to conduct unauthorized transactions. In this type of fraud, perpetrators typically acquire credit card information through various cybercriminal activities, such as data breaches, phishing schemes, or malware attacks. Once armed with the necessary details, including card numbers, expiration dates, and CVVs, fraudsters engage in online shopping, subscription fraud, or other electronic transactions without the need for the physical card. The anonymity and global reach of the internet make it challenging to trace and apprehend those responsible for online credit card fraud. To combat this threat, security measures such as encryption, multi-factor authentication, and real-time transaction monitoring are crucial in detecting and preventing fraudulent activities in the virtual space.
- **Offline Credit Card Fraud**: Offline credit card fraud, in contrast, involves the use of physical credit cards that have been unlawfully obtained. Criminals may steal physical cards from individuals, wallets, or conduct skimming operations at points of sale to copy card details. The stolen cards are then used for in-person transactions, such as making purchases at retail stores, restaurants, or withdrawing cash from ATMs. Unlike online fraud, offline credit card fraud leaves a tangible trail, and law enforcement may use surveillance footage and transaction records to investigate and apprehend perpetrators. However, the increasing prevalence of contactless payment methods and chip technology has led to advancements in card security, aiming to mitigate the risk of offline fraud by making it more challenging for criminals to clone or use stolen physical cards. Figure 1.9 depicts a man using an ATM to take out cash.

Fig. 1.9 Man withdrawing cash from an ATM

1.4.2 Telecommunication Fraud

Over the past ten years, the telecommunications sector has expanded significantly on a global scale, especially with the introduction of various phone-based technologies [66]. Due to the widespread usage of phone technology, there is a corresponding rise in mobile phone fraud worldwide. This worldwide issue results in large yearly losses for numerous corporations, enterprises, and communication service providers. The easiest fraud to commit and the one with the least chance of illicit financial gain for perpetrators is telecommunication fraud. Subscription fraud and overlaid fraud are the two categories of telecommunication fraud. Subscription fraud occurs when criminals get hold of an account which functions similarly to a phone number without intending to pay the fee. Therefore, all of the transactions using the phone number that was received will be fake. Most likely, fraud via phone calls or other illegal activities are employed to sell calls using these accounts. Meanwhile, superimposed fraud is the act of obtaining a valid account through theft. In this instance, the atypical use is placed on top of the regular use by real clients. Cloning cells is an example of overlaid deception. Moreover, insider fraud in the telecommunications industry can happen when a worker sells confidential information to a dishonest opponent for illicit gain. Since our solution focuses on detecting odd payment activity through a machine learning algorithm and halting any possible fraudulent transaction, it should safeguard credit card consumers against both types of telecommunication fraud. Additionally, the suggested method use the unique credit card number for each online transaction. Because the information on these cards would be meaningless, even insiders would not be able to sell the users' information to anybody.

1.4.3 Computer Intrusion

Computer intrusion is a type of cybercrime that involves gaining unauthorized access to personal computers, mobile phones, and other electronic devices, as well as altering or stealing data for illicit uses [67]. Stated differently, it represents an unsanctioned endeavor to obtain data. Usually, people are the target of this kind of crime. An insider who is familiar with the system's architecture and infrastructure frequently commits computer intrusions. An outsider (hacker) may also be the intruder. In order to prevent this type of fraud, the suggested system also aims to identify the origin (location) of every payment transaction. This allows the system to block any transactions that appear suspicious. Conversely, our method adds an extra layer of security by using the one-time credit card information.

1.4.4 Bankruptcy Fraud

The purpose of bankruptcy is to allow a person or business to reorganize and resolve their debts [68]. In this context, using a credit card by someone who intends to file for bankruptcy is considered bankruptcy fraud. Since the credit card issuer cannot be certain of a customer's financial situation, they are obligated to cover their losses in the event of bankruptcy, making bankruptcy fraud one of the trickiest scams to anticipate. Verifying the consumers' financial history by contacting the credit bureau is one way to prevent this type of scam. The credit bureau assists credit card companies and banks in looking into the financial backgrounds of people seeking credit.

1.4.5 Theft Fraud

Here, using a credit or debit card that is not your own is considered theft fraud. Prior to the real customer reporting unusual transactions on their account and asking the card issuer to block the card, the fraudster attempts to use the victim's credit card as many times as feasible. The quicker the victimized customer reports, the quicker the bank responds. Our method can also be used to combat fraud of this type. Our method makes use of computer and mobile applications to provide a virtual credit card. In order to avoid theft fraud, consumers can conduct purchases online without a real card.

1.4.6 Application Fraud

When someone applies for a credit card using false information, it is known as application fraud [69]. Phua et al. conducted a study that looked at over 300 million applications for fraudulent accounts. The study found that 88% of those accounts were opened through

identity fraud methods [70]. It is possible to distinguish between two distinct cases of application fraud: identity thieves and duplicate applications. Applications from the same users with the same information are considered duplicate applications. In this instance, credit card applications employ cross-matching techniques to identify such duplication by producing a suspicious numerical score based on implicit ties to one another in real-time. Applications from various people with identical features constitute identity fraud. To check credit card eligibility, most banks, however, require applicants to complete a form with specified information. Identification, address, phone number, personal information, and other pertinent data are all included in this data. For the objectives of identification and duplication search, the majority of the necessary data is employed. Since machine learning (ML) can be used to identify potentially harmful applications in the future, ML algorithms can generally prevent this form of fraud.

1.5 Fraud detection process

As seen in figure 1.10, the transactions are first verified at the terminal point to see if they are genuine or not. Certain necessary requirements, such having a sufficient balance and a valid PIN (personal identification number), are checked at the terminal point, and the transactions are filtered based on the results. Following the scoring of all valid transactions, the predictive model determines if the transactions are real or fraudulent. Every bogus alert is looked into by the investigators, who also give the predictive model input to help it work better [71]. The prediction model is the only topic covered in this thesis.

1.6 Imbalance Classification Challenge and Solution

Addressing imbalanced datasets in classification tasks has posed enduring challenges. A dataset is deemed imbalanced when one or more classes exhibit a significantly larger number of instances compared to other classes within the dataset. Various approaches exist to quantify the imbalance, one of which is the imbalanced ratio (IR). As outlined by S. Kotsiantis et al. [72] and W. Lee et al. [73], the IR represents the proportion of samples in the majority class (negative class) to the minority class (positive class). In a binary dataset with classes Cmajority and Cminority, the IR is calculated as (Cmajority/Cminority). A higher IR indicates a more imbalanced dataset, with balance achieved when the IR equals 1.

Classifiers often exhibit strong performance for dominant classes but tend to yield weaker results for minority classes. Consequently, a classifier may be erroneously deemed effective based solely on high accuracy scores, primarily attributed to its accuracy on dominant classes.

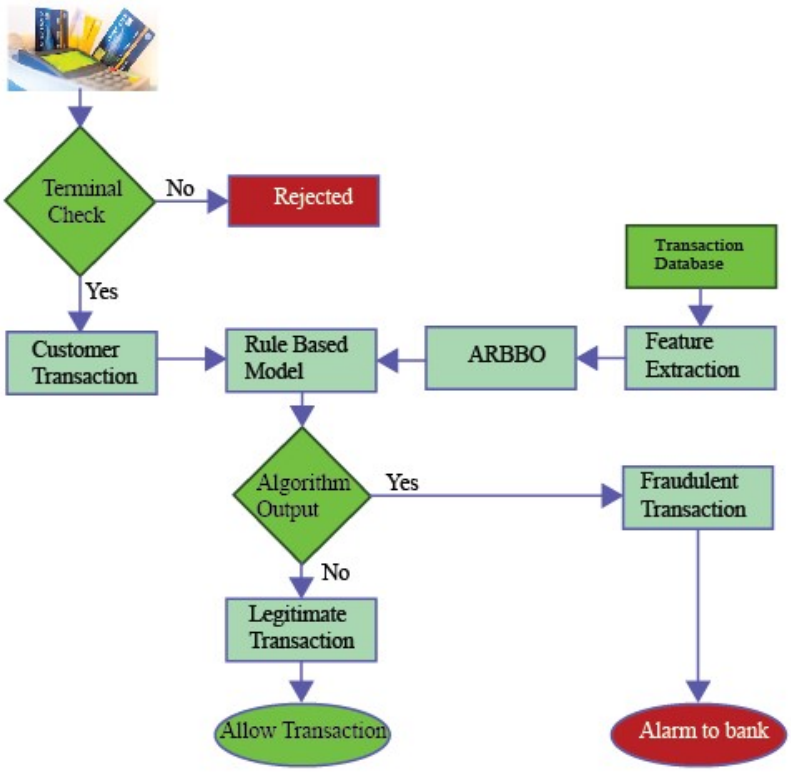

Fig. 1.10 The process of identifying and preventing fraud.

However, in practical applications, the minority classes, though outnumbered, hold greater significance in domains such as business decisions and medical diagnoses. A. Fern´aNdez et al.[74] demonstrated that this challenge is prevalent in binary classification tasks, such as fraud detection, and extends to multi-class classification tasks, including depression detection. Multi-class datasets, featuring several dominated classes, compound the complexity of the problem.

A. Fern´aNdez et al. [75] asserted that many classifiers assume balanced datasets during training, resulting in ineffective training and suboptimal performance in predicting minority labels, despite achieving a commendable overall score. Several factors contribute to such failures:

- Ineffective Training Assumptions: Many classifiers are built under the assumption that datasets are balanced, leading to suboptimal performance when faced with imbalanced data.
- Neglect of Minority Classes: Classifiers may prioritize accuracy on majority classes, neglecting the minority classes that are often more crucial in real-world applications.

- Underestimation of Imbalance Impact: The impact of dataset imbalance on classifier performance may be underestimated, leading to inadequacies in addressing the challenges posed by minority classes.
- Failure to Adapt to Multi-Class Imbalance: In multi-class datasets with multiple dominated classes, classifiers may struggle to adapt, exacerbating the difficulty of achieving balanced predictions across all classes.

Recognizing and addressing these issues is imperative for developing effective classifiers that perform well across all classes, particularly in scenarios where minority classes hold significant importance. Ongoing research and advancements in the field aim to devise strategies to overcome the hurdles posed by imbalanced datasets and enhance the robustness of classification models. As the challenge of data imbalance remains a focal point for various communities, numerous methods have been devised to address it. Broadly, [76] categorized these methods into three groups:

- Data re-sampling.
- Cost-sensitive learning.
- Ensemble techniques.

1.6.1 Data Re-sampling

Data resampling refers to the process of modifying the original dataset by either adding or removing instances to create a more balanced distribution of classes, especially in the context of imbalanced datasets. Imbalanced datasets occur when one class significantly outnumbers the other, leading to challenges in training machine learning models that may exhibit biased behavior towards the majority class. There are three main types of data resampling:

- **Oversampling**: This involves increasing the number of instances in the minority class by either duplicating existing instances or generating synthetic data points. The goal is to balance the class distribution and provide the model with more examples of the minority class to learn from. Common oversampling techniques include Random Oversampling, Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN). Figure 1.11 shows the over sampling process.
- **Undersampling**: This entails reducing the number of instances in the majority class by randomly removing some of them. The objective is to level the class distribution and prevent the model from being biased towards the majority class. Undersampling methods include Random Undersampling and NearMiss. Figure 1.12 shows the under sampling process.

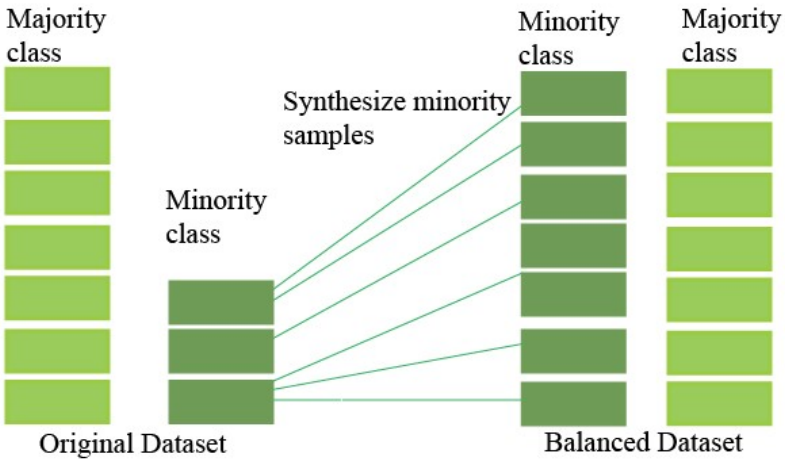

Fig. 1.11 Random oversampling process

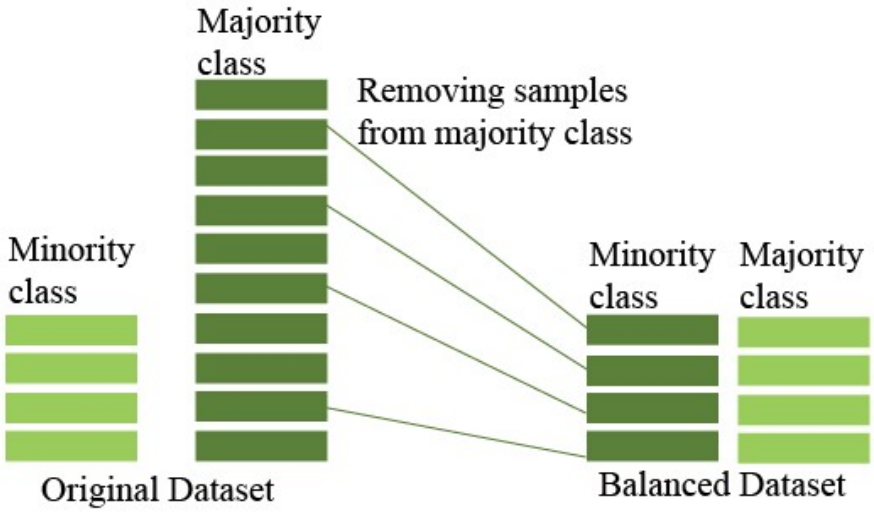

Fig. 1.12 Random undersampling process

- **Hybrid-sampling**: Hybrid sampling is an approach that combines elements of both oversampling and undersampling techniques to address class imbalance in datasets. The goal of hybrid sampling is to create a more balanced dataset by simultaneously adjusting the proportions of instances in both the majority and minority classes. This approach aims to capture the benefits of oversampling (providing the model with more instances of the minority class) and undersampling (reducing the dominance of the majority class) while mitigating their respective drawbacks. In a hybrid-sampling strategy, various techniques may be employed to achieve a balanced distribution, depending on the characteristics of the dataset and the desired outcome.
  Some hybrid-sampling methods involve applying oversampling to the minority class

and undersampling to the majority class in a coordinated manner. The objective is to strike a balance that allows the model to learn from sufficient instances of the minority class without overwhelming it with redundant information from the majority class. The effectiveness of hybrid-sampling techniques often depends on the specific nature of the data and the machine learning algorithm being used. Researchers and practitioners may experiment with different combinations of oversampling and undersampling methods to find the most suitable approach for a given problem, aiming to enhance the model's generalization and predictive capabilities on imbalanced datasets. Figure 1.13 shows the Hybrid resampling process.

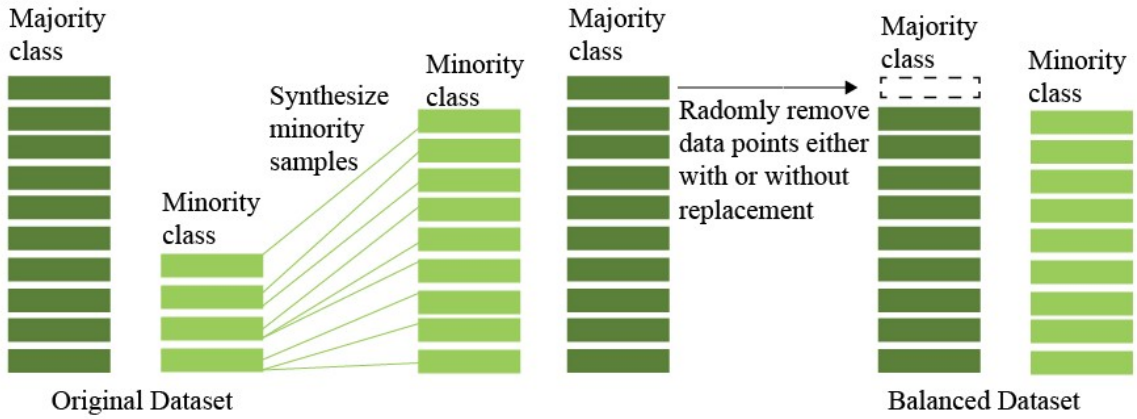

Fig. 1.13 Hybrid resampling process

In the literature, [39] introduced SMOTE (Synthetic Minority Oversampling Technique), a widely embraced data resampling method aimed at addressing the challenge of imbalanced datasets. Belonging to the oversampling category, SMOTE stands out for its remarkable performance, particularly in enhancing instances through duplication. Illustrated in Figure 1.14 is a depiction of a two-dimensional dataset characterized c x zz by class imbalance, with the minority class represented by the rectangular points.

With the SMOTE approach, as demonstrated in Figure 1.15, the algorithm begins by identifying the K-nearest-neighbors of each minority sample. Subsequently, a new sample is synthesized at a random location along the connected line between the focal sample and its nearest neighbors. The green dot in the figure symbolizes the synthesized sample. SMOTE has served as a foundation for several extensions, each tailored to specific needs. For instance, Borderline-SMOTE, proposed by [40], and Safe-level-SMOTE, developed by [77], are among the extensions that have built upon the original SMOTE methodology. These extensions introduce refinements and variations to further enhance the effectiveness of SMOTE in different contexts.

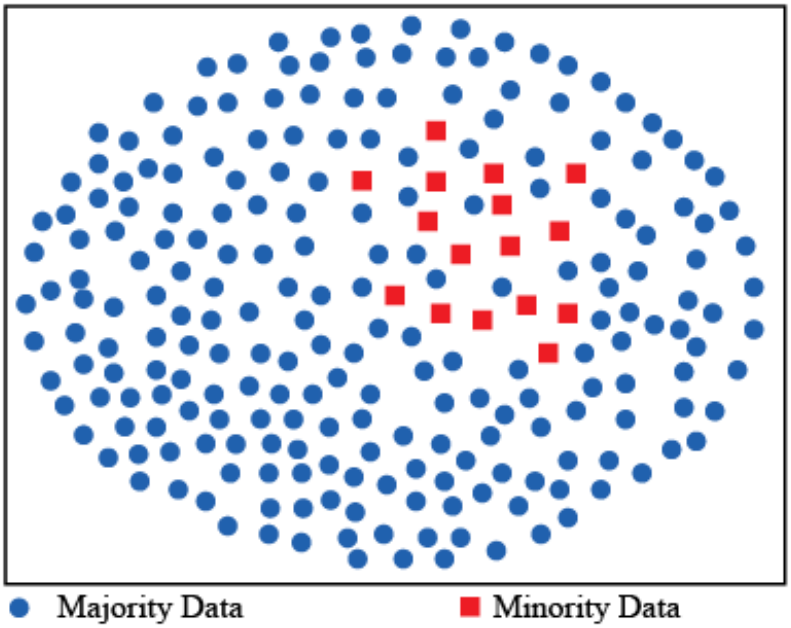

Fig. 1.14 Example of Imbalance Data

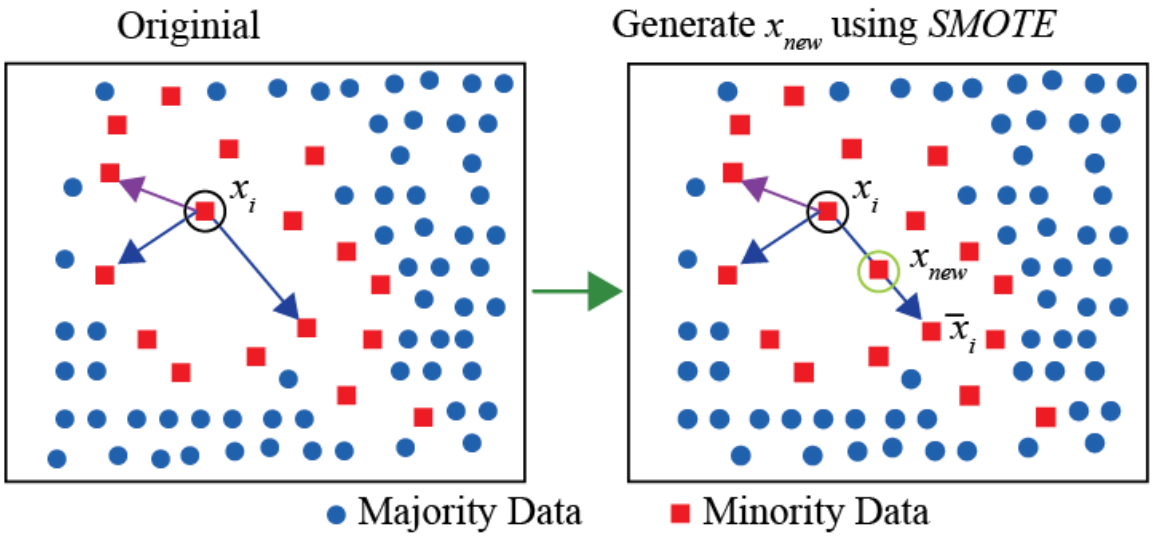

Fig. 1.15 Example of Synthetic Data using SMOTE

1.6.2 Cost-sensitive learning

The primary way that data resampling techniques contribute is through their work on data level adjustments. P. Domingos et al. [78], however, notes that cost-sensitive learning combines both data level and algorithmic level initiatives. The primary concept of cost-sensitive learning is summed up by [78] as accounting for the costs associated with each inaccurate classification. Prior to the classifiers learning, a cost matrix is constructed, and the

cost values inside it are defined by task experts or by using references. Those cost values will be considered when learning to minimize the classification loss. A larger cost will be imposed for each minority class instance that is incorrectly classified since the minority class is more significant than the majority class. A classifier should be able to categorize instances into the class with the lowest cost given a cost matrix. Let's take an example where Figure

	Actual Negative	Actual Positive
Predicted Negative	0	92
Predicted Positive	10	0

Fig. 1.16 Example of a cost matrix for financial fraud detection

is a cost matrix for a task involving fraud detection. When a prognosis is accurate, there is no expense. However, if a positive sample is anticipated to be negative, there will be a $92 fee, and if a negative sample is predicted to be positive, there will be a $10 cost. A total cost matrix can be defined the equation 2.1 as follows based on these costs:

$$C = C_{FN} \times N_{FN} + C_{FP} \times N_{FP} \tag{1.9}$$

where N is the quantity of samples. The goal of a cost-sensitive learning algorithm is to reduce C's value.

Overcoming data imbalance is made easier with the help of the cost matrix concept. However, there are two significant drawbacks that prevent customers from utilizing it. To begin with, it is typically challenging to accurately describe or obtain a high-quality cost matrix. Using medical categorization tasks as an example, physicians can offer valuable insights toward detecting diseases. However, they are unable to calculate the price of a mistaken classification. If inaccurate classifications result in the loss of life, then ethical concerns can also arise. Second, it is uncommon for machine learning algorithms to be created expressly for learning that is cost-sensitive. It is necessary to make specific adjustments to each algorithm. The time required for its development and testing may be considerable.

1.6.3 Ensemble techniques

R. Polikar [79] introduced an ensemble strategy that uses different classifiers to independently learn from the same training set. Next, an aggregate of all the classifiers' predictions is calculated. This method aims to outperform individual classifiers in the crowd by using crowd-classifiers. Three main categories of ensemble approaches exist:

- **BAGGing (Bootstrap AGGregating)**: The suggestion comes from Leo Breiman [80]. Several learning methods are taken in parallel by BAGGing and fitted individually; the training process occurs concurrently. The Random Forest technique is often used in BAGGing. The process of bootstrapping a given dataset into many sub-samples is shown in figure 1.17. A decision tree is trained using each subsample. Next, using the chosen aggregation technique, the predictions from each decision tree are combined to get the final prediction.

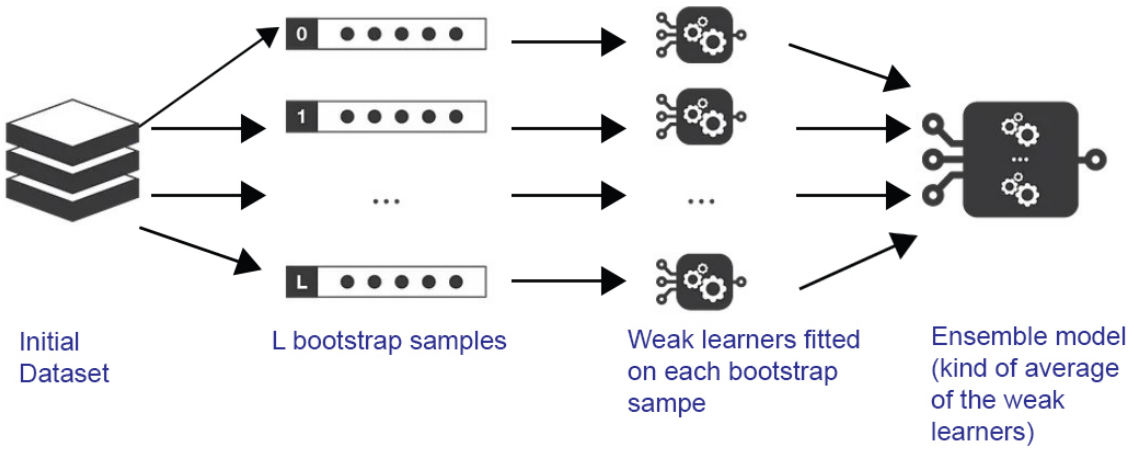

Fig. 1.17 Bagging method

- **Boosting**: Figure 1.18 illustrates how Boosting uses various learning algorithms in a sequential fashion. In contrast to BAGGing, which attempts to lower variance, boosting provides the current learner with dataset observations that the prior learner handled incorrectly. Adaptative boosting and gradient boosting are two common boosting techniques.
- **Stacking**: Stacking is similar to BAGGing in that it takes multiple learners simultaneously. The original dataset will first be divided into two folds, as figure 1.19 illustrates. The first fold will be used to train L weak learners. Subsequently, the skilled students forecast the second fold. Lastly, using the predictions from the previous stage, the second fold is used to train a meta-model. .

A series of studies were done by V. L´opez et al. [76], who summarized many particularities for the three groups mentioned above. While the accuracy of the ensemble approaches is good, their running times might be lengthy. More importantly, consumers typically find it challenging to understand the learned procedures. As accurate as ensemble methods are cost-sensitive approaches. Its use is nonetheless restricted by the need for a pre-defined

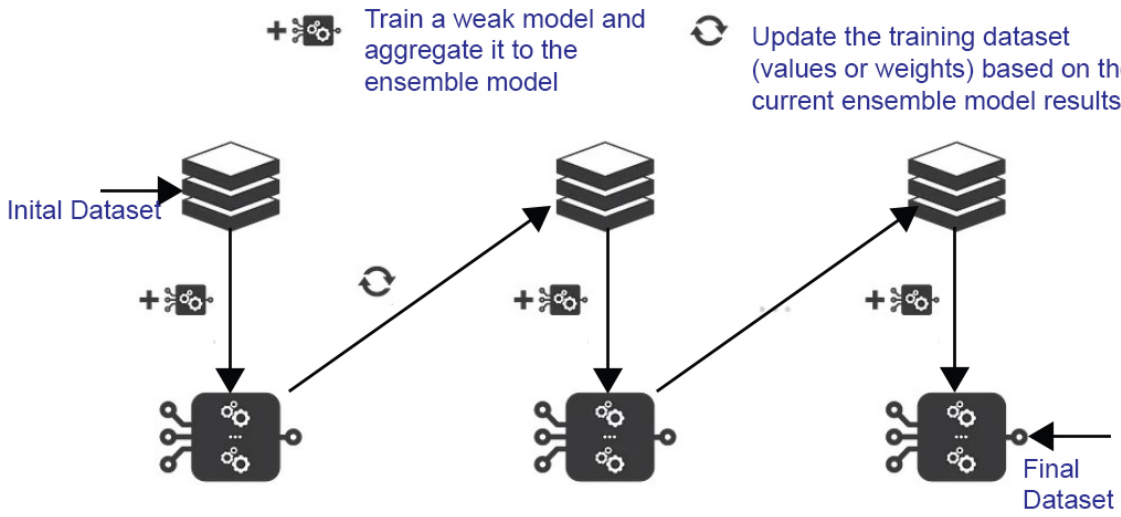

Fig. 1.18 Boosting method

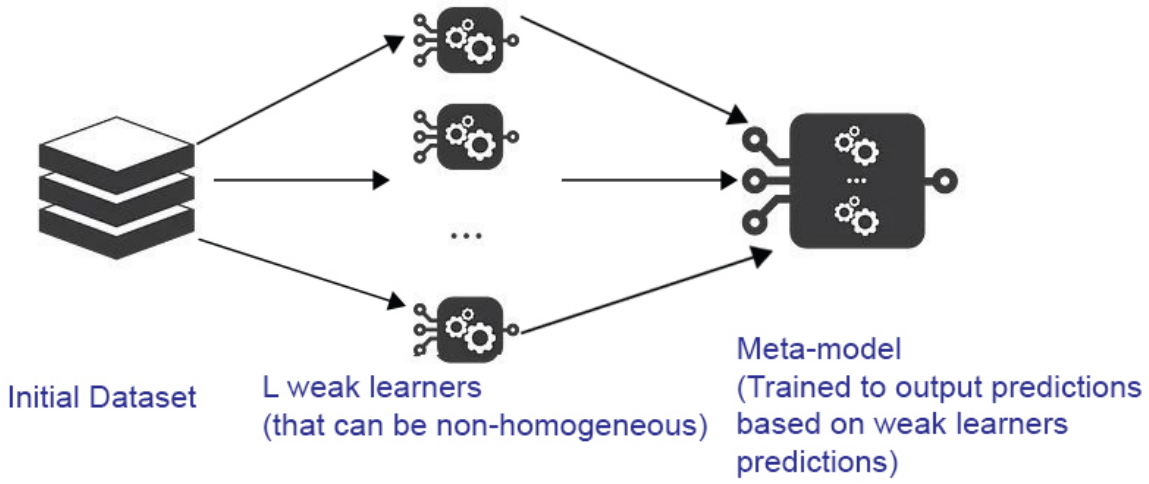

Fig. 1.19 Stacking method

cost-matrix. Finally, resilience and good outcomes are demonstrated by the data resampling procedures. Re-sampling data might be considered the typical method for resolving data imbalance, according to V. L´opez et al. [76].

