



# Comparing Machine Learning Algorithms on Credit Card Fraud Problem

*Fatih Can*

*Dissertation submitted in partial fulfilment of the requirements for the degree of  
M.Sc. in Artificial Intelligence at Dublin Business School*

*August 2023*

*Supervisor: Dr. Vivek Kshirsagar*

# Declaration

I declare that this Applied Research Project that I have submitted to Dublin Business School for the award of Master of Science in Artificial Intelligence is the result of my investigations, except where otherwise stated, where it is acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: **Fatih Can**

Student Number: **10600616**

Date: **14/08/2023**

# Acknowledgements

I would first like to mention that this project could not have been possible without the guidance and support from my supervisor, Dr Vivek Kshirsagar I will be forever thankful for his help. Dr Vivek Kshirsagar was always open to help me whenever I was in trouble or had questions about my research or writing. He consistently ensured that my master's research paper was unique and my own work. He also corrected me at times when he felt any problems or issues in my research artefact or writing. He was always open for any type of discussions regarding the research. Every meeting with him was a great learning experience. He was always reachable by e-mail.

In addition, my family deserves endless gratitude: my wife who supported me in anyway possible during my education and dissertation project. Thanks to her I can fully focused on my dissertation project and be able to finish on time with ease.

# Abstract

The challenge of efficient fraud detection remains a top priority in the financial sector, especially given the potential consequences for both consumers and institutions. This paper investigates the use of machine learning models to detect fraudulent activities, utilizing a dataset containing transaction records. The study used a rigorous analysis methodology to compare a variety of algorithms, including Naive Bayes, Random Forest, Xgboost, and their respective Kfold variations. With an execution time of just over 4 minutes, Xgboost consistently emerged as a top performer, particularly in precision, recall, and F1 score metrics. The Kfold variant of Random Forest took the longest, while Naive Bayes was the quickest. When the results were weighted based on Recall and ROC AUC Score, Xgboost consolidated its position as the most capable model for detecting the vast majority of fraudulent activities. The paper also included detailed visual insights in the form of figures, which provided comparative performance metrics for the models. According to the findings, financial institutions should consider deploying the Xgboost algorithm as a critical component in their fraud detection systems, while also balancing considerations of execution time and the business ramifications of false positives and negatives.

**Keywords:** Machine Learning, Fraud Analytic, Fraud Detection, Xgboost, Naive Bayes, Random Forest, K-Fold Cross-Validation, Execution Time, ROC AUC Score, Precision, Recall, F1 Score, Time-Based Split, Model Performance, Machine Learning, Financial Transactions

## Table of Contents

List of Figures .....	6
List of Tables.....	7
1. Introduction.....	8
1.1. Road Map .....	10
1.2. Author’s Purpose.....	10
1.3. Research Question.....	11
1.4. Aim & Objective .....	11
2. Literature Review .....	12
2.1. Related Work.....	12
3. Methodology.....	23
3.1. Environment Evaluation .....	23
3.2. Research Architecture and Design .....	23
3.3. Data Collection and Pre-processing .....	24
3.3.1. Data Collection .....	24
3.3.2. Pre-Processing.....	24
3.4. Machine Learning Technique Review .....	26
3.4.1. Logistic Regression .....	26
3.4.2. Random Forest .....	27
3.4.3. XGB Boost Gradient Boosting.....	28
3.4.4. SGDClassifier.....	28
3.4.5. Decision Tree .....	29
3.4.6. Naive Bayes .....	30
3.4.7. AdaBoost .....	30
3.4.8. Summary of Machine Learning Techniques .....	31
3.5. Variables Review.....	31
3.5.1. Variable Description and Data Types.....	31
3.5.2. Feature Importance.....	34
3.6. Model Training.....	34
3.6.1. Logistic Regression .....	35
3.6.2. Random Forest .....	37
3.6.3. XGBoost Gradient Boosting.....	38
3.6.4. SGD Classifier.....	39
3.6.5. Decision Tree .....	40
3.6.6. Naive Bayes .....	41

3.6.7.	AdaBoost .....	41
3.7.	Model Evaluation and Comparison .....	42
3.8.	Environment Setup .....	44
3.9.	Methodology Summary .....	45
4.	Results, Discussion and Future Work .....	48
4.1.	Performance of Machine Learning Algorithms .....	49
4.1.1.	Key Results .....	49
4.1.2.	Execution Time .....	50
4.1.3.	Results for Time Based Split .....	51
4.1.4.	Results for Kfold Split .....	53
4.2.	Discussion .....	55
4.3.	Consideration .....	55
5.	Conclusion and Future Work .....	56
5.1.	Conclusions .....	56
5.2.	Future Work .....	56
6.	References .....	58

## List of Figures

Figure 1: Research Architecture ..... 11

Figure 2: Execution Time of Models ..... 11

Figure 3: Combined Comparison of All Models and Metrics

Figure 4: Separate Comparison of All Models and Metrics

Figure 5: Combined Comparison of All Models and Metrics(Kfold)

