

# National Economics University

Faculty Of Economic Mathematics

—o0o—



## Fraud Detection in New Bank Account

Instructor: **Luong Van Thien**

Class: **DSEB64A**

Student:	<b>Hoang Thuy Duong</b>	<b>11221551</b>
	<b>Nguyen Hoai An</b>	<b>11220032</b>
	<b>Pham Thi Ha</b>	<b>11221964</b>
	<b>Nguyen Thi Tuyet May</b>	<b>11224144</b>
	<b>Vo Thi Minh Phuong</b>	<b>11225327</b>

**HA NOI, 2024**

# Contributions

Student ID	Group Member	Contribute
11221551	Hoàng Thùy Dương	20%
11220032	Nguyễn Hoài An	20%
11221964	Phạm Thị Hà	20%
11224144	Nguyễn Thị Tuyết Mây	20%
11225327	Võ Thị Minh Phương	20%

## Abstract

Fraud detection in financial transactions has become a critical challenge for the banking and financial sectors due to the increasing sophistication of fraudulent activities. In this study, we leveraged machine learning techniques to detect fraudulent bank account transactions using a dataset comprising customer demographics, transaction history, and behavioral attributes. The dataset is highly imbalanced, with fraudulent cases representing a small proportion of the total records, making the task particularly challenging.

We began by conducting an exploratory data analysis (EDA) to understand the data distribution, identify key patterns, and uncover correlations between features. Key insights guided the data preprocessing pipeline, including handling missing values, outlier treatment, feature selection, and normalization of skewed distributions. Advanced feature engineering techniques were applied to create meaningful features that enhance the predictive power of the models.

Several machine learning models were employed, including Random Forest, Logistic Regression, K-Nearest Neighbors, and Gaussian Naive Bayes. Each model was tuned for optimal performance using hyperparameter optimization, and metrics such as F1 Score, precision-recall AUC (PR-AUC), and confusion matrices were used for evaluation. The Random Forest model emerged as the best-performing algorithm, demonstrating strong predictive capabilities even without resampling techniques.

To address the dataset's imbalance, resampling methods such as SMOTE-ENN were explored to improve the detection of minority-class fraudulent transactions. Furthermore, we discussed the potential for incorporating unsupervised learning techniques, such as autoencoders and clustering algorithms, to enhance fraud detection in future work.

The results of this study highlight the importance of rigorous preprocessing, feature engineering, and model selection in building effective fraud detection systems. Our findings provide actionable insights and a robust framework that can be applied to real-world scenarios, enabling financial institutions to detect and mitigate fraudulent activities with greater precision.

## Acknowledgements

The authors would like to express their deepest gratitude to the many individuals and organizations whose continuous support and collaboration played a significant role in the successful completion of this study. First and foremost, we would like to express our sincere gratitude to Dr. Luong Van Thien for his generous support in providing the infrastructure and computational resources necessary to conduct this comprehensive study. We are particularly grateful to our partners in the banking sector for generously sharing their anonymized data sets and information, which played a significant role in our fraud detection research.

We are extremely grateful to the anonymous reviewers whose constructive feedback significantly improved the methodological rigor and scientific clarity of the manuscript. Their thoughtful suggestions helped refine our research methodology and strengthen the overall scholarly contribution of this work.

Our sincere thanks go to our colleagues and research assistants who contributed countless hours of data processing, model development, and results validation. Their dedication, intellectual curiosity, and collaborative spirit were fundamental to translating conceptual frameworks into empirical insights.

Finally, we acknowledge the broader academic and industrial communities working tirelessly to combat financial fraud, whose collective efforts inspire and drive innovation in technological security solutions.

# Table of Contents

<b>Abstract .....</b>	<b>1</b>
<b>Acknowledgements .....</b>	<b>2</b>
<b>List of Abbreviations .....</b>	<b>5</b>
<b>1. Chapter 1: Introduction .....</b>	<b>6</b>
<b>2. Chapter 2: Theoretical Background And Literature Review</b>	<b>10</b>
2.1. Theoretical background of Bank Account Fraud Detection .....	10
2.1.1. Overview of Bank Account Fraud situation in Vietnam and globally .....	10
2.1.2. Introduction of Bank Account Fraud detection .....	11
2.1.3. Application of fraud detection in banking industry .....	13
2.2. Literature Review .....	14
2.2.1. Machine Learning Techniques for Bank Account Fraud Detection .....	14
2.2.2. Review of the method for handling imbalanced datasets ..	15
<b>3. Chapter 3: Research Methods .....</b>	<b>18</b>
3.1. Framework .....	18
3.2. Class Imbalance problem .....	19
3.2.1. Oversampling techniques: .....	20
3.2.2. Undersampling: .....	23
3.3. Feature Engineering .....	24
3.4. Machine Learning Models .....	26
3.4.1. Naïve Bayes .....	26
3.4.2. Logistic Regression .....	30
3.4.3. Decision Tree .....	33
3.4.4. Random Forest .....	35
3.4.5. K-Nearest Neighbors (KNN) .....	38
3.5. Evaluation Metrics .....	40
3.5.1. Confusion Matrix .....	40
3.5.2. Precision .....	41
3.5.3. Recall .....	42
3.5.4. F-beta Score .....	42
3.5.5. Area Under the Curve (AUC) .....	43
3.6. Dataset .....	44

3.6.1. Data Description .....	44
3.6.2. Exploratory Data Analysis (EDA) .....	49
3.6.3. Data Preprocessing .....	56
3.6.4. Data Splitting .....	56
<b>4. Chapter 4: Results Analysis and Discussions .....</b>	<b>58</b>
4.0.1. Original Data (Without Resampling): .....	58
4.0.2. Oversampling: .....	58
4.0.3. Undersampling: .....	64
4.0.4. Discussion for results: .....	66
<b>5. Chapter 5: Conclusions and Future Work .....</b>	<b>69</b>
<b>6. References .....</b>	<b>71</b>

## List of Abbreviations

Abbreviation	Full Form
ML	Machine Learning
AI	Artificial Intelligence
ENN	Edited Nearest Neighbors
SMOTE	Synthetic Minority Oversampling Technique
SMOTE-ENN	Synthetic minority oversampling technique and edited the nearest neighbor
AUC	Area Under the Curve
KNN	K-Nearest Neighbors
PR-AUC	Precision-Recall Area Under the Curve
ADASYN	Adaptive Synthetic Sampling Approach for Imbalanced

# 1. Chapter 1: Introduction

## Reasons for choosing research topic

Fraud involving bank accounts has become a growing global issue, leading to significant financial losses for both individuals and institutions. The rapid shift toward digital banking and online transactions has created new vulnerabilities, with fraudsters exploiting advanced techniques to access and misuse bank accounts. Traditional rule-based systems are often inadequate to detect these complex and evolving fraud schemes.

Machine learning offers a promising alternative, enabling the analysis of vast and complex banking datasets to detect suspicious activity and unusual behavior patterns. However, bank account fraud detection presents unique challenges, particularly the issue of imbalanced data, where fraudulent transactions make up only a small proportion of total activity. This imbalance can hinder the accuracy and effectiveness of machine learning models, making it crucial to develop innovative solutions.

Advanced methods such as ensemble techniques, data balancing strategies like SMOTE-ENN, and hyperparameter tuning can significantly enhance model performance. Additionally, using robust evaluation metrics like F1 Score and precision ensures a more accurate assessment of a model's ability to detect fraud in these highly skewed datasets.

Given the critical need to strengthen fraud detection systems for bank accounts, this research focuses on leveraging state-of-the-art machine learning methodologies. The goal is to address these challenges and contribute to the development of more effective and reliable fraud detection solutions, enhancing the security of banking systems in the digital age.

## Research Objectives

The objective of this thesis is to develop and implement supervised machine learning techniques specifically tailored for detecting fraud in bank account transactions. By leveraging advanced methodologies and algorithms, this research aims to improve the accuracy and efficiency of fraud detection systems, ultimately reducing the impact of fraudulent activities on banking operations and safeguarding customer trust.

The study aims to identify the most effective methods for detecting fraudulent transactions while providing insights into the underlying patterns and characteristics of fraudulent behavior in bank account transactions through empirical analysis and experimentation.

The research questions investigated in this thesis are:

- Which machine learning models are most accurate in identifying fraudulent bank account transactions?
- How does the performance of these models compare when applied to highly imbalanced bank account fraud datasets?
- What impact do resampling techniques, such as SMOTE or other balancing methods, have on the detection accuracy of fraudulent transactions in bank accounts?

## **Research Methodology**

To accomplish the application of supervised machine learning for detecting and predicting fraud in bank account transactions, the following research methods will be employed in this study:

- **Exploratory Data Analysis and Feature Engineering:** Given the highly imbalanced nature of the dataset (where fraudulent transactions are a small fraction), exploratory data analysis (EDA) and feature engineering techniques will be applied to better understand the data and create features that enhance model performance.
- **Machine Learning Techniques:** Various machine learning algorithms, including Random Forest (a bagging method) and CatBoost (a boosting algorithm), will be implemented. These models will be tuned using the Optuna Hyperparameter Tuning Framework to optimize their ability to identify fraudulent bank account transactions effectively.
- **SMOTE-ENN Resampling:** To address the class imbalance issue, the SMOTE-ENN (Synthetic Minority Over-sampling Technique - Edited Nearest Neighbors) resampling method will be integrated. This method will help generate synthetic data points for the minority class (fraudulent transactions) and reduce noise, improving model detection accuracy.
- **Evaluation:** A single evaluation framework will be employed to assess the performance of the models. The F1 Score (which focuses on recall for

imbalanced datasets) and PR-AUC (Precision-Recall Area Under the Curve) will be the primary evaluation metrics used in this research, as they are better suited for imbalanced datasets like those encountered in fraud detection.

Programming Language Used: Python

## **Target and Scope of Research**

The target of this study is to detect fraudulent bank account transactions by implementing advanced supervised machine learning techniques. The scope of the research includes analyzing a dataset from the Bank Account Fraud (BAF) suite of datasets, which was published at NeurIPS 2022, focusing on fraud detection in digital banking environments.

## **Structure of the Research**

The research consists of five chapters:

*Chapter 1: Introduction* This chapter introduces the impact of fraud on the financial sector, discusses existing fraud detection methods, and outlines the research objectives. It also details the proposed methodologies and defines the target and scope of the study, focusing on bank account fraud detection.

### *Chapter 2: Theoretical Background and Literature Review*

This chapter provides an overview of fraud detection in the banking industry, with a particular emphasis on bank account fraud. It reviews relevant literature and past research on fraud detection systems and the application of machine learning techniques in the financial sector.

### *Chapter 3: Research Methods*

This chapter describes the methodologies used in the study for detecting fraudulent bank account transactions. It details the machine learning techniques, data preprocessing steps, and evaluation metrics applied throughout the research.

*Chapter 4: Results and Discussions* This chapter presents the experimental results of various machine learning models, compares their performance, and provides a discussion of the findings, including the effectiveness of different techniques in identifying fraudulent bank account transactions.

### *Chapter 5: Conclusions and Future Work*

This chapter summarizes the key findings of the research, highlights its contributions to fraud detection in bank accounts, and offers recommendations for future research and improvements in fraud detection systems.

## **2. Chapter 2: Theoretical Background And Literature Review**

### **2.1. Theoretical background of Bank Account Fraud Detection**

#### **2.1.1. Overview of Bank Account Fraud situation in Vietnam and globally**

Fraud emerged as a threat to the whole world, causing significant financial losses for both cardholders and financial institutions like banks. Bank account fraud has become a significant global issue, driven by the rapid growth of digital banking and online financial services. Fraudsters are increasingly exploiting vulnerabilities in these systems, using sophisticated methods such as phishing, identity theft, account takeovers, and social engineering to access and manipulate bank accounts. The economic impact of fraud is considerable, with billions of dollars lost each year. In 2022, the U.S. alone reported over 3.3 billion in fraud-related losses. Financial institutions worldwide are responding by adopting advanced technologies, such as artificial intelligence (AI), machine learning (ML), and biometric authentication, to detect and prevent fraudulent activities. However, as fraudsters evolve their tactics, including using AI-driven attacks and deepfakes, the challenge of securing digital banking platforms remains increasingly complex.

In Vietnam, the rise of digital banking has brought both convenience and increased vulnerability to fraud. As more people use mobile banking and digital payment platforms, fraudsters have targeted these users through phishing scams, SMS fraud, and social engineering tactics. The State Bank of Vietnam has implemented measures like two-factor authentication (2FA) and enhanced consumer education to combat fraud. However, the growing reliance on digital transactions, especially during the COVID-19 pandemic, has provided new opportunities for fraudsters. Although progress is being made in addressing fraud risks, there is still a need for greater public awareness and more robust security measures to protect consumers and the financial sector from the increasing threat of bank account fraud in the country.

### **2.1.2. Introduction of Bank Account Fraud detection**

#### **What is Fraud?**

In different domains, there is a wide variety of definitions for fraud. Fraud is a form of deception and trickery; fraudsters are people who cause losses to others, then benefit from them by tricking. ACFE defines fraud as follows (Wells J.T., 2011): “Fraud can encompass any crime for gain that uses deception as its principal modus operandi. Of the three ways to illegally relieve a victim of money – force, trickery, or larceny- all offenses that employ trickery are frauds. Although all frauds involve some form of deception, not all deceptions are necessarily frauds. “. The term “fraud” includes many crimes and is defined differently across different domains. However, the common thread among these definitions is the act of deceiving others to achieve personal gain.

#### **Types of Fraud**

Up to this point, researchers have proposed several classifications of the frauds. For example, ACFE classifies occupational frauds 2012: three classes (The Association of Certified Fraud Examiners (ACFE), 2012): corruption, asset misappropriation and financial statement fraud. Nevertheless, there is another classification that provides a more general overview (Bezerra F. & Wainer J., 2008). In this classification, frauds are classified into two classes: internal fraud and external fraud. Internal fraud includes two main categories: financial statement fraud and transaction fraud.

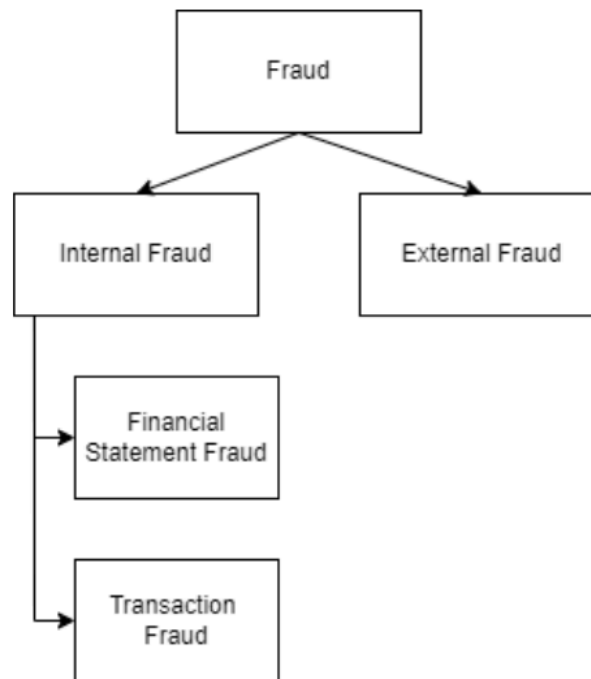


Figure 2.1. Fraud Classification

A bank account fraud detection model is a machine learning algorithm designed to identify fraudulent activities within bank account transactions. By analyzing large volumes of transaction data, these models can detect suspicious patterns and behaviors that may indicate fraud. The model is typically trained using historical transaction data, including features such as transaction amounts, time, frequency, location, and account activity. These features are used to classify transactions as either legitimate or fraudulent.

The goal of the bank account fraud detection model is to prevent unauthorized access and protect customers from financial loss. Similar to bank account fraud detection, the model learns from past fraud cases and applies this knowledge to identify potential threats in real-time. Machine learning techniques such as decision trees, random forests, support vector machines (SVM), and neural networks are commonly used in these models. In the face of imbalanced datasets, where fraudulent transactions are much less frequent than legitimate ones, resampling methods like SMOTE or ensemble learning can help improve detection accuracy. By accurately identifying fraudulent transactions, these models play a crucial role in safeguarding financial institutions and their customers.

### **2.1.3. Application of fraud detection in banking industry**

Banks and financial institutions (FIs) use fraud detection in banking technologies and strategies to reduce the risks of fraud to their business. These risks include the financial costs of fraud as well as the reputational damage that it causes. Such firms are the most targeted by fraudsters because of their potential to provide speedy access to and transfer of funds.