1.7 Role of ML in Detection of Financial Fraud

In the field of credit card fraud detection, the application of machine learning proves to be highly effective, enabling the development of systems capable of accurately predicting and identifying risky or abnormal behavior within vast datasets [81]. This efficacy arises from the inherent capability of machine learning to discern patterns and make predictions without explicit programming, leveraging the analysis of extensive data sets.

Machine learning techniques encompass various approaches, broadly classified into four main categories. First, labeled data—which includes both input feature and matching output label—is used to train the algorithm in supervised learning. Second, unsupervised learning works with unlabeled data, meaning that the algorithm can find patterns and structures in the dataset even in the absence of output labels that have been predetermined. Thirdly, semi-supervised learning incorporates a combination of labeled and unlabeled training data, combining aspects of supervised and unsupervised learning. Lastly, reinforcement learning involves the interaction of the learning system with its environment, learning through actions and feedback in the form of errors or rewards [81].

When tackling the classification problem inherent in credit card fraud detection, distinguishing between fraudulent and non-fraudulent transactions, the selection of a machine learning approach is contingent upon the characteristics of the dataset. If the training dataset includes labeled instances of fraud, supervised learning techniques are often employed. This involves training a model to recognize patterns associated with fraudulent transactions based on the labeled examples.

Conversely, in situations where the dataset lacks explicit labels for fraudulent transactions, unsupervised learning techniques can be applied. These methods, such as clustering algorithms, identify patterns and anomalies within the data, helping to flag potentially fraudulent activity. Semi-supervised learning strikes a balance between the two approaches, allowing for the utilization of both labeled and unlabeled data. This flexibility is particularly beneficial in cases where obtaining a fully labeled dataset is challenging or expensive.

According to Abdallah et al. [82], one of the most common data mining approaches for building credit card Fraud Detection Systems (FDS) is the utilization of classification models. This methodology aligns with supervised learning techniques, wherein the model is trained on labeled data to classify transactions into fraudulent or non-fraudulent categories.

1.8 Motivation

Introducing a rule-based model for financial fraud detection and proposing the Anomaly Reduction Boundary Based Oversampling (ARBBO) algorithm to tackle imbalanced data offer innovative solutions to common challenges in machine learning, especially in classification tasks. Over the years, machine learning algorithms have gained widespread popularity for handling classification tasks, but the issue of imbalanced datasets poses a significant hurdle to their effective application. Imbalanced datasets, where one class is underrepresented compared to others, impact supervised classification algorithms, particularly when assigning prior probabilities to each class. This challenge is prevalent in real-world applications, including financial fraud detection and medical diagnosis. Often, the minority class, such as rare diseases or instances of fraud, is of paramount importance, and misclassification can incur serious costs.

Traditionally, solutions for imbalanced datasets involve modifying the dataset itself through techniques like undersampling, where instances from classes with more data are removed. While this can help achieve balanced classes, it may hinder the classifier's generalization ability, especially when the original dataset is relatively small. Instead of altering the dataset, a more promising approach involves addressing the challenge as perceived from the classifier's perspective, enabling it to learn data representations effectively despite imbalance. López et al. [76] highlighted the need for future research to emphasize the detection and measurement of significant data properties, suggesting exploration of classification approaches capable of overcoming data imbalance in various aspects, including small disjuncts, lack of density, class overlap, noisy data, accurate management of borderline examples, and dataset shifts.

Motivated by this, we introduce the ARBBO algorithm as a binary classification procedure designed to handle data imbalance at the algorithm design level. Our approach aims to outperform other classifiers by addressing the nuanced aspects of data imbalance. Through the results obtained in this thesis, ARBBO demonstrates superior performance, providing a more effective solution for classification tasks when dealing with imbalanced data.

Moreover, we introduce an additional consecutive sequence-based rule-based model for financial fraud detection. This model is specifically crafted to bolster both the efficiency in terms of time and the accuracy of fraud detection. By leveraging sequential patterns and rules, this model aims to capture intricate relationships within financial transactions, providing a complementary approach to existing methods.

1.9 Objective

In addressing the challenge of class imbalance in machine learning datasets, we present a novel resampling method known as ARBBO (Anomaly Reduction Boundary Based Oversampling). The primary objective of ARBBO is to mitigate the impact of imbalanced class distributions by oversampling the minority class instances near the decision boundary. This method aims to enhance the model's ability to accurately classify rare or anomalous events, which is particularly crucial in scenarios where certain classes are underrepresented. By strategically oversampling instances close to the decision boundary, ARBBO seeks to improve the model's discriminatory power and reduce the likelihood of misclassifying minority class samples.

Additionally, we propose the development of a rule-based machine learning model specifically tailored for detecting financial fraud. The objective of this model is to leverage machine learning algorithms in conjunction with predefined rules to identify patterns indicative of fraudulent activities within financial datasets. By combining the flexibility of machine learning with the interpretability of rule-based systems, our approach aims to enhance fraud detection capabilities in the financial domain. This hybrid model is designed to capture intricate patterns and anomalies associated with fraudulent transactions, providing a comprehensive and effective solution for mitigating financial risks.

Our research addresses two key objectives in the realm of machine learning:

1. We propose a resampling method named ARBBO to tackle imbalance issue.
2. We propose a machine learning rule-based model to detect financial fraud.

1.10 Scope of Work

The scope of work for Rule-Based machine learning and Anomaly Reduction Boundary Based Oversampling (ARBBO) in fraud detection and machine learning is multifaceted and encompasses several critical elements.

Firstly, within the realm of Rule-Based machine learning, the primary objective is to develop explicit rules that guide decision-making processes, thereby fostering transparent and interpretable models. This involves a meticulous process of identifying, formulating, and validating rules derived from both domain knowledge and data analysis. By leveraging insights from experts in the field and analyzing historical data, Rule-Based systems aim to encode decision logic in a manner that is easily understandable and auditable.

On the other hand, the focus of ARBBO oversampling techniques is to tackle the challenge of class imbalance prevalent in fraud detection tasks. By generating synthetic samples

strategically around the boundaries of the minority class, ARBBO aims to enhance the robustness of fraud detection models. This approach is geared towards creating a more balanced training set, thereby enabling machine learning algorithms to better discern between fraudulent and legitimate transactions.

The scope of implementing ARBBO algorithms extends beyond mere application to include fine-tuning and optimization. This involves tailoring the oversampling process to suit the specific characteristics of the dataset and the nuances of the fraud detection problem at hand. Furthermore, the efficacy of ARBBO techniques is evaluated comprehensively, with a focus on metrics such as precision, recall, and overall predictive accuracy. Through rigorous experimentation and validation, the goal is to ascertain the effectiveness of ARBBO in improving model performance compared to conventional methods.

Moreover, this work involves thorough comparison with existing methodologies to determine the superiority and practicality of the proposed approach. By conducting experiments across diverse datasets and operational contexts, researchers and practitioners seek to assess the generalizability and robustness of ARBBO in mitigating fraud risks effectively.

Finally, the scope of work for Rule-Based machine learning and ARBBO in fraud detection encompasses the development, implementation, fine-tuning, evaluation, and comparison of methodologies aimed at enhancing the transparency, interpretability, and performance of fraud detection systems. Through a combination of domain expertise, data-driven insights, and innovative techniques, the objective is to build more reliable and effective defenses against fraudulent activities across various domains and industries.

1.11 Research Questions

In our research endeavor, we aim to address two pivotal questions at the intersection of data science and financial analysis. Firstly, we delve into the intricate challenge of balancing imbalanced data prior to rule generation, a critical step in ensuring the robustness and reliability of our subsequent analyses. Imbalanced data poses a significant hurdle, potentially skewing the outcomes and impeding the efficacy of rule generation algorithms. Our investigation seeks to explore innovative methodologies and techniques to mitigate this imbalance, enhancing the quality and representativeness of our dataset. Secondly, we delve into the realm of generating data-driven rules from financial data, recognizing the importance of extracting actionable insights from vast and complex datasets. By leveraging advanced machine learning algorithms and domain-specific knowledge, we endeavor to develop a framework that effectively captures patterns and trends inherent in financial data, facilitating the formulation of robust rules that can inform decision-making processes in various financial

contexts. Through rigorous experimentation and analysis, we aim to contribute valuable insights to both the academic community and practitioners in the financial domain, ultimately fostering greater efficiency and accuracy in decision-making processes.

Based on the preceding discourse, we can outline our research questions as follows:

1. How to balance imbalance data before rule generation?
2. How to effectively generate data-driven rules from financial data?

1.12 Contribution of This Thesis

This thesis makes notable contributions in the realm of financial fraud detection from two distinct perspectives.

Firstly, it introduces a data-driven rule-based model tailored for the identification of financial fraud. This model leverages a data-driven approach to generate rules that encapsulate distinctive patterns associated with fraudulent activities within financial datasets. The adaptability, real-time detection capabilities, and interpretability of the system make it a valuable tool for effectively discerning and responding to potential fraudulent transactions.

Secondly, the thesis introduces the Anomaly Reduction Boundary Based Oversampling (ARBBO) method as a solution to the prevalent problem of imbalance in financial datasets. ARBBO, designed as a binary classification procedure, strategically utilizes oversampling techniques to generate synthetic instances of the minority class, effectively balancing the class distribution. Unlike traditional methods, ARBBO enhances the classifier's generalization ability by allowing it to learn effective data representations in the presence of imbalanced data. Furthermore, the method comprehensively addresses various aspects of data imbalance, as suggested by López et al. [76], ensuring improved performance across multiple imbalance-related challenges. The results obtained from the thesis demonstrate the superior performance of ARBBO compared to other classifiers, affirming its efficacy in handling imbalanced financial datasets. Together, these contributions propel the field of financial fraud detection forward by introducing practical and effective methodologies that bolster interpretability, real-time detection, and overall performance.

1.13 Thesis Organization

The subsequent sections of this thesis report are meticulously organized to provide a comprehensive exploration of the research work conducted:

- **Literature Review (Chapter 2):** In Chapter 2, an exhaustive literature review is undertaken. This review encompasses a broad range of relevant studies, providing a comprehensive overview of existing research and insights related to the thesis topic. The chapter critically examines prior work in the field, identifying key theories, methodologies, findings, and gaps in knowledge. Through this rigorous analysis, the chapter aims to establish a solid theoretical framework and inform the subsequent research methodology and analysis, examining similar research endeavors that have been implemented and tested in the realm of financial fraud detection. A meticulous distinction between our work and existing studies is meticulously presented, showcasing the unique contributions of our approach. Additionally, relevant theories that underpin our methodology are outlined, providing a theoretical foundation for the subsequent chapters.

- **Proposed Methodology (Chapter 3):** Chapter 3 delves into the heart of our research, elucidating the proposed methodology for financial fraud detection. The chapter meticulously details the approach, algorithms, and strategies employed in our data-driven rule-based model and the Anomaly Reduction Boundary Based Oversampling (ARBBO) method. This section serves as a crucial guide for understanding the intricacies of our innovative contributions.

- **Experimental Results (Chapter 4):** Chapter 4 presents the empirical validation of our proposed methodology using the paysim dataset. This section encompasses a thorough analysis of accuracy measurements, providing insights into the effectiveness of our approach. Furthermore, a comparative study is conducted, juxtaposing our results with existing work that shares similar objectives. The outcomes of the experiment verify the strength and excellence of our contributions.

- **Conclusion and Future Directions (Chapter 5):** The final chapter, Chapter 5, serves as the conclusion of this thesis. Key findings are summarized in the concluding remarks, which also highlight the importance of our contributions to the field of financial fraud detection. Limitations encountered during the study are candidly acknowledged, and potential avenues for future research are suggested. This chapter not only encapsulates the culmination of our work but also provides a springboard for future investigations.

1.14 Conclusion

This chapter serves as a concise introduction to the background and objectives of the thesis. It outlines the scope and significance of the research, providing clarity on its focus and

importance. The upcoming chapter will delve into a comprehensive review of relevant literature, laying the foundation for the study's theoretical framework and methodology.

Chapter 2

Related Work

2.1 Existing Oversampling Methods

The severe disparity between minority and majority class membership leads to biases in classifier decision-making, which is the fundamental issue with unbalanced learning. The bulk of minority instances in this scenario could be simply categorized into the majority group, making minority instance recognition challenging. Numerous assessment indicators indicate that classifiers are not performing well, with a low detection rate of minority classes. This does not exclude classifiers from learning from datasets that are unbalanced, though. In order to achieve a more balanced distribution, the dataset is modified through a few different ways when sampling techniques are applied to unbalanced learning. While both oversampling and undersampling are preprocessing steps, this work focuses solely on oversampling. We will cover a number of well-known sampling techniques in this chapter, including adaptive synthetic sampling technique (ADASYN), synthetic minority oversampling technique (SMOTE), and random replacement oversampling.

2.1.1 Random Oversampling with Replacement

Random oversampling with replacement is the process of adding a set E sampled from the minority class using the subsequent mechanism: Replicate the cases and add them to S to enlarge the initial set S, given a set of randomly selected minority occurrences in Smin [83]. By doing this, the class distribution becomes more balanced because the size of all examples in Smin will increase by |E|. Just a subset of the minority class is replicated using this curious strategy in order to raise the weights of those cases. Due to the complete randomness of the replacement process, there is no distinct demarcation between the two classes in this manner. This approach's primary drawback is that it duplicates some of the cases that were already

present in the initial minority class, which could result in overfitting [84]. The oversampling technique's core idea is this procedure. Based on this technique, numerous more widely used oversampling algorithms for practical applications have been created.

2.1.2 Synthetic Minority Oversampling Technique (SMOTE)

In 2002 [39], Chawla introduced the Synthetic Minority Oversampling Technique (SMOTE) in equation 2.1. The SMOTE algorithm oversamples the minority class by generating synthetic cases, as opposed to randomly oversampling, which results in oversampling by replacement. The SMOTE algorithm creates artificial instances [83] by employing feature space similarities between current minority examples rather than data space similarities. By joining some or all of the K nearest neighbors of the minority class with line segments, these fictitious instances are produced. Based on the intended amount of oversampling, neighbors are randomly chosen from the K nearest neighbors. Specifically, let $S_{min} \in$ S represent the minority class. Determine the K-nearest neighbors for each example $x_i \in S_{min}$, given a specified K. The K-nearest neighbors are those K elements of S_{min} whose euclidian distance to x_i in feature space X has the least magnitude. To begin a new sample, randomly select one of the K-nearest neighbors, and then determine the difference between the selected sample and its closest neighbor. Multiply this difference by a value between 0 and 1 that is consistently generated. This factor can be changed by moving from a uniform distribution to a different distribution, depending on the application. Lastly, incorporate this vector into the chosen sample x_i.

$$x_{syn} = x_i + (x_i' - x_i) \times \lambda \tag{2.1}$$

where $\lambda \in [0, 1]$ is a randomly generated number, $x_i \in S_{min}$ is the selected instance from the minority class, and $x_i \in S_{min}$ is one of x_i's K-nearest neighbors. A SMOTE method example is shown in Figure 2.1. K = 4 is the number of K closest neighbors. The red line section between x_i and x_i' is where the synthetic examples are formed, as seen in the picture. The original class distribution is more evenly distributed thanks to these artificial examples, which generally greatly enhances learning. However, there are a few issues with the SMOTE method, such as an oversimplification of the minority class space. Other SMOTE-based algorithms have been developed from the original SMOTE technique over time, and some of them have been demonstrated to significantly improve uneven learning performance.

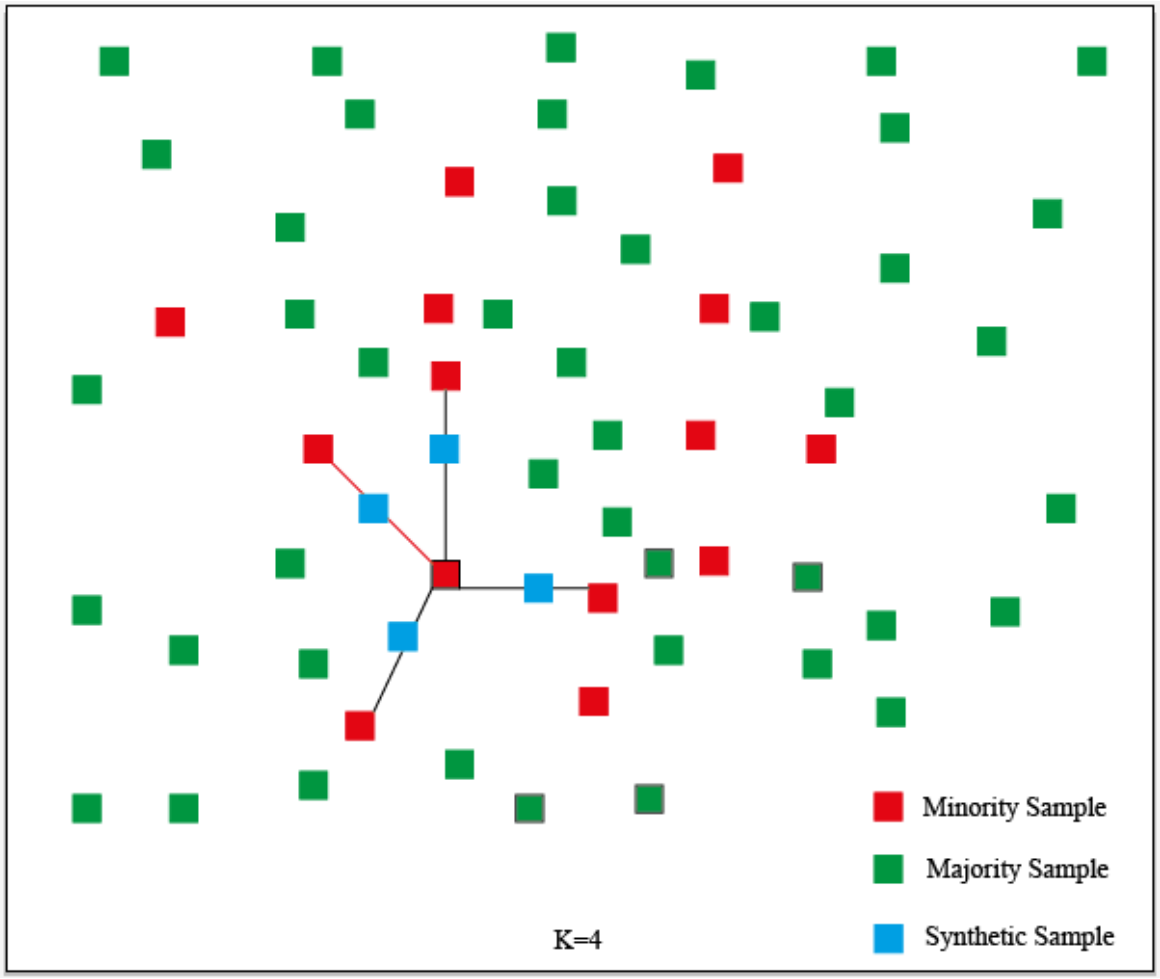

Fig. 2.1 Example of synthetic data generation using SMOTE

2.1.3 Adaptive Synthetic Sampling Approach (ADASYN)

Later, a number of oversampling techniques, including SMOTEBoost, Borderline-SMOTE [40], and Adaptive Synthetic Sampling (ADASYN), were created based on the concept of SMOTE. These techniques have all demonstrated improvements in unbalanced learning on diverse datasets. The methods utilized to discover minority samples with these adaptive algorithms are very interesting. Let's examine the Borderline-SMOTE instance first, as it offers the fundamental concept for the construction of ADASYN, before moving on to ADASYN.

The following is the result of this algorithm. The quantity of closest neighbors who are members of the majority class, or $|S_{i:m} \cap S_{\text{maj}}|$, is determined by first determining the set of nearest neighbors for each $x_i \in S_{\text{min}}$, denoted as $S_{i:m}$ (2.2), where m is the number of minority class. After that, it divides each xi into three categories: "Danger," "Safe," and

"Noise" according to how many majority examples are present in its K-nearest neighbors. x_i is included in "Danger" if:

$$\frac{m}{2} \leq |S_{1:m} \cap S_{maj}| < m \tag{2.2}$$

These instances serve as representations of the marginalized class. x_i is classified as "Noise" if | $S_{1:m} \cap S_{maj}$ | = m, meaning that all K of its nearest neighbors are majority examples; otherwise, it is classified as "Safe." In contrast to SMOTE, Borderline-SMOTE only generates synthetic examples for cases that are close to the boundary; for "Noise" instances, no synthetic examples should be generated. ADASYN, on the other hand, adjusts the Borderline-SMOTE idea and produces different amounts of synthetic instances for minority classes according to their distribution. The number of a minority example's majority nearest neighbors determines how many synthetic examples of that example must be created, according to the ADASYN algorithm [41]. There will be more synthetic examples produced the more majority nearest neighbor there is. More specifically, the total number of synthetic data samples, G (2.3), is determined by the parameter $\beta \in [0, 1]$, which is used to designate the balanced level after the synthetic procedure.

$$G = (|S_m aj| - |S_m in|) \times \beta \tag{2.3}$$

Next, determine each example's K-nearest neighbors ($x_i \in S_{min}$). For a synthetic process, Define the density distribution as the weight of x_i. (2.4) Γ_i.

$$\Gamma_i = \frac{\frac{\Delta_i}{K}}{Z}, i = 1, ..., |S_{min}| \tag{2.4}$$

where Z is a normalized constant, Γ_i is a probability mass function; that is, $\sum \Gamma_i$ = 1; Δ_i is the number of majority occurrences among the K-nearest neighbors of x_i. Then, determine how many synthetic examples (g_i) are needed for every $x_i \in S_{min}$. (2.5).

$$g_i = \Gamma_i \times G \tag{2.5}$$

Ultimately, g_i synthetic examples are produced for each $x_i \in S_{\min}$ utilizing the SMOTE algorithm. based on the idea of the SMOTE algorithm, ADASYN was created. The distinction is that SMOTE gives every minority example an equal chance of being chosen for the synthetic process, whereas ADASYN determines the number of synthetic cases based on the density distribution Υ. Because ADASYN, in contrast to BorderlineSMOTE, does not identify "Noise" situations, a lot of bogus data can be created around those cases, thereby giving learners an inflated minority space.

2.1.4 Other Oversampling Techniques

A number of alterations based on the idea of the original SMOTE algorithm have been developed in addition to the ADASYN algorithm. Chawla [39] later proposed SMOTE-NC (Synthetic Minority Oversampling Technique Nominal Continuous) and SMOTE-N (Synthetic Minority Oversampling Technique Nominal), which are upgraded versions of SMOTE handling datasets with nominal characteristics. Furthermore, SMOTEBoost has been proposed [85], which adjusts the updating weights and skewness compensates. The F-measure indicates a noteworthy improvement in its performance. H. Han [40] suggested two new oversampling techniques: borderline-SMOTE1 and borderline-SMOTE2. These two strategies significantly improve the difference between the two classes by selectively oversampling only the minority samples along the border. Their research indicates that these two algorithms perform better in terms of F-value and TP rate than SMOTE. Borderline oversampling has been widely used to categorize unbalanced data [86]. Usually, the result of data extraction from medical images in data categorization was geometric complexity. Moreover, they are not linearly separable in Euclidean space. To solve this problem, Juanjuan Wang et al. have created a unique method that combines the locally linear embedding algorithm (LLE) with the SMOTE algorithm[14]. In order to enable SMOTE to oversample the data, this approach maps the high-dimensional data into a low-dimensional space. Data obtained from medical imaging was usually the result of geometric complexity in data classification. In addition, they cannot be separated linearly in Euclidean space. By merging the locally linear embedding algorithm, or LLE algorithm, with the SMOTE algorithm, Juanjuan Wang et al. have created a novel way to deal with this problem [87]. SMOTE allows for oversampling of the data by projecting the high-dimensional data onto a low-dimensional environment.

2.1.5 Machine Learning Based Approach

Financial fraud is found through a variety of ways. As a result, the primary methodologies can be divided into groups, like deep learning (DL) and machine learning (ML), ensemble and feature ranking, and user authentication approache. In [88], According to the authors, Thanks to developments in e-commerce and communication technologies, credit card use as a means of payment has expanded, and transaction-related fraud is also on the rise. They employed a better light gradient booster device (LightGBM), which combines Bayesian-based hyper-parameter optimization with LightGBM parameter tuning. The suggested system yields impressive performance metrics, with an accuracy of 98.40%, precision of 92.88%, and an area under the receiver operating characteristic curve (AUC) of 97.34%. Additionally,

the AUC stands at 56.95In [89], According to the writers, major financial losses are caused by credit card theft. Based on factors including sensitivity, area under the precision-recall curve (AUCPR), and accuracy, they have determined that the top algorithms are ANN, SVM, C5.0 decision tree approach, and linear regression (LR). Randhawa and colleagues [**?**] presented a credit card fraud detection engine that uses these two strategies. The authors of this study used the European Cardholders dataset. The authors also considered the AdaBoost methodology in conjunction with machine learning methods such as Support Vector Machine (SVM). The results demonstrated that the AdaBoost-SVM achieved an MCC of 0.044 and an accuracy of 99.959%. Riffi et al. [90] used the Multilayer Perceptron (MLP) and Extreme Learning Machine (ELM) algorithms to construct a credit card fraud detection engine. The European Cardholders dataset, created in 2013, was used by the study's authors. The results demonstrated the 97.84% accuracy rate of the MLP technique. By contrast, the ELM's accuracy rate in identifying credit card fraud was 95.46%. In Jiang et al. [91], the cardholders' transactions are first collected, after which they are aggregated based on their behavior, the dataset is classified, the model is trained and tested, and lastly the transactions are collected again. The approach presented by Xuan et al. [53] combined random forest models based on random trees and random forest models based on CARTs.

Their research made use of historical transaction data and was based on the behavioral traits of both genuine and fraudulent transactions. The dataset of a Chinese e-commerce corporation includes over 30,000,000 transactions, 82,000 of which are fraudulent occurrences, and 62 attributes. A 5:1 normal-to-abnormal transaction ratio was used. The results of the study revealed a 98.67% recall rate, a 32.68% precision rate, and an accuracy rate of 98.67%. Large amounts of data can be analyzed using logistic regression and the stacked auto-encoders approach, however the findings are imprecise and not useful for practical purposes. A study by A.A.E. Naby et al. [92] provided evidence in favor of this. Awoyemi et al.'s [55] detection of fraudulent credit activity using naive Bayes, K-Nearest Neighbor, and logistic regression with accuracy rates of 97.92%, 97.69%, and 54.86%, respectively. Taha and Malebary [88] proposed an enhanced light gradient boosting machine (LightGBM) to detect fraud in credit card transactions. To optimize the LightGBM's hyperparameters, a Bayesian-based approach must be used. They compared a number of classifiers, such as naive Bayes, SVM, decision trees, random forests, and categorical boosting (CatBoost). The updated LightGBM surpassed the benchmarked classifiers with an accuracy of 98.40%, precision of 97.34%, and area under the receiver operating characteristic curve (AUC) of 92.88%.

For credit card fraud detection, Ileberi et al. [93] proposed a feature selection model based on genetic algorithms (GAs). The study showed that the GA-based feature selection

enhanced the performance of the various ML classifiers, with the random forest obtaining the highest accuracy of 99.98%. Furthermore, in order to build trustworthy credit card fraud detection (CCFD) models, Salekshahrezaee et al. [94] examined the effects of feature extraction and data resampling on the following machine learning classifiers: CatBoost, Random Forest, Extreme Gradient Boosting (XGBoost), and LightGBM. The features were extracted using principal component analysis (PCA) and convolutional autoencoder (CAE), and the data was resampled using SMOTE, random undersampling (RUS), and SMOTE Tomek methods. The experimental findings demonstrated that the classifiers' performance was greatly enhanced when feature extraction and resampling were done using the RUS and CAE techniques, respectively. Additionally, for multidimensional and partial rectification analysis, extensive feature selection, financial fraud detection, and importance grading of variables, the authors of [95] recommended utilizing a random forest model. Different machine learning algorithms were built by Khatri et al. [96] in order to detect credit card fraud. In their investigation, the authors employed the following methodologies: Random Forest (RF), Decision Tree (DT), k-Nearest Neighbor (kNN), Logistic Regression (LR), and Naive Bayes (NB). The results showed that the kNN method yielded the best results, with 91.11% precision and 81.19% sensitivity, respectively. Rajora et al. conducted a comparative study on machine learning methods for credit card fraud detection [97].