Figure 6: Separate Comparison of All Models and Metrics(Kfold)

## List of Tables

Table 1: Variable Types and Descriptions

Table 2: Logistic Regression Features

Table 3: Logistic Regression HyperParameters

Table 4: Random Forest Features

Table 5: Random Forest HyperParameters

Table 6: XGBoost Features

Table 7: XGBoost HyperParameters

Table 8: SGD Classifier Features

Table 9: SGD Classifier HyperParameters

Table 10: Decision Tree Features

Table 11: Decision Tree HyperParameters

Table 12: AdaBoost Features

Table 13: AdaBoost HyperParameters

## 1. Introduction

The digital era has brought with it a wide range of benefits as well as challenges, as evidenced by the growing popularity of online transactions. The security of online transactions, particularly card payments, is one of the fundamental issues in this area. Unfortunately, the convenience and pervasiveness of online payments make them a double-edged sword that fraudsters find to be extremely tempting. This dissertation explores the panorama of credit card fraud in recognition of the significance of this problem.

According to a 2015 issue of The Nilson Report, fraud losses associated with general purpose and private label credit, debit, and prepaid cards totaled a staggering \$16.31 billion on a global card volume of \$28.844 trillion. To put this in context, for every \$100 in volume, approximately \$5.65 was lost due to fraud (The Nilson Report, 2015). Such losses highlight not only the magnitude of the problem, but also the urgency and necessity of robust fraud detection systems. The rate of fraud growth, which increased by 19%, actually outpaced the rate of volume growth, which increased by 15%. Among the major contributors to this alarming figure were counterfeiting, card-not-present (CNP) scenarios, fraudulent applications, and lost or stolen cards.

One of the most shocking revelations was the United States' disproportionate share of fraud losses. Despite accounting for only 21.4% of global card volume, the United States bore 48.2% of gross card fraud losses, totaling \$7.86 billion. This disproportionately high figure, equivalent to 12.75 for every \$100 in total volume, was nearly quadruple the fraud rate of the other regions combined, which was 3.73 for every \$100. Such anomalies are largely due to the United States' lack of EMV-compliant infrastructure, a critical

security layer that protects against losses caused by counterfeit cards (The Nilson Report, 2015).

The European Central Bank (2023) conducted a thorough analysis that uncovered some important patterns in card fraud across Europe. Card fraud in Europe saw a discernible decline in 2021 despite the COVID-19 pandemic, which understandably had an impact on spending habits and transaction dynamics. Card fraud specifically decreased to its lowest level since 2008 in 2021, accounting for just 0.028% of the total value of payments made using SEPA-issued cards (European Central Bank, 2023). Another striking finding is the decline in card-not-present fraud as well as the significant decline in card-present fraud involving fake cards in 2020 and 2021. However, the size and significance of the problem at hand are highlighted by the fact that the value of card transactions in SEPA in 2021 was a staggering €5.40 trillion, with fraudulent transactions totalling €1.53 billion (European Central Bank, 2023).

With the aid of cutting-edge computational methods, this dissertation aims to investigate how machine learning can be used to spot card fraud. While conventional analysis has clarified the decline in card fraud and the significance of regulatory measures, machine learning presents a novel strategy to improve the mechanisms for detection and prevention. We aim to create robust models that can adapt to changing fraud techniques by training algorithms on patterns and anomalies within transaction data, particularly in the wake of the pandemic and its subsequent impact on spending patterns. The objective is to not only comprehend card fraud as it exists today but also to anticipate potential threats and arm the financial sector with preventative tools to deal with them.

## 1.1. Road Map

The strategic plan for the research project is the following:

- Introduction: contains sections on background information, machine learning techniques, the author's motivation, the research question, and the aim and objective.
- Literature review: a comparison of earlier research on the subject of this dissertation.
- Methodology: Describe the techniques employed to assess and choose the most suitable Machine Learning technique, the variables, and the features used to construct the Fraud Detection Model.
- Results: analysis of results and investigation of the parameters.
- Discussion: an evaluation of the results, the project's conclusion, and some details on the project's limitations and scope. Give proposals for additional work as well.

## 1.2. Author's Purpose

Credit cards are now used more frequently as the primary method of payment in the digital age. Convenience and quick transaction times have resulted from this, but it has also made it easier for bad actors to exploit flaws and commit fraud. The goal of this research is to thoroughly examine the issue of credit card fraud, comprehend its complexities, and apply cutting-edge machine learning techniques to identify and potentially stop these frauds. By doing this, the author hopes to help create a more secure environment for digital transactions and reduce financial losses for both individuals and institutions.

### 1.3. Research Question

What are the best algorithms or methods that maximize detection rates while minimizing false positives, and to what extent can machine learning models accurately detect credit card fraud given the historical transaction data?