As a result, banks and FIs are constantly evaluating how to stay one step ahead of fraudsters by strengthening their fraud detection in banking tools and implementing prevention solutions to protect their assets, systems and customers. This can be quite a challenge with systems needing to be robust while remaining user friendly for genuine customers. It comprises various applications that banks use to combat fraud.

- **Artificial Intelligence (AI) and Machine Learning (ML)**

AI and ML components such as cognitive fraud analytics and predictive models play a vital role in enhancing fraud detection within banking systems. Cognitive fraud analytics involves continuously updating behavioral profiles with each transaction to anticipate individual behaviors and execute fraud detection strategies at scale. Meanwhile, predictive models leverage historical data gathered through machine learning algorithms to anticipate the likelihood of specific behaviors.

Risk assessment is another critical aspect, as it involves computing real-time risk scores based on behavioral analysis and other data, enabling banks to make informed decisions for each customer.

- **Behavioral Analytics and Pattern Recognition**

Banking fraud detection software uses behavioral analytics and pattern recognition algorithms to analyze customer behavior, spending habits, and device usage. Banks can leverage behavioral analytics to develop profiles of customer behavior based on their transaction patterns and device interactions. From the data, banks can identify unusual or suspicious activities that may indicate fraudulent behavior.

- **Real-Time Transaction Monitoring**

Banks use real-time transaction monitoring to identify fraudulent activities as they happen, protecting customer assets and reducing the risk of financial losses.

Transaction monitoring analyzes various data points, such as transaction amounts, frequencies, and locations, to pinpoint patterns that may indicate potential fraud. This includes identifying multiple transactions from different locations within a short period, unusually large transactions, or transactions that deviate from a customer's normal spending behavior.

## **2.2. Literature Review**

Bank account fraud detection is an increasingly critical area of research, particularly as digital banking services expand and fraudsters continue to exploit vulnerabilities in online financial systems. The rise in online banking and digital payment platforms has led to an increase in fraud activities, which poses significant risks to both financial institutions and consumers. This literature review synthesizes key findings and methodologies employed in bank account fraud detection, highlighting the challenges, advancements, and the role of machine learning in combating fraud.

### **2.2.1. Machine Learning Techniques for Bank Account Fraud Detection**

In this comprehensive analysis of the detection of bank account fraud using machine learning, several researchers have highlighted the way to improve fraud detection. The study titled *“An Ensemble-based Fraud Detection Model for Financial Transaction Cyber Threat Classification and Countermeasures”* was authored by Asma A. Alhashmi, Abdullah M. Alashjaee, Abdulbasit A. Darem, Abdullah F. Alanazi, and Rachid Effghi in 2023 propose ensemble technique to improve bank account fraud detection. The accuracy of each model is as follows:

Voting Classifier: 0.97 Random Forest: 0.96 Gradient Boosting: 0.96 Logistic Regression: 0.96 Neural Networks: 0.96 Stacking: 0.98. The study highlights the significance of precision-recall trade-offs in fraud detection and underscores the potential of ensemble methods to enhance the resilience and efficacy of security systems in the banking industry.

In another research paper titled *“Financial Fraud Detection Using Value-at-Risk With Machine Learning in Skewed Data”* authored by Abdullahi U. Usman, Sunusi Bala Abdullahi, Yu Liping, Bayan Alghofaily, Ahmed S. Almasoud, and Amjad Rehman and published in 2024, a method is proposed that incorporates value-at-risk (VaR) as a risk measure to handle

the skewness of fraud instances. The paper introduces a novel detection rate metric that considers risk fraud features to measure the performance of the fraud detection model. The paper reports an improved fraud detection model using KNN with a true positive rate of 0.95 and a detection rate of 0.9406. The study focuses on establishing data-driven criteria for fraud risk management by considering potential losses and skewness in fraud datasets.

Figures in the paper illustrate the proposed method's performance evaluation and decision boundaries for KNN. The results show that KNN outperforms other models like binary logistic regression (BLR) and Naïve Bayes (NB) in terms of accuracy and detection rates.

### **2.2.2. Review of the method for handling imbalanced datasets**

Imbalanced datasets are a common challenge in machine learning, particularly when one class (the minority class) is significantly underrepresented compared to another (the majority class). This imbalance can lead to biased models that favor the majority class, resulting in poor performance for detecting or predicting rare events. In medical diagnostics, this issue is especially critical because the minority class often represents diseased individuals who require accurate identification for timely treatment.

Esenogho et al. (2022) introduced an innovative approach to bank account fraud detection by leveraging a combination of advanced machine learning techniques: a neural network ensemble classifier and a hybrid data resampling method. This approach was designed to address two critical challenges in bank account fraud detection:

- *Class Imbalance Problem:* Fraudulent transactions are significantly fewer than legitimate ones, making it difficult for machine learning models to detect fraud effectively.
- *Dynamic Nature of Fraudulent Behavior:* The behavior of fraudulent transactions evolves over time, requiring adaptable and robust detection mechanisms.

The proposed model combines Long Short-Term Memory (LSTM) networks with Adaptive Boosting (AdaBoost) and uses the SMOTE-ENN hybrid resampling technique to balance the dataset. SMOTE is an oversampling technique that balances the class distribution by adding synthetic samples

to the minority class, whereas ENN is an undersampling method that removes some samples from the majority class. SMOTE-ENN uses both oversampling and undersampling to create a balanced dataset. However, the authors did not investigate the impact of different hyperparameter settings or variations in the neural network architecture on the performance of the proposed method.

Building upon this foundation, Prasetyo et al. (2021) conducted a study that specifically addressed this issue by evaluating the performance of bank account fraud detection algorithms on imbalanced datasets, with a focus on metrics such as recall and F2 score. These metrics are particularly important in fraud detection because:

Recall measures the proportion of actual fraudulent transactions correctly identified by the model, which is crucial for minimizing missed detections. F2 Score places more emphasis on recall than precision, making it suitable for scenarios where identifying fraud is more critical than avoiding false positives. To mitigate dataset imbalance, Prasetyo et al. utilized the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE is an oversampling method that generates synthetic samples for the minority class (fraudulent transactions) by interpolating between existing minority class instances. This approach helps balance the dataset and improves model training.

The researchers experimented with different oversampling proportions using SMOTE, ranging from 20% to 100% of the majority class size. Their findings revealed that:

The choice of oversampling proportion significantly impacts algorithm performance. Generating synthetic data equivalent to 60% of the majority class size yielded the best results, achieving an F2 score of 85.34. This result highlights SMOTE's effectiveness in improving model performance on imbalanced datasets when applied judiciously. The study underscores that while SMOTE can enhance algorithm performance, selecting an appropriate oversampling proportion is critical for optimizing results. Oversampling too little may not sufficiently address imbalance issues, while excessive oversampling could lead to overfitting or introduce noise into the dataset.

Vietnamese researchers, particularly Dang et al. (2021), have made significant contributions to addressing the challenges posed by imbalanced datasets in bank account fraud (CCF) detection systems. To address the imbalanced issue, Dang et al. utilized two widely recognized resampling techniques: SMOTE (Synthetic Minority Oversampling Technique) and ADASYN (Adaptive Synthetic Sampling). These methods aim to balance the dataset by generating synthetic samples for the minority class (fraudulent transactions). Several ML models based on SMOTE such as Random Forest, Logistic Regression, AdaBoost, Deep Neural Network (DNN) achieved F1 Score at 99.99%, 94.34%, 97.21%, and 99.8% respectively. Similarly, models based on ADASYN have scores of 99.98%, 88.73%, 90.34% and 99.69% corresponding to Random Forest, Logistic Regression, AdaBoost, DNN in order. These findings underscored the importance of handling imbalanced data in developing robust CCF detection systems and show the efficiency of resampling techniques when applied to this case.

Despite notable progress in credit card fraud detection research yielding promising predictive outcomes, there remains a lack of studies focusing on ensemble techniques that incorporate a variety of methods. This is particularly true when leveraging approaches such as SMOTE-ENN, hyperparameter tuning, and employing F2 Score and PR-AUC as primary evaluation metrics to address imbalanced datasets. Therefore, the objective of this study is to explore ensemble approaches using multiple models and methodologies, applying these techniques and metrics to identify the most effective model for detecting credit card fraud.

## 3. Chapter 3: Research Methods

### 3.1. Framework

This project aims to develop an effective machine learning model to detect fraudulent activities with high accuracy and reliability. The dataset undergoes feature engineering, including scaling numerical data and encoding categorical variables, to create meaningful features that highlight patterns associated with fraudulent behavior. The data is divided into training and testing sets, with resampling techniques applied exclusively to the training set. Both oversampling techniques (e.g., SMOTE, ADASYN) and undersampling methods are utilized to address the issue of class imbalance. A variety of machine learning algorithms, including Logistic Regression, Random Forest, KNN, and others, are trained on the resampled training data and evaluated for performance. Key metrics such as Accuracy, ROC-AUC, and Log Loss are used to compare models and determine the best-performing one. Finally, the selected model is thoroughly evaluated to ensure its reliability for real-world applications in fraud detection. The workflow for this methodology is illustrated in the figure above.

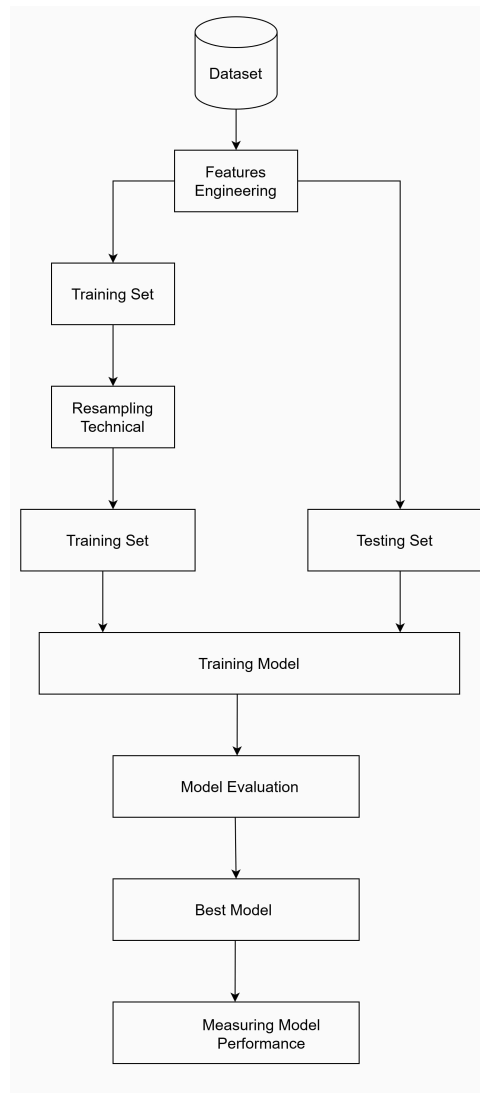


Figure 3.1. Framework of credit card fraud detection model

### 3.2. Class Imbalance problem

In a dataset, class imbalance occurs when one class label significantly outweighs the other(s), leading to an uneven distribution of data. This issue is common in many real-world applications where the majority of observations belong to one class, while the minority class is underrepresented. A prime example of class imbalance is observed in fraud detection problems, where the number of fraudulent observations is often much smaller than non-fraudulent ones. Addressing this imbalance is crucial for developing effective machine learning models, as failing to do so can lead to biased predictions that favor the majority class. This report focuses on tackling the class imbalance issue using resampling methods,

specifically oversampling and undersampling techniques, which will be explored in detail in the following sections.

## Resampling Method

Predictive models often struggle to perform effectively when faced with imbalanced class distributions. Therefore, it is crucial to carry out specific data preprocessing steps before feeding the data into the model. Addressing class imbalance typically involves employing a data-level approach, such as resampling techniques. These techniques can be categorized into three main types: oversampling, undersampling, and hybrid methods, each offering a unique strategy to balance the dataset.



Figure 3.2. Undersampling and Oversampling Techniques

### 3.2.1. Oversampling techniques:

#### SMOTE:

On the other hand, oversampling addresses class imbalance by increasing the amount of data in the minority class, typically by creating additional instances until it matches the size of the majority class. A widely used approach for achieving this is the Synthetic Minority Oversampling Technique (SMOTE).

Developed by Chawla (Bowyer et al., 2002), SMOTE generates synthetic examples of the minority class by interpolating between its nearest neighbors rather than merely duplicating existing instances. This process, as illustrated in figure 3, reduces the risk of overfitting that may arise from simple replication. The number of nearest neighbors used for interpolation is selected randomly and depends on the desired level of oversampling

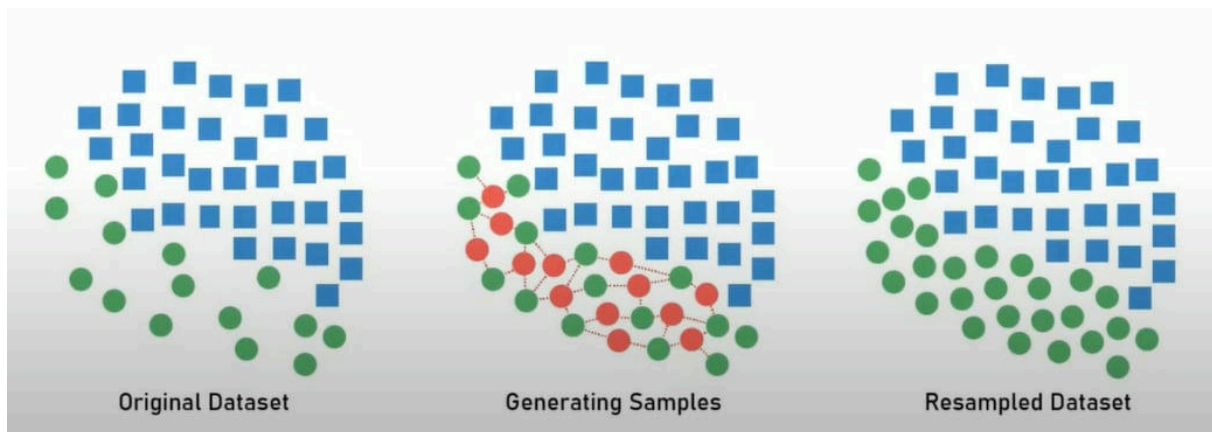


Figure 3.3. SMOTE Method

**SMOTETomek:**

SMOTETomek is a hybrid resampling technique that combines the strengths of oversampling through SMOTE and undersampling via Tomek links removal. This approach begins by applying SMOTE to generate synthetic examples for the minority class by interpolating between its nearest neighbors, effectively increasing the representation of the minority class. Next, Tomek links are identified and removed; these are pairs of instances from opposite classes that are very close to each other, often lying on the class boundary. Removing these pairs not only cleans the dataset but also enhances class separation. Together, SMOTETomek balances the dataset while reducing noise and ambiguity, improving the model's ability to generalize.

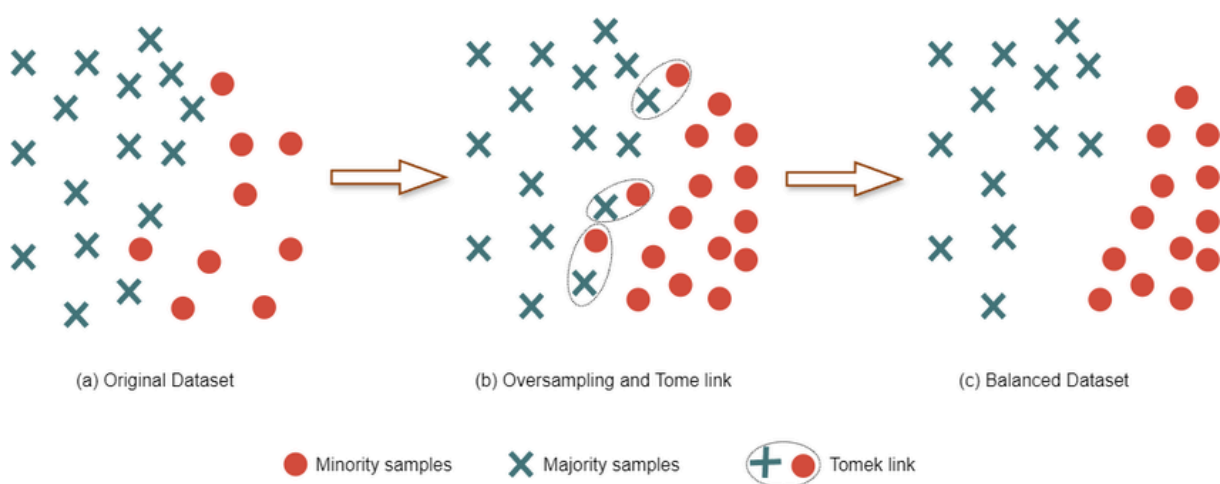


Figure 3.4. SMOTETomek Method

**ADASYN:**

ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning) is an advanced oversampling technique that aims to address class imbalance by creating synthetic examples of the minority class in a more targeted manner. Unlike SMOTE, which generates synthetic samples uniformly, ADASYN focuses on generating new samples in regions where the minority class is underrepresented or where the learning is more challenging.