Among the methods that were examined were the RF and KNN approaches. The results demonstrated the accuracy of 94.9% and AUC of 0.94 of the RF algorithm. In comparison, the accuracy and AUC of the KNN were 93.2% and 0.93, respectively. While the results are promising, this study did not address the issue of class imbalance that exists in the dataset used. Trivedi et al. [98] presented an efficient credit card fraud detection engine using ML techniques. The authors tested various methods using the European Cardholders dataset. The tests' outcomes showed that the GB had 94.01% accuracy and 93.99% precision, respectively. In contrast, the RF achieved precision and accuracy of 95.98% and 94.00%, respectively. Tanouz et al. [99] provided a framework for ML-based credit card fraud detection. Additionally, in order to address the issue of class imbalance present in the dataset utilized, the authors employed an undersampling method. In this study, two machine learning techniques are considered: LR and RF. Accuracy was the main performance indicator that the researchers used.

The results demonstrated the RF technique's 91.24% fraud detection accuracy rate. In comparison, the accuracy of the LR approach was 95.16%. The results showed that more investigation is required to resolve the class imbalance problem in the European credit cardholder dataset. Riffi et al. [90] used the Multilayer Perceptron (MLP) and Extreme Learning Machine (ELM) algorithms to construct a credit card fraud detection engine. Although

they are both artificial neural networks (ANNs), the ELM and the MLP have different basic designs. The principal performance metric utilized by the authors was the accuracy of fraud detection. The results demonstrated the 97.84% accuracy rate of the MLP technique. By contrast, the ELM's accuracy rate in identifying credit card fraud was 95.46%. Although the ELM is less sophisticated than the MLP, this investigation found that the MLP performed better than the ELM.

2.1.6 Deep Learning Based Approach

Using the synthetic minority oversampling-edited closest neighbor approach and AdaBoost as the base learner, E. Esenogho et al. [100] created a complicated ensemble of LSTM. The authors used a dataset of European credit card users' transactions over two days in September 2013 for their studies. The final result has a sensitivity of 0.996, specificity of 0.998, and AUC of 0.990, outperforming more well-known algorithms like SVM, MLP, decision trees, just LSTM, and AdaBoost. Alghofaili et al. [101] examined the use of long short-term memory in 2020 to detect credit card fraud by comparing it to an auto-encoder and traditional machine learning models, such as logistic regression, random forest, and support vector machines. Ibtissam Benchaji et al. [102] performed another LSTM analysis using a local bank-provided dataset of 594,643 transactions from November 2012 to April 2013. It was certain that the card had been used by the cardholder if the data fit the pattern; otherwise, there was a significant risk of fraud.

They intend to create another Recurrent Neural Networks variant-based model in the future to test its competency in comparison to the existing model. To anticipate fraud, BiLSTM-MaxPooling-BiGRU-MaxPooling was utilized by Najadat et al. [103]. The authors also employed a naive base, voting, AdaBoost, random forests, decision trees, and logistic regression to compare the results of each model. The extremely uneven class of the dataset utilized in this study makes it unique. To resolve the imbalance in the dataset, the authors used the random under-sampling, random over-sampling, and synthetic minority over-sampling procedures. The outcomes demonstrated the models of deep learning using the three sample methods outperformed machine learning models in terms of accuracy by a wide margin. The machine learning-based models had an accuracy of 81% as their maximum value. To obtain 91.37% accuracy, The authors used a random oversampling technique to combine BiLMST-MaxPooling with BiGRU-MaxPooling.

In a different study, Mubalaike and Adali [104] investigated the use of deep learning to detect fraud with a high degree of accuracy. With accuracy of 90.49%, 80.52%, and 91.53%, respectively, they integrated restricted Boltzmann machines (RBM), stacked auto-encoders (SAE), and decision tree models (EDT). . In comparison to the EDT and RBM, the SAE

was lower. I. D. Mienye et al. [105] had proposed a robust solution that incorporated deep learning, utilizing an MLP as a meta-learner and LSTM and GRU neural networks as basic models. In addition, the class imbalance in the dataset had been addressed using the SMOTE-ENN approach. Experimental results had demonstrated superior performance, attaining a specificity of 0.997 and a sensitivity of 1.000 in comparison to other widely used machine learning techniques in the literature. To accomplish sequential modeling To accomplish sequential modeling and ensure enhanced fraud detection, Benchaji et al. [102] suggested a technique for detecting credit card fraud using an LSTM network. In order to find the most pertinent qualities, Combining the uniform manifold approximation and projection (UMAP) with the LSTM network method and an attention mechanism. The testing outcomes showed that the suggested method was reliable for spotting credit card fraud.

We can see from the aforementioned literature evaluations that no researchers employ Rule-Based models for identifying financial fraud. Therefore, We provide a live binary classification technique that utilizes the combination of Rule-Based models and anomaly reduction boundary based oversampling (ARBBO) models in order to close the gap between the two categories of models used today for fraud detection, including approaches based on deep learning and machine learning.

2.2 Research Challenges

Developing Fraud Detection Systems (FDSs) employing Dynamic Data Models (DDMs) based on Machine Learning algorithms poses significant challenges, including:

1. Fraudulent transactions represent a minority portion of daily transactions [106].
2. The distribution of fraudulent activities changes over time, influenced by seasonality and the emergence of new attack strategies [107].
3. In many cases, the true class of the majority of transactions is only determined several days after the transaction occurs, as investigators promptly examine only a limited number of transactions [108].

The initial obstacle, commonly referred to as the unbalanced problem (citation), emerges due to the skewed distribution of transactions favoring the genuine class. Not only do the distributions of genuine and fraud samples lack equilibrium, but they also display overlap. Many Machine Learning algorithms are inadequately equipped to address both unbalanced and overlapping class distributions (citation). Concept drift, resulting from changes in fraudulent activities and customer behavior, introduces non-stationarity in transaction streams [109].

FDSs must be continually updated, leveraging recent supervised samples and discarding outdated information to remain effective without being misled. Strategies failing to adapt or revisit frequently tend to lose predictive accuracy over time [106].

The third challenge pertains to the impracticality of scrutinizing all transactions in real-world scenarios due to human labor constraints. These limitations restrict the number of alerts generated by the Fraud Detection System (FDS) that investigators can verify. Investigators evaluate FDS alerts by reaching out to cardholders, gathering feedback to determine whether the alerts are associated with fraudulent or legitimate transactions. These feedback responses, received for only a small portion of daily transactions, provide real-time information for training or updating classifiers. The classification (fraudulent/non-fraudulent) for the remaining transactions typically remains unknown until several days later, often being automatically assigned based on a predefined timeframe for customers to identify and report frauds. Traditional FDSs, which do not incorporate feedback from investigators, tend to produce less accurate alerts compared to those effectively utilizing both feedback and other available supervised samples [108].

2.3 Problem Statement

The problem statement at the core of the contributions from the data-driven rule-based model and the ARBBO method revolves around the intricacies of financial fraud detection. A predominant challenge lies in the highly imbalanced nature of the dataset, where instances of fraud constitute a minority compared to non-fraudulent transactions. Addressing this imbalance is crucial for effective machine learning algorithms, ensuring accurate identification of fraudulent activities. Real-time processing emerges as a critical requirement, emphasizing the need for swift detection and response to potential fraud in the dynamic landscape of financial transactions. Minimizing false positives, a key concern, underscores the delicate balance required between high sensitivity to detect fraud and low rates of falsely flagging legitimate transactions. Additionally, model explainability takes precedence, aiming to provide transparency in decision-making processes, fostering trust among users and meeting regulatory compliance. Notably, the problem statement recognizes the perpetual adaptation of fraudsters, necessitating models that can dynamically counter evolving tactics. The proposed data-driven rule-based model prioritizes interpretability, while the ARBBO method, designed at the algorithmic level, enhances the classifier's generalization ability, making it more resilient to emerging fraud patterns. Together, these contributions address the multifaceted challenges inherent in financial fraud detection, providing innovative solutions to advance the field.

In the context of balancing imbalanced data before rule generation, the highly skewed distribution between fraudulent and non-fraudulent transactions poses a significant challenge. Techniques such as oversampling of the minority class, undersampling of the majority class, or more sophisticated methods like the ARBBO algorithm are utilized to address this issue. These methods aim to create a more balanced dataset, ensuring that the rule generation process is not biased towards the majority class and effectively captures patterns from both classes.

Regarding the effective generation of data-driven rules from financial data, it involves extracting meaningful patterns and relationships from the dataset to construct rules that accurately distinguish between fraudulent and non-fraudulent transactions. This process often entails feature engineering, where relevant attributes of the transactions are identified and transformed to facilitate rule generation. Additionally, various rule induction algorithms such as decision trees, association rule mining, or sequential pattern mining are employed to automatically generate rules from the data. The resulting rules are then evaluated based on their interpretability, predictive performance, and ability to capture fraud patterns accurately.

Chapter 3

Proposed Methodology

The training dataset D comprises N transactions, denoted as $T = [T_1, T_2, T_3, ...T_N]$, with each transaction defined by a set of attributes $A = [A_1, A_2, A_3...A_m]$. Each attribute in D has a predetermined limit. For instance, if we consider feature A, a limit L is established for A values. This limit serves as a condition, such that if a value is less than L or greater than L, the class value assigned to the transaction is either 0 or 1. Here, 0 represents a non-fraudulent transaction, while 1 signifies a fraudulent one. The process involves extracting relational rules from an imbalanced financial dataset to identify fraudulent transactions. The proposed Rule-Based approach, depicted in Figure 3.2, incorporates an Anomaly Reduction Boundary Based Oversampling (ARBBO). To detect fraud using the rule based model with ARBBO resampling technique from dataset, the following steps are followed:

The dataset is clustered based on transaction types, including CASH_IN, CASH_OUT, TRANSFER, DEBIT, and PAYMENT. In the paysim dataset, for instance, these five transaction types result in five distinct clusters for subsequent rule generation.

3.1 Feature Selection

Feature selection is crucial for removing irrelevant data from large datasets, improving model efficiency. This study proposes a dynamic feature selection strategy that iteratively traverses the dataset to assess feature importance and automatically filters out less significant features. This approach allows the model to retain only relevant and important features, enhancing its accuracy and efficiency. Among the many feature selection methods used are Random Forest and Decision Tree. First, Random Forest selects the top 80% of important features, followed by Decision Tree refinement for optimal results. An automated loop evaluates feature importance and eliminates less significant ones without user intervention, effectively reducing the number of features from 11 to 9.

3.2 Cluster

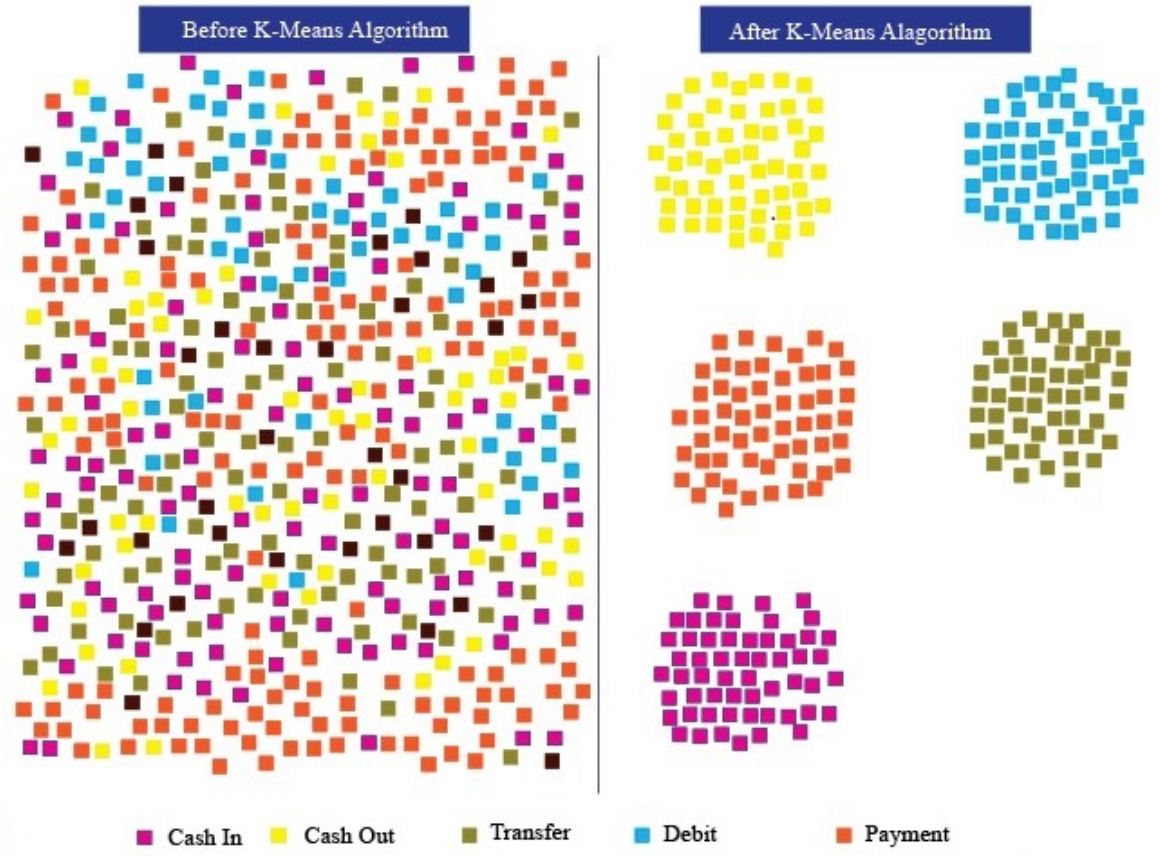

Fig. 3.1 Details Schematic Diagram of Proposed Model

The dataset undergoes a partitioning process utilizing the K-means clustering algorithm, whereby it is divided into distinct clusters based on transaction types. Each cluster corresponds to a specific transaction category, such as CASH_IN, CASH_OUT, TRANSFER, DEBIT, and PAYMENT. This segmentation facilitates a more focused analysis and rule generation for each transaction type.

To illustrate, consider the paysim dataset, which encompasses diverse financial transactions, each falling into one of five categories: CASH_IN, CASH_OUT, TRANSFER, DEBIT, and PAYMENT. The K-means clustering algorithm is applied to this dataset, resulting in the creation of five distinct clusters, each representing transactions of a particular type.

For instance, one cluster may predominantly comprise CASH_IN transactions, while another focuses on CASH_OUT transactions. This segregation enables a more nuanced exploration of patterns and behaviors specific to each transaction type, as the algorithm groups together transactions with similar characteristics.

By dividing the dataset into these clusters 3.1, subsequent rule generation processes can be tailored to the unique attributes and dynamics of each transaction category. The model's capacity to recognize patterns and relationships unique to different transaction types is improved by this focused approach, thereby contributing to a more refined and effective rule set for fraud detection or other analytical purposes. Figure 3.2 displays a detailed schematic diagram of the suggested model and Algorithm 1 shows the procedure of the proposed model.

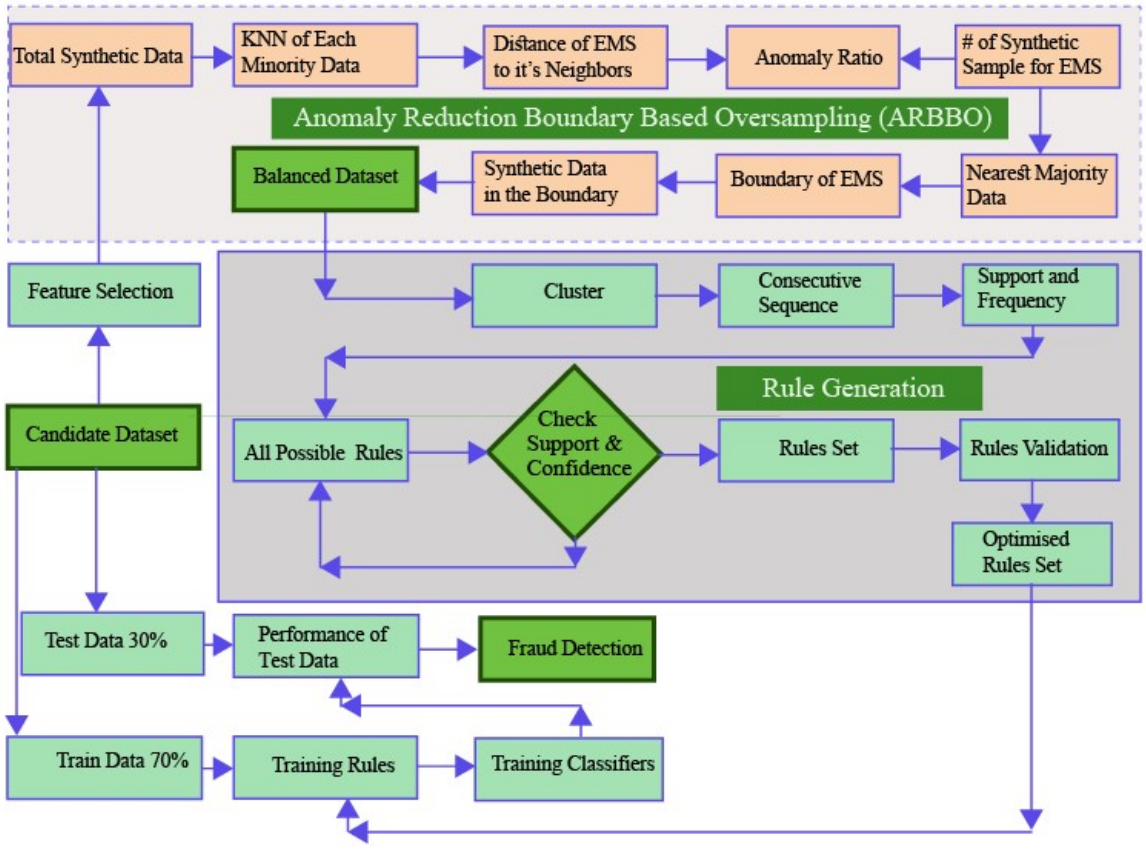

Fig. 3.2 Details Schematic Diagram of Proposed Model

3.3 Anomaly Reduction Boundary-Based Oversampling (ARBBO)

Within this segment, we outline the steps of the ARBBO method, which comprises three stages: Anomaly ratio calculation, safe boundary computation, and synthetic data generation. Addressing class imbalance in data, the ARBBO approach involves oversampling minority

Algorithm 1: Proposed model Algorithm

Data: Input Dataset DS, a dataset containing n financial transactions
Result: Fraud and non-fraud transactions
if $DS.dataTaype \neq numeric$ **then**
  convert data into numeric;
if $featureImportance \leq minimumThreshold$ **then**
  eliminate feature;
//Anomaly Reduction Boundary Based Oversampling(ARBBO) begins here;
while *i in transactions* **do**
  if $isFraud==1$ **then**
    if $type==Cluster[key]$ **then**
      Cluster[key].append(i)

$ts_{synthetic} \leftarrow (N_{maj} - N_{min}) \times \theta$; where θ is the balance level. **while** x_i *in* N_{mi} **do**
  $K_{x_i} \leftarrow Find\ the\ \mathbf{k}\ neighbors\ of\ x_i$
  $td_{x_i} = \sum_{x_j \in N_{min} \wedge x_j \in K_{x_i}} d(x_i, x_j)$
  Anomaly ratio of x_i : $ar_{x_i} = \frac{log(e^{\frac{1}{td_{x_i}}})}{\sum_{x_j \in N_{min}} td_{x_j}}$
  $tss_{xi} = r_{x_i} \times Sample_{sync}$
  while $j \leftarrow 1$ *to* tss_{x_i} **do**
    for $k \leftarrow 1$ *to* $NLimit(j)$ **do**
      for $l \leftarrow 1$ *to minorAttributesSafeBorder* **do**
        $Sample_{new} \leftarrow$ *Two synthetic samples in opposite direction*
        D'=$D' \cup Sample_{new}$

$ConsecutiveSequence()$*Find Consecutive sequence from each cluster*
while *custerIndex in Cluster* **do**
  while *amount in ConsecutiveSequence* **do**
    if $confidence \geq ConfidenceThreshold$ **then**
      Confidence[cI][amount].append(confidence)
    if $support \geq SupportThreshold$ **then**
      Support[cI][amount].append(support)

$AllPossibleRules()\leftarrow$ Create all possible relational rules for each category;
while *ruleIndex in AllPossibleRules* **do**
  $support()\leftarrow$ Calculate support of rule
  $Confidence()\leftarrow$ Confidence of rule
  if $confidence \geq confidenceThreshold$ && $support \geq supportThreshold$ **then**
    RuleSet.append(rule)

prediction $\leftarrow$ *Each* transaction is valided by the rule set;
return *prediction*;

class samples to enhance transaction support. Figure 3.3 presents the suggested ARBBO model's block diagram.

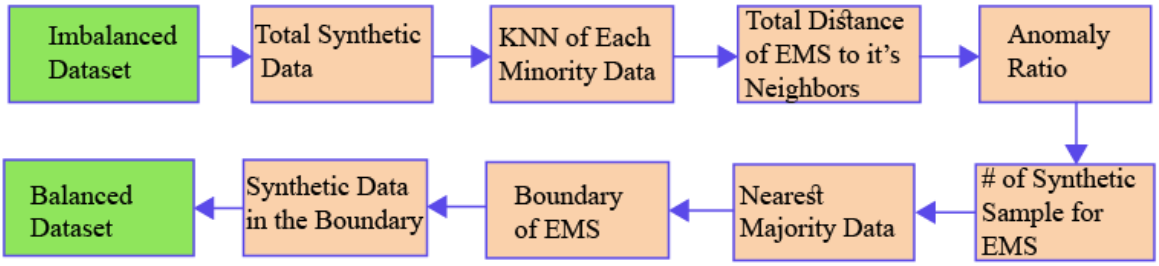

Fig. 3.3 Block diagram of proposed ARBBO model.

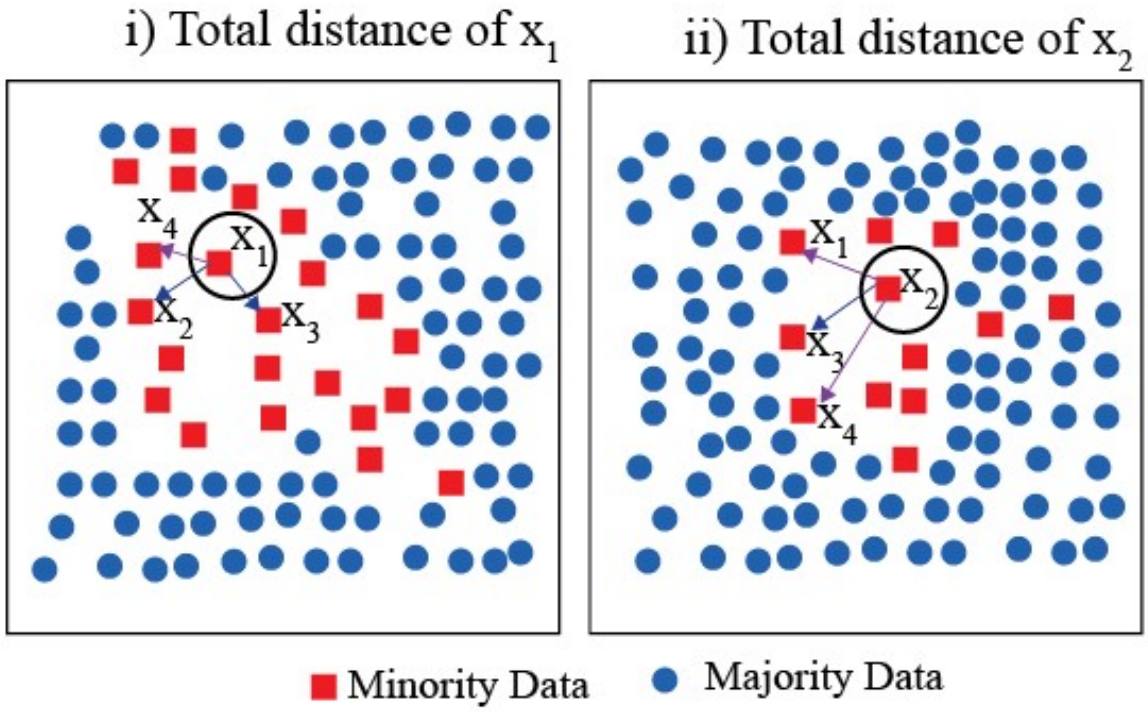

Fig. 3.4 Total distance of x_1 and x_2 with their neighbors.

3.3.1 Anomaly Ratio Based Total Synthetic Data

The model evaluates the level of class imbalance and calculates the quantity of synthetic samples needed to get a reasonable imbalance ratio. If the degree is below the specified threshold, subsequent steps are taken to create a balanced dataset. For a dataset with N_{maj} samples of the majority class and N_{min} samples from minority classes, the degree of imbalance in

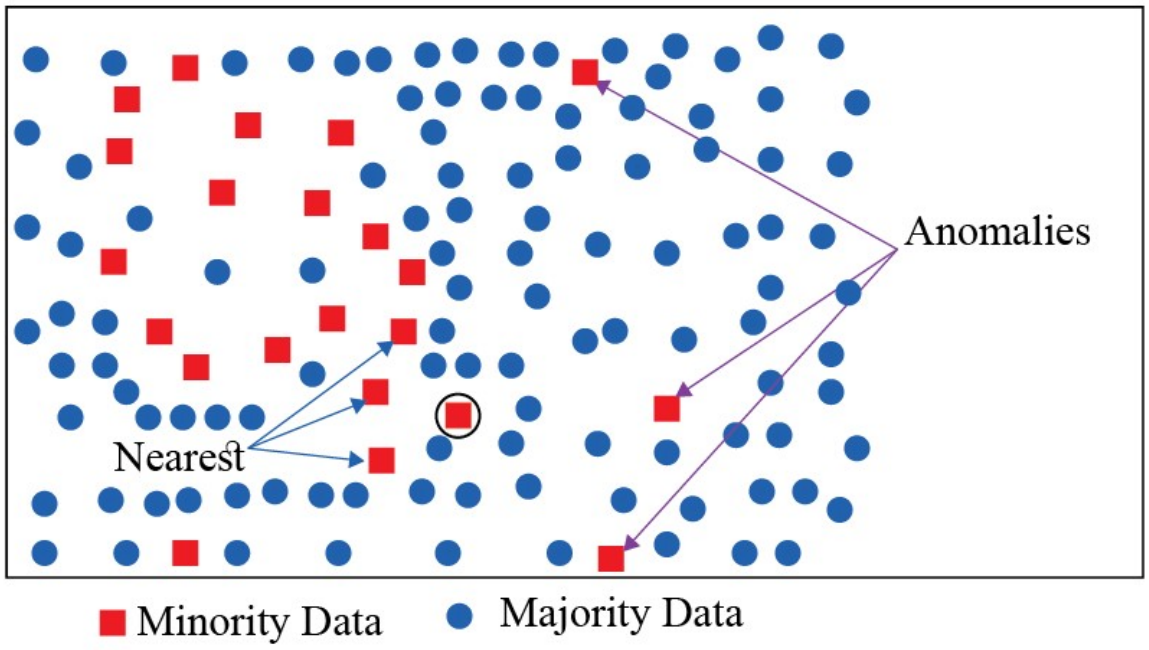

Fig. 3.5 Illustration of an anomaly within the dataset.

equation (3.1) and the total quantity of synthetic samples needed, $ts_{synthetic}$, are computed in equation (3.2):

$$degree = \frac{N_{min}}{N_{maj}} \tag{3.1}$$

$$ts_{syncthetic} = (N_{maj} - N_{min}) \times \theta \tag{3.2}$$

For every sample $x_i \in N_{min}$, the nearest k neighbors k_{x_i} that are members of the minority class are found. The distance between this sample and its closest k neighbors is known as the total distance of x_i, and it is computed as follows:

$$td_{x_i} = \sum_{x_j \in N_{min} \wedge x_j \in K_{x_i}} d(x_i, x_j) \tag{3.3}$$

Where $d(..)$ represents the distance between the two data samples in equation (3.3). The following formula is used to find the anomalous ratio of x_i:

$$ar_{x_i} = \frac{log(e^{\frac{1}{td_{x_i}}})}{\sum_{x_j} \in N_{min} td_{x_j}} \tag{3.4}$$

Note that $\sum_{ar_i} = 1$ as equation (4) has been utilized for normalization. In Figure 3.4, the aggregate lengths of two samples from minority classes, x_1 and x_2, are illustrated. The sum of the distances to x_2, x_3, and x_4 determines the overall distance of x_1 in Figure 3.4 left, while that of x_2 is the sum of distances to x_1, x_3, and x_4 in Figure 3.4 right. Notably, td_{x_1} is smaller than td_{x_2}, showing that the overall distance is reduced for a sample that is closer to its k nearest neighbors. Equation (3.4) is used to compute the anomaly ratio for each minority class sample; a larger total distance results in a smaller value, and vice versa. This method works well with anomalous and imbalanced datasets because it produces more synthetic data for samples that are near their k nearest neighbors and less for those that are farther away. Equation (3.5) is used to determine the number of synthetic data samples for each $x_i \in N_{min}$:

$$tss_{x_i} = r_{x_i} \times Sample_{sync} \tag{3.5}$$

3.3.2 Boundary Calculation

In the context of categorizing sample data into SAFE and ANOMALY, the proposed method diverges from the traditional SMOTE approach. Although the closest minority data points are connected with an interpolation line in the standard SMOTE method to create synthetic data, the KNN approach's precision in locating these nearby points may lead to potential data overlap between majority and minority classes. This paper suggests a change to address this worry: using the safe border distance in place of the nearest neighbors parameter in the SMOTE approach.