### 1.4. Aim & Objective

This research suggests using machine learning techniques to show the extent to which fraud detection systems work with data from credit card transactions. Understanding how these algorithms can identify patterns indicative of fraudulent behaviour is the main goal. The objective is to demonstrate the algorithm's ability to accurately detect suspicious activities, which means minimizing false positives while efficiently catching real frauds, rather than just identifying every possible fraudulent transaction.

The following are the goals:

- Examine Machine Learning methods for detecting credit card fraud.
- Select the best technique to use for this research.
- Perform tests to show how effectively the system can identify fraudulent transactions.
- Validate and review the developed model's accuracy.

## 2. Literature Review

This chapter provides a summary of current knowledge in Machine Learning techniques applied to Fraud Data, allowing readers to identify relevant approaches, hypotheses, and problems in previous studies. It focuses on prior research that the author has identified as answering the project question. The studies are examined at the end of this section, outlining the benefits and drawbacks of their methodologies.

### 2.1. Related Work

The importance of data security in businesses has grown in recent years, particularly with the rise in credit card fraud, as evidenced by a report of €1.8 billion in fraudulent transactions in 2016 (Varmedja et al., 2019). This study was built on Kaggle's "Credit Card Fraud Detection" dataset, which includes transactions from September 2013. To counterbalance the data, the researchers used preprocessing techniques such as feature selection tools and the Synthetic Minority Oversampling Technique (SMOTE). To determine the efficacy of various machine learning algorithms in detecting fraudulent transactions, extensive testing was performed on Logistic Regression, Naive Bayes, Random Forest, and Multilayer Perceptron. Notably, while most algorithms demonstrated high accuracy rates, the Random Forest model was the most promising, achieving an astounding 99.96% accuracy. In conclusion, the research suggests that, depending on the circumstances, conventional algorithms may be as effective as deep learning models in detecting fraud (Varmedja et al., 2019).

Credit card fraud detection is an urgent concern in today's digital age, particularly with the increase in online transactions and e-commerce activities (Adepoju, Wosowei, & Jaiman, 2019). The effectiveness of detecting such fraudulent activities is heavily

dependent on the dataset used, the variables chosen, and the detection techniques used. Adepoju et al. (2019) used a Kaggle dataset with 3,075 transaction entries to rigorously evaluate four machine learning techniques: Support Vector Machine, Naive Bayes, K-Nearest Neighbor, and Logistic Regression. Although each algorithm performed admirably, the Logistic Regression classifier outperformed its peers with an impressive accuracy rate of 99.07%. Notably, the findings of this study indicate that Logistic Regression is particularly adept at detecting fraudulent credit card transactions, providing invaluable insights for credit card companies looking to improve their fraud prevention mechanisms.

In a similar vein, Awoyemi et al. (2017) investigated the use of three machine learning classifiers for credit card fraud detection: Naive Bayes, k-Nearest Neighbor, and Logistic Regression. Using a dataset of European cardholder transactions from September 2013, the researchers addressed the issue of data imbalance with a hybrid approach that included both over-sampling and under-sampling techniques. Accuracy, sensitivity, specificity, precision, Matthews correlation coefficient, and balanced classification rate were among the evaluation metrics. The results suggested that the k-Nearest Neighbour classifier performed well across most metrics, particularly with the 10:90 data distribution. The study sheds light on the effectiveness of these techniques in detecting fraud and paves the way for future research into meta-classifiers and meta-learning techniques.

Furthermore, Yee, Sagadevan, and Malim (2018) investigate credit card fraud detection using machine learning and data mining techniques. Their methodology incorporates the widely used WEKA tool for classifier performance measurement as well as the 10-

fold cross validation technique. The researchers then evaluate classifier performance using a variety of metrics, including True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F-Measure, and accuracy. The team used two datasets for their experiments: a raw dataset and one that had undergone data transformation and reduction. The TAN classifier outperformed other classifiers in several metrics in Experiment 1, which used the raw dataset. However, it was discovered that raw data with a high level of noise had a negative impact on model evaluations. All classifiers performed better in Experiment 2, which used the transformed dataset, with many achieving accuracies greater than 95.0%. This study emphasizes the importance of data preprocessing in improving classification results, as well as the effectiveness of Principal Component Analysis in refining datasets for credit card fraud detection. Furthermore, when the data was subjected to rigorous preprocessing, the Bayesian classifiers—K2, Nave Bayesian, TAN, Logistics, and J48—were shown to have superior predictive capabilities for credit card fraud. Finally, Yee et al. demonstrated that when given filtered datasets, Bayesian classifiers perform significantly better for credit card fraud detection, emphasizing the importance of data preprocessing and transformation.

Similarly, Lakshmi and Kavilla (2018) evaluated various machine learning algorithms for their effectiveness in detecting credit card fraud. Their research, which was published in the International Journal of Applied Engineering Research, compared algorithms such as Logistic Regression, Decision Trees, and Random Forest. Reading the dataset, random sampling, and feature selection were all part of the methodology before evaluating performance with metrics derived from a confusion matrix, such as Accuracy and Sensitivity. The Random Forest classifier performed the best, achieving an accuracy of 95.5% in fraud detection, outperforming both Logistic Regression (90.0%) and Decision

Tree (94.3%). Such discoveries highlight the importance of algorithm selection in improving the accuracy of fraud detection systems (Lakshmi, S.V.S.S., and Kavilla, S.D., 2018).