This is achieved by adaptively determining the number of synthetic instances needed for each minority sample based on the distribution of its nearest neighbors. As a result, ADASYN dynamically emphasizes difficult-to-learn examples, ensuring that the model pays more attention to regions where the class imbalance is most severe. This approach not only balances the dataset but also improves the classifier's ability to handle complex decision boundaries.

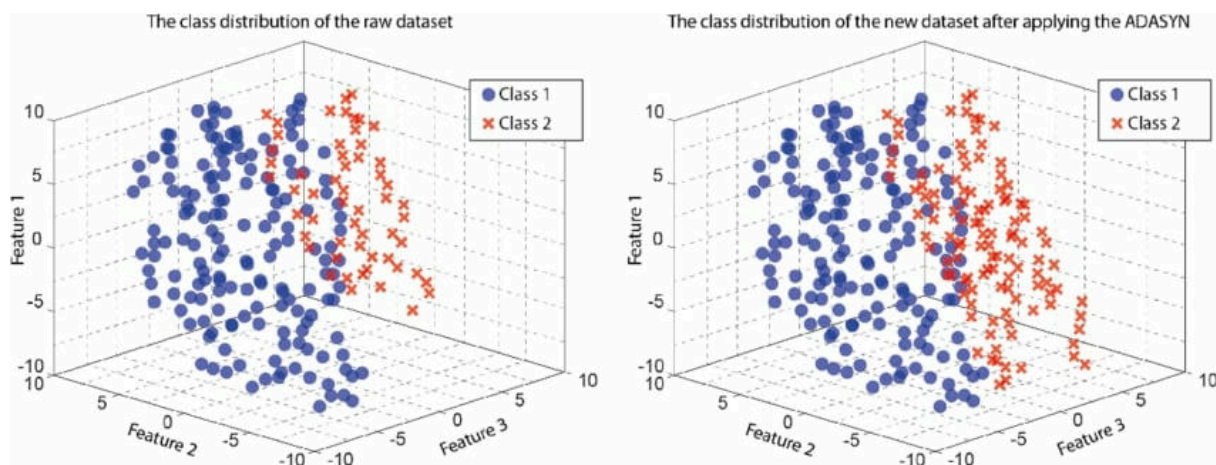


Figure 3.5. ADASYN Method

### SMOTEEN:

SMOTEENN (Synthetic Minority Oversampling Technique and Edited Nearest Neighbors) is a hybrid resampling method that combines oversampling and undersampling to address class imbalance effectively. This technique starts with SMOTE to generate synthetic samples for the minority class by interpolating between existing samples and their nearest neighbors, increasing the representation of the minority class. Following this, Edited Nearest Neighbors (ENN) is applied as an undersampling step to clean the dataset by removing noisy or ambiguous samples from

both classes that do not agree with the majority class of their k-nearest neighbors.

By combining these two techniques, SMOTEENN not only balances the dataset but also enhances its quality by eliminating noisy points that could adversely affect the model's performance. This makes it particularly effective in improving the generalization of machine learning models on imbalanced datasets.

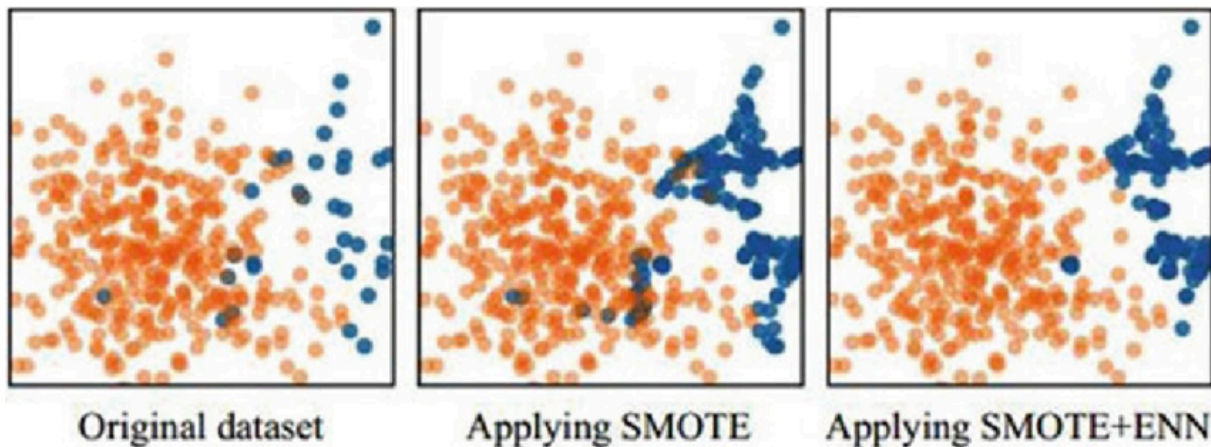


Figure 3.6. SMOTEEEN Method

### 3.2.2. Undersampling:

Conversely, undersampling addresses class imbalance by reducing the number of majority class instances until it matches the quantity of the minority class. One common technique for achieving this is the `RandomUnderSampler`. This method randomly selects a subset of samples from the majority class, discarding the rest, to balance the class distribution.

`RandomUnderSampler`, a straightforward and computationally efficient method, works by randomly eliminating instances from the majority class without any replacement. This approach reduces the overall size of the dataset, making it more manageable and enabling the model to learn from both classes more equally. However, by removing majority class samples, it may also discard valuable information, potentially leading to a loss of important patterns and underfitting. The number of majority class samples selected for removal depends on the desired balance between the classes.

As shown in figure , after applying the RandomUnderSampler, the majority class becomes more balanced with the minority class, but at the cost of potentially reducing the dataset's diversity.

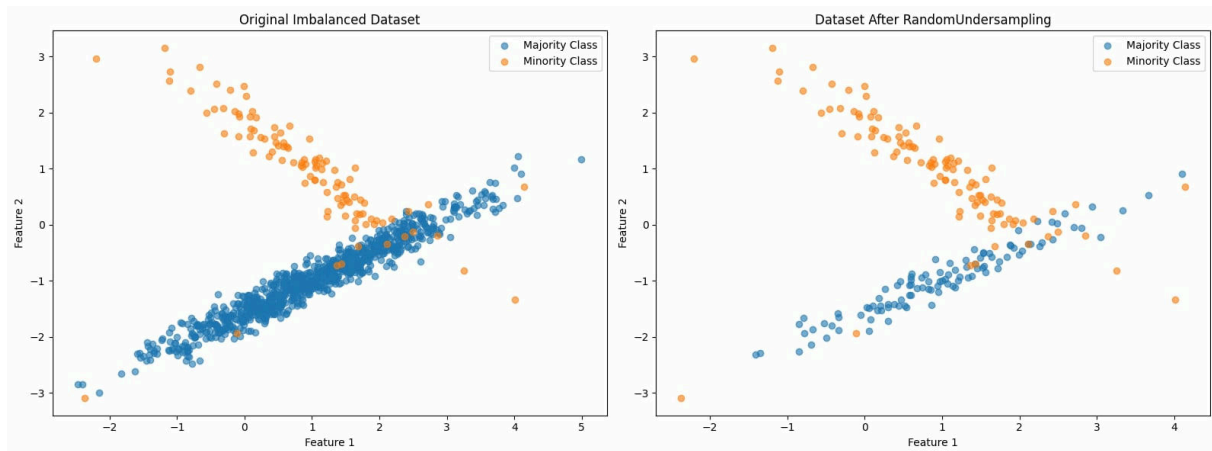


Figure 3.7. RandomUnderSampler Method

### 3.3. Feature Engineering

Feature engineering is one of the most important steps in the machine learning pipeline, as it involves transforming raw data into meaningful features that can enhance the performance of the model. In this project, various feature engineering techniques were applied to prepare the dataset for training machine learning models. These steps ensured that the data was clean, balanced, and properly formatted, allowing the model to effectively learn patterns and relationships that are indicative of fraudulent activities.

#### Dropping Irrelevant Features:

The first step in feature engineering was to eliminate irrelevant or redundant features that do not contribute to predicting fraud. By removing unnecessary features, we reduce the complexity of the model, speed up computation, and prevent overfitting.

The target variable, `fraud_bool`, which indicates whether a transaction is fraudulent, was separated from the feature set, as it is only used during model evaluation and not for training. Additionally, several other columns were deemed irrelevant to the model or had no significant impact on the target variable. These features included:

- `income`: While income may be useful in certain fraud detection scenarios, it was not deemed relevant for this dataset or problem context.

- `name_email_similarity`: This feature, which measures the similarity between a user's name and email address, was removed as it was considered redundant or noisy.
- `email_is_free`, `phone_home_valid`, `phone_mobile_valid`, `has_other_cards`, `foreign_request`, `keep_alive_session`, and `month`: These features were either less informative or already captured by other features and were therefore dropped to simplify the model.

By removing these irrelevant columns, the dataset was reduced to a more manageable size and focused on the features that have a higher likelihood of contributing to predicting fraudulent transactions.

### **Scaling Numerical Features**

Many machine learning models are sensitive to the scale of input features, especially algorithms such as logistic regression, k-nearest neighbors (KNN), and support vector machines (SVM). To ensure that numerical features with different units or ranges did not disproportionately influence the model, all numerical features were scaled to bring them into a similar range.

In this project, the `RobustScaler` was used to scale the numerical features. The `RobustScaler` is particularly effective when dealing with outliers because it scales the data based on the interquartile range (IQR) rather than the mean and standard deviation, which can be heavily influenced by extreme values. This makes the `RobustScaler` more robust to outliers compared to other scaling methods like `StandardScaler`, which uses the mean and standard deviation to scale the data.

By scaling the data, each feature was transformed into a comparable range, allowing the model to process all features equally and learn from them without any single feature dominating due to its scale. This is crucial for models like KNN, where the distance between data points affects the model's predictions, and for models like logistic regression, where regularization plays a key role.

### **Handling Categorical Variables:**

Categorical variables, such as user identifiers, transaction types, and other non-numeric features, need to be transformed into numerical

representations before they can be used in machine learning models. One of the most common techniques for handling categorical variables is One-Hot Encoding, which creates binary columns for each category in a categorical feature.

For example, if a feature contains three unique categories, such as “transaction\_type” with the values [“Online”, “In-store”, “Mobile”], One-Hot Encoding will create three new columns: “transaction\_type\_Online”, “transaction\_type\_In-store”, and “transaction\_type\_Mobile”, each containing a binary value (1 or 0) indicating the presence of that category.

To avoid the issue of multicollinearity (where one category can be perfectly predicted from the others), we used the drop=‘first’ parameter during the encoding process. This drops the first category in each feature, ensuring that only the remaining categories are included as binary columns. This step prevents the model from learning redundant information and helps improve the stability and interpretability of the model.

After One-Hot Encoding, the categorical variables were transformed into numerical representations, which were then integrated into the dataset. This allowed the model to process categorical features as numerical inputs, making them compatible with machine learning algorithms.

### **3.4. Machine Learning Models**

In the context of detecting credit card fraud, machine learning algorithms play an important role in recognizing fraudulent transactions. This section will look at that excel at dealing with

#### **3.4.1. Naïve Bayes**

A Naive Bayes classifiers, a family of algorithms based on Bayes’ Theorem. Despite the “naive” assumption of feature independence, these classifiers are widely utilized for their simplicity and efficiency in machine learning.

Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset.

One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities.

Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimension (as each word represent one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc. The advantage of using naïve Bayes is its speed. It is fast and making prediction is easy with high dimension of data.

This model predicts the probability of an instance belongs to a class with a given set of feature value. It is a probabilistic classifier. It is because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other. In real world, this condition satisfies rarely. It uses Bayes theorem in the algorithm for training and prediction

The naïve Bayes classifier provides a rather different kind of algorithm, one based on estimating the probabilities of class membership. Application of Bayes' theorem, together with the assumption that the features  $x_i$  are conditionally independent of one another given the output class  $y$ , leads to the formula

$$P(y|x_1, x_2, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y).$$

The decision rule is to assign a test instance to the class with the highest estimated probability. Undoubtedly, the assumption of conditional independence of the features is not strictly valid. Nonetheless, naïve Bayes often performs well enough to be competitive with other machine-learning methods, and has the advantage of conceptual simplicity compared to most. As early as 1974, Cramer et al. published a method of computing the conditional probability of a molecule being bioactive given the fragments it contained.<sup>81</sup> Naïve Bayes classifiers are now frequently used in chemoinformatics, usually for predicting biological rather than physicochemical properties, naïve Bayes often being used alongside and compared against other classifiers. This has been done in studies

of athletic performance enhancement,<sup>42</sup> toxicity,<sup>66</sup> the mechanism of phospholipidosis,<sup>82</sup> and also for protein target prediction and bioactivity classification for drug-like molecules.<sup>83–85</sup> It is in principle possible to use naïve Bayes for regression,<sup>86</sup> but this is rarely seen in chemoinformatics

### Types of Naive Bayes Model

There are three types of Naive Bayes Model: Gaussian Naive Bayes classifier, Multinomial Naive Bayes, Bernoulli Naive Bayes.

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

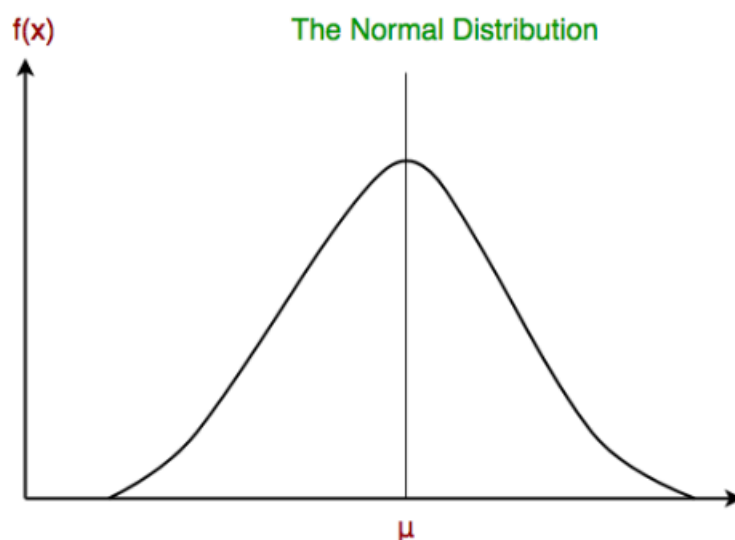


Figure 3.8. The Normal Distribution

In Multinomial Naive Bayes, feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence (i.e. a word occurs in a document or not) features are used rather than term frequencies (i.e. frequency of a word in the document).

## Advantages of Naive Bayes

- **Implement and Computationally Efficient:** The Naive Bayes algorithm is straightforward to implement due to its reliance on simple probabilistic principles. It involves minimal computations, making it highly efficient, especially for large datasets or real-time applications.
- **Effective with a Large Number of Features:** Naive Bayes works well with high-dimensional data, such as text classification tasks where the feature space is extensive (e.g., word frequencies in documents). It remains computationally feasible even when the number of features grows significantly.
- **Performs Well with Limited Training Data:** The algorithm requires relatively small amounts of training data to estimate the parameters necessary for classification. Its probabilistic foundation allows it to generalize effectively in low-data environments.
- **Handles Categorical Features Effectively:** Naive Bayes is particularly strong in datasets with categorical variables since it computes probabilities based on feature occurrences within each class. This makes it a popular choice for tasks like spam detection or sentiment analysis.
- **Assumes Normal Distribution for Numerical Features:** When dealing with numerical data, Naive Bayes assumes that the values follow a Gaussian (normal) distribution. This assumption simplifies the model and allows for straightforward probability calculations for continuous variables.

## Disadvantages of Naive Bayes

- **Assumption of Independence Among Features:** A key limitation of Naive Bayes is its assumption that all features are conditionally independent given the target class. In many real-world scenarios, features are often correlated (e.g., age and income), and violating this assumption can degrade performance.
- **Influence of Irrelevant Attributes:** Irrelevant or redundant features can impact the performance of Naive Bayes since it treats all features as equally important. This limitation can lead to noisy predictions if irrelevant attributes dominate the dataset.