Finding the closest majority data point from the chosen minority sample defines the safe border distance, which then confines all newly created synthetic data points inside the bounds this distance establishes. Moreover, a circular shape formula is applied to data generated within this boundary, as depicted in equations (3.6), (3.7), and (3.8), where a two-dimensional vector example is illustrated.

$$|x - c| \leq dist^2 \tag{3.6}$$

$$\sum_{i=1}^{n} (x_{ij} - c_{ij})^2 \leq dist^2 \tag{3.7}$$

$$dist^2 = \sum_{i=1}^{n} (c_j - mp_j)^2 \tag{3.8}$$

The nearest majority point from the circle's center is shown as'mp' with coordinates (mp_1, mp_2, mp_3, ...mp_n). The circle's center, or 'c', is represented by the coordinates

$(c_1, c_2, c_3, ...c_n)$. The newly generated synthetic data under a safe radius distance is expressed as x_i with a first-direction interpolation scheme, having components $(x_1, x_2, x_3, ...x_n)$ for i = 1...n. Similarly, y_i represents synthetic data using a second-direction interpolation scheme with components $(y_1, y_2, y_3, ...y_n)$ for i = 1...n. The distance between 'c' and 'mp', $dist^2$ is determined by equations (3.7) and (3.8), as depicted in Figure 3.6.

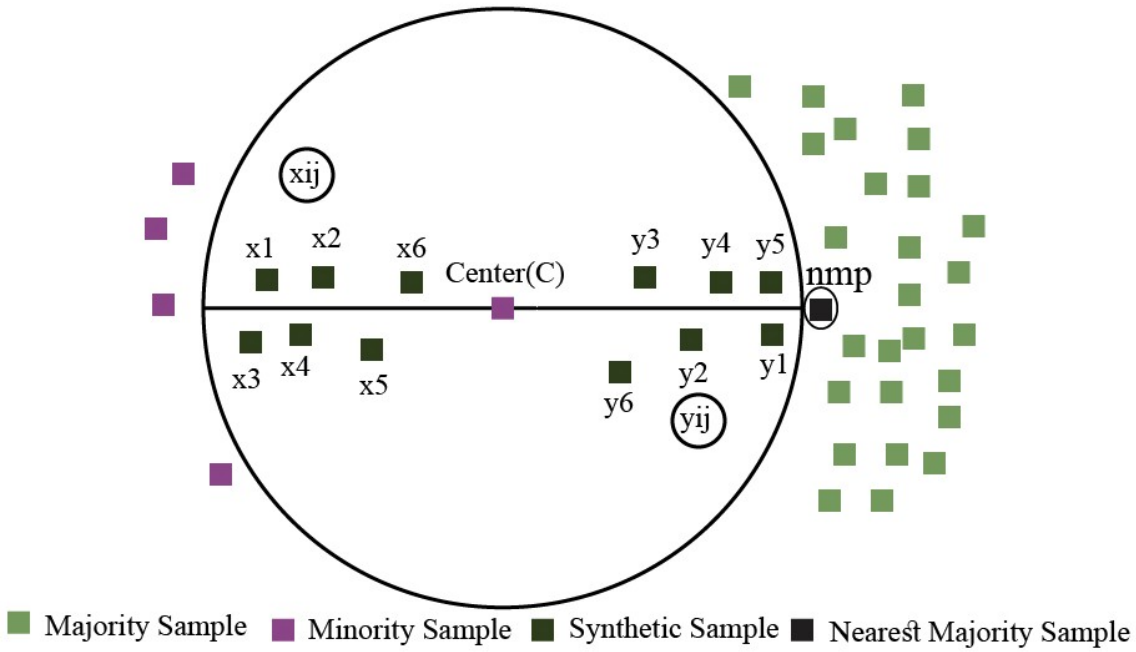

Fig. 3.6 Boundary of a selected minority sample.

3.3.3 Synthetic Data Generation

The creation of fresh synthetic data is the main goal of the last stage. The distance between each minority sample that has been chosen and all of the majority data points is calculated using the Euclidean distance formula. Finding the closest majority data point is essential; among the chosen minority samples, it is the one with the least distance in relation to the overall distance. This process is illustrated in equation (3.9).

$$dist_{ij} = \min \sum_{i=1}^{n} \sum_{i=1}^{n} \sqrt{(c_j - mp_j)^2} \tag{3.9}$$

$dist_{ij}$ in equation (3.9) denotes the smallest distance between minority sample 'j' and majority sample 'i'. Once these nearest majority data points are identified, the generation of synthetic data is executed along an interpolation line connecting these two points. Notably,

the synthetic data generation is performed in two directions, $dist_{ij}$ and $-dist_{ij}$, as elaborated in equations (3.10) and (3.11).

$$x_{ij} = center_j + \left(rand\ 0,1\ X\ \left(dist_{ij} - center_j\right)\right) \tag{3.10}$$

$$y_{ij} = center_j + \left(rand\ 0,1\ X\ \left(center_j - dist_{ij}\right)\right) \tag{3.11}$$

In equations (3.10) and (3.11), x_{ij} and y_{ij} represent the components of the synthetic data point along a line of interpolation connecting the minority sample to the closest majority sample. The generation involves random scaling factors determined by the distance between these two points in the respective directions, contributing to the diversity of the synthetic data.

3.4 Consecutive Sequence Based Rule Generation

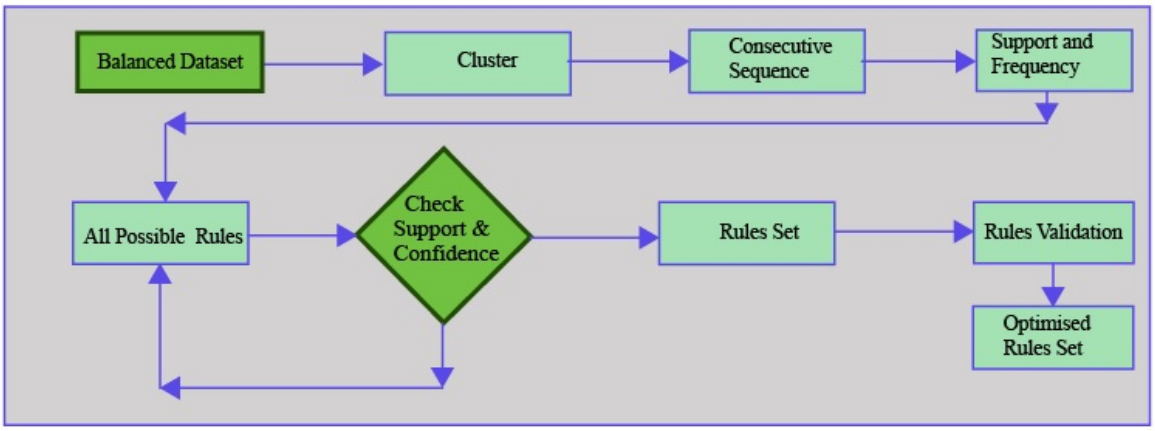

Fig. 3.7 Blog Diagram of Rule Generation Process.

3.4.1 Consecutive Sequence

In this work, we employ the Consecutive Sequence Mining Algorithm described in Algorithm 2. Figure 3.7 describes the rule generation process. The consecutive sequences are calculated from each type of transaction from each cluster. Let's think about a set of values of amount attribute:
1000, 1001, 2000, 3000, 5000, 4000, 5001, 1003, 2001, 3001, 4001, 2003, 2004, 2004, 5003, 5004, 5005, 4001, 3002, 1004, 4003, 1004, 6000, 3003.

Algorithm 2: Consecutive Sequence Mining Algorithm

Initialization:

1. Maintain two empty lists: *current_sequence* and *sequences*.
2. Set a variable *previous_number* to `None`.
3. Sort the data.
4. Divide the input data into N chunks.

Iterate through each of N chunks:

1. For each number n in the list:
2. If *previous_number* is `None` or n is equal to *previous_number* + 1:
    - Append n to *current_sequence*.
3. Otherwise:
    - If the length of *current_sequence* is greater than 1 (meaning it's a valid sequence),
        - Append it to *sequences*.
    - Reset *current_sequence* to a new list containing only n.
4. Update *previous_number* to the current number n.

Finalize:

- Return *sequences*.

From the above values, the consecutive sequences are as follows:
1. 1000, 1001, 1004, 1004, **2.** 2000, 2001, 2003, 2004, 2004, **3.** 3000, 3001, 3002, 3003, **4.** 4000, 4001, 4001, 4003, **5.** 5000, 5001, and **6.** 6000. To generate relational rules that set the proposed model apart from other models such as Apriori and FP-Growth, each sequence's maximum and minimum bounds are computed. Sequences 1, 2, 3, 4, and 5 have minimum and maximum limits of 1000 and 1004, 2000 and 2004, 3000 and 3003, 4000 and 4003, 5000 and 5001, respectively, while sequence 6 has a single value for its minimum and maximum limits of 6000. As a result, the rule terms of amount attribute are 1000 <= amount && 1044>=amount, 2000 <= amount && 2004>=amount, 3000 <= amount && 3003>=amount, 4000 <= amount && 4003>=amount, 5000 <= amount && 5001>=amount, and amount=6000.
Similarly, let's consider about a set of values of the oldbalanceOrg attribute:
9203, 9200, 9201, 6099, 7000, 7001, 6301, 6302, 6303, 5501, 5502, 5503, 5504, 5000, 5001, 5003, 5004, 5005, 4001, 4001, 4003, 1003, 1004, 1000, 1001, 1002, 999.
The following are the sequential sequences derived from the values above:
1. 999, 1000, 1001, 1002, 1003, 1004, **2.** 4001, 4001, 4003, **3.** 5000, 5001, 5003, 5004, 5005, **4.** 5501, 5502, 5503, 5504, **5.** 6301, 6302, 6303, **6.** 6099, 7000, 7001,**7.** 9200, 9201, 9203. The rule terms of the oldbalanceOrg(OBO) property are constructed as follows after determining the lowest and maximum values of each sequence:
999 <= OBO && 1004>=OBO, 4001 <= OBO && 4003>= OBO, 5000 <= OBO && 5005>= OBO, 6301 <= OBO && 6303>= OBO, 6099 <= OBO && 7001>= OBO, and 9200 <= OBO && 9203>= OBO.
The rule words for other attributes, including newbalanceOrig, oldbalanceDest, and newbalanceDest, are produced by adhering to the amount and oldbalanceOrg attributes mentioned above. From the experiment dataset, the rule terms of amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, and newbalanceDest are 578, 566, 10, 415, and 585 respectively. If we apply ${}^{n}C_{r}$ on each attribute of the dataset to form the rules, then according to the proposed model, the possible minimum number of rules is ${}^{578}C_1 \times {}^{566}C_1 \times {}^{10}C_1 \times {}^{415}C_1 \times {}^{585}C_1$.

3.4.2 Support of Consecutive Sequence

The assessment of the support for every rule term is an important part of the financial dataset analysis process. The procedure entails applying each potential rule term to the transactions in the dataset in order to determine its frequency. Subsequently, the count of satisfactory outcomes, referred to as support, is computed for each rule term. For instance, consider a financial dataset containing various transactions representing customer purchases. One rule

term could be the combination of specific items or products that tend to co-occur frequently in these transactions. The support for this rule term would be the number of transactions in which this combination of items is observed. The entire dataset undergoes this process for each possible rule term generated by the output of a consecutive sequence mining algorithm. The support for each rule term is calculated based on the cumulative support of its occurrences across all transactions. The equation (3.12) represents the support of a pattern.

After establishing the support for all possible rule terms, the subsequent step involves combining sequences whose support exceeds a predefined threshold, typically greater than 0. These sequences are essentially the building blocks of rules, representing patterns or associations discovered within the financial dataset. In summary, the analysis entails systematically assessing the support of individual rule terms through the examination of transactional data, and subsequently forming rules by aggregating sequences with significant support. This method allows for the identification of meaningful patterns and associations within the financial dataset, facilitating insights into customer behavior, market trends, or other pertinent financial aspects.

3.4.3 Rule Generation

The suggested model employs association rules in the form of A $\rightarrow$ C, where A represents the antecedent comprising rule terms connected by "and," and C signifies the consequent denoting fraud. These rules are formulated by combining rule terms associated with each attribute and their respective transaction types in a financial dataset.

For example, consider a rule: "If the transaction involves a high-value amount (A1) and occurs during non-business hours (A2), then classify it as fraud (C)." Here, A1 and A2 are rule terms representing attributes like transaction amount and time, and C denotes the consequence of fraud.

To ensure that only pertinent rules are considered, the model applies minimum support and confidence thresholds. In other words, a rule is deemed eligible only if it surpasses these predefined thresholds. This selective approach aims to generate a refined set of rules that exhibit both sufficient support and confidence, enhancing their efficacy in identifying fraudulent transactions within the financial dataset.

For instance, the minimum support threshold may require that a rule must be applicable to a certain percentage of transactions in the dataset, ensuring its relevance. Similarly, the confidence threshold might dictate that a rule must have a minimum accuracy level in predicting fraud, adding a layer of reliability to the selected rules.

The model assesses candidate rules by comparing their support and confidence scores against user-defined thresholds. This evaluation process ensures that only rules meeting the

specified criteria are considered for inclusion in the final set. The outcome is a collection of efficient rules that have demonstrated both statistical significance and predictive accuracy in identifying instances of fraud within the financial dataset.

In summary, the proposed model leverages association rules to systematically combine rule terms related to various attributes and transaction types. By applying minimum support and confidence thresholds, the model refines this rule set, ensuring that only relevant and reliable rules contribute to the identification of fraudulent transactions in the financial dataset.

3.4.4 Support and Confidence

The analysis of each sequence of clusters in the proposed model involves a careful examination of their frequency of occurrence. Subsequently, the support and confidence of these sequences are calculated. If the computed support and confidence values meet or surpass the user-specified threshold, the corresponding rule is considered final and is then integrated into the system's knowledge base. This crucial step ensures that only the most pertinent and trustworthy rules become part of the model, contributing to optimal performance.

To illustrate with a financial example, consider a sequence of clusters representing patterns of transaction behavior in a credit card dataset. A rule derived from such a sequence might be: "If a customer makes multiple high-value transactions within a short time frame (sequence), then flag the transactions for further scrutiny as potential fraud (rule)." The frequency of occurrence of this pattern is analyzed, and if it consistently exhibits both support 3.12 (the frequency of the pattern's occurrence in the dataset) and confidence 3.13 (the accuracy of the rule in identifying fraud) above a specified threshold, it becomes a finalized rule.

To refine the rule set specifically for fraud detection, the model employs a selection process based on the least amount of confidence and support. Minimum support ensures that a rule occurs frequently enough to be considered relevant, while minimum confidence ensures that the rule is consistently correct in its predictions. By eliminating unreliable rules that fall below these thresholds, the model captures meaningful patterns indicative of fraudulent behavior. This selective approach leads to the development of a more accurate and dependable fraud detection model, as it focuses on rules with a strong empirical basis and predictive capability. Association rule mining frequently uses representations of confidence and support. The equations (3.12) and (3.13) show a rule's confidence and support, respectively.

$$\text{Support}(X) = \frac{\text{Number of transactions containing } X}{\text{Total number of transactions}} \tag{3.12}$$

$$\text{Confidence}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)} \tag{3.13}$$

3.4.5 Rule Set

By choosing the most pertinent rules based on the minimum support and confidence notions, the rule set can be refined. But not every regulation has the same weight, and some might even be deceptive. The model assesses each rule according to its confidence and support to make sure the set of rules is effective and efficient. The confidence indicates how frequently the rule is accurate, whereas the minimal support indicates how often a rule appears in the dataset. In order to ensure that the rules capture significant trends in the data and avoid false or unreliable rules, the selection procedure of the rules is based on minimum support and confidence. The refined rule set is then produced by choosing just those rules that do so, based on the user-defined minimal support and confidence levels. Therefore, a rule-based machine learning model that is more reliable and accurate could be able to detect fraudulent financial transactions with more precision.

3.4.6 Rule Validation

Ensuring the accuracy and efficacy of rules in a fraud detection system is critical, and rule validation serves as a pivotal step in achieving this objective. Two essential methods, namely rule structure verification and rule consistency verification, play integral roles in this validation process.

Rule structure verification ensures that each rule strictly adheres to the IF-THEN structure. For example, in a financial dataset, a rule might state: "IF a transaction involves an unusually large sum of money (condition), THEN flag it for further investigation (consequence)." Rule structure verification ensures that such rules are properly formulated and adhere to the logical structure required for effective fraud detection.

On the other hand, rule consistency verification is concerned with assessing the alignment of a given rule with other association rules within the repository. This involves examining antecedent (IF part) and consequent (THEN part) constraints across multiple rules to identify any conflicts or inconsistencies. In the context of a financial dataset, if one rule identifies a suspicious transaction based on a certain criterion, Verification of rule consistency guarantees that this decision aligns seamlessly with other rules governing similar transactions.

By meticulously implementing these validation methods, the reliability and effectiveness of the rules are fortified. This is crucial in the context of fraud detection within a financial dataset, where inaccurate or inconsistent rules could lead to both false positives and false

negatives. In summary, rule validation is a key safeguarding measure, ensuring that the rules governing the detection of fraudulent activities are not only logically sound but also consistent with other rules in the system.

3.4.7 Rule Optimization

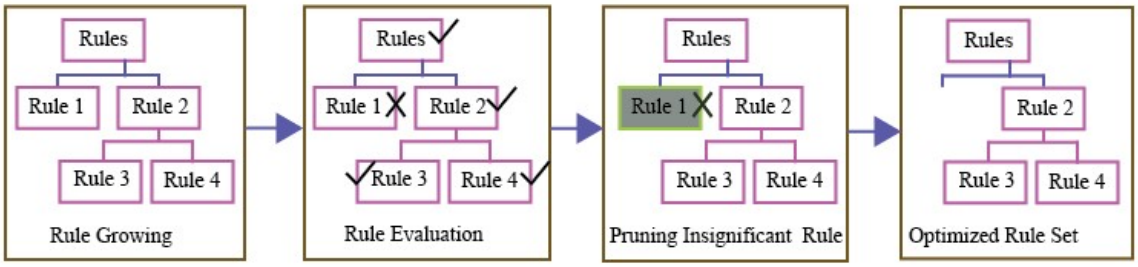

Fig. 3.8 Diagram depicting the procedural steps of the rule optimization process.

Rule optimization is a crucial step aimed at improving the effectiveness of a fraud detection classifier by eliminating unnecessary rules. This process involves iterative exploration, assessing the significance of rule support, confidence, and redundancy. Lowering thresholds may lead to the inclusion of redundant rules, while maintaining a confidence threshold between 50% and 100% is found to yield optimal fraud prediction and minimize redundancy. Figure 3.8 shows the details schematic diagram of proposed model.

For example, let's consider a rule (r1) stating that "IF a transaction involves an unusually large amount," and another rule (r2) stating "IF the transaction occurs during non-business hours." Both rules individually may have lower confidence, but when combined as [(r1$\wedge$r2)$\vee$ r3], where r3 represents another rule, the confidence level can be significantly increased. This merging process allows for the creation of more impactful rules, contributing to improved fraud detection accuracy.

The optimized rule, [(r1$\wedge$r2)$\vee$ r3] $\rightarrow$ Fraud, serves to eliminate redundant rules while enhancing the classifier's accuracy. This consolidated rule encapsulates the combined conditions of r1 and r2, maximizing confidence in identifying fraudulent transactions.

Subsequently, an proposed rule-based model is implemented, utilizing a set of IF-ELSE statements derived from the optimized rules. These statements act as decision criteria, applying the streamlined and unified rule set to categorize data elements as either fraudulent or non. The IF-ELSE structure allows for a systematic evaluation of transaction attributes, leading to more accurate and efficient fraud detection within the dataset.

In summary, rule optimization involves refining the rule set by considering support, confidence, and redundancy, ultimately leading to the creation of more impactful rules. The

optimized rules are then implemented in an Rule-Based model, forming a set of IF-ELSE statements that enable the classifier to make accurate and reliable predictions regarding the fraudulent nature of financial transactions.

3.5 Fraud Detection

The recently created Rule-based model produces a set of rules with maximum support and confidence based on a financial dataset. These rules are then incorporated in the form of IF-ELSE statements within a predictive function designed for Fraud Detection. In this predictive function, if a customer transaction in the test dataset satisfies the condition specified by any rule, it is classified as fraudulent; otherwise, it is categorized as non-fraudulent.

For instance, consider a rule stating: "IF a transaction involves an unusual combination of high-value transactions from different geographical locations (condition), THEN classify it as fraud (consequence)." To generate predictions on the test dataset, the Rule-based model employs a collection of these rules.

The predictions made by the Rule-based model are stored in a list named $model_{predictions}$. As the model iterates through each rule in the testing dataset, the predicted values are appended to this list. Subsequently, the accuracy, True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) scores of the new rule-based classifier are computed by comparing its predictions with the actual values.

To examine the rule-based model's performance in more detail, the $model_{predictions}$ list is passed to different methods that generate key evaluation metrics. These metrics include the confusion matrix, providing a breakdown of correct and incorrect predictions; the classification report, which provides an F1-score, precision, and recall; and the ROC-curve, which shows how well the model can distinguish between fraud and non-fraud cases.

In summary, the Rule-based model utilizes IF-ELSE statements derived from rules with high support and confidence to predict fraud in customer transactions. The model's predictions are then evaluated against actual values, and key performance metrics are computed for a comprehensive assessment of its effectiveness in fraud detection within the financial dataset.

Chapter 4

Experiments and Evaluation

4.1 Dataset Description

The experiment's chosen dataset is available on Kaggle.com. It is made up of 6,362,620 transaction records, all of which were produced by the PaySim financial mobile money simulator. In order to build this synthetic dataset, PaySim, which mimics real financial records, utilized a one-month financial log from a mobile money service provider that contained a sample of real transactions. The company's name remained undisclosed, but it was described as a multinational enterprise that offered services to mobile phone users so they could use their phones' electronic wallets to send and receive money. This synthetic dataset has been reduced by the data supplier to a quarter of its original size. There are a total of 11 attributes: 7 independent variables that reflect a transaction's features, 1 dependent variable that represents a transaction's state, and 3 variables that are eliminated in section V since they are not deemed to be significant influencers. Table 1 shows a list of all the variables. Some of the variables are further described in the paragraphs that follow in order to provide clarification. Step is the number of hours in a month.

For instance, a record with step = 1 indicates that it occurred in the first hour on the first day of the experiment, and a record with step = 744 indicates that it happened in the final hour of the month. Five distinct transaction kinds are involved in the category variable "type," which is CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER. TRANSFER is the act of moving money between users. CASH-IN signifies that an increase in cash inflow led to an increase in the customer's account balance; Cash outflow reduces the account balance, making it the opposite of cash in; Transferring funds from a mobile service (electronic wallet) to a bank account is known as DEBIT. The process of paying merchants for products or services is known as PAYMENT, and it results in lower balances on client accounts and higher balances on merchant accounts. An extra feature that the mobile service provider employed

Table 4.1 Characteristics, Illustration, and Overview of the Paysim Dataset

Variable Name	Example	Description
step	5	Each step is an hour of time in the real world.
type	PAYMENT	CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER
amount	8424.74	Transaction amount in local currency
nameOrig	C1000001725	Customer who started the transaction
oldbalanceOrig	351422.72	The initial balance of the sender before the transaction
newbalanceOrig	257557.59	The new balance of the sender after the transaction
nameDest	M1974356374	Customer/Merchant who received the transaction
oldbalanceDest	526950.37	The initial balance of the receiver before the transaction
newbalanceDest	771436.84	The new balance of the receiver after the transaction
isFraud	1	The status of a transaction (0 as legitimate and 1 as fraudulent)
isFlaggedFraud	0	The status that the system identified

to regulate large transfers was isFlaggedFraud. The specification states that isFlaggedFraud would be set to 1 in the event that exceeding 200,000 units (in local money) were attempted to be transferred in a single transaction; otherwise, it would be set to 0. It isn't specifically related to isFraud.

The dataset contains no missing or mismatched values. Nevertheless, as Figure 4.1 illustrates, there is a significant disparity in the records between the two classes, with 6,354,407 (99.87%) genuine cases and just 8213 fraudulent cases (0.13%). We have utilized 20% of the data from the main dataset due to its large volume, with 70% allocated for training and 30% for testing. Using the proposed ARBBO, subsamples with an equal distribution of fraudulent and non-fraudulent transactions can be constructed to reduce the impact of skewness. Figure 4.2 shows the transactions according to the transaction types. Figure 4.3 shows the correlation heat-map among the attributes of the paysim dataset.

4.2 Evaluation Measure

We conducted experiments using the original datasets to evaluate how well the suggested model performed in comparison to a number of classifiers, including RF, DT, MLP, KNN, NB, and LR. The tests were conducted using the Python programming language and its machine-learning modules. Seventy percent of the samples in the dataset were used for training, and the remaining thirty percent were used to assess the performance outcomes of the proposed model. Using metrics like accuracy in equation 4.1, precision in equation 4.2, recall in equation 4.3, f1-score in equation 4.4, confusion matrix, AUC of the Receiver Operating Characteristic curve (ROC-AUC), and AUC, the performance of machine learning

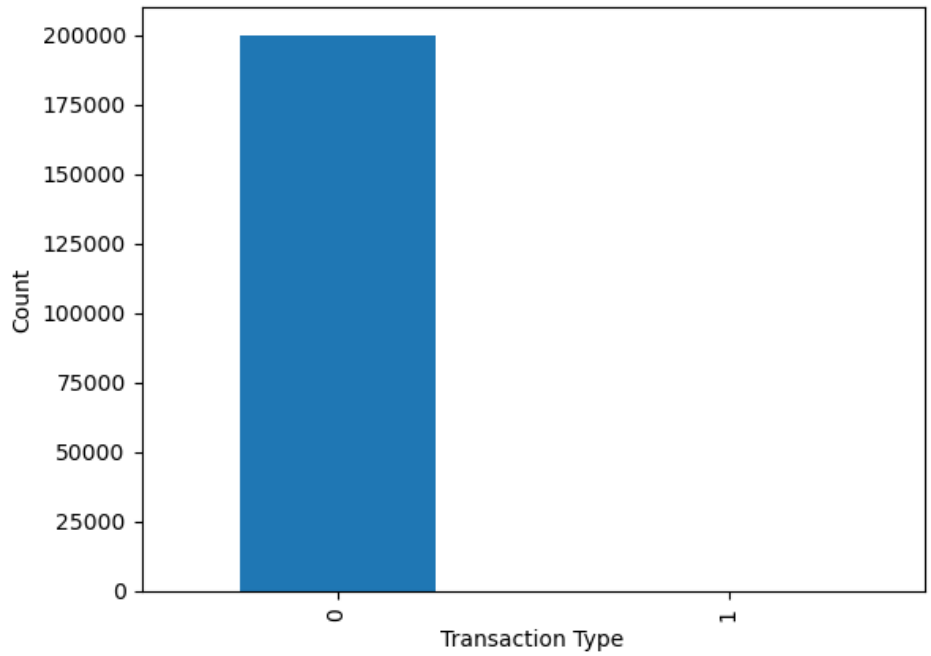

Fig. 4.1 Fraud vs Non-Fraud Transactions in Paysim dataset.

classification algorithms is evaluated following their training on the dataset. It is impossible to find the perfect measure to assess a model's efficacy. The closer a classifier's AUC value is to 1, the better, as a perfect model is indicated by an AUC value of 1. The ratio of true positives to false positives at various threshold levels is compared using the ROC curve. A confusion matrix contains information about the expected and actual classifications of a classifier, such as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

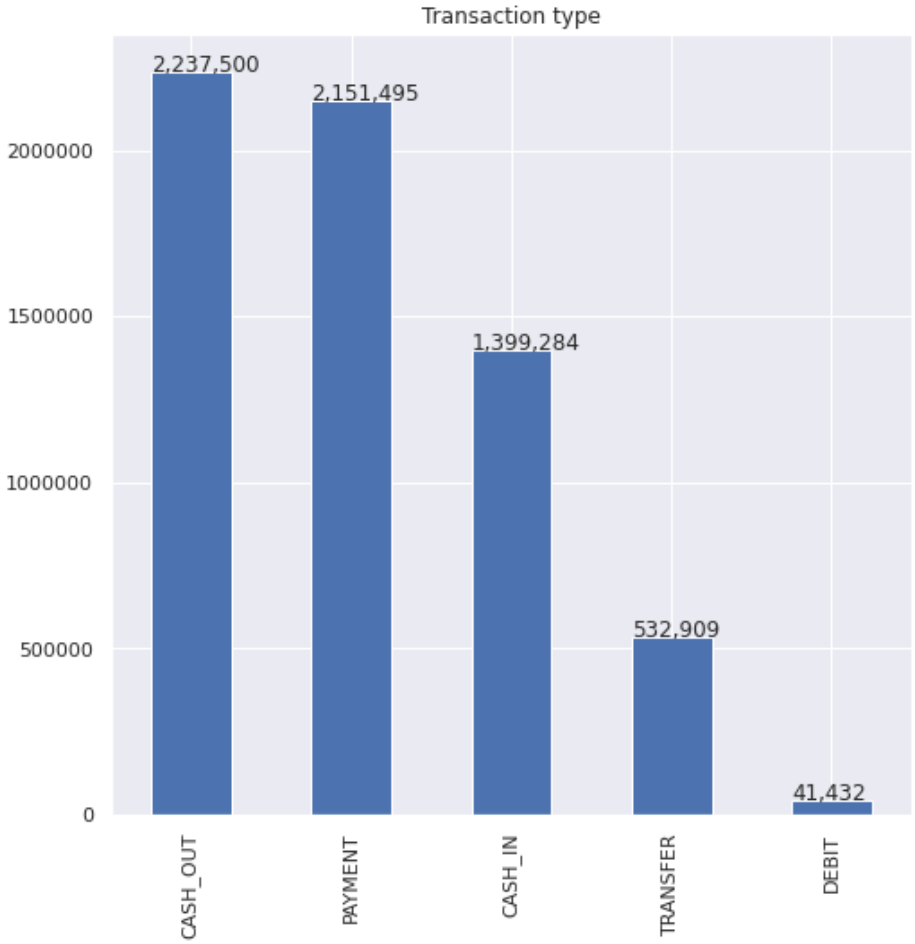

Fig. 4.2 Fraud vs Non-Fraud Transactions in Paysim dataset.