Khatri et al. (2020) evaluated various supervised machine learning algorithms in the field of credit card fraud detection to determine the most efficient and accurate method for distinguishing between genuine and fraudulent transactions. They used an unbalanced dataset of 284,807 European cardholder transactions, only 492 of which were fraudulent, to assess the sensitivity, precision, and processing time of models such as kNN, Naive Bayes, Decision Tree, Logistic Regression, and Random Forest. The study discovered that, while kNN had superior sensitivity, the Decision Tree model was more suitable for fraud detection due to its rapid prediction capability due to its longer data testing time. Their findings highlight the importance of selecting algorithms not only for accuracy but also for efficiency, especially in high-stakes industries like banking where timely fraud detection is critical. Thus, Khatri et al.'s work suggests a preference for Decision Tree models in credit card fraud detection, but also acknowledges the potential value of exploring unsupervised machine learning techniques as a comparative avenue in future research endeavors.

Maniraj et al. (2019), on the other hand, present a comprehensive approach to credit card fraud detection based on the most recent machine learning algorithms. The authors use datasets from Kaggle in their study, which is published in the International Journal of Engineering Research. The datasets consist of 31 columns, primarily featuring anonymized transaction details, with columns such as "Time", "Amount", and "Class" indicating the gap between transactions, the transacted amount, and the transaction

classification (valid or fraudulent). They use the Local Outlier Factor and Isolation Forest Algorithm from the sklearn package to detect fraudulent activities. Despite achieving an impressive accuracy of more than 99.6%, their approach had a precision of 28% when only a tenth of the dataset was considered. When the entire dataset was used, the precision increased to 33%. The precision disparity is attributed to the dataset's significant imbalance between valid and fraudulent transactions. The authors emphasize that the system's efficacy could improve further over time with more data input. However, real-world implementation challenges remain, owing in part to banking confidentiality and their unwillingness to share data. Still, the findings of the paper point in a promising direction for future innovations in fraud detection, particularly because the incorporation of additional algorithms and larger datasets has the potential to improve the system's accuracy (Maniraj, S.P., Saini, A., Ahmed, S., and Sarkar, S., 2019).

Mittal and Tyagi (2019) investigate the applicability and efficacy of supervised and unsupervised machine learning techniques in their seminal study on credit card fraud detection. Their research emphasizes the adaptability of supervised learning algorithms, with a focus on Random Forest for its robustness against noise and outliers, Neural Networks for pattern recognition, Deep Learning for multi-layer perceptron networks, and the Support Vector Machine for linear classification. The research also looks at hybrid supervised methods such as Extended Gradient Boosted Tree and Quadratic Discriminant Analysis. Their work on unsupervised learning emphasizes the importance of techniques capable of clustering data and highlighting anomalies. Self-Organizing Maps, which configure neurons to mirror input data topologies, and the K-means clustering method are particularly effective. The study also delves into more novel

methods like the Isolation Forest, a regressor that isolates anomalies, and the Local Outlier Factor, which estimates density using nearest neighbors. Mittal and Tyagi's research sheds light on the potential of machine learning in improving fraud detection amidst the complexities of voluminous genuine transactions, pointing to an optimistic future for fraud prevention efforts.

Tanouz et al.'s 2021 study delves into the ever-increasing need for advanced fraud detection systems in light of the growing prominence of credit card payments.

Recognizing the inherent difficulty of heavily imbalanced datasets, where fraudulent transactions dwarf genuine ones, they propose a set of machine learning classification algorithms, including logistic regression, random forest, and Naive Bayes. Their strategy is based on a meticulous preprocessing regime in which they address outliers and correlate various features for optimal data health. Surprisingly, despite the Decision Tree classifier's impressive accuracy score of 99.9%, the authors highlight the dangers of overreliance on raw accuracy, pointing to the significant number of False Positives and Negatives as potential sources of significant fiscal impact. According to the study's findings, the Random Forest classifier is the best performer, with an accuracy of 96.7741%, a precision rate of 100%, and a recall rate of 91.1111%. Nonetheless, the paper ends on a cautiously optimistic note, recognizing the consistent performance of the tested algorithms while emphasizing the potential benefits of additional real-world training data and algorithmic fusion for improved accuracy. The authors also hint at future endeavors involving genetic algorithms, emphasizing their commitment to pushing the boundaries of fraud detection even further.