- **Zero Probability Problem for Unseen Events:** If the classifier encounters a feature value in the test data that was not present during training, it assigns a probability of zero, which prevents accurate predictions. This is commonly mitigated by techniques like Laplace smoothing, which adjusts probabilities to handle unseen events.

### **3.4.2. Logistic Regression**

Indeed, logistic regression is one of the most important analytic tools in the social and natural sciences. In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. A neural network can be viewed as a series of logistic regression classifiers stacked on top of each other. Logistic regression can be used to classify an observation into one of two classes (like ‘positive sentiment’ and ‘negative sentiment’), or into one of many classes[4].

Logistic regression is a supervised machine learning algorithm widely used for binary classification tasks, such as identifying whether an email is spam or not and diagnosing diseases by assessing the presence or absence of specific conditions based on patient test results. This approach utilizes the logistic (or sigmoid) function to transform a linear combination of input features into a probability value ranging between 0 and 1. This probability indicates the likelihood that a given input corresponds to one of two predefined categories. The essential mechanism of logistic regression is grounded in the logistic function’s ability to model the probability of binary outcomes accurately. With its distinctive S-shaped curve, the logistic function effectively maps any real-valued number to a value within the 0 to 1 interval. This feature renders it particularly suitable for binary classification tasks, such as sorting emails into “spam” or “not spam”. By calculating the probability that the dependent variable will be categorized into a specific group, logistic regression provides a probabilistic framework that supports informed decision-making

Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1. Moreover, if the output of the sigmoid function (estimated probability) is greater than a predefined threshold on the graph, the model predicts that

the instance belongs to that class. If the estimated probability is less than the predefined threshold, the model predicts that the instance does not belong to the class.

The sigmoid function is referred to as an activation function for logistic regression and is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x} = 1 - \sigma(-x).$$

where  $e$  is base of natural logarithms and value is numerical value one wishes to transform

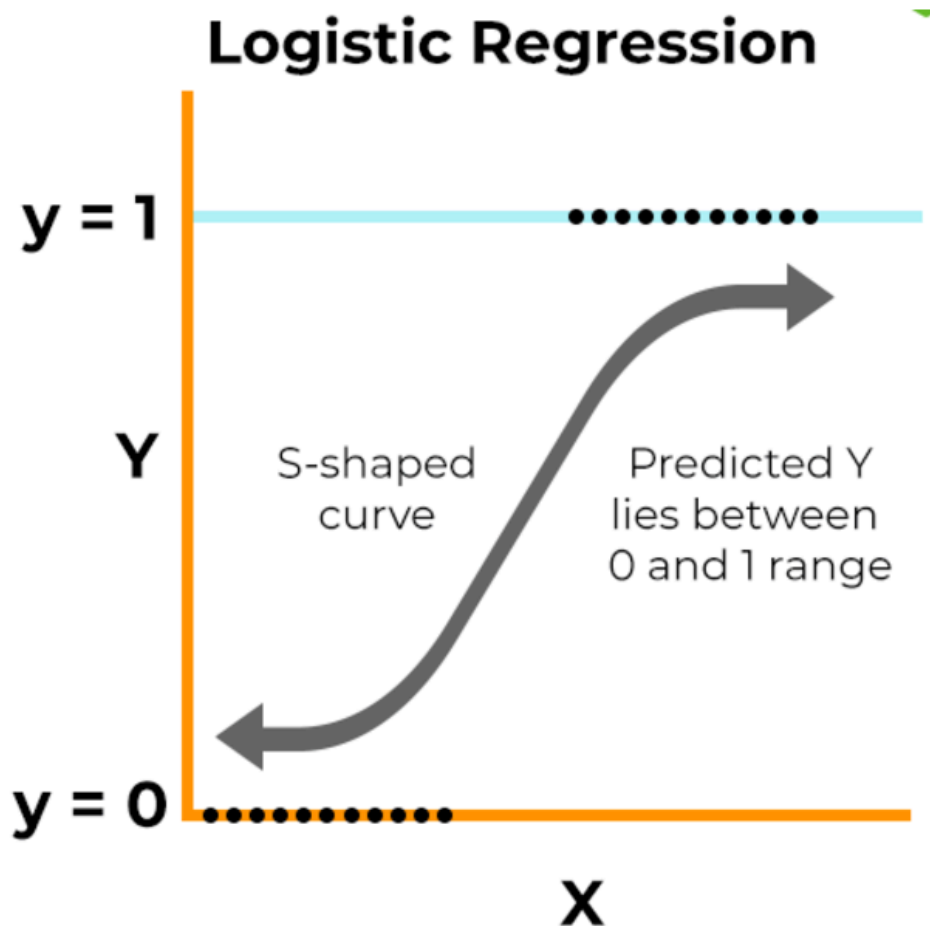


Figure 3.9. Key Assumptions for Implementing Logistic Regression

### Advantage of Logistic Regression

- Easier to implement machine learning methods: A machine learning model can be effectively set up with the help of training and testing. The training identifies patterns in the input data (image) and associates

them with some form of output (label). Training a logistic model with a regression algorithm does not demand higher computational power. As such, logistic regression is easier to implement, interpret, and train than other ML methods.

- Provides valuable insights: Logistic regression measures how relevant or appropriate an independent/predictor variable is (coefficient size) and also reveals the direction of their relationship or association (positive or negative).
- Suitable for linearly separable datasets: A linearly separable dataset refers to a graph where a straight line separates the two data classes. In logistic regression, the y variable takes only two values. Hence, one can effectively classify data into two separate classes if linearly separable data is used.

### **Disadvantage of Logistic Regression**

- Logistic regression fails to predict a continuous outcome. Let's consider an example to better understand this limitation. In medical applications, logistic regression cannot be used to predict how high a pneumonia patient's temperature will rise. This is because the scale of measurement is continuous (logistic regression only works when the dependent or outcome variable is dichotomous).
- Logistic regression assumes linearity between the predicted (dependent) variable and the predictor (independent) variables. Why is this a limitation? In the real world, it is highly unlikely that the observations are linearly separable. Let's imagine you want to classify the iris plant into one of two families: *setosa* or *versicolor*. In order to distinguish between the two categories, you're going by petal size and sepal size. You want to create an algorithm to classify the iris plant, but there's actually no clear distinction—a petal size of 2cm could qualify the plant for both the *setosa* and *versicolor* categories. So, while linearly separable data is the assumption for logistic regression, in reality, it's not always truly possible.
- Logistic regression may not be accurate if the sample size is too small. If the sample size is on the small side, the model produced by logistic regression is based on a smaller number of actual observations. This can result in overfitting. In statistics, overfitting is a modeling error which occurs when the model is too closely fit to a limited set of data because of

a lack of training data. Or, in other words, there is not enough input data available for the model to find patterns in it. In this case, the model is not able to accurately predict the outcomes of a new or future dataset.

### **3.4.3. Decision Tree**

Decision tree learning is a method commonly used in data mining.[3] The goal is to create a model that predicts the value of a target variable based on several input variables.

A decision tree is a simple representation for classifying examples. For this section, assume that all of the input features have finite discrete domains, and there is a single target feature called the “classification”. Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target feature or the arc leads to a subordinate decision node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes, signifying that the data set has been classified by the tree into either a specific class, or into a particular probability distribution (which, if the decision tree is well-constructed, is skewed towards certain subsets of classes).

A tree is built by splitting the source set, constituting the root node of the tree, into subsets—which constitute the successor children. The splitting is based on a set of splitting rules based on classification features.[4] This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT)[5] is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.[6]



- **Overfitting:** Decision trees are prone to overfitting, especially when they grow too deep or when the dataset is noisy. Deep decision trees can memorize the training data, leading to poor generalization on unseen data. Techniques like pruning or limiting the tree depth can mitigate this issue.
- **High variance:** Decision trees have high variance, meaning small changes in the training data can result in significantly different trees. Ensemble methods like Random Forest or Gradient Boosting are often used to reduce variance and improve performance.
- **Instability:** Decision trees are sensitive to small variations in the data, which can lead to different splits and, consequently, different trees. This instability makes them less reliable compared to some other algorithms.
- **Bias towards features with many levels:** Features with a large number of levels (i.e., high cardinality) tend to be favored over features with fewer levels in decision tree splits. This bias can affect the performance of the model, especially if the high-cardinality features are not truly informative.
- **Difficulty in capturing linear relationships:** Despite being able to capture non-linear relationships, decision trees struggle with capturing linear relationships between features and the target variable. Other algorithms like linear regression may perform better in such cases.

#### **3.4.4. Random Forest**

Random Forest is indeed an ensemble learning technique that is applicable to both regression and classification tasks. It is essentially an extension of bagging which approach combines several weak learners. Weak learners in a random forest are decision trees. First, we'll cover the fundamentals of decision trees before diving into the random forest algorithm.

A decision tree is a supervised learning technique that may be applied to both regression and classification. However, it is primarily utilized in problems related to classification. It is made up of numerous internal nodes, each representing a test in an attribute (for example, whether the weather tomorrow will be sunny, overcast, or rainy). Each branch in the tree represents the test outcome, whereas the leaf nodes represent the ultimate outcome (class label). It entails splitting down a training set into many subsamples.

In order to build a decision tree, it is necessary to divide the entire training data into subsets, a process executed at each internal node according to specific criteria. On the basis of metrics such as Gini impurity and information gain, a decision tree algorithm determines how best to divide each node. Gini impurity gauges how often a randomly selected item from the set would be inaccurately labeled if it were randomly assigned a label according to the subset's label distribution. Meanwhile, information gain is employed to determine which feature should be utilized for splitting at each stage of tree development. The process of splitting will continue until the inner node is assigned a class label value. The splitting process persists until each internal node is assigned a class label value. While decision trees are often interpretable and effective in certain datasets, their variance can be high due to the algorithm's greedy approach, consistently choosing the best split at each level without considering distant nodes. Consequently, here is a risk of overfitting, where the model only performs better in the training set but struggles on test sets.

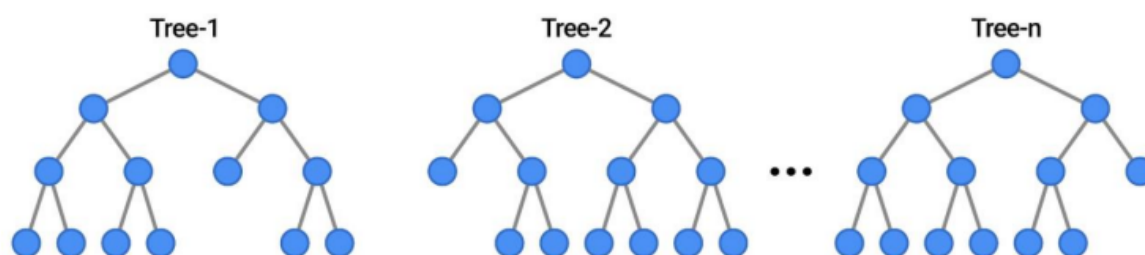


Figure 3.11. Overview of Decision Tree

To improve the overall performance of the model, Random Forest builds a number of decision trees in a simple language and combines them. As previously discussed, creating random samples of the training data with replacement. Random forest bootstrap is used to train each decision tree with various subsamples of the data. In addition, Random Forests use a subset of features on each tree. E.g. When there are 50 features in the initial data, 10 features will be selected on each tree to be trained with. Hence, for each tree, there will be 10 randomly selected features available for training, including determining the best splitting between nodes. Once a collection of decision trees has been set up, the results are aggregated into an overall prediction that can be made by means of voting. This ensemble approach

ensures robustness and generalization as multiple decision trees play a role in decisions being made. Furthermore, each tree is tuned to a set of separate data subsets to improve the model's ability to derive from unknown data. From the given input data, figure 3.10 provides a clear explanation of how Random Forest is constructed.

### **Advantages of Random Forest**

- **Robustness:** The Random Forest method can manage outliers and noisy data. It can generalize effectively to new data since it is less likely to overfit the data.
- **Accuracy:** It performs well with both continuous and categorical variables and can handle both regression and classification issues.
- **Speed:** It is a quick method that can handle enormous datasets, despite its complexity. To expedite training, it can also be readily parallelized.
- **Feature Importance:** Random Forest offers a feature importance metric that can be used to choose features and better comprehend data.
- **Excellent Results Without Extensive Parameter Tuning:** Random Forest aims to provide excellent results even with default parameters. Users can achieve competitive performance without spending significant time on parameter tuning.

### **Disadvantages of Random Forest**

- **Overfitting:** Random Forest can overfit the data even if it is less likely to do so than a single decision tree. This is especially true if the forest has an excessively high or deep number of trees.
- **Interpretability:** Because Random Forest uses numerous decision trees, it may be harder to interpret than a single decision tree. Understanding how the algorithm arrived at a specific forecast can be challenging.
- **Training Period:** Compared to other algorithms, Random Forest may require a longer training period, particularly if there are a lot of trees and they are deeply rooted.
- **Memory Usage:** Because Random Forest retains several trees, it uses more memory than other algorithms. If the dataset is large this can be an issue

### 3.4.5. K-Nearest Neighbors (KNN)

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The distinction between these terminologies is that “majority voting” technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don’t necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%. The University of Wisconsin-Madison summarizes this well with an example here ([link resmaged outside ibm.com](https://www.cs.wisc.edu/~daneel/poli.png)).

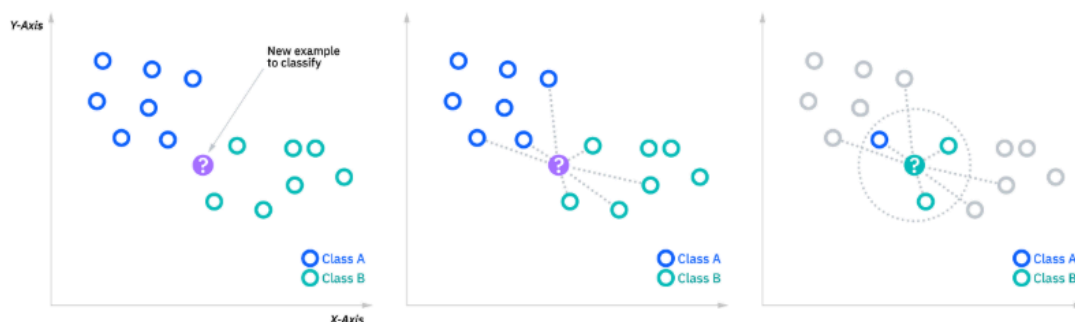


Figure 3.12. Overview of KNN

It’s also worth noting that the KNN algorithm is also part of a family of “lazy learning” models, meaning that it only stores a training dataset versus undergoing a training stage. This also means that all the computation occurs when a classification or prediction is being made. Since it heavily relies on memory to store all its training data, it is also referred to as an instance-based or memory-based learning method.

Many studies use some kind of internal validation to optimize the value of  $k$ , with the optimum value dependent upon the dataset at hand.  $k$ NN has been used in bioactivity studies of anticonvulsants and dopamine D1 agonists of kinase inhibition, cannabinoid psychoactivity, steroids, anti-inflammatories and anti-cancer drugs, athletic performance enhancement, and estrogen receptor agonists. Studies of toxicological and pharmacological relevance have looked at drug clearance, mutagenic potency, and percutaneous drug absorption.  $k$ NN has been used to predict the odor characteristics of compounds. Studies using  $k$ NN to investigate physicochemical properties have considered melting point, boiling point,  $\log P$ , aqueous solubility, and the analysis of mixtures.

### **Advantages of KNN**

- Easy to implement: Given the algorithm's simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.
- Adapts easily: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.
- Few hyperparameters:  $k$ NN only requires a  $k$  value and a distance metric, which is low when compared to other machine learning algorithms.

### **Disadvantages of KNN**

- Does not scale well: Since  $k$ NN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective. More memory and storage will drive up business expenses and more data can take longer to compute. While different data structures, such as Ball-Tree, have been created to address the computational inefficiencies, a different classifier may be ideal depending on the business problem.
- Curse of dimensionality: The  $k$ NN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs. This is sometimes also referred to as the peaking phenomenon, where after the algorithm attains the optimal number of features, additional features increases the amount of classification errors, especially when the sample size is smaller.

- Prone to overfitting: Due to the “curse of dimensionality”, KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of  $k$  can also impact the model’s behavior. Lower values of  $k$  can overfit the data, whereas higher values of  $k$  tend to “smooth out” the prediction values since it is averaging the values over a greater area, or neighborhood. However, if the value of  $k$  is too high, then it can underfit the data.