4.2.1 Receiver Operating Characteristics (ROC) Curves

Two assessment metrics that are employed in the receiver operating characteristic curve, or ROC curve, are the true positive rate and the false positives rate, as shown in Figure 4.4.

$$\text{The FP percentage, or F P \%} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.5}$$

The equation 4.5 can be used to calculate the x-axis of a ROC curve. It is the same as one minus specificity. Plotted on the y-axis, the recall, or TP %, is calculated using the following formula 4.6.

$$\text{TP \%} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.6}$$

The ROC curve shows the TP percentage across the FP percentage to visually represent the tradeoff between benefits (TP%) and costs (FP%). A ROC curve's points represent how well

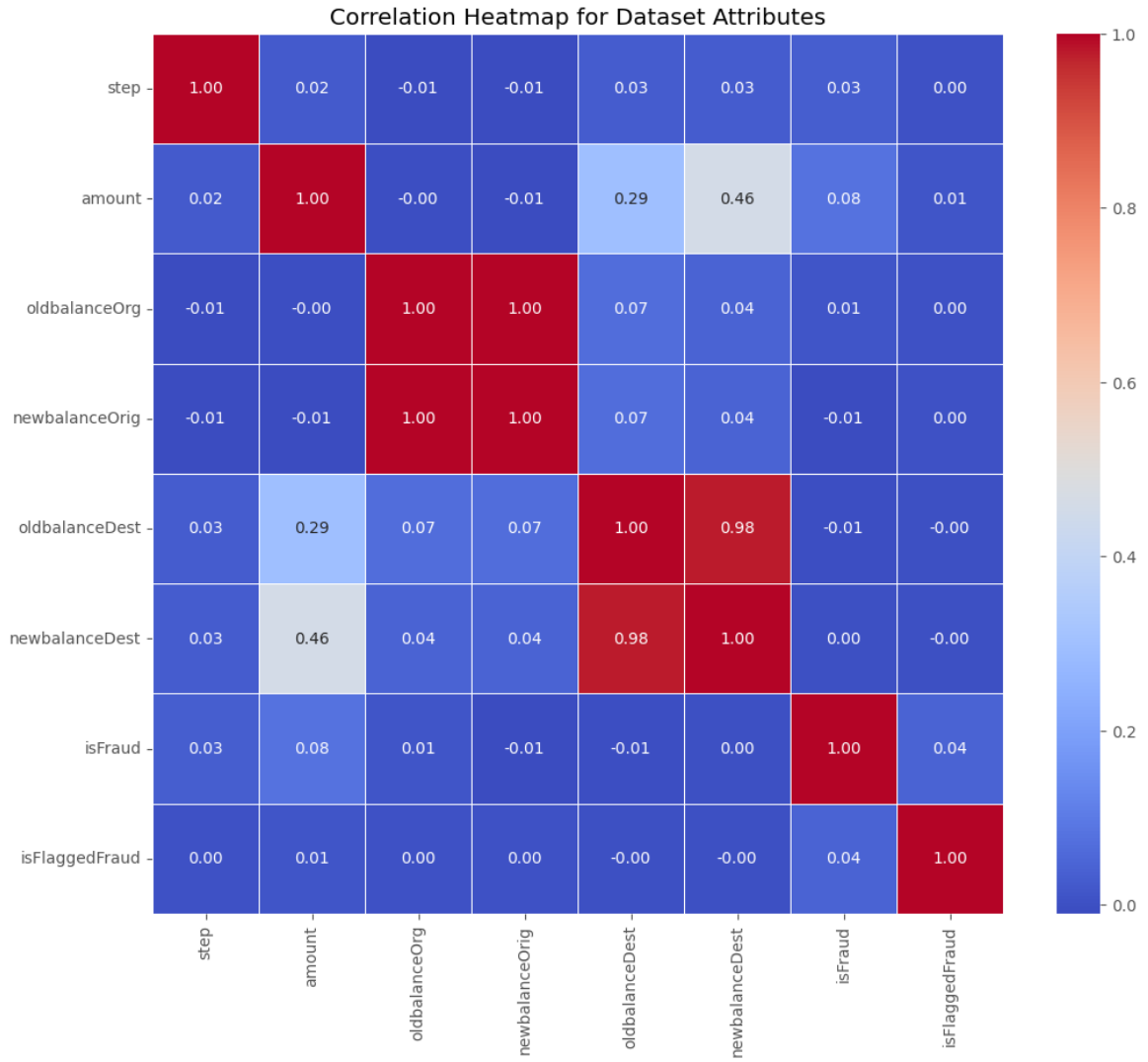

Fig. 4.3 Correlation Heat-map for Dataset Attributes.

a single classifier performs on a certain distribution. The case of guessing the class at random is shown by the line y = x. Since it is difficult for a classifier to identify positive examples in an unbalanced dataset if there was a very little initial disparity between the minority and majority classes, we typically get low true positive rates and low false positive rates. The point is often at the lower left corner in this instance. The corresponding point of a classifier will shift around the curve according to changes in the training set's distribution. By either undersampling the majority class or oversampling the minority class, we can alter the training set's distribution.

For a particular classification learning, our goal is to attain a high true positive rate and a low false positive rate. The ideal point is shown in the upper left corner (0,100). Even

though it is nearly impossible to reach this point, we should work to devise a strategy that will allow us to get a curve that is as near to the ideal point as we can. This raises an interesting question: if the ROC curves intersect, how can we figure out one categorization learning strategy works better than the others. AUC, or the Area Under the ROC Curve, provides a full answer to this query. This statistic is a common method for comparing how well classification learners perform. AUC measures both classifiers' overall performances given various class distributions, although it is challenging to choose between the classifiers for green and cyan based only on ROC curves, as shown in Figure 4.4. We can determine which classifiers are more dominant by comparing their AUC. All of the experiments in this paper make use of a number of the assessment criteria covered above. One statistic may demonstrate a classification approach's superiority over others, while another may suggest that it is less desirable. As a result, evaluation should be taken into account using a variety of measures, and an individual can choose a suitable metric by considering their familiarity with the program.

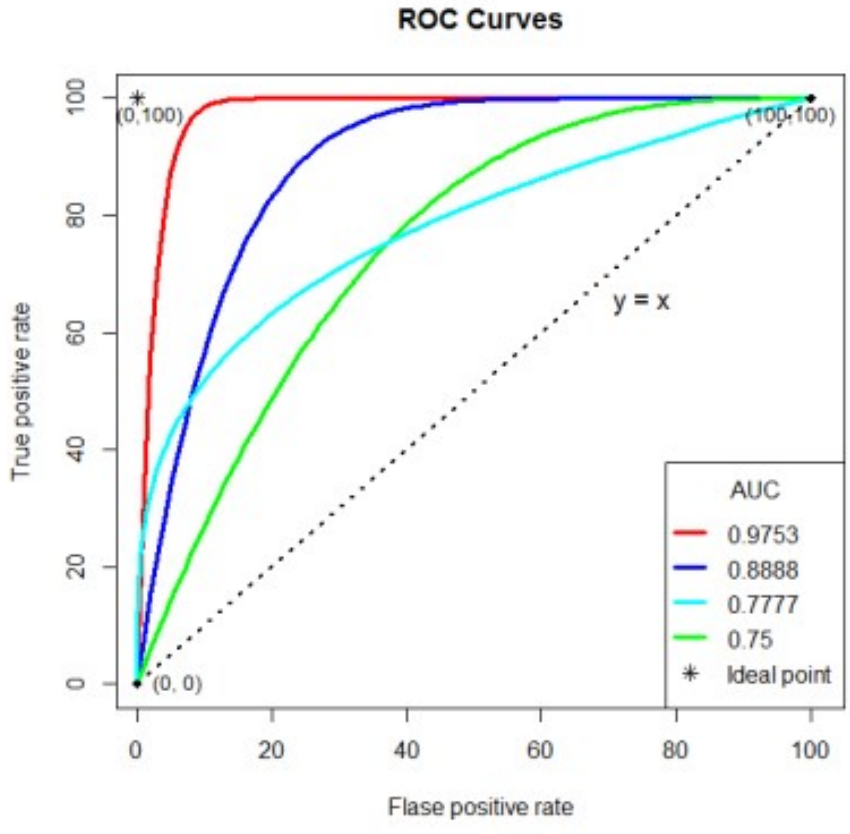

Fig. 4.4 Example of ROC Curve.

4.2.2 Confusion Matrix

A common tool used to record performance is the confusion matrix, which may be used to calculate most of the metrics stated above, as shown in Figure 4.5. various categorization tools. The projected class is represented in the matrix's columns, and the actual class is represented in its rows. True positives (the proportion of correctly categorized negative samples, with a similar criteria for the remaining samples), in the confusion matrix, the letters TN, FN, FP, and TP stand for false negatives, false positives, false negatives, and true positives, respectively. Because a learning algorithm's overall evaluation should be based on a group of measures rather than a single statistic when there are uneven learning conditions. The performance of classification algorithms will be assessed using a collection of evaluation criteria linked to ROC graphs that are created from confusion matrices in the next section.

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

Fig. 4.5 Example of Confusion Matrix.

4.3 Results and Analysis

Figures 4.6, 4.7, 4.8, 4.9, 4.10 show the some code segments of proposed rule-based model. Figures 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 show the some code segments of proposed ARBBO oversampling model.

This section provides a thorough analysis, detailing the results of the proposed Rule-Based and ARBBO models in comparison to other well-established classifiers, including DT, LR, NB, KNN, MLP, RF, and the SMOTE oversampling method. The evaluation metrics considered encompass True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), and Receiver Operating Characteristic AUC.

Furthermore, the comparison extends to include benchmarking against relevant published works. This comprehensive assessment aims to provide a thorough understanding of The effectiveness of the suggested Rule-Based and ARBBO models within the context of existing methodologies and standards in the area of identifying financial fraud.

```
#Storing each list into specific dict key according to type
num_transactions=len(transactions)
conditions={
    "CASH_IN":[],
    "CASH_OUT":[],
     "DEBIT":[],
    "PAYMENT":[],
    "TRANSFER":[],
}
#Going through each datapoint and only those values will be appended that has Fraud==1
for i in range(0,len(transactions)):
    if transactions[i][6]==1:
        if transactions[i][0]=='CASH_IN':
            conditions['CASH_IN'].append(transactions[i])

        if transactions[i][0]=='CASH_OUT':
            conditions['CASH_OUT'].append(transactions[i])

        if transactions[i][0]=='DEBIT':
            conditions['DEBIT'].append(transactions[i])

        if transactions[i][0]=='PAYMENT':
            conditions['PAYMENT'].append(transactions[i])

        if transactions[i][0]=='TRANSFER':
            conditions['TRANSFER'].append(transactions[i])
```

Fig. 4.6 Frist Code Segment for Rule Generation.

Table 4.2 The PaySim Dataset was used to generate certain relational association rules.

Relational Association Rule	SUP	CON
{oldbalanceOrg = amount AND type = CASH_OUT} → {Fraud}	72%	98%
{56000<= *oldbalanceOrg*<= 56900**AND***type* == *CASH_OUT***AND**1<= *newbalanceDest*<= 105} → {*Fraud*}	49%	13%
{*oldbalanceDest* == *newbalanceDest* == 0**AND***oldbalanceOrg*>0**AND***type* == *CASH_OUT*} → {*Fraud*}	64%	82%
{*oldbalanceOrg* == *amount***AND***type* == *TRANSFER*} → {*Fraud*}	45%	12%
{56000<= *oldbalanceOrg*<= 56900**AND***type* == *TRANSFER***AND**1<= *newbalanceDest*<= 105} → {*Fraud*}	20 %	47%

The Table 4.2 presents some relational associative rules obtained using our new consecutive sequence mining Algorithm 2. A notable distinction between our consecutive sequence-based relational rules and the existing rules generated by Apriori and FP-growth lies in the nature of the rules.

Conventional association rule mining techniques, such FP-growth and Apriori, are made to find frequently occurring itemsets in transactional databases. However, when dealing with datasets characterized by consecutive sequences, these algorithms may exhibit longer iteration times due to the nature of the candidate dataset.

In our approach, the consecutive sequence mining algorithm is specifically tailored to handle sequential patterns, and its efficiency in generating relational rules from such datasets

```
def write_rules(rules):
    final_rules=""
    if len(rules[1])>1:
        final_rules+="{ "+rules[1][0]
        final_rules+=" ,type=="+str(rules[0])
        final_rules+=" ,"+rules[1][1]
        final_rules+=" ,"+rules[1][2]
        final_rules+= " Support : {0:2.7f}".format(rules[2])
        final_rules+=" confidence : {0:2.7f}  ".format(rules[3])
        final_rules+=" } -> {FRAUD}"
    else:
        final_rules=" "
        final_rules+="{ "+rules[1][0]
        final_rules+=" , type=="+str(rules[0])
        final_rules+= " Support : {0:2.7f}".format(rules[2])
        final_rules+=" confidence : {0:2.7f}  ".format(rules[3])
        final_rules+=" } -> {FRAUD}"

    return final_rules

def check_threshold(length,rules,count,name,type_name,min_support,min_confidence):
    confidence=count/length
    support=count/num_transactions
    if confidence>=min_confidence and support>= min_support:
        rules.append(write_rules([type_name,name,support,confidence]))
        return rules
    return rules
```

Fig. 4.7 Second Code Segment for Rule Generation.

contributes to reduced iteration times. This is especially helpful for datasets where the association rules heavily depend on the consecutiveness of the sequences.

By focusing on consecutive sequences, our algorithm provides insights into patterns that might be overlooked by traditional frequent pattern-based methods. This adaptability makes it a valuable tool in scenarios where the consecutive order of items holds significance, such as time-series data or sequential transactions.

Finally, the consecutive sequence mining algorithm introduced in Algorithm 2 stands out in scenarios where Apriori and FP-growth may experience prolonged iteration times due to

Table 4.3 PaySim Dataset using ARBBO was used to construct a few relational association rules.

Relational Association Rule	SUP	CON
$\{oldbalanceOrg == amount \textbf{AND} type == CASH\_OUT\} \rightarrow \{Fraud\}$	92%	100%
$\{56000 <= oldbalanceOrg <= 56900 \textbf{AND} type == CASH\_OUT \textbf{AND} 1 <= newbalanceDest <= 105\} \rightarrow \{Fraud\}$	75%	86%
$\{oldbalanceDest == newbalanceDest == 0 \textbf{AND} oldbalanceOrg > 0 \textbf{AND} type == CASH\_OUT\} \rightarrow \{Fraud\}$	84%	96%
$\{oldbalanceOrg == amount \textbf{AND} type == TRANSFER\} \rightarrow \{Fraud\}$	85%	76%
$\{56000 <= oldbalanceOrg <= 56900 \textbf{AND} type == TRANSFER \textbf{AND} 1 <= newbalanceDest <= 105\} \rightarrow \{Fraud\}$	65 %	84%

```
def get_rules(conditions,min_support,min_confidence):
    oldbalanceOrg_limit=56900
    amount_limit=1160000
    newbalanceOrg_limit=12
    newbalanceDest_limit=105
    case=conditions[0][0] #getting transaction type name
    oldbalanceOrg_greater="oldbalanceOrg > "+str(oldbalanceOrg_limit)
    oldbalanceOrg_less="oldbalanceOrg <="+str(oldbalanceOrg_limit)

    newbalanceOrg_greater="newbalanceOrg > "+str(newbalanceOrg_limit)
    newbalanceOrg_less="newbalanceOrg <="+str(newbalanceOrg_limit)

    newbalanceDest_greater="newbalanceDest > "+str(newbalanceDest_limit)
    newbalanceDest_less="newbalanceDest <="+str(newbalanceDest_limit)

    newbalanceDest_greater="newbalanceDest > "+str(newbalanceDest_limit)
    newbalanceDest_less="newbalanceDest <="+str(newbalanceDest_limit)

    amount_greater="amount > "+str(amount_limit)
    amount_less="amount <="+str(amount_limit)
    amount_equal="amount == oldbalanceOrg"

    oldbalanceDestequalsnew="oldbalanceDest==newbalanceDest"
    oldnewDestequalszero="oldbalanceDest==newbalanceDest==0 and oldbalanceOrg >0 "
    oldbalanceequalsamount_limit="oldbalanceOrg=="+str(amount_limit)
    oldbalanceequalsamount="oldbalance==amount"
    newbalanceDest_less="newbalanceDest<="+str(newbalanceDest_limit)
```

Fig. 4.8 Third Code Segment for Rule Generation.

the distinctive characteristics of the candidate dataset. This approach offers a specialized solution for association rule mining in datasets with consecutive sequences, showcasing its versatility in handling diverse data patterns.

The Table 4.3 provides a glimpse into the relational association rules obtained through the application of our proposed ARBBO oversampling method. Notably, the introduction of ARBBO to the association rule mining process has led to the generation of more robust and compelling rules. In comparison to the rules extracted without the ARBBO oversampling, the rules obtained post-application exhibit increased strength and significance. The ARBBO technique was created to address class disparities and enhance the representativeness of minority classes, contributes to a more comprehensive and insightful rule set. By oversampling instances using the ARBBO technique, the algorithm effectively ensures a more balanced representation of the dataset, particularly beneficial when dealing with imbalanced classes. The creation of association rules depends critically on this balanced representation, as it provides a more true representation of the underlying patterns in the data. The strengthened association rules obtained with ARBBO showcase the effectiveness of incorporating oversampling techniques tailored to the characteristics of the dataset. This not only enhances

```
oldbalanceDestequalsnew="oldbalanceDest==newbalanceDest"
oldnewDestequalszero="oldbalanceDest==newbalanceDest==0 and oldbalanceOrg >0 "
oldbalanceequalsamount_limit="oldbalanceOrg=="+str(amount_limit)
oldbalanceequalsamount="oldbalance==amount"
newbalanceDest_less="newbalanceDest<="+str(newbalanceDest_limit)

#dict named type will store the count of each transaction that satisfies the following
Type= {
        oldbalanceequalsamount : 0,
        oldbalanceDestequalsnew : 0,
        oldnewDestequalszero:0,
        oldbalanceequalsamount_limit :0,
        oldbalanceOrg_less:
            {
                newbalanceOrg_greater:
                {
                    amount_less: 0,amount_greater: 0
                },
                newbalanceOrg_less:
                {
                   amount_less: 0,amount_greater: 0
                },
                newbalanceDest_greater :
                {

                  amount_less: 0,amount_greater: 0,amount_equal:0,'count':0
                },
                 newbalanceDest_less:{
```

Fig. 4.9 Fourth Code Segment for Rule Generation.

the reliability of the rules but also contributes to a more meaningful interpretation of the relationships within the data. In conclusion, the Table 4.3 illustrates the positive impact of integrating the ARBBO oversampling method into the association rule mining process. The resulting rules stand out for their increased robustness, providing valuable insights into the underlying associations within the dataset, especially in scenarios characterized by class imbalance.

4.3.1 Analysis of Results with an Imbalanced Dataset

The confusion matrix depicted in Figure 4.17 for Decision Tree (DT) reveals a higher count of False Positives (FP). Concurrently, the ROC curve illustrated in Figure 4.18 provides insight into the overall performance of DT, indicating an AUC value of 0.99. The higher count of False Positives in the confusion matrix suggests that the DT model is more prone to misclassifying instances as positive when they are actually negative. This could be attributed to factors such as model complexity or imbalances in the dataset. However, the ROC curve's AUC value of 0.99 indicates that, overall, the DT model exhibits excellent discriminative the capacity to distinguish between favorable and unfavorable instances. Despite the higher FP count, the AUC value signifies a strong ability to distinguish between the two classes. In

```
            }
            },

        oldbalanceOrg_greater:
               {
                newbalanceOrg_greater:
                   {
                     amount_less: 0,amount_greater: 0,'count':0
                 },
                newbalanceOrg_less:
                   {
                     amount_less: 0,amount_greater: 0},
                newbalanceDest_less :
                   {
                     amount_less: 0,amount_greater: 0 ,amount_equal:0
                   },
                newbalanceDest_greater:
                   {
                     amount_less: 0,amount_greater: 0, amount_equal :0
                   }
               },


            }
#This will iterate through values of each key
for i in range(0,len(conditions)):
    amount=conditions[i][1]
    oldbalanceOrg=conditions[i][2]
    newbalanceOrg= conditions[i][3]
```

Fig. 4.10 Fifth Code Segment for Rule Generation.

summary, while the confusion matrix highlights a specific challenge with False Positives, the high AUC value in the ROC curve indicates that the overall classification performance of the DT model is good.

The confusion matrix depicted in Figure 4.19 for Multilayer Perceptron (MLP) reveals a higher count of False Positives (FP) and False Negative(FN), lower count of True Negative(TN), True Positive(TP). Concurrently, the ROC curve illustrated in Figure 4.20 provides insight into the overall performance of MLP, indicating an AUC value of 0.75.

The confusion matrix presented in Figure 4.21 for k-Nearest Neighbors (KNN) reveals elevated counts of FP and FN, coupled with a lower count of True Positives (TP). These findings imply that there are difficulties for the KNN model in accurately recognizing both positive and negative cases, leading to a less favorable classification outcome. Furthermore, the Receiver Operating Characteristic (ROC) curve shown in Figure 4.18 for KNN indicates an AUC value of 0.91. The AUC value reflects the model's the capacity to distinguish between

```
import numpy as np
from sklearn.neighbors import NearestNeighbors


class ARBBOVersampling:
    def __init__(self,
                 ratio=100,
                 k_neighbors=6,
                 random_state=None):
        # check input arguments
        if ratio > 0 and ratio < 100:
            self.ratio = ratio
        elif ratio >= 100:
            if ratio % 100 == 0:
                self.ratio = ratio
            else:
                raise ValueError(
                    'ratio over 100 should be multiples of 100')
        else:
            raise ValueError(
                'ratio should be greater than 0')

        if type(k_neighbors) == int:
            if k_neighbors > 0:
                self.k_neighbors = k_neighbors
```

Fig. 4.11 Frist Code Segment for ARBBO.

positive and negative instances. While an AUC of 0.91 suggests reasonable discriminative performance, the higher counts of FP and FN in the confusion matrix indicate room for improvement in the model's precision and recall. In summary, the elevated counts of FP and FN in the confusion matrix, combined with an AUC value of 0.91 in the ROC curve, suggest that the KNN model may benefit from adjustments to enhance its classification performance, particularly in terms of minimizing false positives and false negatives.

The confusion matrix depicted in Figure 4.23 for Logistic Regression (LR) reveals elevated counts of both False Positives (FP) and True Negatives (TN). This indicates that the LR model has a higher tendency to incorrectly classify instances as positive (FP) and correctly classify instances as negative (TN), possibly leading to imbalanced classification outcomes. Moreover, the Receiver Operating Characteristic (ROC) curve illustrated in Figure 4.24 for

```
        else:
            raise TypeError(
                'Expect integer for k_neighbors')

        if type(random_state) == int:
            np.random.seed(random_state)

    def _randomize(self, samples, ratio):
        length = samples.shape[0]
        target_size = length * ratio
        idx = np.random.randint(length, size=target_size)

        return samples[idx, :]

    def _calculate_total_synthetic_samples(self, Nmaj, Nmin, theta):
        return int((Nmaj - Nmin) * theta)

    def _calculate_total_distance(self):
        total_distances = []
        for idx, neighbors in enumerate(self.knn):
            distance_sum = 0.0
            for neighbor_idx in neighbors:
                if neighbor_idx != idx:  # Exclude the sample itself
                    distance = np.linalg.norm(self.samples[idx] - self.samples[neighbor_idx])
                    distance_sum += distance
            total_distances.append(distance_sum)
```

Fig. 4.12 Second Code Segment for ARBBO.

```
def _populate(self, idx, nnarray, total_synthetic_samples):
    for i in range(self._calculate_total_synthetic_samples_for_sample(self.anomaly_ratios[idx], total_synthetic_samples)):
        nn = np.random.randint(low=0, high=self.k_neighbors)
        nearest_majority_point = self.samples[nnarray[nn]]

        # Calculate distance between minority class sample and nearest majority point
        distance = np.linalg.norm(nearest_majority_point - self.samples[idx])

        # Generate two synthetic samples inside the boundary of the circle
        synthetic_sample_pos, synthetic_sample_neg = self._generate_synthetic_samples(self.samples[idx], nearest_majority_point, distance)

        # Update synthetic samples
        self.synthetic[self.newidx] = synthetic_sample_pos
        self.newidx += 1

        self.synthetic[self.newidx] = synthetic_sample_neg
        self.newidx += 1

def _generate_synthetic_samples(self, center, nearest_majority_point, distance):
    direction_vector = np.random.rand(2)

    # Calculate synthetic sample in the positive direction
    synthetic_sample_pos = center + direction_vector * (distance - center)
```

Fig. 4.13 Third Code Segment for ARBBO.

LR displays a commendable AUC value of 0.97. The AUC value reflects the model's ability to discriminate between positive and negative instances. While an AUC of 0.97 suggests strong discriminative performance, the higher counts of FP and TN in the confusion matrix

```
        # Calculate synthetic sample in the negative direction
        synthetic_sample_neg = center + direction_vector * (center - distance)

        return synthetic_sample_pos, synthetic_sample_neg

    def _calculate_anomaly_ratio(self, total_distances):
        anomaly_ratios = []

        for total_distance in total_distances:
            exp_term = np.exp(1 / total_distance) if total_distance > 0 else 0
            anomaly_ratio = np.log(exp_term) / np.sum(total_distances)
            anomaly_ratios.append(anomaly_ratio)

        return np.array(anomaly_ratios)

    def oversample(self, samples, merge=False):
        """Perform oversampling using SMOTE
        Parameters
        ----------
        samples : list or ndarray, shape (n_samples, n_features)
            The samples to apply SMOTE to.
        merge : bool, optional (default=False)
            If set to true, merge the synthetic samples to original samples.
        Returns
```

Fig. 4.14 Fourth Code Segment for ARBBO.

signal a potential imbalance that could impact the precision and recall of the model. In summary, the elevated counts of FP and TN in the confusion matrix, coupled with an AUC value of 0.97 in the ROC curve, indicate that the LR model may benefit from adjustments to address potential imbalances and further optimize its classification performance.

The confusion matrix presented in Figure 4.25 for Random Forest (RF) exhibits a notably higher count of False Positives (FP), suggesting a tendency for the model to incorrectly classify instances as positive. Interestingly, the overall performance of RF, as indicated by the precision and recall metrics, is reported to be better than Decision Trees (DT), Logistic Regression (LR), and K-Nearest Neighbors (KNN). Furthermore, the Receiver Operating Characteristic (ROC) curve displayed in Figure 4.26 demonstrates an impressive AUC value of 1. This perfect AUC value suggests that the RF model achieves optimal discrimination between positive and negative instances. However, the higher count of FP in the confusion matrix could indicate potential areas for improvement, as FP instances contribute to misclassifications. In conclusion, while RF exhibits exceptional discriminative performance with an AUC of 1, attention may be required to address the higher count of FP observed in the confusion matrix, ensuring a more balanced and accurate classification.

```
output : ndarray
    The output synthetic samples.
"""

# Calculate total distance for each sample to its k nearest neighbors
total_distances = self._calculate_total_distance()

# Calculate anomaly ratio for each sample based on Equation (3)
anomaly_ratios = self._calculate_anomaly_ratio(total_distances)

# Calculate total synthetic samples for each minority class sample based on Equation (1)
total_synthetic_samples = self._calculate_total_synthetic_samples(
    len(self.synthetic), len(self.samples), theta
)

if type(samples) == list:
    self.samples = np.array(samples)
elif type(samples) == np.ndarray:
    self.samples = samples
else:
    raise TypeError(
        'Expect a built-in list or an ndarray for samples')

self.numattrs = self.samples.shape[1]
```

Fig. 4.15 Fifth Code Segment for ARBBO.

The confusion matrix illustrated in Figure 4.27 for the proposed Rule-Based model indicates a higher count of False Negatives (FN) and True Positives (TP), accompanied by lower counts of True Negatives (TN) and False Positives (FP). This pattern suggests that the Rule-Based model may be more sensitive to positive instances but may misclassify some negative instances. Examining the Receiver Operating Characteristic (ROC) curve in Figure 4.28, the model achieves a commendable AUC value of 0.997. This AUC value indicates strong discriminative performance, with a high probability of ranking positive instances higher than negative instances. In summary, the proposed Rule-Based model demonstrates a good ability to discriminate between positive and negative instances, as evidenced by the high AUC value. However, the higher count of FN suggests a potential area for improvement, and it would be beneficial to explore ways to reduce false negatives for a more balanced classification.