Minastireanu and Mesnita's 2019 research investigates the pressing issue of click fraud, a scourge in online advertising that jeopardizes the integrity of e-business and marketing initiatives. The authors emphasize the critical need for powerful detection mechanisms by drawing attention to online advertisers' tireless efforts to combat these fraudulent clicks. In their exploration of machine learning algorithms, they highlight LightGBM, a Gradient Boosting Decision Tree-type method, to sift through a massive dataset containing 200 million clicks spread over four days. Their methodology heavily relies on the experimental validation of LightGBM, with the primary goal of identifying suspicious IP addresses that log multiple clicks without resulting app installations. Their results are astounding; the algorithm achieves an astounding 98% accuracy rate. In order to delve deeper into their methodology, the researchers use a Kaggle public dataset and meticulously engineer 26 features from the original set, 19 of which are cherry-picked for the main model. But Minastireanu and Mesnita's research is more than just numbers. It's a nuanced examination of the complexities of machine learning. They use k-fold cross-validation, specifically with  $k = 5$ , to assess the precision of the algorithm, emphasizing the importance of strategic feature engineering and validation. They also meticulously tune the LightGBM parameters, taking into account potential pitfalls such as overfitting. Their evaluation tools are equally thorough, revolving around feature importance and the ROC curve to determine the algorithm's efficacy. When comparing LightGBM to other machine learning algorithms, the authors highlight its categorical variable handling, gradient-based one-side sampling (GBOSS), and histogram-based methodologies. Furthermore, when compared to other gradient boosting algorithms such as XGBoost and Stochastic Gradient Boosting, LightGBM's superior speed and memory efficiency are highlighted, though the authors acknowledge

the limitations posed by resource constraints in their experiment. Minastireanu and Mesnita conclude their research by advocating for continued innovation in the field of click fraud detection. They emphasize the evolving nature of fraudulent strategies, implying that, while their current methods have made significant strides, the battle against click fraud is far from over, and urging future research to adapt to and counter more complex fraudulent behaviors.

Raghavan and El Gayar (2019) conducted a comprehensive evaluation of machine learning and deep learning models for fraud detection across three distinct datasets, including the European, Australian, and German datasets, as the research landscape evolves. The size of these datasets varied, with the European dataset being significantly larger and more imbalanced, encompassing 284,807 transactions over two days in September 2013, only 492 of which were fraudulent. Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Random Forest, Convolutional Neural Networks (CNN), Autoencoders, Restricted Boltzmann Machine (RBM), and Deep Belief Networks (DBN) were among the models tested. The study discovered that SVM consistently outperformed all other methods across all datasets. Furthermore, CNNs outperformed other deep learning techniques, especially on the larger European dataset. Ensemble methods, which combined SVM, Random Forest, and KNN, performed better on smaller datasets. While supervised models such as CNN, KNN, and Random Forest produced compelling results, the dynamic nature of fraud necessitates regular retraining of these models to maintain efficacy. Autoencoders trained solely on normal transactions presented a promising alternative by detecting fraud as deviations from established patterns, proving useful for data labeling and subsequently supporting supervised model retraining.

Trivedi et al. (2020) presented a powerful credit card fraud detection model based on machine learning methodologies and accompanied by a feedback system in their paper. When applied to a dataset of 284,807 transactions from European account holders, this mechanism improves the detection rate and cost-effectiveness of classifiers. The researchers compared performance metrics such as precision, recall, F1-score, accuracy, and False Positive Rate (FPR) to various classifiers such as Random Forest, Decision Trees, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive Bayes, Logistic Regression, and Gradient Boosting. Among these, supervised learning techniques such as ANN and SVM demonstrated notable effectiveness in discerning patterns and categorizing data, whereas unsupervised techniques such as K-means and the Hidden Markov Model investigated inherent data structures. Credit card fraud detection is fraught with difficulties, such as massive daily data inflows, highly imbalanced data, and evolving fraudster strategies. Trivedi et al.'s research highlighted the Random Forest technique's superior performance, with an accuracy of 95.988%, indicating its potential as a primary tool in combating credit card fraud, though the application of machine learning remains complex and requires considerable expertise.

Further exploring this area, Thennakoon et al. (2019) presented a comprehensive methodology for detecting real-time credit card fraud. The authors used a combined dataset derived from two data sources: a fraud transactions log file that included all online credit card fraud occurrences, and a general transactions log file that included all bank transactions in a specific timeframe. The data, which was initially skewed due to the distinction between legitimate and fraudulent transactions, was subjected to extensive preparation, including cleaning, transformation, integration, and reduction. The imbalanced nature of the dataset was addressed using resampling techniques,

specifically Synthetic Minority Oversampling Techniques (SMOTE) and under-sampling methods such as condensed nearest neighbor (CNN) and random under-sampling (RUS). To analyze various fraud patterns, four different machine learning algorithms were used: Support Vector Machine, Naive Bayes, K-Nearest Neighbor, and Logistic Regression. The proposed real-time detection system includes three major components: an API Module, fraud detection models, and a data warehouse. This methodical approach ensures that fraudulent transactions are detected and communicated as soon as possible, increasing the efficiency of fraud monitoring teams. Finally, the study emphasizes the importance of resampling techniques in optimizing classifier performance and identifies machine learning models with the highest accuracy rates for specific fraud patterns. The system's future prospects are also highlighted, particularly its potential application in detecting location-based fraud.