## **3.5. Evaluation Metrics**

### **3.5.1. Confusion Matrix**

The confusion matrix is a vital metric for evaluating the accuracy and precision of predictions made by a model, especially in classification tasks with multiple classes. It’s depicted as an  $N \times N$  matrix, where  $N$  is the number of class labels being classified. This matrix offers an overview of how well a predictive model performs when applied to a dataset. It’s also known as an error matrix.

Typically, the confusion matrix measures the errors or incorrect predictions when one class (e.g., class A) is misclassified as another class (e.g., class B), or vice versa. Each column in this matrix represents a predicted class, while each row corresponds to the actual class. A confusion matrix is typically represented as a 2x2 table with the following statistical measures:

- True Positive (TP): The model correctly predicted the positive class
- True Negative (TN): The model correctly predicted the negative class
- False Positive (FP): The model incorrectly predicted the positive class
- False Negative (FN): The model incorrectly predicted the negative class

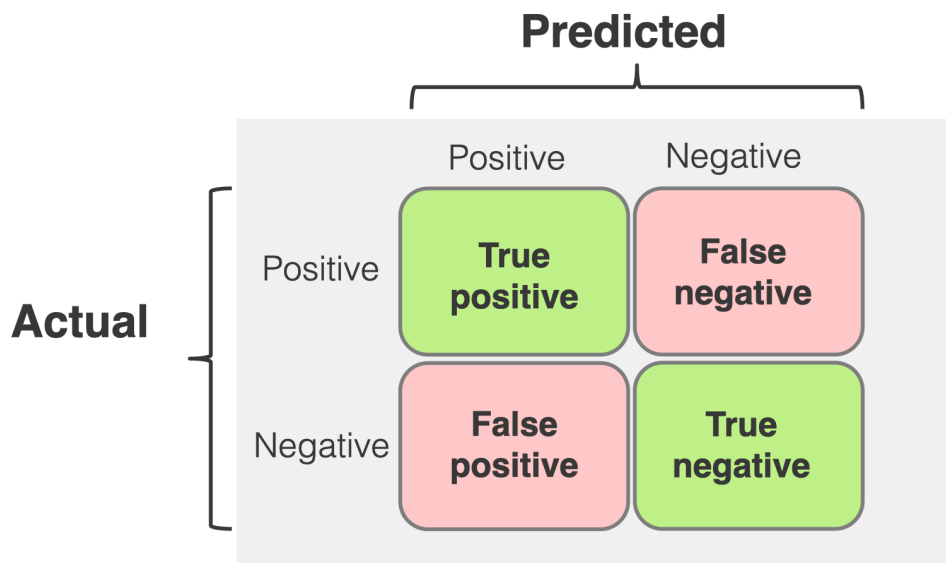


Figure 3.23. Confusion Matrix

In an ideal situation, where a classifier achieves perfection, its confusion matrix would solely contain non-zero values along its main diagonal, which extends from the top left to the bottom right. This implies that the classifier accurately identifies all true positives and true negatives while avoiding false positives and false negatives.

		Predicted	
		Non-Fraud	Fraud
Actual	Non-Fraud	True Negative (TN)	False Positive (FP)
	Fraud	False Negative (FN)	True Positive (TP)

Figure 3.24. Table of Confusion matrix for non-fraud and fraud

### 3.5.2. Precision

While the confusion matrix provides comprehensive insights, concise metrics are often easier to interpret. Precision, a vital metric, measures the accuracy of positive predictions. It is defined as follows:

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

, where

- TP: the number of True Positives
- FP: the number of False Positives

### 3.5.3. Recall

One could theoretically achieve perfect precision by making just one correct positive prediction (prediction = 1, actual = 1, resulting in 100% precision). However, this approach is impractical since it would overlook all other positive instances. Therefore, precision is typically used in conjunction with another metric, recall or True Positive Rate (TPR), which gauges the proportion of correctly identified positive instances:

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

, where

- TP: the number of True Positives
- FN: the number of False Negatives

### 3.5.4. F-beta Score

The F-beta score is a versatile metric for evaluating performance in use cases with both balanced and imbalanced data. It extends the concept of the F1 score by incorporating a parameter,  $\beta$ , where setting  $\beta$  to 1 yields the traditional F1 Score. FBeta considers both Precision and Recall and blends them through a weighted Harmonic mean. In fraud detection, both precision and recall are important. HighRecall guarantees that fraudulent transactions are accurately identified, while high Precision avoids wrongly labeling legitimate transactions as fraudulent.

Achieving high precision is vital to prevent wrongly labeling legitimate transactions as fraudulent. Additionally, F-beta is a thresholded metric, indicating that we need to apply thresholds initially to obtain actual binary predictions

$$F_{\beta} = \frac{(1+\beta^2)*Precision*Recall}{\beta^2*Precision+Recall}$$

In this section, we will focus and delve into F1 score. F1 score represents the harmonic mean of precision and recall, offering a balanced assessment of a

model's performance. It shines in scenarios where both precision and recall hold equal significance in a classification task. Ranging from 0 (poorest) to 1 (optimal), a higher F1 Score denotes better overall performance in maintaining a balance between precision and recall.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

### **3.5.5. Area Under the Curve (AUC)**

One could theoretically achieve perfect AUC (Area Under the Curve) by having a model that always ranks the positive instances higher than the negative ones, resulting in a perfect separation between the classes. However, this scenario is rare and unrealistic in practical applications, where class overlap and misclassifications are inevitable. Therefore, AUC is often used in conjunction with other metrics, such as ROC (Receiver Operating Characteristic) curve, to evaluate the performance of a classifier. The AUC measures the ability of the model to distinguish between positive and negative classes, with a higher AUC indicating better model performance. An AUC score of 1 represents perfect classification, while a score of 0.5 suggests random guessing, indicating that the model has no discriminatory power. AUC is particularly useful when dealing with imbalanced datasets, as it provides an aggregate measure of the model's ability to rank predictions across different thresholds.

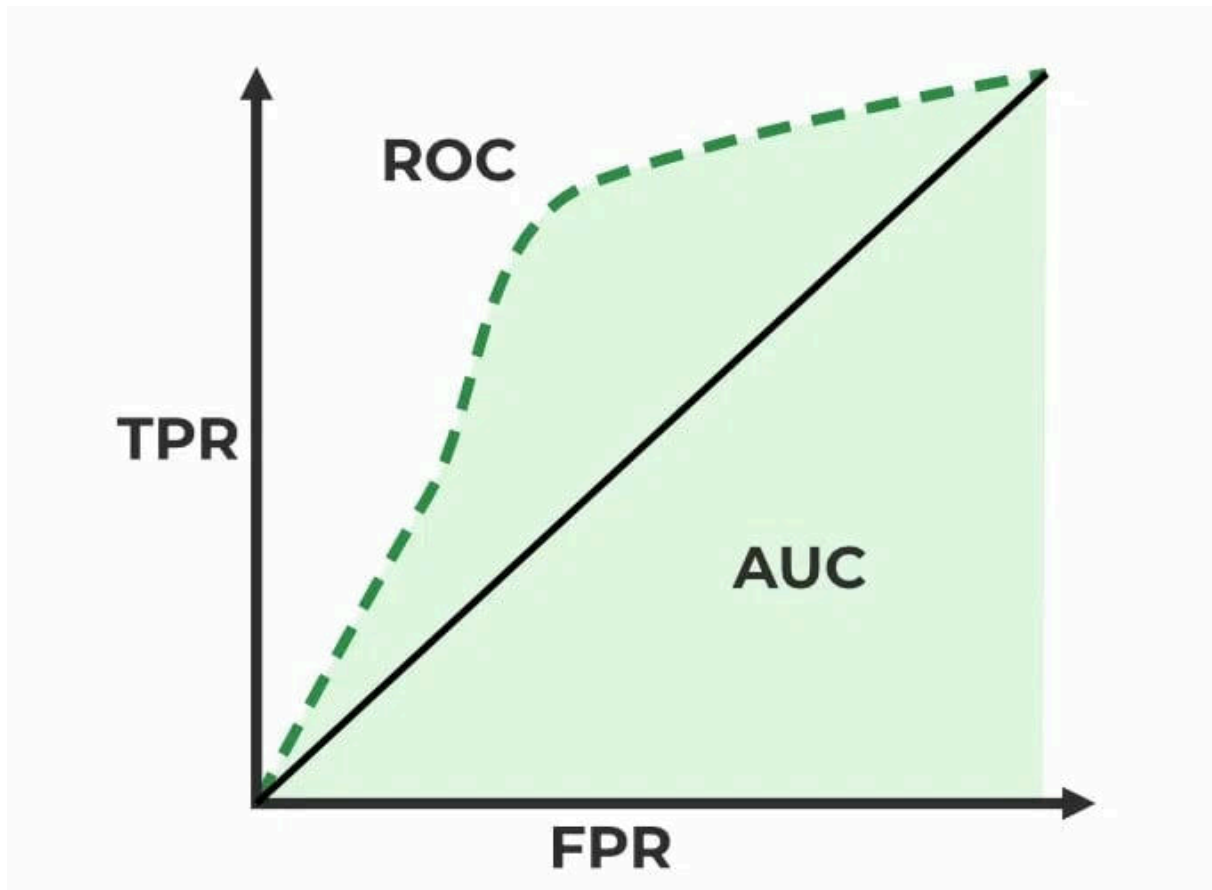


Figure 3.25. ROC-AUC Classification Rating Index

## 3.6. Dataset

### 3.6.1. Data Description

The dataset used in this thesis, specifically the **Base.csv** file from the BAF dataset, contains bank account-related transactions and attributes designed for fraud detection analysis. This dataset simulates transactional data to support the identification of fraudulent activities in bank accounts. It includes detailed customer and transaction-level information for modeling and evaluation purposes. The dataset consists of a large number of records, capturing various features that describe account activities, demographic details, and historical behavior.

The **Base.csv** file contains structured data with attributes such as income levels, account activity velocity, days since certain events, and categorical features like payment types and employment status. It is a highly imbalanced dataset, where the fraudulent transactions account for only a small proportion of the total. The imbalanced nature of the dataset

reflects real-world scenarios, where fraudulent activities are significantly rarer compared to legitimate ones.

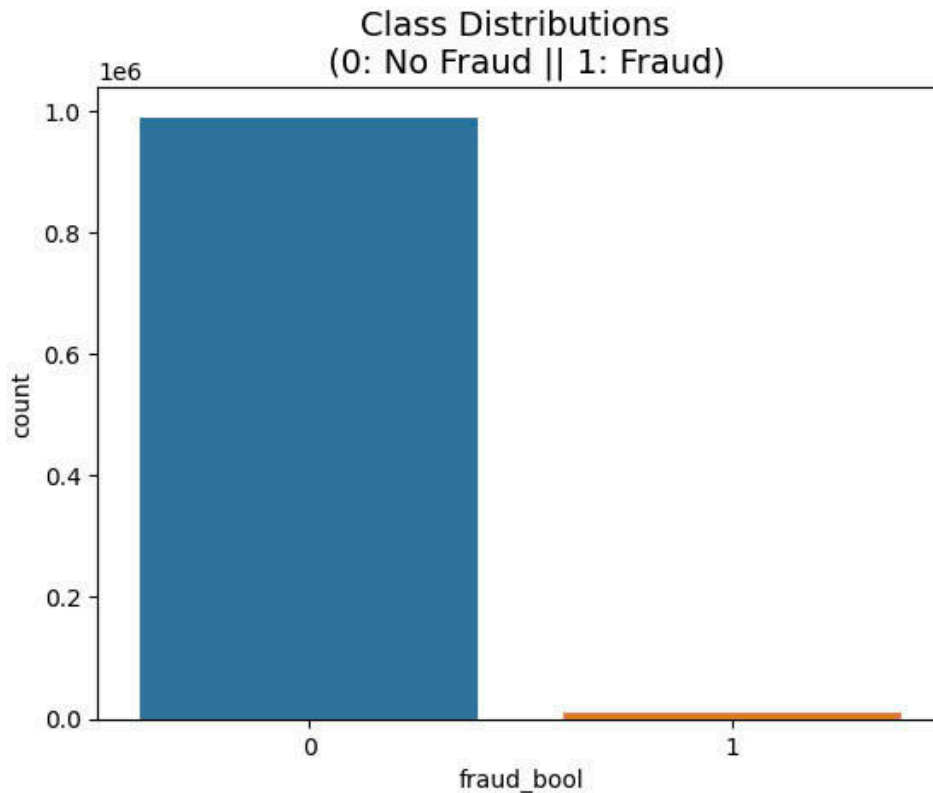


Figure 3.26. A comparison of the number of Fraud and Non-fraud transactions

Variable	Data Type	Description
income	numeric	Annual income of the applicant (in decile form). Ranges between [0.1, 0.9]
name_email_similarity	numeric	Metric of similarity between email and applicant's name. Higher values represent higher similarity. Ranges between [0, 1]
prev_address_months_count	numeric	Number of months in previously registered address of the

		applicant, i.e., the applicant's previous residence, if applicable. Ranges between $[-1, 380]$ ( $-1$ is a missing value).
current_address_months_count	numeric	Months in currently registered address of the applicant. Ranges between $[-1, 429]$ months ( $-1$ is a missing value).
customer_age	numeric	Applicant's age in years, rounded to the decade. Ranges between $[10, 90]$ years.
days_since_request	numeric	Number of days passed since application. Ranges between $[0, 79]$ days.
intended_balcon_amount	numeric	Initial transferred amount for application. Ranges between $[-16, 114]$ (negatives are missing values).
payment_type	categorical	Credit payment plan type. 5 possible (anonymized) values.
zip_count_4w	numeric	Number of applications within the same zip code in the last 4 weeks. Ranges between $[1, 6830]$
velocity_6h	numeric	Velocity of total applications made in the last 6 hours, i.e., average number of applications per hour in the last 6 hours. Ranges between $[-1175, 16818]$

velocity_24h	numeric	Velocity of total applications made in the last 24 hours, i.e., average number of applications per hour in the last 24 hours. Ranges between [1297, 9586]
velocity_4w	numeric	Velocity of total applications made in the last 4 weeks, i.e., average number of applications per hour in the last 4 weeks. Ranges between [2825, 7020]
bank_branch_count_8w	numeric	Number of total applications in the selected bank branch in the last 8 weeks. Ranges between [0, 2404]
date_of_birth_distinct_emails_4w	numeric	Number of applications with the same date of birth in the last 4 weeks. Ranges between [0, 39]
employment_status	categorical	Employment status of the applicant. 7 possible (anonymized) values
credit_risk_score	numeric	Numeric score of application risk. Ranges between [−191, 398]
phone_home_valid	binary	Validity of provided home phone
phone_mobile_valid	binary	Validity of provided mobile phone
bank_months_count	numeric	How old is the previous account (if held) in months. Ranges between

		$[-1, 82]$ (-1 is a missing value)
has_other_cards	binary	If the applicant has other cards from the same banking company
proposed_credit_limit	numeric	Applicant's proposed credit limit. Ranges between $[200, 2000]$
foreign_request	binary	If the origin country of request is different from the bank's country
source	categorical	Online source of application. Either browser (INTERNET) or app (TELEAPP)
session_length_in_minutes	numeric	Length of user session on the banking website in minutes. Ranges between $[-1, 107]$ (-1 is a missing value)
device_os	categorical	Operating system of the device that made the request. Possible values are: Windows, macOS, Linux, X11, or other
keep_alive_session	binary	User option on session logout
device_distinct_emails_8w	numeric	Number of distinct emails in the banking website from the used device in the last 8 weeks. Ranges between $[-1, 2]$ (-1 is a missing value)
device_fraud_count	numeric	Number of fraudulent applications with the used

		device. Ranges between [0, 1]
month	numeric	Month where the application was made. Ranges between [0, 7]
fraud_bool	binary	If the application is fraudulent or not

**Table 3.3. Summary of raw features of dataset**

The `Base.csv` file from the BAF dataset contains a mix of numeric and categorical variables, representing various aspects of credit applications and applicant profiles. The dataset includes 18 features, such as applicant income, email similarity, address history, and application velocity metrics. These features are not transformed but represent specific characteristics of the applicant and the application process. Key features include ‘income’, indicating the applicant’s annual income in deciles, and ‘velocity\_24h’, which reflects the average number of applications per hour in the last 24 hours. These features are crucial for understanding application patterns and identifying fraudulent behavior. The dataset provides a rich foundation for fraud detection, with detailed, interpretable features that allow for exploratory analysis and machine learning model development.

### 3.6.2. Exploratory Data Analysis (EDA)

In this section, we explore and understand the dataset and extract some valuable insights within it, which can support Feature Engineering and Model Development phases.