The results presented in Table 4.4 provide an extensive comparative evaluation of the suggested Rule-Based model against other existing classifiers, including DT, LR, RF, KNN, and MLP. The metrics considered for the evaluation include precision, recall, f1-score, accuracy, and ROC-AUC values.

```
        if self.ratio < 100:
            ratio = ratio / 100.0
            self.samples = self._randomize(self.samples, ratio)
            self.ratio = 100

        self.N = int(self.ratio / 100)
        new_shape = (self.samples.shape[0] * self.N, self.samples.shape[1])
        self.synthetic = np.empty(shape=new_shape)
        self.newidx = 0

        self.nbrs = NearestNeighbors(n_neighbors=self.k_neighbors)
        self.nbrs.fit(samples)
        self.knn = self.nbrs.kneighbors()[1]

        for idx in range(self.samples.shape[0]):
            nnarray = self.knn[idx]
            self._populate(idx, nnarray, total_synthetic_samples)

        if merge:
            return np.concatenate((self.samples, self.synthetic))
        else:
            return self.synthetic
```

Fig. 4.16 Sixth Code Segment for ARBBO.

The proposed Rule-Based model outperforms other classifiers in terms of precision (0.998), recall (0.450), accuracy (0.995), and ROC-AUC (0.991). This suggests that the Rule-Based model achieves a harmonious combination of accurately recognizing positive instances (precision) and accumulating a substantial amount of real positive instances (recall). The high accuracy and ROC-AUC values further indicate the model's overall effectiveness in classification.

Comparatively, other classifiers exhibit varying performance across the metrics. For instance, DT shows lower precision but higher recall, while MLP has moderate precision but very low recall. KNN and NB demonstrate competitive precision, but the Rule-Based model stands out with superior performance.

In summary, the proposed Rule-Based model showcases a strong performance across multiple metrics, making it a promising choice for handling imbalanced datasets and achieving robust classification.

The results presented in Table 4.5 offer a comparative assessment with other published and existing works, showcasing The effectiveness of the suggested Rule-Based model and

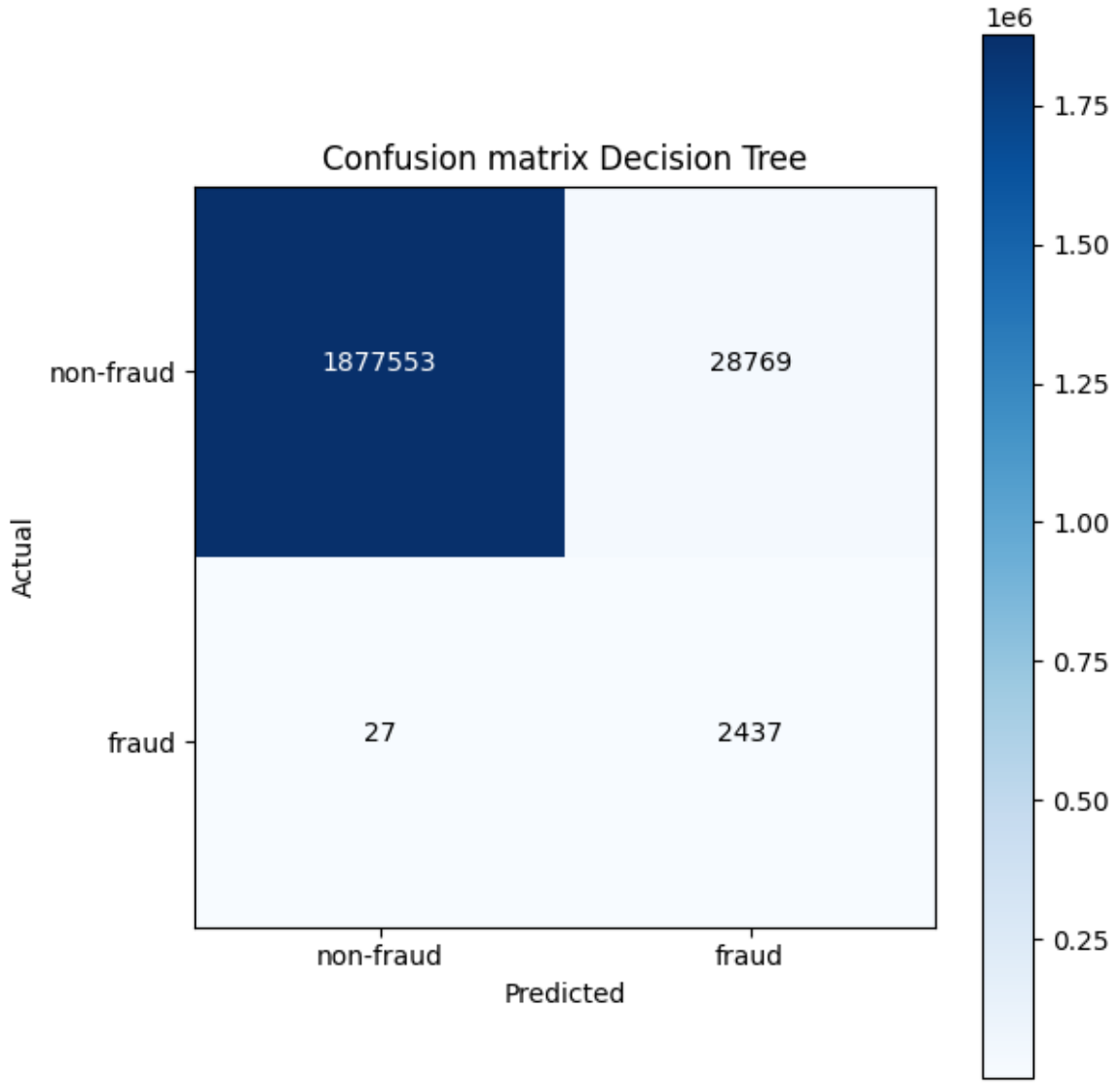

Fig. 4.17 Confusion matrix of Decision Tree classifer.

various methods. The Rule-Based model exhibits commendable performance with precision (0.998), recall (0.450), f1-score (0.995), accuracy (0.995), and ROC-AUC (0.991).

Comparatively, the Rule-Based model outperforms several existing methods. For instance, the method of Pumsirirat et al.[110] demonstrates a precision of 0.8534, recall of 0.8015, f1-score of 0.826, and accuracy of 0.9994. Local Outlier Factor et al.[111] achieves a precision of 0.8966, recall of 0.9257, f1-score of 0.9109, accuracy of 0.9095, and ROC-AUC of 0.97. Self-training LSTM prediction et al.[118] and Deep Symbolic Classification et al.[112] exhibit varying levels of precision, recall, f1-score, accuracy, and ROC-AUC.

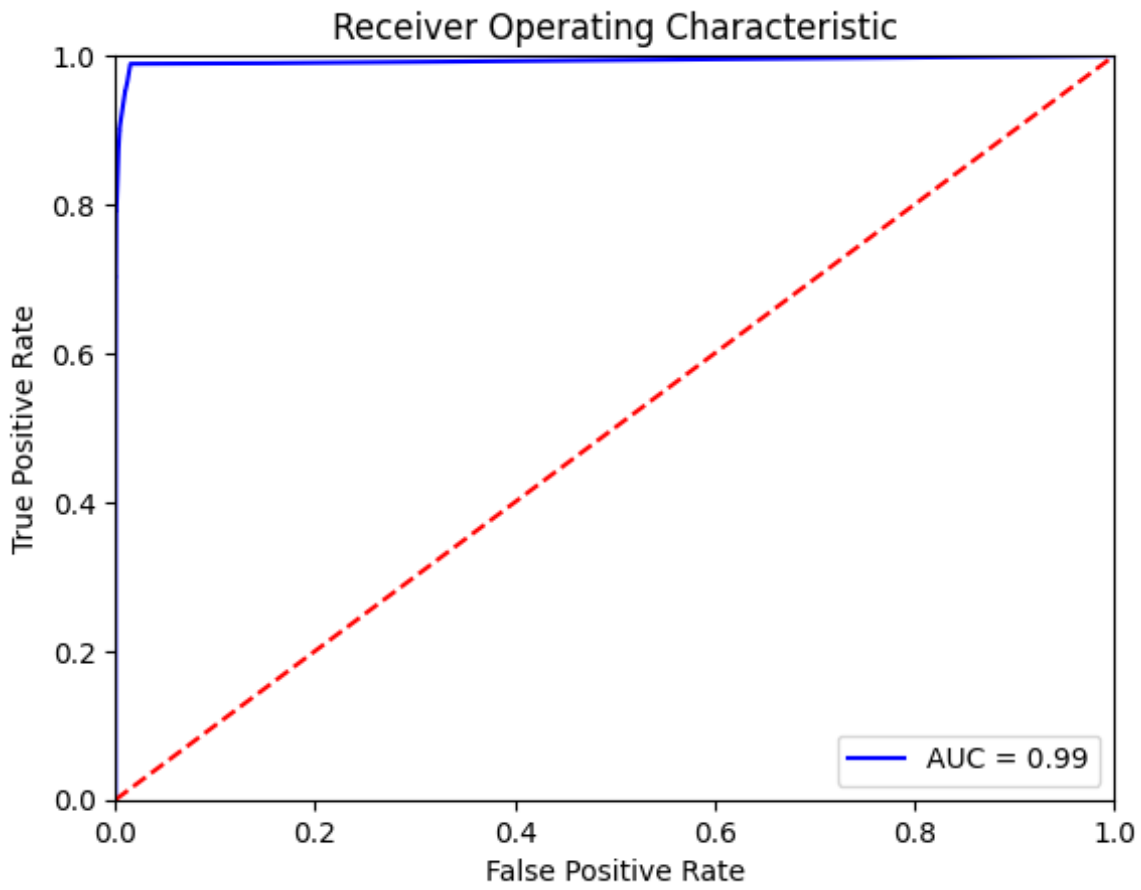

Fig. 4.18 ROC curve of Decision Tree classifer.

The proposed Rule-Based model stands out as a robust and effective approach, surpassing or matching the performance of existing methods across key evaluation metrics. This suggests its suitability for financial fraud detection tasks, making it a noteworthy contribution to the field.

4.3.2 Analysis of Results with a Balanced Dataset

The outcomes of the suggested model will be shown in this part using a balanced dataset. Figure 4.29 illustrates the confusion matrix for Decision Tree (DT) with Synthetic Minority Over-sampling Technique (SMOTE) and the proposed Anomaly Reduction Boundary Based Oversampling (ARBBO). The predictions achieved with ARBBO demonstrate superior performance compared to SMOTE, particularly in terms of TP, TN, FP, and FN. Figure 4.30 depicts the Receiver Operating Characteristic (ROC) curve. The AUC for DT with ARBBO is 0.998, showcasing a slight improvement compared to the AUC of 0.9947 achieved SMOTE.

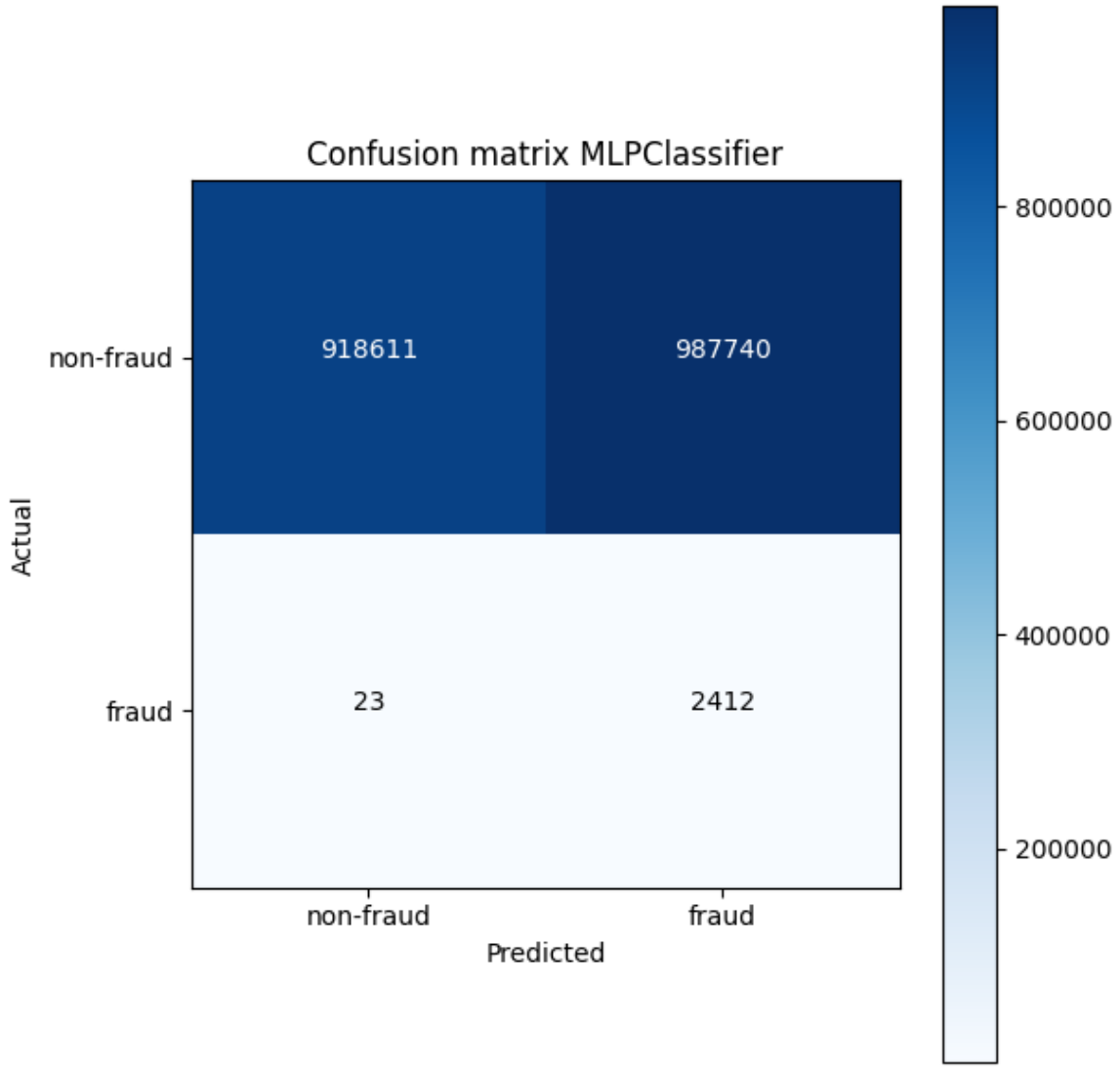

Fig. 4.19 Confusion matrix of Multilayer Perceptron classifer.

In Figure 4.31, the confusion matrix of K-Nearest Neighbors (KNN) with SMOTE and ARBBO is presented. The results with ARBBO exhibit better performance than SMOTE in TP and TN, with higher TP and slightly lower TN. The ROC curve in Figure 4.32 further emphasizes the superiority of ARBBO over SMOTE for KNN, as the AUC is 0.9969 for ARBBO, surpassing the AUC of 0.9875 achieved with SMOTE.

Figure 4.33 illustrates the confusion matrix of Logistic Regression (LR) with SMOTE and ARBBO. The predictions with ARBBO demonstrate improved performance, with TP and TN, and FP compared to SMOTE. Specifically, ARBBO achieves higher TP and slightly lower FP, resulting in an enhanced model.

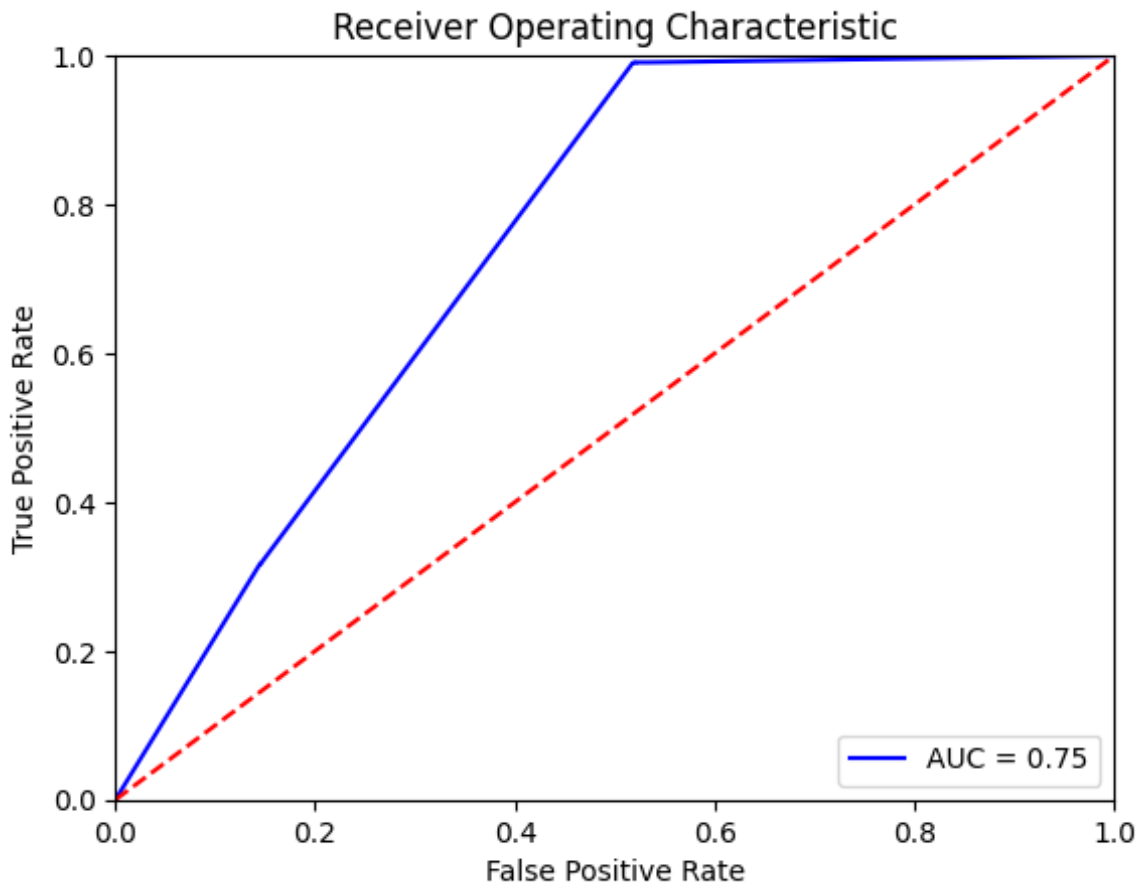

Fig. 4.20 ROC curve of Multilayer Perceptron classifer.

The ROC curve in Figure 4.34 provides further insights into the performance of LR. The AUC for LR with ARBBO is 0.888, outperforming the AUC of 0.7387 obtained with SMOTE. This signifies that the proposed ARBBO method contributes to a better AUC value, indicating improved model performance in terms of sensitivity and specificity.

Figure 4.35 displays the confusion matrix of Naive Bayes (NB) with SMOTE and the proposed ARBBO. The predictions with ARBBO exhibit higher TP and lower TN compared to SMOTE. Particularly, ARBBO achieves superior TP and slightly lower TN, suggesting improved model performance.

The ROC curve depicted in Figure 4.36 provides a visual representation of NB's performance. The AUC for NB with ARBBO is 0.754, outperforming the AUC of 0.501 obtained with SMOTE. This indicates that the proposed ARBBO oversampling method contributes to a better AUC value, signifying enhanced model performance in terms of sensitivity and specificity.

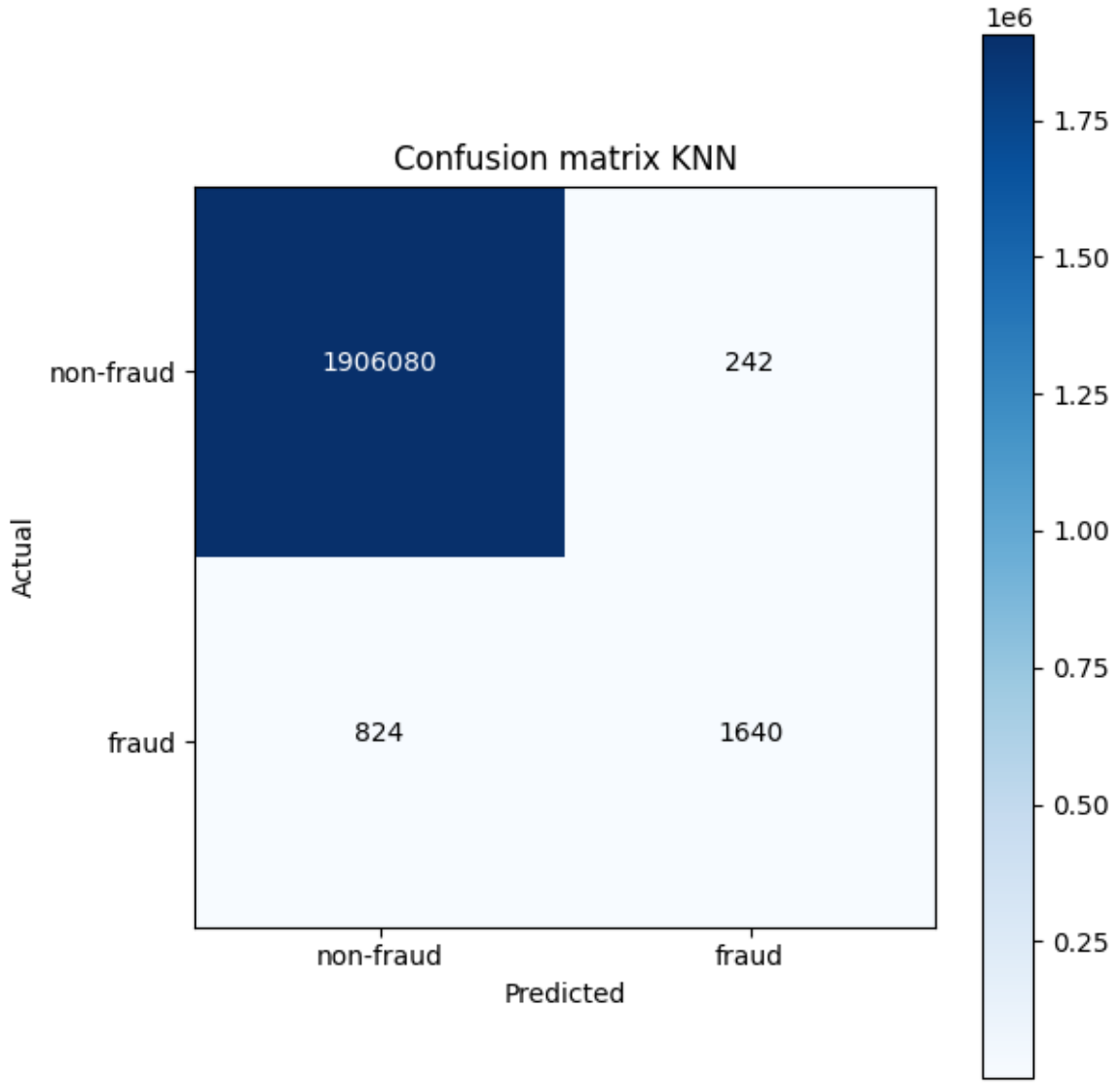

Fig. 4.21 Confusion matrix of K-Nearest Neighbors classifer.

Figure 4.37 illustrates the confusion matrix of Random Forest (RF) with SMOTE and the proposed ARBBO methods. The TN, TP, and FP values with ARBBO are observed to be superior to those with SMOTE. Specifically, TP is higher, while TN and FP are lower compared to SMOTE, indicating improved performance of the model with ARBBO.

In Figure 4.38, the ROC curve for RF with both SMOTE and ARBBO is presented. The AUC values are calculated as 0.888 and 0.738 for ARBBO and SMOTE, respectively. This substantial difference in AUC values underscores the improved functionality of the suggested ARBBO oversampling method compared to SMOTE, demonstrating its effectiveness in enhancing the entire effectiveness of the Random Forest model.

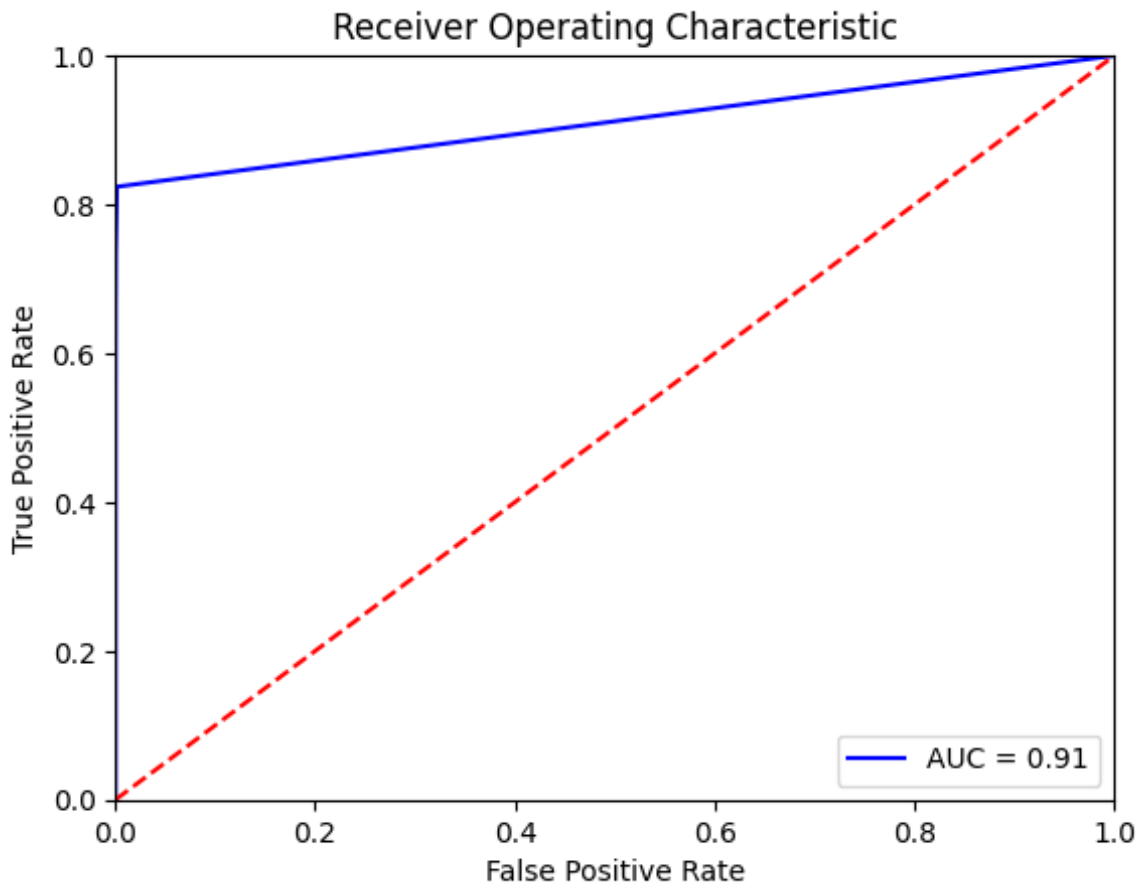

Fig. 4.22 ROC curve of K-Nearest Neighbors classifer.

Table 4.6 offers an extensive synopsis of the results obtained for Decision Tree (DT), K-Nearest Neighbors (KNN), Naive Bayes (NB), Random Forest (RF), Logistic Regression (LR), and the proposed rule-based model using both SMOTE and ARBBO oversampling methods. The metrics for assessment considered include accuracy, precision, recall, and F1-score.

The results highlight that the precision, F1-score, and accuracy of LR with ARBBO outperform those achieved with SMOTE. Similarly, RF with ARBBO demonstrates superior precision, recall, F1-score, and accuracy compared to SMOTE. KNN with ARBBO achieves remarkable recall, F1-score, and accuracy, all exceeding 0.996, showcasing its superior performance.

For NB with ARBBO, recall, F1-score, and accuracy reach 0.548, 0.690, and 0.753, respectively, indicating notable improvements over the results obtained with SMOTE.

Furthermore, DT with ARBBO exhibits excellent precision, recall, F1-score, and accuracy, all surpassing the corresponding metrics obtained with SMOTE. The proposed

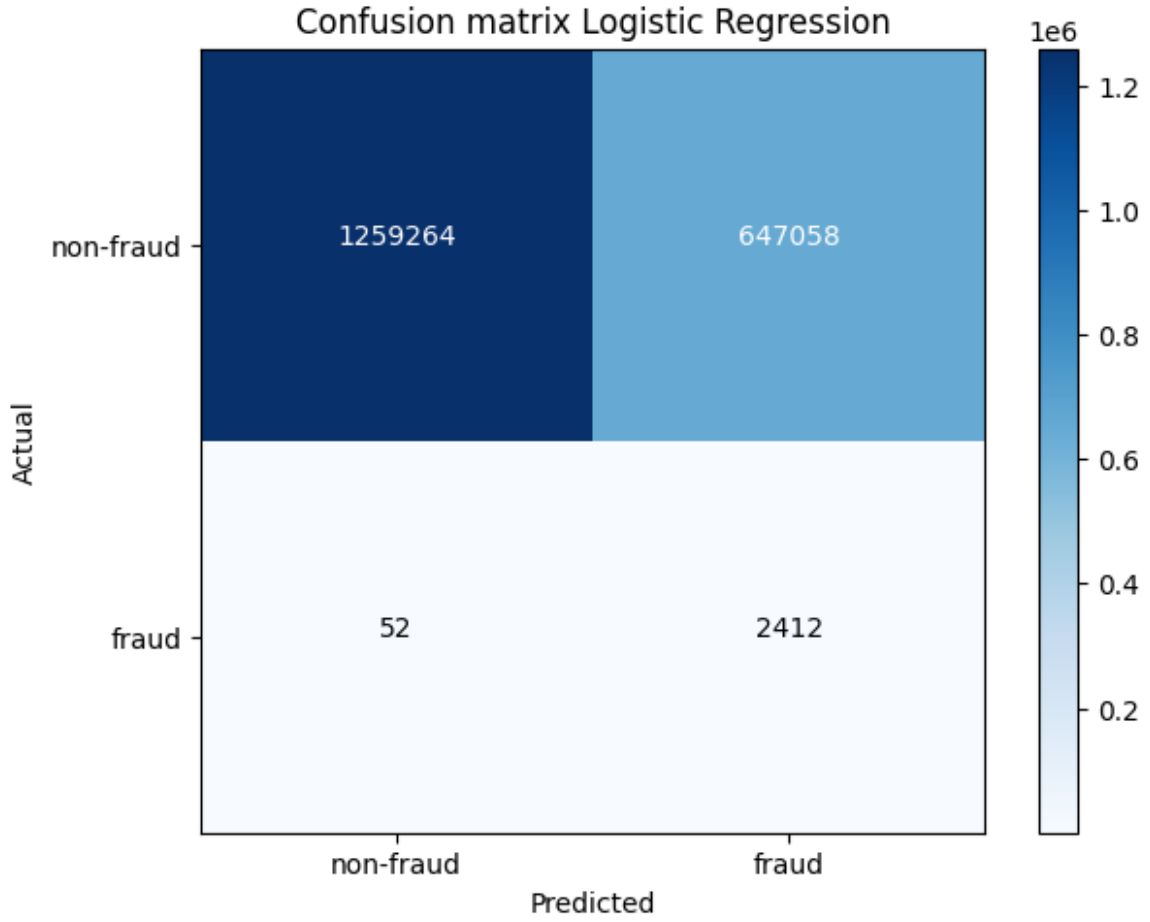

Fig. 4.23 Confusion matrix of Logistic Regression classifier.