Finally, Sailusha et al. (2020) investigated credit card fraud detection using machine learning techniques, focusing primarily on the Random Forest and Adaboost algorithms. The researchers used a Kaggle dataset that contained transaction data from a European credit card company from September 2013. This dataset included 284,807 transactions, with only 0.172% of them being fraudulent. The goal of their research was to use the algorithms to classify both fraudulent and non-fraudulent transactions and then compare their effectiveness. From data splitting and model training to model deployment and evaluation, a comprehensive procedure was outlined. The evaluation criteria were based on metrics derived from a confusion matrix, such as accuracy, precision, recall, and the F1-score. While both algorithms performed similarly in terms of accuracy, the Random Forest algorithm outperformed Adaboost in terms of precision, recall, and F1-score. Despite these findings, the authors point out that there

is no single algorithm that is 100% effective at detecting credit card fraud. They suggest that future research could look into the potential of deep learning techniques to improve fraud detection accuracy.

## 3. Methodology

### 3.1. Environment Evaluation

The study was conducted using a personal computer with the following specifications:

- Operating System: Windows 11 Home 64-bit (10.0, Build 22621) (22621.ni\_release.220506-1250), with regional language settings in English.
- System Manufacturer: ASUSTeK COMPUTER INC.
- System Model: ASUS TUF Dash F15 FX517ZE\_FX517ZE
- BIOS Version: FX517ZE.315 (type: UEFI)
- Processor: 12th Gen Intel(R) Core(TM) i5-12450H (12 CPUs), operating at ~2.0GHz.
- Memory: A total of 16GB (16384MB) RAM was installed with 16006MB available for OS tasks.
- Page File Memory: A total of 10116MB was used, with 14545MB available.
- DirectX Version: DirectX 12

### 3.2. Research Architecture and Design

In order to find best model for Fraud Detection we followed these steps:

1. Data Generation
2. Preprocessing
3. First implementation of models
4. Getting proper Hyperparameters and Important features for each model
5. Second implementation of models
6. Getting final results
7. Comparing models and suggesting best model

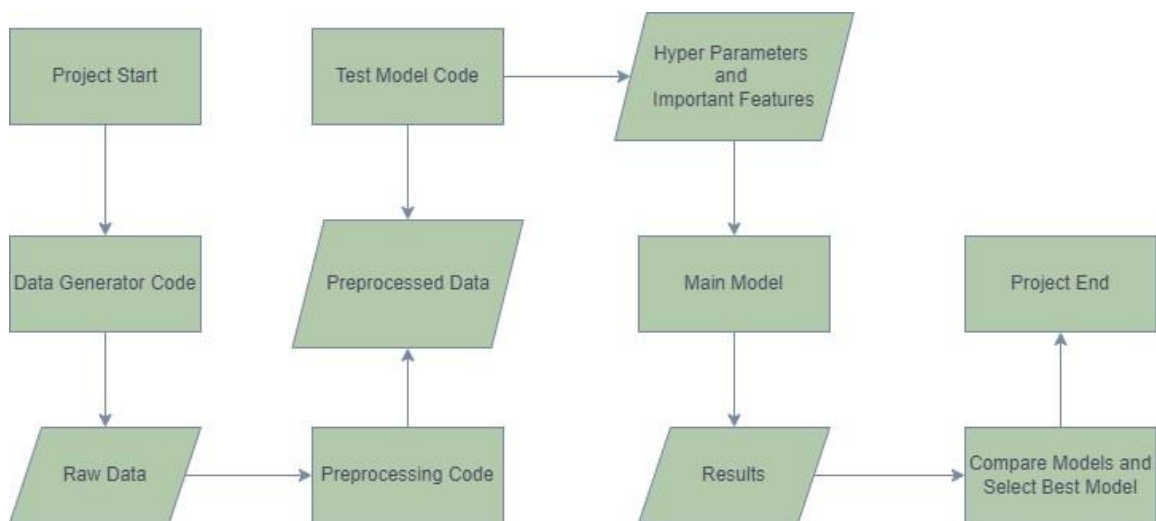


Figure 1: Research Architecture

### 3.3. Data Collection and Pre-processing

#### 3.3.1. Data Collection

The dataset for this research was generated using the data generation codes provided by *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook* (Le Borgne et al., 2022). Original code and dataset can be found on Kaggle website. This Dataset is heavily used for Fraud Analytic researches. I modified the original code to suit the specific needs of this study. The generated data encompasses transactions from January 1, 2022, to July 2, 2023.

#### 3.3.2. Pre-Processing

After data generation, the dataset went through several preprocessing steps to improve its suitability for modeling:

##### 1. Feature Creation:

- TX\_DURING\_WEEKEND was created to determine if a transaction occurred during the weekend.