#### I. Correlations of all variables

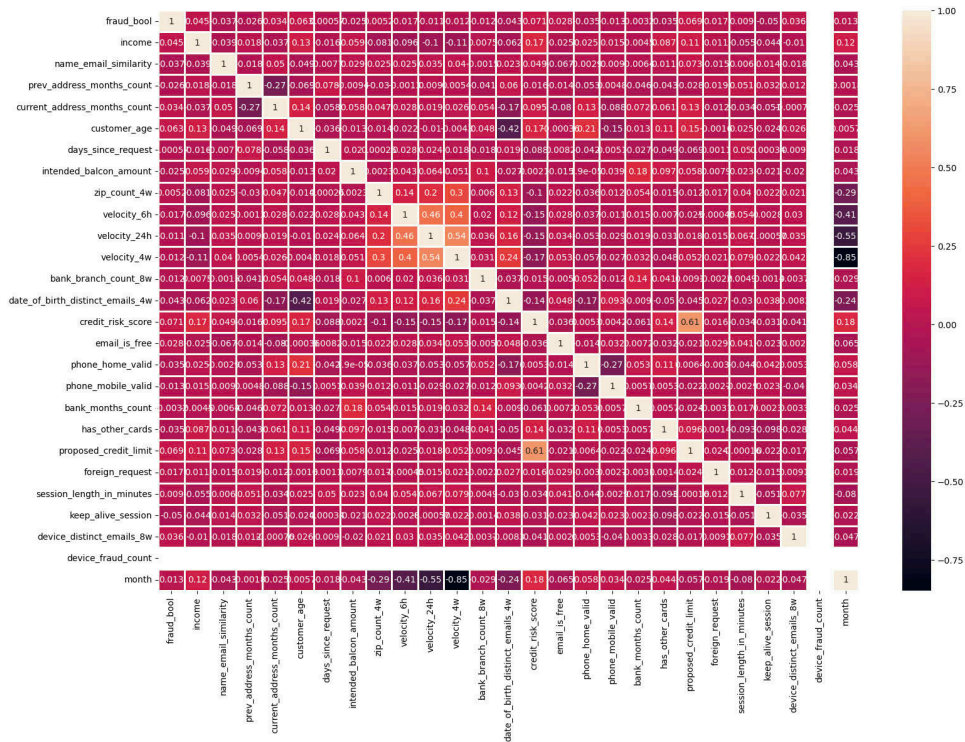


Figure 3.27. Heatmap for Correlation

The image provided depicts a correlation heatmap that visually represents the relationships between various features in the dataset. Correlation values range from  $-1$  to  $1$ , where:

- $+1$  indicates a perfect positive correlation: as one feature increases, the other also increases proportionally.
- $-1$  indicates a perfect negative correlation: as one feature increases, the other decreases proportionally.
- $0$  indicates no correlation.

The heatmap is color-coded, with lighter shades indicating stronger correlations (positive or negative) and darker shades representing weaker or no correlation.

## Key Observations from the Heatmap:

### 1. Target Variable (fraud\_bool):

- The fraud\_bool column represents the binary target variable, likely used to predict fraudulent behavior.
- Most features have a weak correlation with fraud\_bool, as indicated by the low correlation values (close to 0). This suggests that predicting

fraud may require combining features rather than relying on individual ones.

## **2. Strongly Correlated Features:**

- `velocity_4w` and `velocity_24h` (0.85): These two features, representing transaction velocity over different timeframes, exhibit a very strong positive correlation. It may indicate redundancy, and one could potentially be dropped or transformed during preprocessing.
- `month` and `velocity_4w` (0.80): These features show a strong relationship, possibly due to seasonal trends in velocity patterns.

## **3. Moderate Correlations:**

- `velocity_6h` and `velocity_24h` (0.54): Indicates that transaction activity within shorter timeframes aligns moderately with longer periods.
- `zip_count_4w` and `velocity_4w` (0.30): Suggests a connection between geographic activity (zip code changes) and transaction patterns.

## **4. Weak or No Correlation:**

- Most features, such as `email_is_free`, `phone_home_valid`, and `session_length_in_minutes`, exhibit weak correlations with other variables, suggesting they may provide unique or non-linear contributions to the target variable.

## **5. Potential Feature Groups:**

- Velocity-related Features: (`velocity_6h`, `velocity_24h`, `velocity_4w`) show consistent interrelationships, hinting at their importance in capturing transaction patterns over time.
- User Profile Features: Features like `income`, `customer_age`, and `current_address_months_count` have limited correlations with `fraud_bool` or other features, which may suggest they act as secondary predictors.

## **Implications for Feature Selection and Model Building:**

### **1. Feature Redundancy:**

Strongly correlated features, such as `velocity_4w` and `velocity_24h`, may introduce multicollinearity in the model. Dimensionality reduction

techniques like Principal Component Analysis (PCA) or selecting one representative feature could mitigate this issue.

## 2. Non-linear Relationships:

The weak correlations of most features with `fraud_bool` suggest the possibility of non-linear relationships. Advanced models like tree-based algorithms or neural networks may capture these interactions better than linear models.

## 3. Interaction Effects:

Certain features, such as `zip_count_4w` and `velocity_4w`, might provide more predictive power when combined, indicating a need for interaction term creation.

## 4. Further Analysis:

While the heatmap gives an overview, analyzing feature distributions, outliers, and class imbalances (if any) will provide a more comprehensive understanding of the dataset.

## II. Density Plot: “prev\_address\_months\_count”

The density plot for `prev_address_months_count` reveals the distribution of the number of months a customer has lived at their previous address.

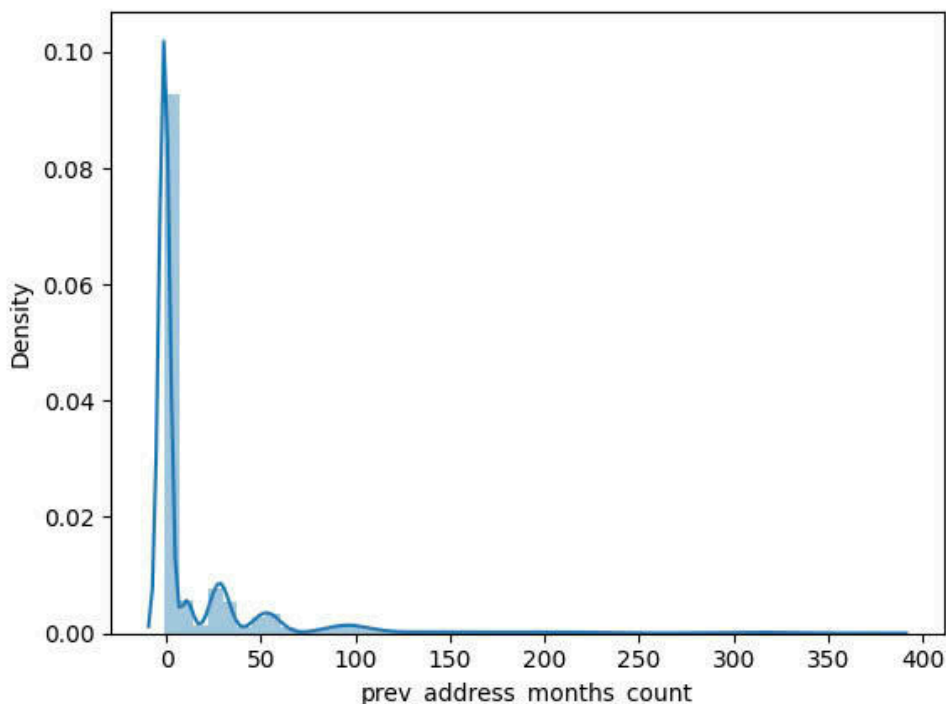


Figure 3.28. The density plot for `prev_address_months_count`

**1. Distribution Shape:**

The distribution is right-skewed, with a high concentration of values between 0–50 months. A small number of values extend beyond 100 months, indicating the presence of potential outliers.

**2. Customer Behavior:**

Most customers tend to move frequently or have not stayed long at their previous address, as evidenced by the sharp peak near lower values. Longer durations at a single address are less common, possibly signaling more stable individuals.

**3. Potential Issues:**

The long tail in the distribution suggests the presence of outliers (e.g., values  $>100$  months), which could disproportionately impact model training if not addressed. The skewness may also affect algorithms sensitive to non-normal distributions.

**4. EDA conclusion:**

- This feature may reflect the stability of the customer's address, which is an important factor for risk assessment.
- Outliers need to be handled and data normalized to ensure that this feature contributes effectively to the prediction model.

**III. Density Plot: “customer\_age”**

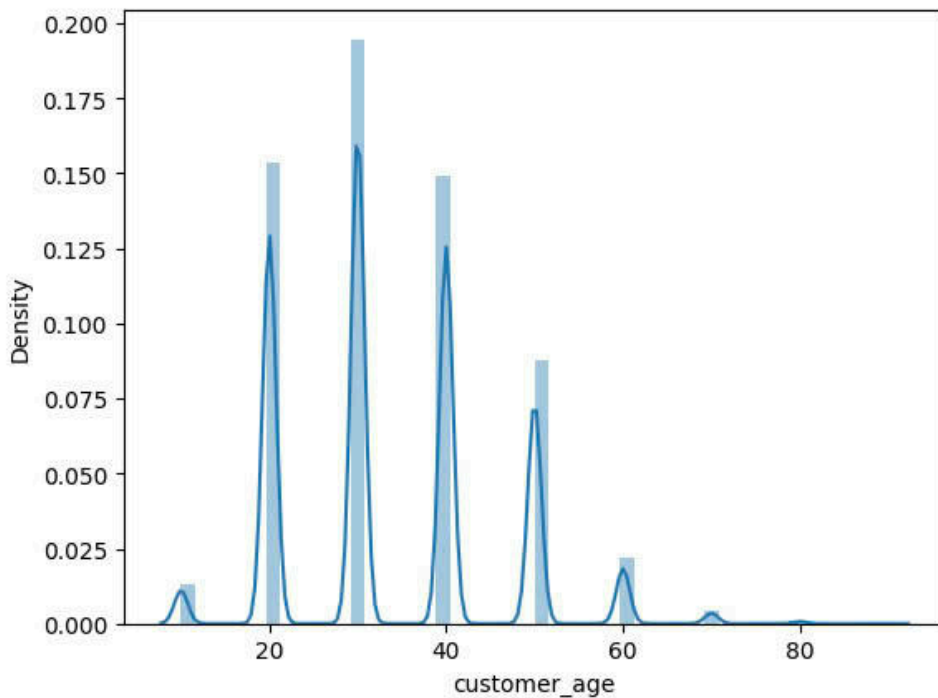


Figure 3.29. The density plot for customer\_age

### 1. Distribution Shape:

The distribution of the customer\_age variable is multimodal, with prominent peaks around 20, 30, 40, and 50 years old. There are very few values beyond 60 years, indicating a sharp decline in higher age groups.

### 2. Customer Behavior:

Most customers are concentrated in age groups like 20s, 30s, 40s, and 50s, possibly reflecting critical life stages when individuals actively engage in financial or service-related activities. There are significantly fewer customers in the above-60 age group, suggesting that older individuals are not the primary target audience.

### 3. Potential Issues:

Multiple peaks could pose challenges for algorithms that do not handle non-normal distributions well, such as linear models. A slight skewness (due to the lower representation of older age groups) might impact the balance of the dataset.

### 4. EDA Conclusion:

This feature likely reflects the target age groups of the customers, helping to analyze and predict behavior based on age. It might be necessary to

handle the less-represented age groups (e.g., by grouping or smoothing the multimodal distribution) to improve the efficiency of the predictive model.

#### IV. Density Plot: “days\_since\_request”

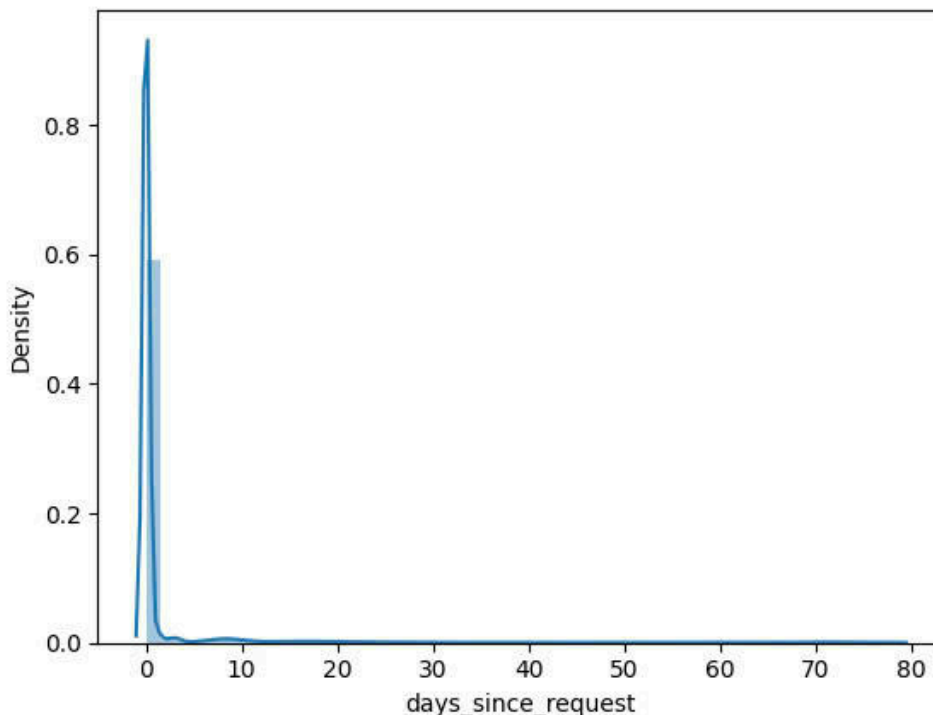


Figure 3.30. The density plot for days\_since\_request

##### 1. Distribution Shape:

The distribution of the days\_since\_request variable is highly right-skewed, with the majority of values concentrated between 0 and 5 days. A long tail extends beyond 10 days, indicating a small number of instances with significantly higher values.

##### 2. Customer Behavior:

Most requests are recent, occurring within a few days, as shown by the sharp peak at lower values. There are fewer cases with longer time spans since the request, which might represent exceptions or delayed responses.

##### 3. Potential Issues:

The long tail in the distribution suggests the presence of outliers (e.g., values greater than 10 days), which could disproportionately influence the training of certain machine learning models. The extreme skewness

might require transformations (e.g., logarithmic) to better suit algorithms sensitive to non-normal data.

#### 4. EDA Conclusion:

This feature is likely indicative of how quickly requests are processed or acted upon, making it a potentially important predictor. To enhance its contribution to the predictive model, consider handling outliers and applying normalization or transformation to reduce skewness.

#### 3.6.3. Data Preprocessing

Data preprocessing is a very important step to ensure the quality of input data, thereby optimizing the performance of the machine learning model. In this problem, numerical features are normalized using the **RobustScaler** tool from the **sklearn.preprocessing** library. The reason for using **RobustScaler** instead of other methods such as **StandardScaler** or **MinMaxScaler** is because it is less sensitive to outliers. This is especially useful when the dataset contains columns with large variations or unusual values, helping to minimize the impact of deviant data points.

**The normalization process is performed by:**

- Defining the list of numeric columns to be normalized (`num_col_scale`).
- Using **RobustScaler**'s `fit_transform` to transform the values, bringing them to a more stable distribution, improving the learning ability of the algorithms.

The final result is input data with normalized features, ensuring that the model can learn effectively from the dataset without being affected by scale or outliers.

#### 3.6.4. Data Splitting

Splitting the data into training and testing sets is an essential step for model evaluation. In this problem, the data is split based on the **month** column, a time feature that represents the month in which the event occurred. The time-based split method is highly practical, as the previous months (0-5) are used as the training set, while the later months (6-7) are used as the test set.

**The data split is performed as follows:**

- **Feature separation and target label:** The input variable (X) is split from the entire data by removing the `fraud_bool` column, while this column is retained as the target label (y).
- **Splitting the data by month:** Rows with `month < 6` are used as the training set (X\_train, y\_train), while rows with `month >= 6` form the test set (X\_test, y\_test).
- **Remove month column:** After splitting, the month column is removed to ensure that the model does not use time information directly, avoiding data leakage.

This time-based splitting method not only helps to evaluate the model more realistically, but also ensures that the test data is not affected by any information in the training set. This is especially important for time-sensitive problems, such as fraud detection or time series forecasting.