Rule-Based model with ARBBO achieves precision, recall, F1-score, and accuracy values of 0.998, 0.995, 0.997, and 0.998, respectively, indicating its superior performance.

In conclusion, the above discussion emphasizes that both the proposed Rule-Based model and the ARBBO oversampling method outperform existing methods across various performance metrics.

Table 4.4 Result Analysis with imbalance dataset.

Classifier	Precision	Recall	F1-Score	Accuracy	ROC-AUC
Rule-Based-Model	**0.450**	**0.996**	**0.620**	**0.998**	**0.991**
DT	0.112	0.993	0.202	0.989	0.991
MLP	0.003	0.990	0.005	0.482	0.837
KNN	0.022	0.960	0.043	0.946	0.980
NB	0.019	0.365	0.038	0.976	0.779
LR	0.006	0.982	0.013	0.812	0.976

Table 4.5 Result Analysis with existing works.

Model	Precision	Recall	F1-Score	Accuracy	ROC-AUC
Rule-Based-Model	**0.450**	**0.996**	**0.620**	**0.998**	**0.991**
Method of Pumsirirat et al.[110]	0.853	0.801	0.826	0.996	
Interpretable Autoencoders et al.[111]	0.896	0.925	0.910	0.909	0.97
Self-training LSTM prediction et al.[113]	0.987	0.590	0.778	0.836	0.891
Deep Symbolic Classification et al.[112]	0.95	0.67	0.78	0.99	0.78

Table 4.6 Result Analysis with balanced dataset.

Oversampler	Classifier	Precision	Recall	F1-Score	Accuracy
ARBBO	LR	**0.912**	0.859	**0.885**	**0.888**
SMOTE	LR	0.670	0.941	0.783	0.738
ARBBO	RF	**0.998**	**0.999**	**0.998**	**0.998**
SMOTE	RF	0.997	0.996	0.996	0.996
ARBBO	KNN	0.995	**0.998**	**0.996**	**0.996**
SMOTE	KNN	0.997	0.977	0.987	0.987
ARBBO	NB	0.930	**0.548**	**0.690**	**0.753**
SMOTE	NB	0.999	0.451	0.621	0.725
ARBBO	DT	**0.998**	**0.999**	**0.998**	**0.998**
SMOTE	DT	0.995	0.994	0.995	0.995
ARBBO	MLP	**0.993**	**0.992**	**0.993**	**0.993**
SMOTE	MLP	0.852	0.996	0.707	0.823
SMOTE	Rule-Based	0.995	0.994	0.995	0.995
ARBBO	Rule-Based	**0.998**	**0.995**	**0.997**	**0.998**

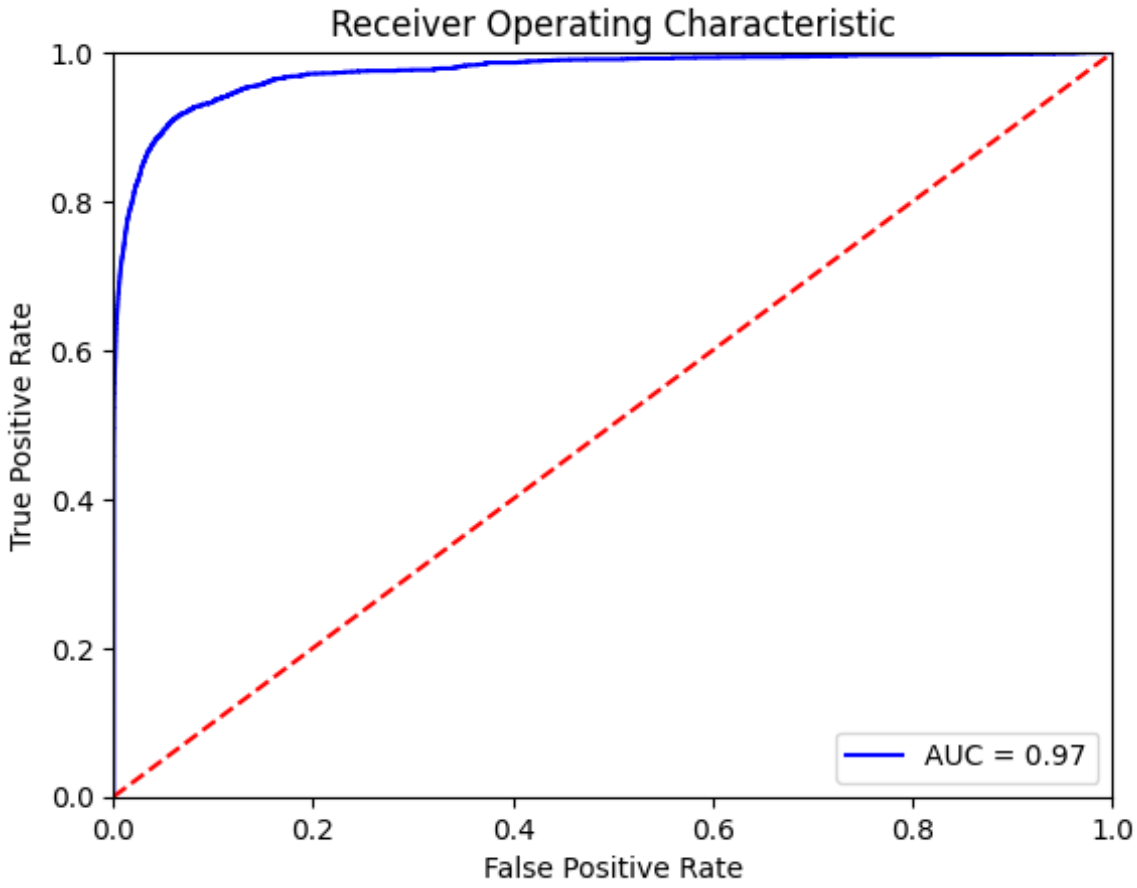

Fig. 4.24 ROC curve of Logistic Regression classifier.

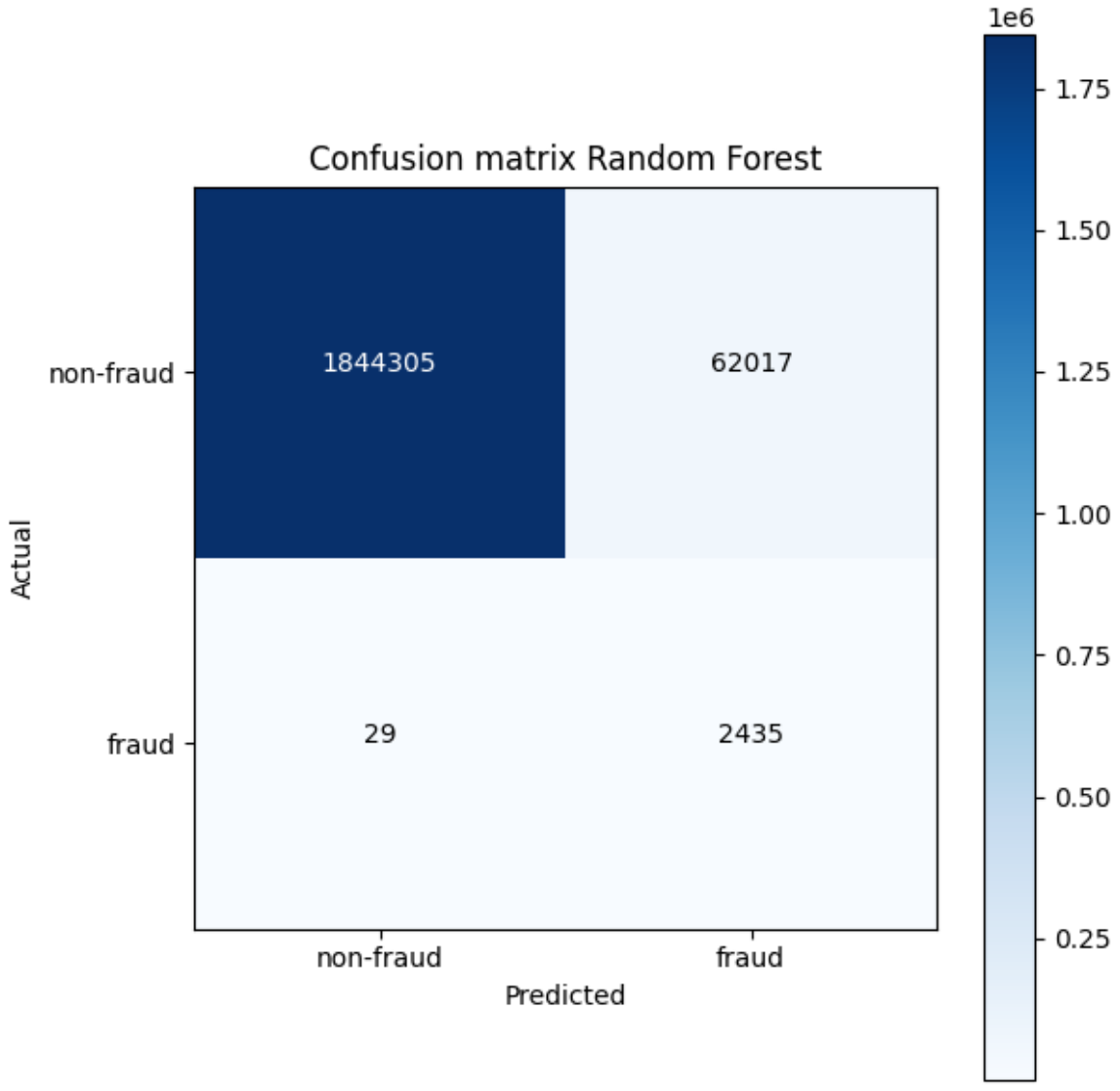

Fig. 4.25 Confusion matrix of Random Forest classifier.

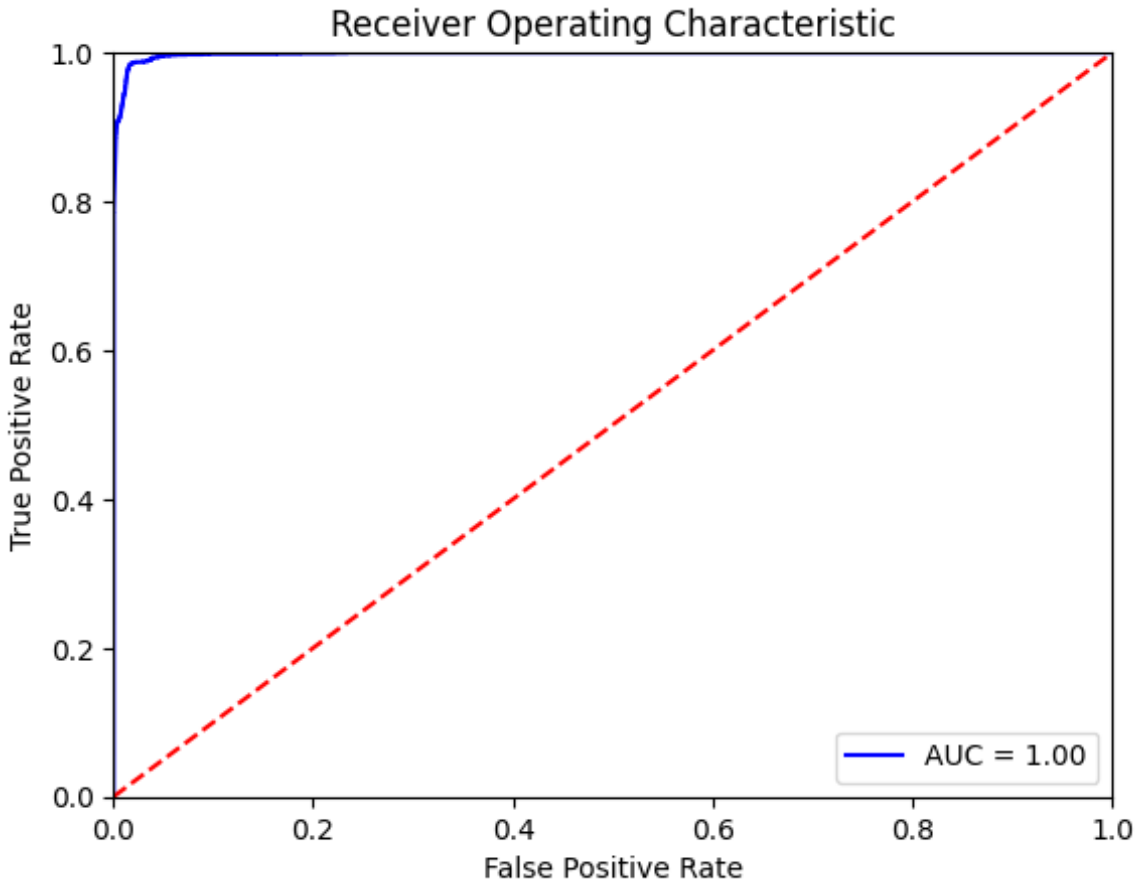

Fig. 4.26 ROC curve of Random Forest classifier.

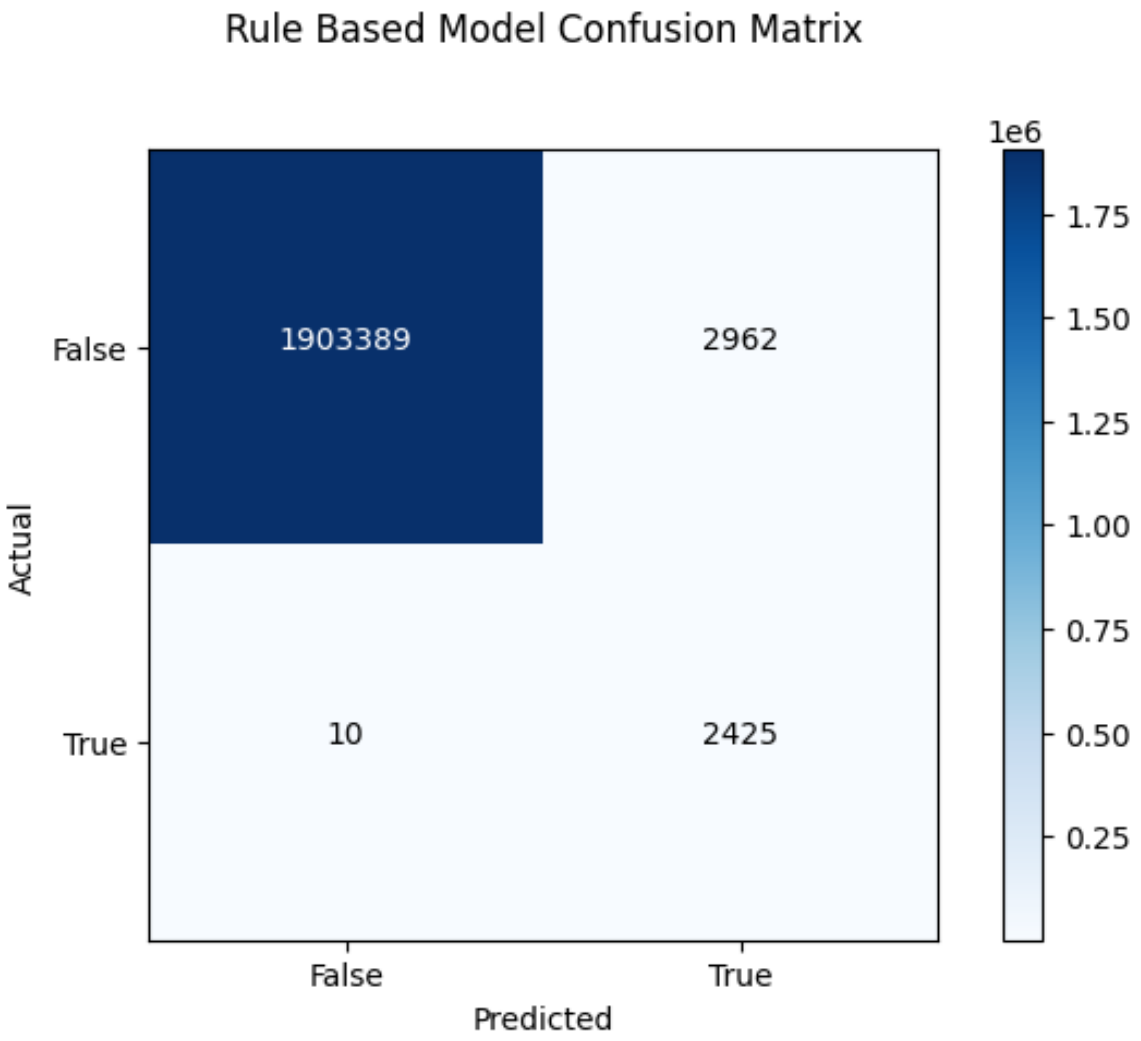

Fig. 4.27 Confusion matrix of proposed Rule-Based model.

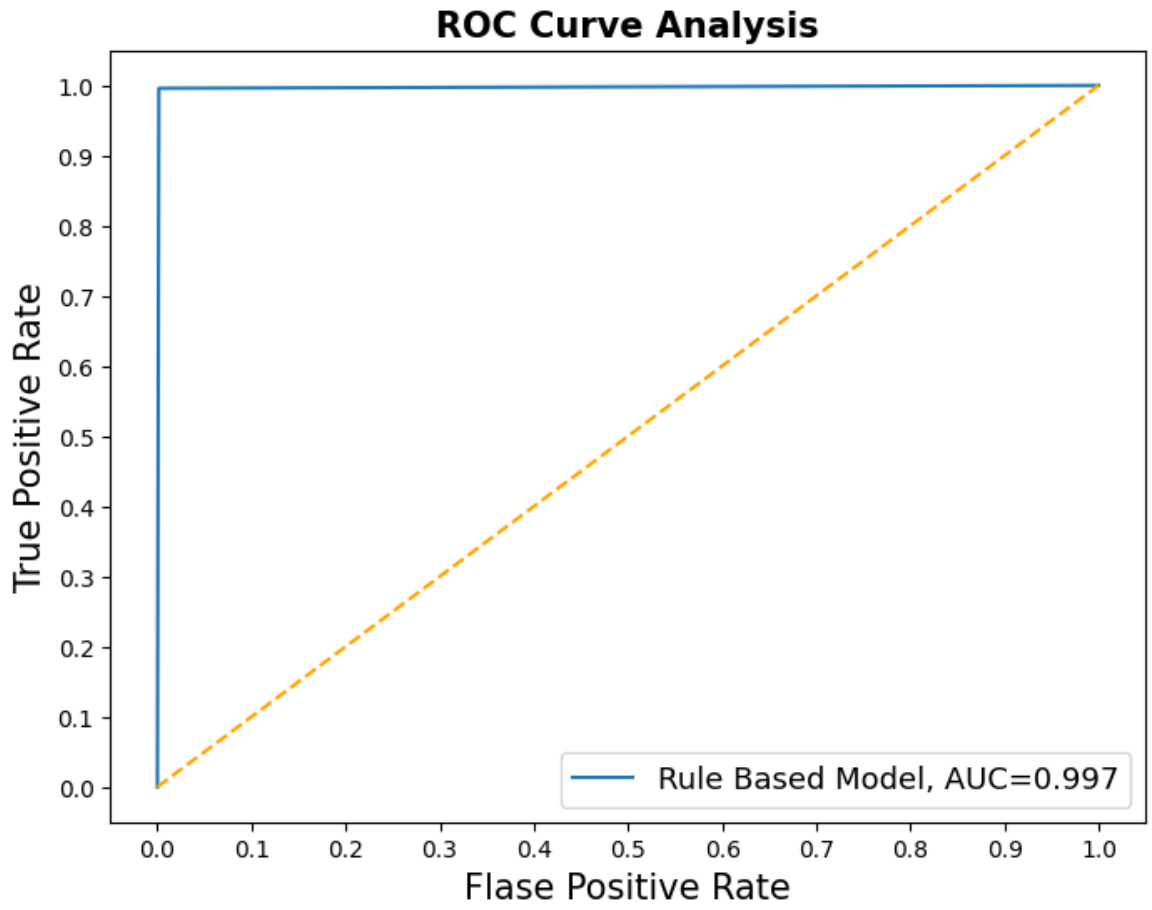

Fig. 4.28 ROC curve of proposed Rule-Based model.

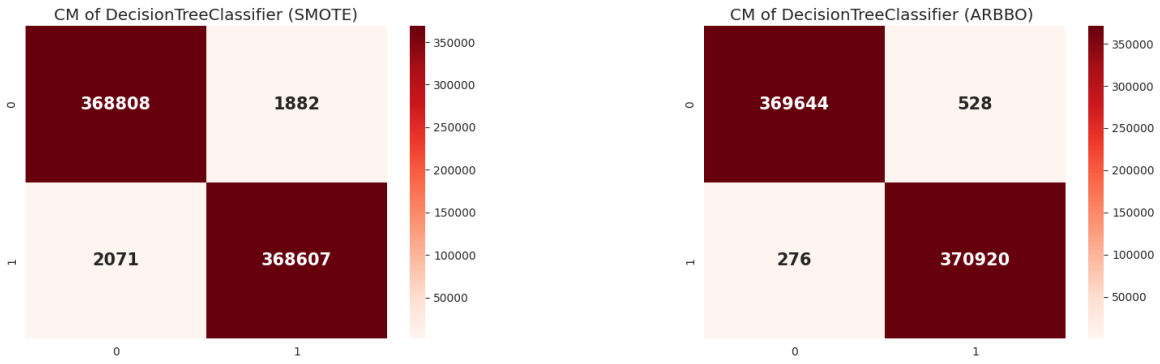

Fig. 4.29 Confusion matrix of DT classifier with SMOTE and proposed ARBBO.

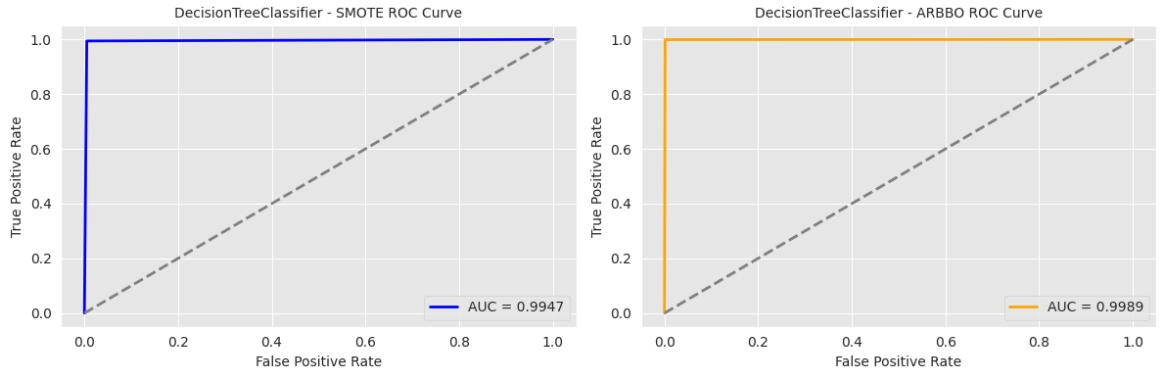

Fig. 4.30 ROC curve of DT classifier with SMOTE and proposed ARBBO.

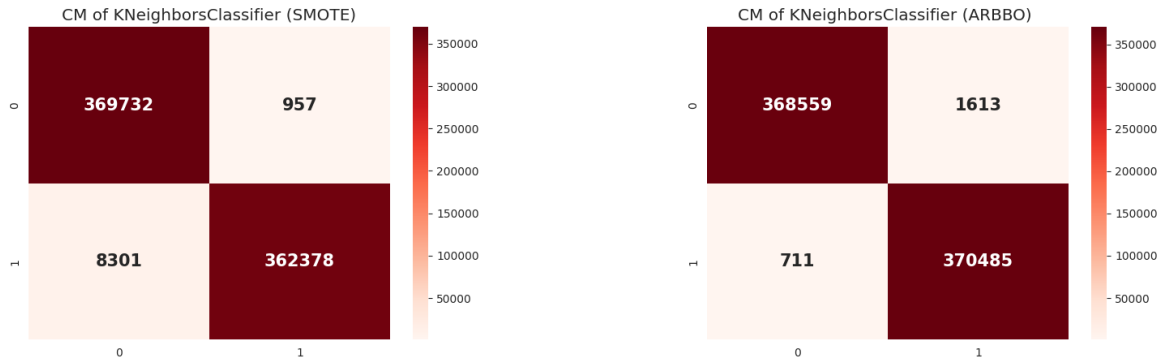

Fig. 4.31 Confusion matrix of KNN classifier with SMOTE and proposed ARBBO.

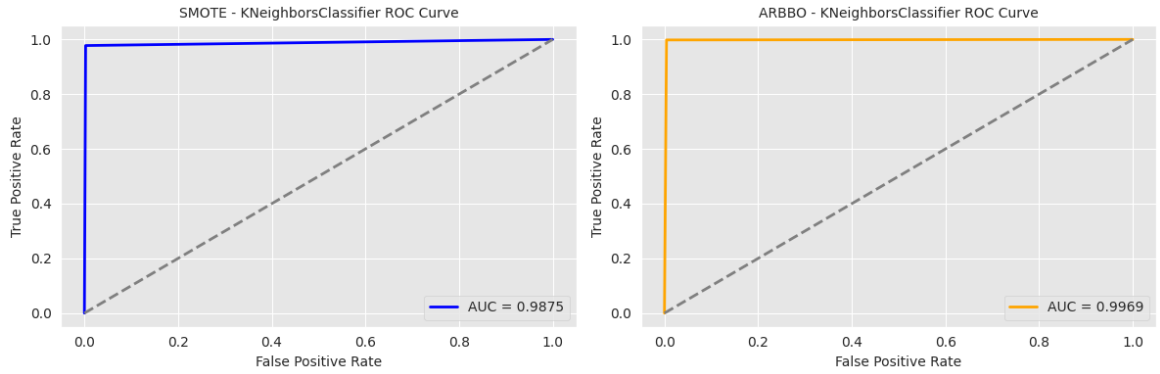

Fig. 4.32 ROC curve of KNN classifier with SMOTE and proposed ARBBO.

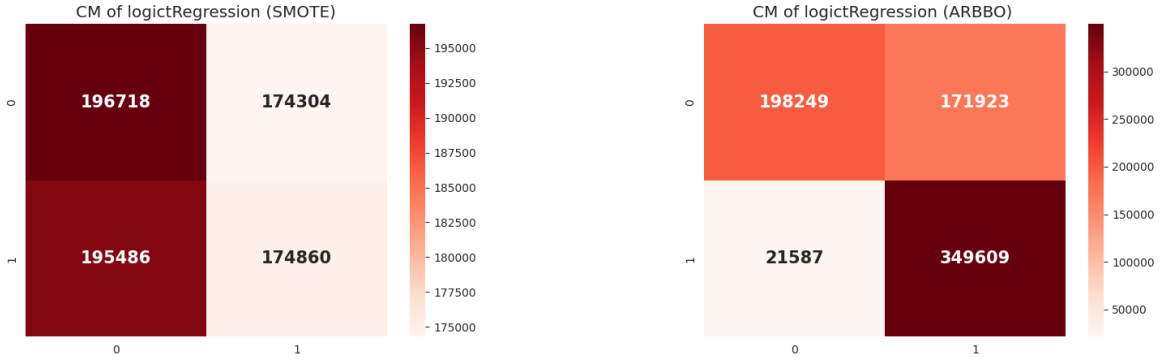

Fig. 4.33 Confusion matrix of LR classifier with SMOTE and proposed ARBBO.