- TX\_DURING\_NIGHT was established to check if a transaction took place during nighttime.
- Several other features were extracted related to customer behavior, terminal risk, and time-based features. These include 'CUSTOMER\_ID\_NB\_TX\_1DAY\_WINDOW', 'CUSTOMER\_ID\_AVG\_AMOUNT\_1DAY\_WINDOW', and 'TERMINAL\_ID\_NB\_TX\_7DAY\_WINDOW' among others.

## **2. Temporal Feature Extraction:**

The data also went through extraction of temporal features like year, month, day, hour, minute, second, and day of the week.

## **3. Dropping Unnecessary Columns:**

The column TX\_FRAUD\_SCENARIO was deemed unnecessary for the modeling process and hence was removed from the dataset. Another reason is it leads to overfitting every time we use that column.

## **4. Data Scaling:**

The StandardScaler from sklearn.preprocessing was applied to standardize features, ensuring they possess a mean of 0 and a standard deviation of 1.

## **5. Data Splitting: The dataset was split based on the transaction date:**

- Training Data: Transactions between 2022-01-01 to 2023-05-15.
- Test Data: Transactions between 2023-05-15 to 2023-07-02.

A k-fold cross-validation strategy with k=5 was used for model evaluation during the training phase in addition to this time based split.

In short, the preprocessing enhanced the original columns ('TRANSACTION\_ID', 'TX\_DATETIME', 'CUSTOMER\_ID', 'TERMINAL\_ID', 'TX\_AMOUNT', 'TX\_TIME\_SECONDS', 'TX\_TIME\_DAYS', 'TX\_FRAUD') by adding features that could potentially provide more insights into transaction patterns and behavior.

### 3.4. Machine Learning Technique Review

Machine Learning techniques provide a set of algorithms that can be applied to a wide range of problems, from simple linear regression to complex high-dimensional classification tasks. The application of these techniques in the domain of credit card fraud detection is useful in distinguishing between legitimate and fraudulent transactions based on patterns and anomalies in the data. This section provides an overview of selected machine learning techniques, including theoretical formulations and their application to credit card fraud detection.

#### 3.4.1. Logistic Regression

Logistic regression is a statistical method for analyzing datasets with binary outcomes. Given a set of input features  $X$ , the probability  $P(Y=1)$  of a transaction being fraudulent is modeled as:

$$P(Y=1) = 1 / (1 + e^{-z}) \text{ Where } z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n.$$

Logistic regression generates a probability that the given input point belongs to a specific class, which is then converted into a binary outcome using a threshold (e.g., 0.5).

**Pros:**

- Clarity: Provides a clear probabilistic interpretation.
- Effective: Requires fewer computational resources.
- A good starting point: Provides a quick first approach to any binary classification problem.

**Cons:**

- Linearity: Assumes a linear decision boundary, which may miss complex data patterns.
- Features: Extensive feature engineering and preprocessing are required.

### 3.4.2. Random Forest

Random Forest is a method of ensemble learning that combines multiple decision trees to produce a more accurate and stable prediction. It introduces randomness into the model by building each tree with a subset of the data and a subset of the features. The final prediction is either an average (in the case of regression) or a majority vote (in the case of classification).

**Pros:**

- Nonlinear: Capable of modeling complex decision boundaries.
- Robustness: Averaging makes it less prone to overfitting.
- Feature importance: Indicates which features are most useful in making predictions.

**Cons:**

- Cost of computation: Can be computationally expensive, especially with large trees.

- Interpretability: This model is less interpretable than simpler models.

### 3.4.3. XGB Boost Gradient Boosting

Gradient Boosting is an iterative ensemble method, and XGBoost is a distributed gradient boosting library that has been optimized. XGBoost boosts the model's performance and speed. Using gradient descent, a new tree is added at each iteration to minimize the loss function.

#### **Pros:**

- Accuracy: Often outperforms other algorithms in terms of performance.
- Regularization: Includes L1 (Lasso) and L2 (Ridge) regularization.
- Versatility: Can be used with a variety of loss functions.

#### **Cons:**

- Computational Cost: Training on large datasets can be time-consuming.
- Hyperparameters: Hyperparameters must be carefully tuned.

### 3.4.4. SGDClassifier

Stochastic Gradient Descent (SGD) is an optimization technique used to update parameters in learning algorithms, particularly in large-scale problems. SGDClassifier is a linear classifier that has been SGD optimized. The loss function and penalties are both adjustable, making it suitable for a wide range of problems.

**Pros:**

- Scalability: Because of incremental learning, it is suitable for large-scale datasets.
- Faster convergence than batch gradient descent.
- Flexibility: Can be used with a variety of loss functions and penalties.

**Cons:**

- Sensitivity: Requires careful data preprocessing and scaling.
- Hyperparameters: For optimal performance, the learning rate and other parameters must be tuned.