## 4. Chapter 4: Results Analysis and Discussions

The performance of all the models in this project was evaluated using several key metrics, including Precision, Accuracy, and AUC (Area Under the Curve, derived from the ROC curve). Among these, AUC was emphasized, as it provides a comprehensive measure of the model's ability to distinguish between the positive (fraudulent) and negative (non-fraudulent) classes across various thresholds. Accuracy was also considered, as it reflects the overall correctness of the model in predicting both classes. However, in imbalanced datasets such as fraud detection, Accuracy alone may not provide a full picture of model performance, which is why other metrics, like Precision, are also crucial.

### 4.0.1. Original Data (Without Resampling):

Without resampling techniques, the models struggled with class imbalance, leading to poor performance in detecting fraudulent transactions. While Precision was relatively high, the models predominantly predicted the majority class (non-fraudulent) correctly, missing many fraud cases and resulting in false negatives. This is problematic in fraud detection, where missing a fraud case is more costly than flagging a non-fraudulent one. Accuracy, though high, was misleading, as it masked the models' inability to effectively detect fraud. Similarly, AUC scores were impacted by the imbalance, as the models could achieve decent AUC by correctly predicting the majority class, without effectively distinguishing fraudulent transactions.

### 4.0.2. Oversampling:

#### SMOTETomek:

When SMOTETomek was applied to handle the class imbalance, the performance of the models significantly improved. SMOTETomek, by combining SMOTE (which generates synthetic samples for the minority class) and Tomek Links (which removes noisy or borderline examples), effectively balanced the dataset. This led to better detection of fraudulent transactions, as the models were no longer biased toward predicting the majority class (non-fraudulent). Precision and Recall improved notably, with models achieving a better balance between identifying

fraudulent transactions (higher Recall) and minimizing false positives (higher Precision). Accuracy, while still useful, no longer masked the true performance of the models, as the balance between classes enabled more reliable predictions. The AUC score also saw an increase, reflecting the model's enhanced ability to distinguish between fraudulent and non-fraudulent transactions across various thresholds. Overall, SMOTETomek helped address the class imbalance issue, leading to improved model performance, especially in detecting the minority class (fraudulent transactions), which is the key objective of fraud detection systems.

### SMOTETomek

Model	Precision	Recall	F1-score	ROC-AUC Score
Logistic Regression	0.85	0.85	0.85	0.93
KNN	0.75	0.71	0.7	0.76
Naive Bayes (Gaussian)	0.77	0.74	0.73	0.88
Naive Bayes (Bernoulli)	0.83	0.83	0.83	0.91
Decision Tree	0.9	0.88	0.88	0.88
Random Forest	0.92	0.9	0.9	0.99

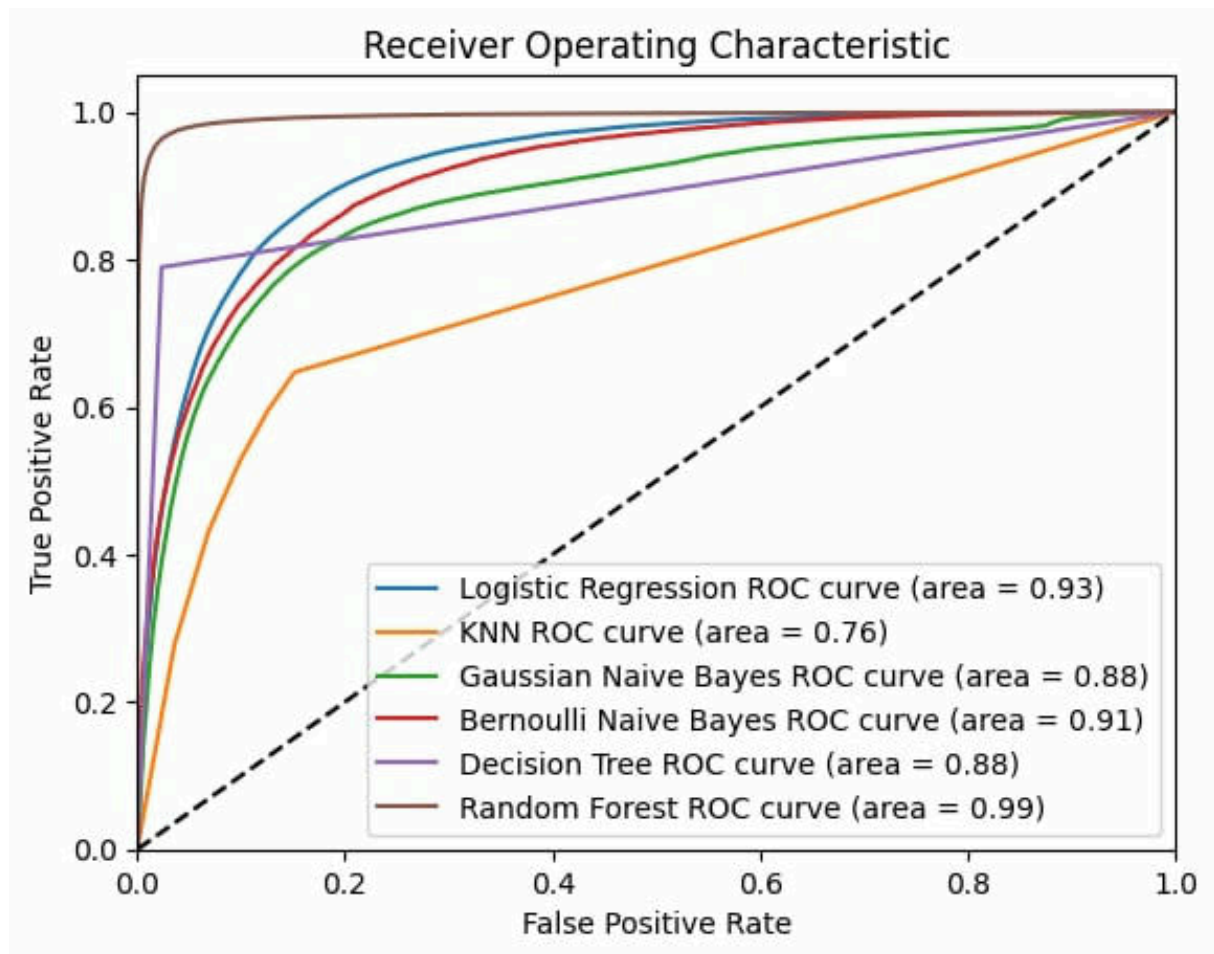


Figure 4.1. Receiver Operating Characteristic using SMOTETomek

### ADASYN:

The application of ADASYN (Adaptive Synthetic Sampling) further enhanced model performance by adaptively generating synthetic samples for the minority class, focusing on harder-to-learn examples. Unlike other resampling techniques, ADASYN places more emphasis on areas where the minority class is underrepresented or difficult to distinguish from the majority class. By improving the representation of the minority class, ADASYN led to a noticeable increase in Recall, enabling the models to identify more fraudulent transactions. While Precision was also improved, the trade-off between Precision and Recall was better balanced compared to the non-resampled models, as ADASYN generated more relevant synthetic samples. The AUC score showed a marked improvement, indicating that the models were better able to differentiate between fraudulent and non-fraudulent transactions.

## ADASYN

Model	Precision	Recall	F1-score	ROC-AUC Score
Logistic Regression	0.85	0.85	0.85	0.92
KNN	0.74	0.71	0.69	0.75
Naive Bayes (Gaussian)	0.77	0.73	0.73	0.87
Naive Bayes (Bernoulli)	0.83	0.83	0.83	0.91
Decision Tree	0.91	0.89	0.89	0.89
Random Forest	0.92	0.9	0.9	0.99

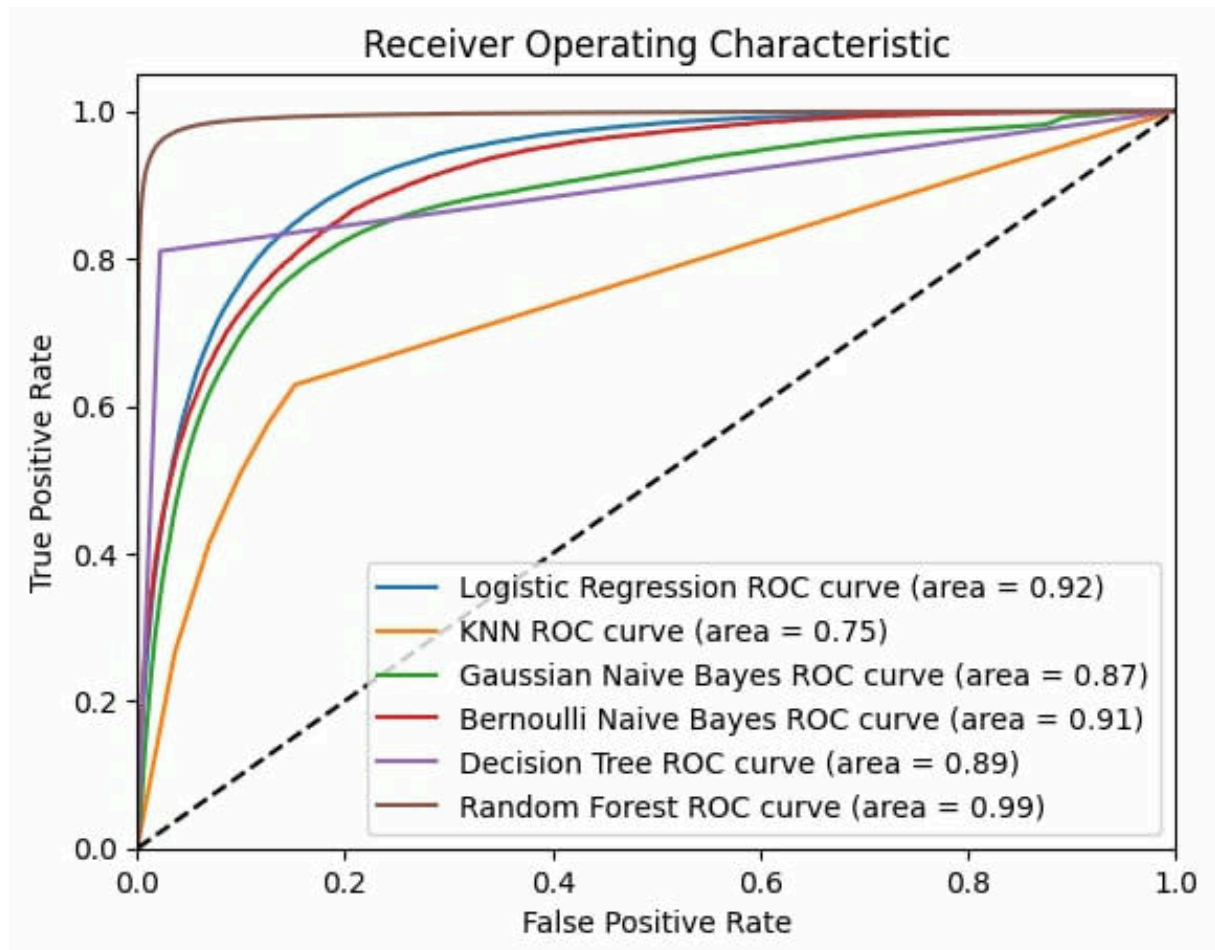


Figure 4.2. Receiver Operating Characteristic using ADASYN

### SMOTEENN:

The use of SMOTEENN (Synthetic Minority Oversampling Technique with Edited Nearest Neighbors) further improved model performance by

combining the benefits of SMOTE for oversampling the minority class with Edited Nearest Neighbors (ENN) to remove noisy, borderline, or misclassified examples. This hybrid approach not only balanced the dataset but also cleaned it by eliminating potentially problematic instances, leading to more accurate model predictions. Recall improved as SMOTEENN generated synthetic samples for the minority class and removed noisy examples, allowing the model to better capture fraudulent transactions. Precision was also enhanced, as the removal of borderline examples helped reduce false positives, improving the reliability of the fraud predictions. The AUC score showed further enhancement, indicating that SMOTEENN helped the models better discriminate between fraudulent and non-fraudulent transactions.

### SMOTEENN

Model	Precision	Recall	F1-score	ROC-AUC Score
Logistic Regression	0.87	0.87	0.87	0.95
KNN	0.78	0.76	0.75	0.8
Naive Bayes (Gaussian)	0.79	0.76	0.76	0.9
Naive Bayes (Bernoulli)	0.85	0.85	0.85	0.93
Decision Tree	0.88	0.87	0.86	0.87
Random Forest	0.92	0.91	0.91	1

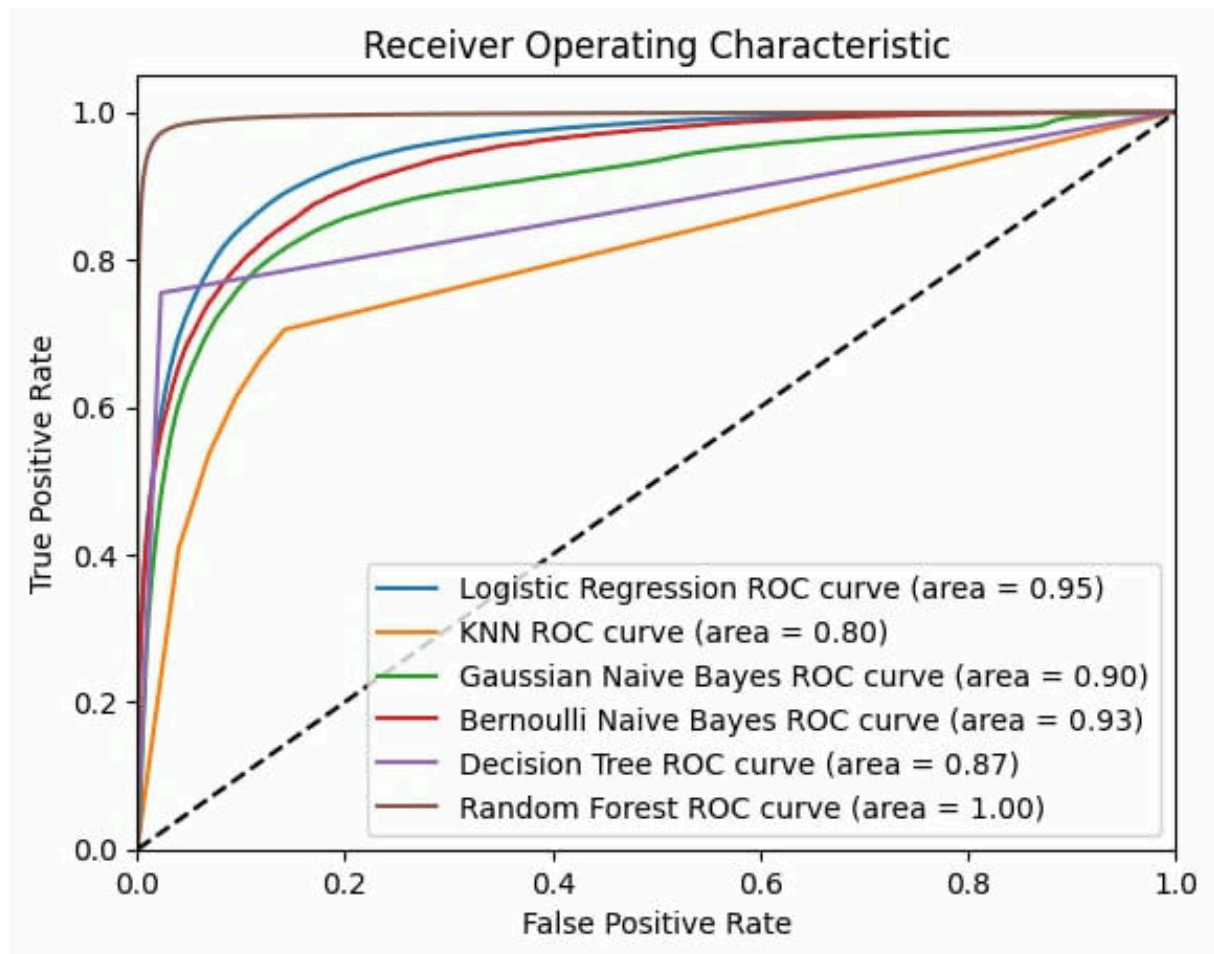


Figure 4.3. Receiver Operating Characteristic using SMOTEENN

### SMOTE:

The application of SMOTE (Synthetic Minority Oversampling Technique) significantly improved the performance of the models by generating synthetic examples for the minority class. SMOTE works by creating new, synthetic instances of the minority class through interpolation between existing minority class samples, rather than simply duplicating them. Recall was notably improved, as SMOTE increased the number of minority class instances, enabling the model to capture more fraudulent transactions. Precision also showed improvements, as SMOTE-generated samples helped the model better differentiate between fraudulent and non-fraudulent transactions. The AUC score increased, reflecting the model's enhanced ability to distinguish between the positive and negative classes across different thresholds.