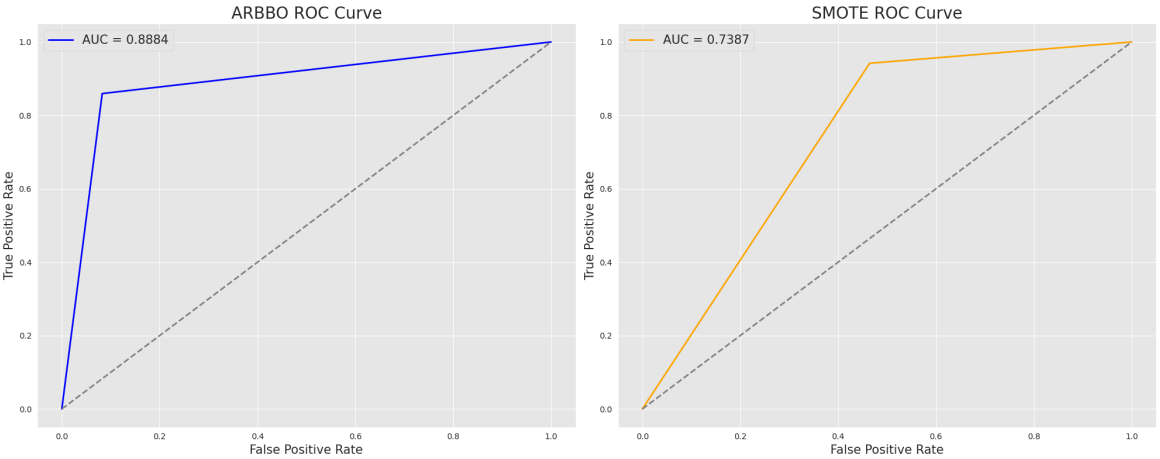

Fig. 4.34 ROC curve of LR classifier with SMOTE and proposed ARBBO.

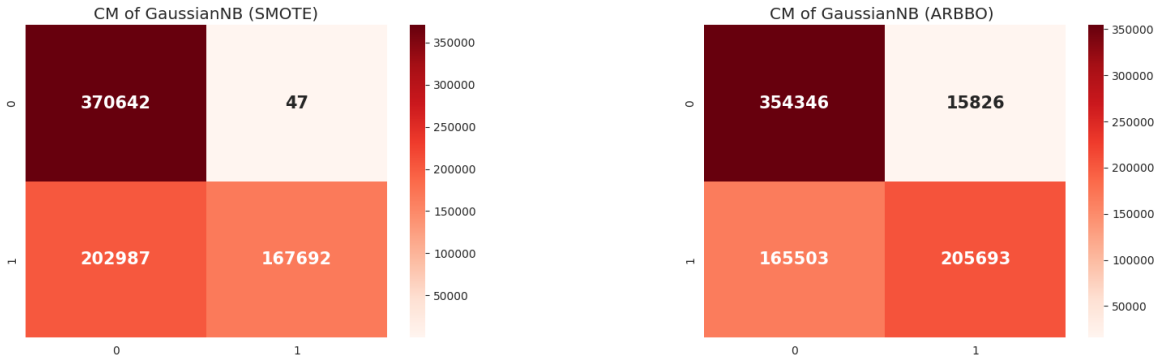

Fig. 4.35 Confusion matrix of NB classifier with SMOTE and proposed ARBBO.

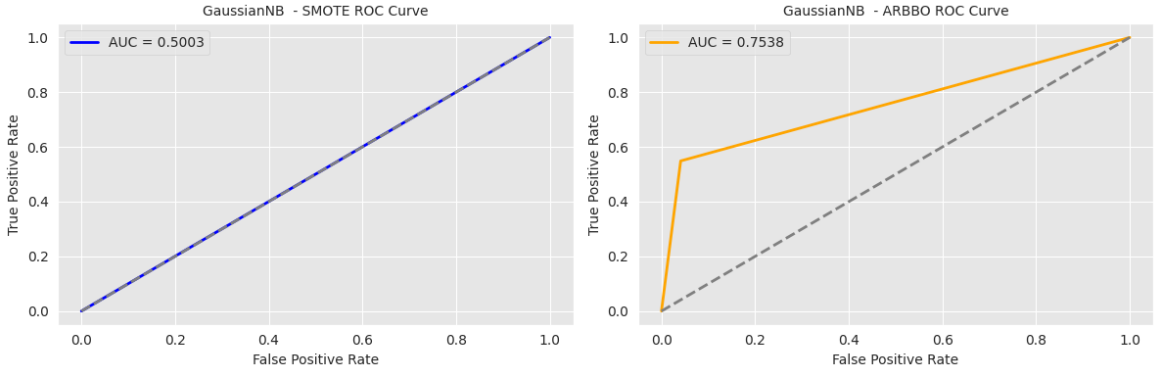

Fig. 4.36 ROC curve of NB classifier with SMOTE and proposed ARBBO.

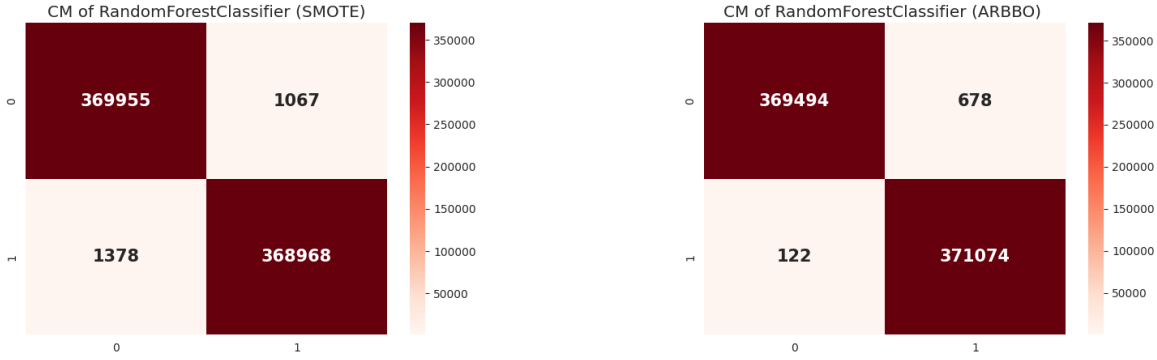

Fig. 4.37 Confusion matrix of RF classifier with SMOTE and proposed ARBBO.

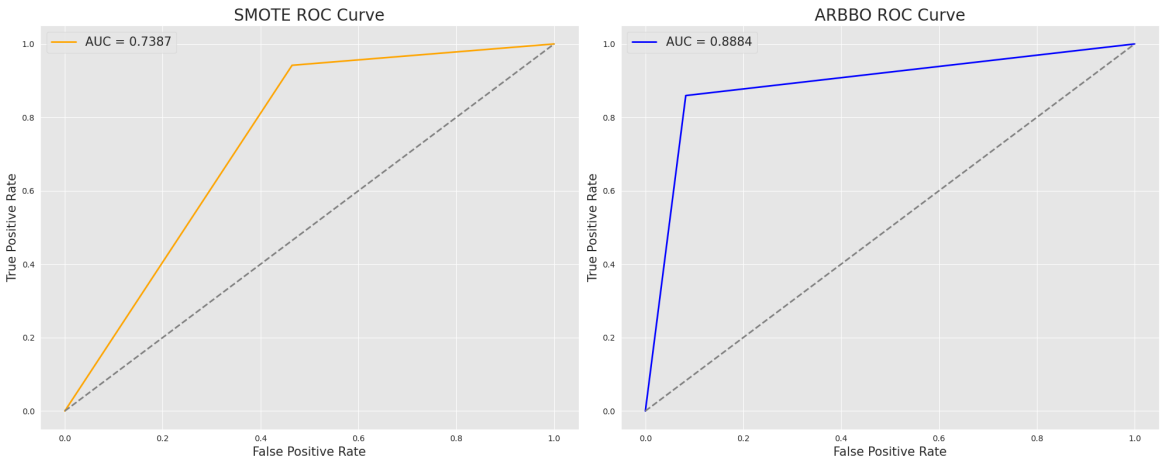

Fig. 4.38 ROC curve of RF classifier with SMOTE and proposed ARBBO.

Chapter 5

Conclusion and Future Direction

5.1 Answer of Research Questions

Question 1: How to balance imbalance data before rule generation?
Answer: We have proposed anomaly reduction boundary based oversampling technique (ARBBO) to address the class imbalance issue.

Question 2: How to effectively generate data-driven rules from financial data?
Answer: We have proposed consecutive sequence based relational rules to detect financial fraud.

5.2 Conclusion

In conclusion, the pervasive threat of financial fraud, affecting both individuals and commercial organizations, continues to impose significant economic burdens, amounting to billions of dollars annually. Among the various forms of fraud, financial fraud stands out as the most prevalent and costly, triggering widespread concern. Despite the alarming nature of this challenge, recent advancements in machine learning have proven instrumental in detecting and mitigating financial fraud, although issues of class inequality have posed significant obstacles.

This research has contributed an effective method for detecting financial fraud, addressing the critical issue of imbalanced datasets. The utilization of the ARBBO (Adaptive Randomized Borderline Over-sampling) approach was the first step, enabling the construction of a balanced dataset. Subsequently, a novel rule-based model based on relational associative

rules was developed. The findings of the experiment showed that the integration of proposed rule-based fraud detection with the ARBBO method significantly improved detection accuracy, outperforming existing methods and showcasing robust performance across various input data scenarios.

The proposed method achieved a commendable detection level of 99%, closely mirroring real-world conditions. This success indicates its potential applicability in diverse financial fraud scenarios, encompassing bank transactions, insurance transactions, share market transactions, and credit/debit card transactions. The combination of the ARBBO data resampling technique and the rule-based model emerges as a productive method for spotting financial transaction fraud.

Looking ahead, future research endeavors will focus on extending and refining the proposed method by incorporating more recent methods to cut down on time required for rule generation and classification processes. The goal of this ongoing effort is to improve the financial fraud detection model's scalability and efficiency, which will aid in the continuous fight against this persistent and changing threat.

5.3 Future Directions

Considering the positive results obtained with the suggested approach to financial fraud detection, there exist compelling avenues for future research and enhancements. The following directions will be pursued to further refine and extend the capabilities of the fraud detection model:

5.3.1 Optimizing Rule Generation and Classification Processes

One key area for improvement involves optimizing the time required for rule generation and classification processes. The current methodology has proven effective, but efforts will be directed towards streamlining and accelerating these processes. The goal of this optimization is to improve the model's efficiency, especially in situations when there are big and changing datasets.

5.3.2 Incorporating Advanced Techniques

Future research endeavors will explore the application of cutting-edge methods in machine learning and data analysis. Incorporating cutting-edge methodologies may improve the generalizability and accuracy of the model. The capacity of methods like ensemble methods

and deep learning to extract complex patterns and relationships from financial datasets will be taken into consideration.

5.3.3 Scalability and Adaptability

To ensure the applicability of the fraud detection model in various domains and under evolving conditions, scalability and adaptability will be focal points of future developments. Efforts will be directed towards designing the model to accommodate diverse financial transaction scenarios, including emerging trends and technologies.

5.3.4 Real-time Fraud Detection

An essential aspect of future research involves exploring real-time fraud detection capabilities. The integration of real-time monitoring and analysis will enable the model to promptly identify and respond to fraudulent activities as they occur. This enhancement is crucial in addressing the dynamic nature of financial fraud.

5.3.5 Collaborative Research and Industry Integration

To validate and enhance the proposed method, collaborative research initiatives will be pursued. Engaging with industry partners, financial institutions, and cybersecurity experts will provide valuable insights and real-world data for continuous refinement. Establishing partnerships with relevant stakeholders ensures the model's alignment with industry needs and standards.

These future directions collectively aim to fortify the proposed financial fraud detection model, contributing to its ongoing evolution and effectiveness in combating the persistent and evolving threat of fraudulent activities within the realm of finance.

References

[1] V. K. Shukla F. Beena, I. Mearaj and S. Anwar. Mitigating financial fraud using data science—'a case study on credit card frauds,'. *IEEE Access*, 6:14277–1428, 2021.

[2] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi. Credit card fraud detection using adaboost and majority voting. *IEEE Access*, 6(2169-3536):14277 – 14284, 2018.

[3] C. Guangquan H. Tingfei and H. Kuihua. Using variational auto encoding in credit card fraud detection. *IEEE Access*, 8:149841–149853, 2020.

[4] Oxford Learner's Dictionaries. Oxford learner's dictionaries. *https://www.oxfordlearnersdictionaries.com/definition/english/fraud*, 2021. Accessed: Oct. 26, 2021.

[5] M. Zareapoor and J. Yang. A novel strategy for mining highly imbalanced data in credit card transactions. *Intelligent Automation and Soft Computing*, 23(4):1–7, 2017.

[6] M. Óskarsdóttir, C. Bravo, C. Sarraute, J. Vanthienen, and B. Baesens. The value of big data for credit scoring: Enhancing financial inclusion using mobile phone data and social network analytics. *Intelligent Automation and Soft Computing*, 74(4):26–39, 2019.

[7] Federal Trade Commission. Credit card fraud statistics to know in 2021. 2022.

[8] V. Lee C. Phua, R. Gayler and K. Smith-Miles. On the communal analysis suspicion scoring for identity crime in streaming credit applications. *European Journal of Operational Research*, 195(2):595–612, 2009.

[9] R. Bolton and D. Hand. Statistical fraud detection: A review. *Statistical Fraud Detection*, 17(3):235–249, 2002.

[10] M. Lahby Y. Abakarim and A. Attioui. An efficient real-time model for credit card fraud detection based on deep learning. *Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications*, pages 1–7, 2018.

[11] Y. Sahin and E. Duman. Detecting credit card fraud by ann and logistic regression. *2011 International Symposium on Innovations in Intelligent Systems and Applications*, 2011.

[12] N. K. Gyamfi and J.D. Abdulai. Bank fraud detection using support vector machine. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 37–41, 2018.

[13] S. Patil, H. Somavanshi, J. Gaikwad, A. Deshmane, and R. Badgujar. Credit card fraud detection using decision tree induction algorithm. *International Journal of Computer Science and Mobile Computing*, 4(4):92–95, 2015.

[14] W. N. Robinson and A. Aria. Sequential fraud detection for prepaid cards using hidden markov model divergence. *Expert Systems with Applications*, 91:235–251, 2018.

[15] A. A. Taha and S. J. Malebary. An intelligent approach to credit card 565 fraud detection using an optimized light gradient boosting machine. *IEEE Access*, 8:25579–25587, 2020.

[16] P. Raghavan and N. E. Gayar. Fraud detection using machine learning and deep learning. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, pages 334–339, 2019.

[17] A. LaTorre D. Molina and F. Herrera. Shade with iterative local search for large-scale global optimization. *2018 IEEE Congress on Evolutionary Computation (CEC)*, 49(8):1–8, 2018.

[18] H.J. Kim J. Kim and H. Kim. Fraud detection for job placement using hierarchical clusters-based deep neural networks. *Int. J. Speech Technol*, 49(8):2842–2861, 2019.

[19] B. Bandaranayake. Fraud and corruption control at education system level: A case study of the victorian department of education and early childhood development in australia. *J. Cases Educ. Leadership*, 17(4):34–53, 2014.

[20] V.N.Dornadula and S. Geetha. Credit card fraud detection using machine learning algorithms. *Procedia Computer Science*, 165:631–641, 2019.

[21] Buchanan and B.G. Artificial intelligence in finance. *Zenodo : London, UK*, 2019.

[22] T. Langsdorf R. Brause and M. Hepp. Neural data mining for credit card fraud detection. *Proceedings 11th International Conference on Tools with Artificial Intelligence*, 2002.

[23] C. Rudin Y. Shaposhnik S. Wang C. Chen, K. Lin and T. Wang. An interpretable model with globally consistent explanations for credit risk. *arXiv*, 2018.

[24] A.Pumsirirat and L.Yan. Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 9(1), 2018.

[25] A. Fernández. Artificial intelligence in financial services. *Banco de España: Madrid, Spain*, 2019.

[26] H. Yan and S. Lin. Machine learning in uk financial services. *Bank of England: London, UK*, 2019.

[27] H. Yan and S. Lin. New trend in fintech: Research on artificial intelligence model interpretability in financial fields. *Open Journal of Applied Sciences*, 2019.

[28] L. D. Wall. Some financial regulatory implications of artificial intelligence. *Journal of Economics and Business*, 100:55–63, 2018.

[29] M. Alhaisoni R. Damaševičius R. Scherer A. Rehman M. A. Khan, I. Ashraf and S. A. C. Bukhari. Multimodal brain tumor classification using deep learning and robust feature selection: A machine learning application for radiologists. *Diagnostics (Basel)*, 10:1–19, 2020.

[30] J. A. Cruz and D. S. Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer Inform*, 2:59–77, 2006.

[31] O. Caelen C. Alippi A. D. Pozzolo, G. Boracchi and G. Bontempi. Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(4):3784 – 3797, 2017.

[32] Y. Taher R. Haque M.S. Hacid S. Makki, Z. Assaghir and H. Zeineddine. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Transactions on Neural Networks and Learning Systems*, 7(18843354):93010 – 93022, 2019.

[33] Y. Taher R. Haque M.S. Hacid S. Makki, Z. Assaghir and H. Zeineddine. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Transactions on Neural Networks and Learning Systems*, 7(18843354):93010 – 93022, 2019.

[34] R. A. Johnson A. D. Pozzolo, O. Caelen and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. *2015 IEEE Symposium Series on Computational Intelligence*, 7(15718469):159–166, 2015.

[35] G. T. Reddy C. Iwendi A. K. Bashir H. Patel, D. S. Rajput and O. Jo. A review on classification of imbalanced data for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 16(1550147720916404):159–166, 2020.

[36] J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal Big Data*, 6(1):1–54, 2019.

[37] J. M. Johnson and T. M. Khoshgoftaar. A survey of predictive modeling on imbalanced domains. *ACM computing serveys*, 49(2):1–50, 2016.

[38] J.Shang G.Mingyun H.Yuanyue G.Haixiang, L.Yijing and G.Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73(2):220–239, 2017.

[39] L. O. Hall N. V. Chawla, K. W. Bowyer and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Artificial Intelligence*, 16(2):321–357, 2002.

[40] W. Y. Wang H. Han and B. H. Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *In Proceedings of the Conference on Aristophanes Upstairs and Downstairs, Magdalen Coll, Oxford, UK*, pages 16–18, 2005.

[41] L. O. Hall N. V. Chawla, K. W. Bowyer and W. P. Kegelmeyer. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. *In Proceedings of the International Joint Conference on Neural Networks, Hong Kong, China*, pages 1–8, 2008.

[42] X. W. Liang, A. P. Jiang, T. Li, Y. Y. Xue, and G. T. Wang. Lr-smote — an improved unbalanced data set oversampling based on k-means and svm. *Knowledge-Based Systems*, 196(105845), 2020.

[43] S. J. Yen and Y. S. Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. *In Proceedings of the International Conference on Intelligent Computing, Kunming, China*, pages 16–19, 2006.

[44] S. J. Yen and Y. S. Lee. Improving identification of difficult small classes by balancing class distribution. *In Proceedings of the 8th Conference on Artificial Intelligence in Medicine in Europe, Cascais, Portugal*, pages 63–66, 2001.

[45] C. M. Bishop. Pattern recognition and machine learning. *Information Science and Statistics*, 4, 2006.

[46] W. Daelemans and A. V. d. Bosch. Memory-based language processing. *Cambridge University Press*, 2005.

[47] D. T. Larose and C.D. Larose. Discovering knowledge in data: an introduction to data mining. *John Wiley Sons*, 4, 2014.

[48] M. Maire H. Zhang, A. C. Berg and J. M. hat. In 2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06). *John Wiley Sons*, 2:2126–2136, 2006.

[49] R. Gandhi. Support vector machine - introduction to machine learning algorithms. *https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47*, 2020.

[50] S. S. Keerthi K. Duan and A. N. Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[51] E. Kremic and A. Subasi. Performance of random forest and svm in face recognition. *The International Arab Journal of Information Technology*, 13(2):287–293, 2016.

[52] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[53] Z. Li L. Zheng S. Wang S. Xuan, G. Liu and C. Jiang. Random forest for credit card fraud detection. *2018 IEEE 15th International Conference on Networking*, pages 1–6, 2018.

[54] G. John and P. Langley. Estimating continuous distributions in bayesian classifiers. *arXiv preprint arXiv*, page 338–345, 1995.

[55] O. Adetunmbi J. Awoyemi and S. Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 2017.

[56] J. James. Bayes' theorem. *Stanford Encyclopedia of Philosophy*, 2003.

[57] Y. Sahin and E. Duman. Detecting credit card fraud by ann and logistic regression. *In 2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 315–319, 2011.

[58] D. Singh S. Kalyanakrishnan and R. Kant. On building decision trees from large-scale data in applications of online advertising. *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 669–678, 2014.

[59] Donna Fuscaldo. Financial fraud, investopedia, https://www.investopedia.com/financial-fraud- 4689710. *Investopedia*, 2022.

[60] M. Bhattacharya J. West and R. Islam. Intelligent financial fraud detection practices: An investigation. *School of Computing Mathematics, School of Computing Mathematics,Charles Sturt University*, 2015.

[61] S. A. Gadsden W. Hilal and J. Yawney. Financial fraud: A review of anomaly detection techniques and recent advances, expert systems with applications. *Expert Systems With Applications*, 2022.

[62] K. Vishwakarma. Credit card fraudulent detection using machine learning. *International Research Journal of Engineering and Technology (IRJET)*, 7(9), 2020.

[63] H. Bodepudi. Credit card fraud detection using unsupervised machine learning algorithms. *International Journal of Computer Trends and Technology*, 69(8):1–3, 2021.

[64] J. Ulzheimer. Credit card fraud detection using unsupervised machine learning algorithms lecture. *How Do Credit Card Issuers Detect Fraud? - Credit Card Insider. YouTube. 2015. URL: https://www.youtube.com/watch?v=N3LjCsVYQ-k*, 2015.

[65] J. Yadav K. Chaudhary and B. Mallick. A review of fraud detection techniques: Credit card. *International Journal of Computer Applications*, 45(1):39–44, 2012.

[66] H. Ghodosi J. Pieprzyk and E. Dawson. Information security and privacy. *Proceedings of 12th Australasian Conference*, 4586:1–11, 2007.

[67] A. Hussein L. Delamaire and J. Pointon. Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, 4(2):57–68, 2009.

[68] P. Richhariya and P. Singh. A survey on financial fraud detection methodologies. *International Journal of Computer Applications*, 45(22):15–22, 2012.

[69] K. Joshi. A review of credit card fraud detection techniques in e-commerce. *Academic Journal of Forensic Sciences*, 1(1), 2018.

[70] V. Lee C. Phua, R. Gayler and K. Smith. On the approximate communal fraud scoring of credit applications. *Proceedings of Credit Scoring and Credit Control*, pages 1–10, 2005.

[71] O. Caelen T. Eliassi-Rad L. Akoglu M. Snoeck V. V. Vlasselaer, C. Bravo and B. Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Proceedings of Credit Scoring and Credit Control*, 75:38–48, 2015.

[72] P. Pintelas S. Kotsiantis, D. Kanellopoulos. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[73] C. Jun W. Lee and J. Lee. Instance categorization by support vector machines to adjust weights in adaboost for imbalanced data classification. *Information Sciences*, 381:92–103, 2017.

[74] M. Galar M. Jesus A. Fern´aNdez, V. L´oPez and F. Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110, 2013.

[75] J. an Luengo E. Bernad´o-Mansilla A. Fern´andez, S. Garc´ıa and F.Herrera. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *IEEE Transactions on Evolutionary Computation*, 14(6):913 – 941, 2010.

[76] S. Garc´ıa V. Palade V. L´opez, A. Fern´andez and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[77] K. Sinapiromsaran C. Bunkhumpornpat and C. Lursinsap. Safelevel-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. *Pacific-Asia conference on knowledge discovery and data mining*, 16:475–482, 2009.

[78] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. *Pacific-Asia conference on knowledge discovery and data mining*, pages 1049–001, 1999.

[79] R. Polikar. Ensemble-based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

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

[81] Expert.ai Team. Expert system team. what is machine learning? *https://www.expert.ai/blog/machine-learning-definition/*, 2022.

[82] M. A. Maarof A. Abdallah and A. Zainal. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90–113, 2016.

[83] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263 – 1284, 2009.

[84] V. Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4), 2012.

[85] L. O. Hall N. V. Chawla, A.Lazarevic and K. W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. *Lecture Notes in Computer Science*, 2838(4):107–119, 2003.

[86] E. W. Cooper H. M. Nguyen and K. Kamei. Borderline over-sampling for imbalanced data classification. *nternational Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

[87] H. Wang J. Wang, M. Xu and J. Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. *2006 8th International Conference on Signal Processing*, 2006.

[88] A. A. Taha and S. J. Malebary. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access*, 8(2169-3536):25579 – 25587, 2020.

[89] Y. Taher R. Haque M. Hacid S. Makki, Z. Assaghir and H. Zeineddine. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access*, 7:93010–93022, 2020.

[90] M. A. Mahraz A. El Yahyaouy F. Z. El Hlouli, J. Riffi and H. Tairi. Credit card fraud detection based on multilayer perceptron and extreme learning machine architectures. *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, page 1–5, 2020.

[91] G. Liu L. Zheng C. Jiang, J. Song and W. Luan. Credit card fraud detection: a novel approach using aggregation strategy and feedback mechanism. *IEEE Internet of Things Journal*, 5(5):3637–3647, 2018.

[92] E. E. D. Hemdan A. A. E. Naby and A. El-Sayed. Deep learning approach for credit card fraud detection. *2021 International Conference on Electronic Engineering (ICEEM)*, 2021.

[93] Y. Sun E. Ileberi and Z. Wang. A machine learning based credit card fraud detection using the ga algorithm for feature selection. *Journal of Big Data*, 9(1):24, 2022.

[94] J. L. Leevy Z. Salekshahrezaee and T. M. Khoshgoftaar. The effect of feature extraction and data sampling on credit card fraud detection. *Journal of Big Data*, 10(1):6, 2023.

[95] S. H. A. Kazmi C. Liu, Y. Chan and H. Fu. Financial fraud detection model: Based on random forest. *International Journal of Economics and Finance*, 2015.

[96] A. Arora S. Khatri and A. P. Agrawal. Supervised machine learning algorithms for credit card fraud detection: A comparison. *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, 2018.

[97] C. Jha N. Bharill O. P. Patel S. Joshi D. Puthal S. Rajora, D. L. Li and M. Prasad. A comparative study of machine learning techniques for credit card fraud detection based on time variance. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, page 1958–1963, 2018.

[98] U. K. Lilhore N. K. Trivedi, S. Simaiya and S. K. Sharma. An efficient credit card fraud detection model based on machine learning methods. *International Journal of Advanced Science and Technology*, 29(5):3414–3424, 2020.

[99] R. Ramesh R. Sailusha, V. Gnaneswar and G. R. Rao. Credit card fraud detection using machine learning. *Procedia Computer Science*, 165:631–641, 2019.

[100] T. G. Swart K. Aruleba E. Esenogho, I. D. Mienye and G. Obaido. A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access*, 10:16400–16407, 2022.

[101] A. Albattah Y. Alghofaili and A. Murad Rassam. A financial fraud detection model based on lstm deep learning technique. *Journal of Applied Security Research*, 15(4):498–516, 2020.

[102] S. Douzi I. Benchaji and B. E. Ouahidi. Credit card fraud detection model based on lstm recurrent neural networks. *Journal of Advances in Information Technology*, 12(2):113–118, 2021.

[103] A. A. Aqouleh H. Najadat, O. Altiti and M. Younes. Credit card fraud detection based on machine and deep learning. *2020 11th International Conference on Information and Communication Systems (ICICS)*, 2020.

[104] A. M. Mubalaike and E. Adali. Deep learning approach for intelligent financial fraud detection system. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, page 598–603, 2018.

[105] I. D. Mienye and A. Sun. A deep learning ensemble with data resampling for credit card fraud detection. *IEEE Access*, 11:30628 – 30638, 2023.

[106] Y. L. Borgne S. Waterschoot A. D. Pozzolo, O. Caelen and G. Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert System Application*, 41(10):4915–4928, 2014.

[107] O. Caelen S. Waterschoot N. V. Chawla A. D. Pozzolo, R. A. Johnson and G. Bontempi. Using hddt to avoid instances propagation in unbalanced and evolving data streams. *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014.

[108] O. Caelen C. Alippi A. D. Pozzolo, G. Boracchi and G.Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. *2015 International Joint Conference on Neural Networks*, 75:1–8, 2015.

[109] A. Bifet M. Pechenizkiy J. Gama, I. Žliobait˙e and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

[110] M. Ghosha S. Misraa, S. Thakura and S. K. Saha. An autoencoder based model for detecting fraudulent credit card transaction. *International Conference on Computational Intelligence and Data Science (ICCIDS 2019*, 167(4):254–262, 2020.

[111] Hofer-Schmitz K. Nahrgang K. Weber A. Badii A. Sundaram M. Jordan B. Stojanović, J.Božić and J.Runevic. Follow the trail: Machine learning for fraud detection in fintech applications. *Sensors*, 21(1594), 2021.

[112] E. Acar S. Visbeek and F. D. Hengst. Explainable fraud detection with deep symbolic classification. *Security and Communication Networks*, 2023.

[113] Z. Xiao and J. Jia. Explainable fraud detection for few labeled time series data. *Security and Communication Networks*, 2021, 2021.

Chapter 6

Appendix

6.1 Related Publications

1. S. Islam, M. M. Haque, A.N.M.R. Karim, "A rule-based machine learning model for financial fraud detection", International Journal of Electrical and Computer Engineering (IJECE), 14(1):759-771, 2024.
2. S. Islam, M. M. Haque, "A Dynamic Resampling Method for Credit Card Fraud Detection in Imbalance Classification", 12th Computing Conference 2024, London, United Kingdom.