## SMOTE

Model	Precision	Recall	F1-score	ROC-AUC Score
Logistic Regression	0.85	0.85	0.85	0.93
KNN	0.75	0.71	0.7	0.76
Naive Bayes (Gaussian)	0.77	0.74	0.73	0.88
Naive Bayes (Bernoulli)	0.83	0.83	0.83	0.91
Decision Tree	0.9	0.88	0.88	0.88
Random Forest	0.92	0.9	0.9	0.99

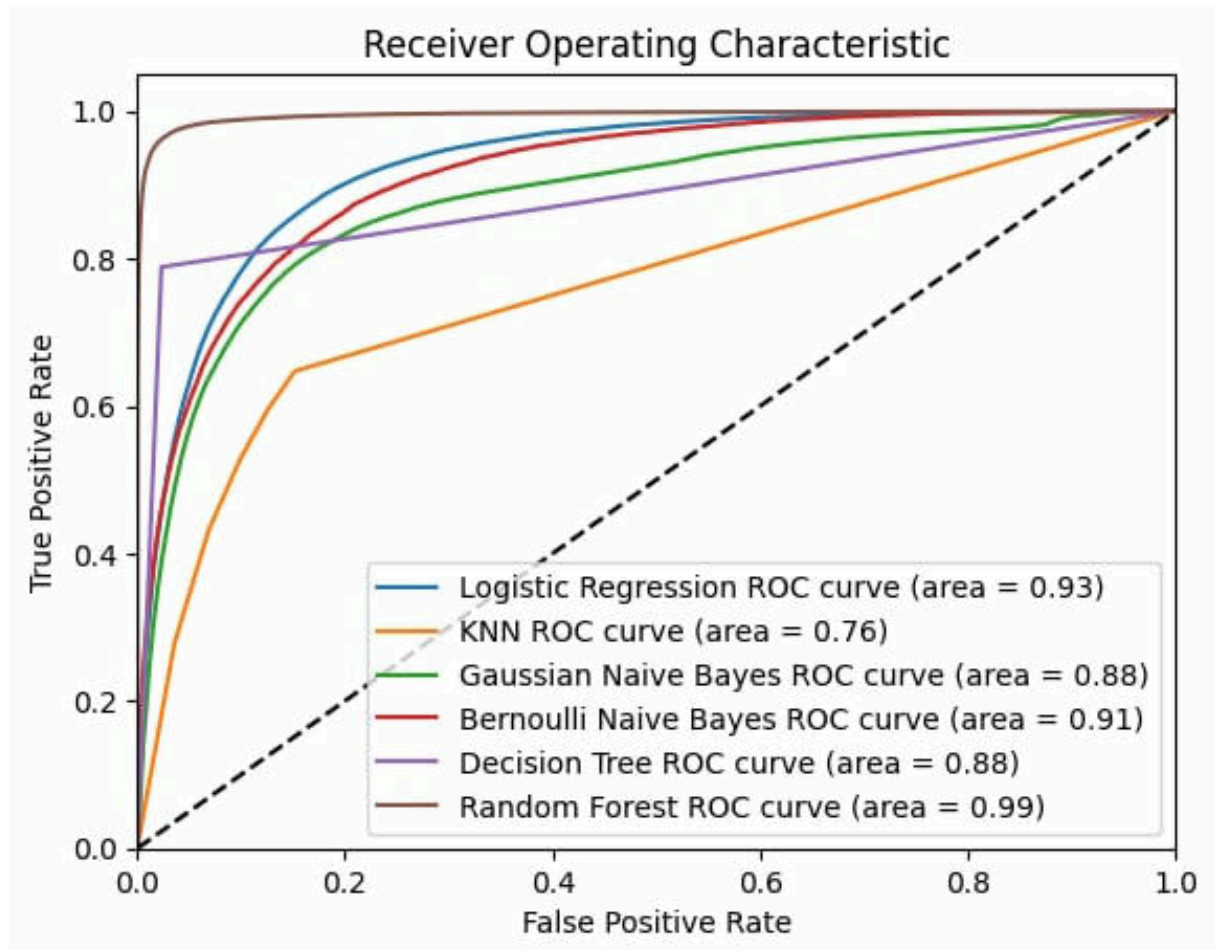


Figure 4.4. Receiver Operating Characteristic using SMOTE

### 4.0.3. Undersampling:

The use of RandomUnderSampler addressed the class imbalance by reducing the majority class instances, leading to a more balanced

dataset. By randomly eliminating some of the majority class samples, the model was better able to focus on detecting the minority class (fraudulent transactions). Recall improved, as the model could better identify fraudulent cases due to the more balanced data. Precision also benefited, with fewer false positives as the model was trained on a more representative distribution of both classes. AUC showed a positive shift, reflecting the model's improved ability to distinguish between the two classes.

### **Random Under Sample**

<b>Model</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-score</b>	<b>ROC-AUC Score</b>
Logistic Regression	0.8	0.8	0.8	0.88
KNN	0.72	0.72	0.72	0.78
Naive Bayes (Gaussian)	0.76	0.76	0.76	0.82
Naive Bayes (Bernoulli)	0.76	0.76	0.76	0.84
Decision Tree	0.67	0.67	0.67	0.67
Random Forest	0.8	0.8	0.8	0.87

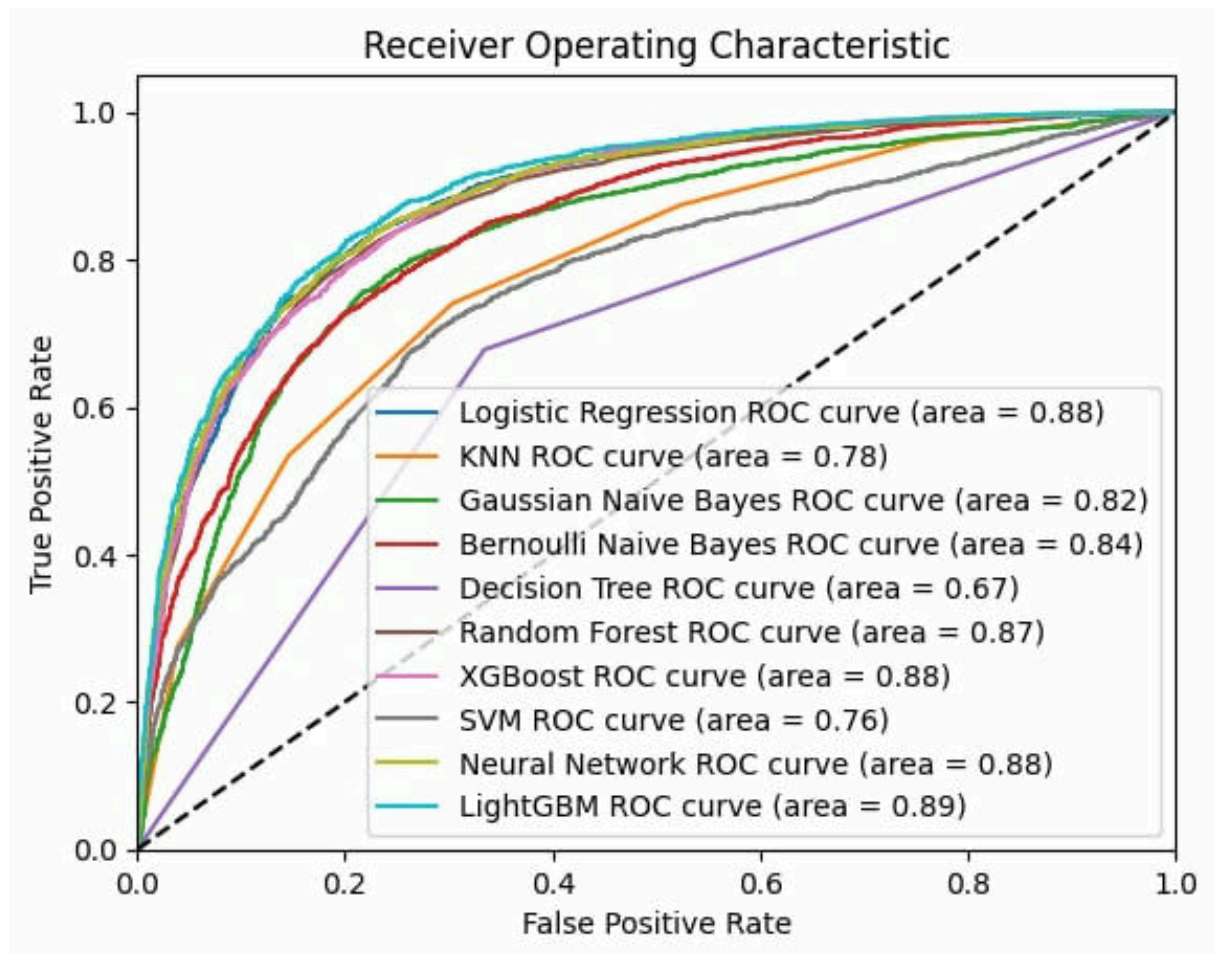


Figure 4.5. Receiver Operating Characteristic using Undersampling

#### 4.0.4. Discussion for results:

##### Rank of models

Model	Rank
Random Forest	1
Logistic Regressio	2
Bernoulli Naive Bayes	3
Decision Tree	4
Gaussian Naive Bayes	5
KNN	6

The results of this study underscore the importance of selecting the right machine learning models for fraud detection. Among the models evaluated, Random Forest emerged as the best performer, ranking first in overall performance. This model demonstrated the highest effectiveness

in detecting fraudulent transactions, reflecting its robust ability to learn complex patterns from the data. The Random Forest model is particularly valuable for financial institutions, as it can significantly reduce the risk of fraud, protecting banks and customers from substantial financial losses.

Additionally, Logistic Regression (ranked second) and Bernoulli Naive Bayes (ranked third) also performed well in identifying fraudulent transactions. These models are simpler and less computationally intensive compared to more complex algorithms like Random Forest, making them suitable for scenarios where computational resources or speed are priorities. However, while they were effective in some cases, they did not perform as well as Random Forest, indicating that more advanced models tend to better handle the complexities of fraud detection.

On the other hand, models such as Decision Tree, Gaussian Naive Bayes, and KNN ranked lower in the evaluation. These models showed some ability to identify fraud, but their performance was not as consistent as that of the higher-ranked models. Specifically, KNN (ranked sixth) demonstrated the weakest performance, highlighting the challenges of using distance-based algorithms in imbalanced datasets like fraud detection, where the minority class (fraudulent transactions) is underrepresented.

The high Recall achieved by Random Forest and Logistic Regression indicates their strong capability to detect fraudulent transactions, minimizing the risk of missing potentially harmful activities. In the context of fraud detection, maximizing Recall is crucial, as missing a fraudulent transaction can lead to significant financial losses and reputational damage for financial institutions.

Moreover, the Precision values of these models ensure that fraudulent transactions are accurately identified without misclassifying too many legitimate transactions as fraud. This balance between Precision and Recall is important for maintaining customer trust and satisfaction, as customers expect minimal disruptions to their legitimate transactions.

Given the emphasis on minimizing false negatives in fraud detection, the F2 Score metric aligns with the strategic goals of financial institutions, focusing on improving fraud detection rates while minimizing the risks associated with missed fraudulent activities. The PR-AUC scores further emphasize

the models' ability to differentiate between fraudulent and non-fraudulent transactions, ensuring that resources can be effectively allocated for further investigation and prevention efforts.

In conclusion, the results highlight that Random Forest is the most suitable model for fraud detection, offering the best balance of high Recall, Precision, and AUC. However, it is important to note that fraud detection is a dynamic challenge, with evolving fraud patterns that require continuous model adaptation and monitoring to maintain high performance. Financial institutions should invest in state-of-the-art machine learning models and regularly reassess their models to stay ahead of emerging fraud techniques and ensure robust protection for their customers.

## 5. Chapter 5: Conclusions and Future Work

In this study, we leveraged machine learning techniques to analyze and detect fraudulent transactions in bank account data. The dataset used contained a variety of customer and transaction-related features, including behavioral, demographic, and historical attributes. The dataset also presented a high degree of imbalance, with fraudulent transactions comprising a small fraction of the total data. This posed significant challenges, requiring careful handling through data preprocessing and modeling techniques.

To address the challenge of highly imbalanced data and the dynamic behavior of fraudsters, we initially conducted exploratory data analysis and feature engineering. These techniques, guided by exploratory analysis, aim to generate new features during the data engineering phase. We then used machine learning models to detect fraudulent transactions such as Random Forest: a bagging method, Logistic Regression: a linear classification algorithm, KNN: a non-parametric, instance-based algorithm, Naive Bayes (Gaussian): a probabilistic classification algorithm. The methods used show that the Random Forest Model performs quite well and gives satisfactory results. . Furthermore, a cutoff of 0.5 was found to be effective in detecting fraudulent transactions, indicating the potential for practical application of this model, especially in the banking sector. There are several key recommendations that can be made to further improve the fraud detection performance in credit card transactions. First, it is important to continue to explore different resampling techniques. These methods, which can have a beneficial impact on various scenarios and datasets, can lead to improvements in model performance at all levels. Second, the need to continuously explore and experiment with new features generated from datasets is underscored by the important role of feature engineering in improving model performance. This process should be guided by insights gained from exploratory data analysis and domain knowledge of fraudulent activities. Finally, there is a need to continuously monitor and tune detection models given the dynamics of fraudulent behaviors. To ensure the effectiveness of these models in detecting new fraud techniques, mechanisms should be introduced to regularly update them with new data and tune their hyperparameters. For future work, in-

depth research on anomaly detection techniques offers promising directions beyond traditional supervised learning approaches. Unsupervised learning approaches, such as autoencoders or segmentation algorithms, offer viable options for detecting anomalous patterns in credit card transactions without the need for labeled data. For example, by leveraging autoencoders, models can learn to reproduce normal transaction patterns and identify deviations that may indicate fraudulent activity. Similarly, segmentation algorithms can group transactions based on their inherent similarities, allowing the detection of outliers that deviate significantly from the norm. The use of these unsupervised techniques has the potential to enhance the ability of fraud detection systems to identify previously unseen fraudulent behavior or emerging trends. Furthermore, it reduces the reliance on labeled data, thus reducing the burden of manual annotation and potentially extending the applicability of the model to new fraud scenarios. Exploring anomaly detection methods in this way opens up exciting possibilities for improving the robustness and efficiency of bank account fraud detection systems, ultimately strengthening security measures in the financial sector.

## 6. References

1. Bank Account Fraud Dataset Suite (**NeurIPS 2022**). [online] Available at: <https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022>
2. Daniel Jurafsky & James H. Martin (2024). **Chapter 5 Logistic Regression**. Available at: <https://web.stanford.edu/~jurafsky/slp3/5.pdf>
3. ABDULLAHI UBALE USMAN, SUNUSI BALA ABDULLAHI... (2024) **Financial Fraud Detection Using Value-at-Risk With Machine Learning in Skewed Data**, (Senior Member, IEEE) Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10507824>
4. An Ensemble-based Fraud Detection Model for Financial Transaction Cyber Threat Classification and Countermeasures Available at: <https://drive.google.com/file/d/1CtlhWFXQ3goIpP4lHGAFY-Atx0VgLXsZ/view>
5. Decoupling Decision-Making In Fraud Prevention Through Classifier Calibration For Business Logic Action Available at: <https://arxiv.org/pdf/2401.05240>
6. geeksforgeeks Available at: <https://www.geeksforgeeks.org/>
7. Machine Learning - Vu Huu Tiep Available at: <https://drive.google.com/file/d/1Dxc2jhiIjPcJUUEudq6otlLCENXT7NBi/view?usp=sharing>
8. The Association of Certified Fraud Examiners (ACFE). (2012). Report to the Nation on Occupational Fraud and Abuse. Austin, TX, ACFE. Available at: <https://www.acfe.com/-/media/files/acfe/pdfs/rtnn/2012/2012-report-to-nations.pdf>
9. MATHEMATICS FOR MACHINE LEARNING - Marc Peter Deisenroth,A. Aldo Faisal, Cheng Soon Ong. Available at: [https://drive.google.com/file/d/1USld2sCVRdimzAaeCqMEk8-sh7k-Zr\\_7/view?usp=sharing](https://drive.google.com/file/d/1USld2sCVRdimzAaeCqMEk8-sh7k-Zr_7/view?usp=sharing)