### 3.4.5. Decision Tree

Based on the feature values, decision trees divided the dataset into subsets. This process is repeated recursively, resulting in a decision-tree-like model. At each node, a feature is selected to split the data based on a criterion such as entropy or Gini impurity, with the goal of maximizing information gain.

**Pros:**

- Interpretability: It is simple to visualize and comprehend.
- No data scaling is required, and it can handle both numerical and categorical data.
- Feature selection: Selects important features automatically.

**Cons:**

- Overfitting: Overfitting is common, especially with deep trees.
- Variability: Minor changes in data can produce different trees.

### 3.4.6. Naive Bayes

Naive Bayes classifiers are a family of probabilistic classifiers based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a set of features, the probability of an event  $A$  is proportional to:  $P(A|X)$  is proportional to  $P(X|A) * P(A)$  Where  $P(A|X)$  is the posterior probability,  $P(X|A)$  is the likelihood, and  $P(A)$  is the prior probability.

**Pros:**

- Effectiveness: Short training and prediction times.
- Simplicity: Based on strong (naive) assumptions, this can be advantageous in simple tasks.
- Appropriate for high-dimensional datasets.

**Cons:**

- Assumption: Assumes that all features are independent, which is not always true.
- Continuous data: Continuous data necessitates additional assumptions and preprocessing.

### 3.4.7. AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble learning method that adjusts an observation's weight based on the most recent classification. If an observation was incorrectly classified, it would be given more weight, ensuring that subsequent models prioritize difficult cases.

**Pros:**

- Accuracy is frequently improved by combining multiple weak classifiers.
- Less prone to overfitting: This is particularly true when the underlying classifier is simple.

- Versatility: Can be used with a variety of base classifiers.

**Cons:**

- Noise sensitivity: Sensitive to noisy data and outliers.
- Cost of computation: Multiple rounds of computation can be time-consuming.

### 3.4.8. Summary of Machine Learning Techniques

When it comes to detecting credit card fraud, each of the machine learning techniques mentioned above has its own set of advantages and considerations. The algorithm of choice is frequently determined by the nature of the data, the computational efficiency required, and the desired predictive accuracy. The sections that follow will go into greater detail about the implementation and performance comparison of these techniques in the context of the dataset at hand.

## 3.5. Variables Review

An extensive list of variables was used to develop a comprehensive fraud detection model. We present a systematic overview of these variables, their data types, and their importance as determined by various machine learning techniques.

### 3.5.1. Variable Description and Data Types

Table below shows that basic information about variables in dataframe.

Table 1: Variable Types and Descriptions

Variable	Data Type	Description
TRANSACTION_ID	int64	Unique identifier for each transaction.
TX_DATETIME	datetime64[ns]	Date and time when the transaction occurred.

<b>CUSTOMER_ID</b>	int64	Unique identifier for each customer.
<b>TERMINAL_ID</b>	int64	Unique identifier for each terminal.
<b>TX_AMOUNT</b>	float64	Amount involved in the transaction.
<b>TX_TIME_SECONDS</b>	int64	Duration since the first transaction, in seconds.
<b>TX_TIME_DAYS</b>	int64	Duration since the first transaction, in days.
<b>TX_FRAUD</b>	int64	Target variable; 1 indicates fraud, 0 indicates not fraud.
<b>TX_DURING_WEEKEND</b>	int64	Indicates if transaction occurred during the weekend.
<b>TX_DURING_NIGHT</b>	int64	Indicates if transaction occurred during the nighttime.
<b>CUSTOMER_ID_NB_TX_1DAY_WINDOW</b>	float64	Number of transactions by customer in the last day.
<b>CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW</b>	float64	Average transaction amount by customer in the last day.
<b>CUSTOMER_ID_NB_TX_7DAY_WINDOW</b>	float64	Number of transactions by customer in the last 7 days.
<b>CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW</b>	float64	Average transaction

		amount by customer in the last 7 days.
<b>CUSTOMER_ID_NB_TX_30DAY_WINDOW</b>	float64	Number of transactions by customer in the last 30 days.
<b>CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW</b>	float64	Average transaction amount by customer in the last 30 days.
<b>TERMINAL_ID_NB_TX_1DAY_WINDOW</b>	float64	Number of transactions at a terminal in the last day.
<b>TERMINAL_ID_RISK_1DAY_WINDOW</b>	float64	Risk score of a terminal based on the last day's transactions.
<b>TERMINAL_ID_NB_TX_7DAY_WINDOW</b>	float64	Number of transactions at a terminal in the last 7 days.
<b>TERMINAL_ID_RISK_7DAY_WINDOW</b>	float64	Risk score of a terminal based on last 7 days' transactions.
<b>TERMINAL_ID_NB_TX_30DAY_WINDOW</b>	float64	Number of transactions at a terminal in the last 30 days.
<b>TERMINAL_ID_RISK_30DAY_WINDOW</b>	float64	Risk score of a terminal based on last 30 days' transactions.
<b>YEAR</b>	int64	Year of the transaction.
<b>MONTH</b>	int64	Month of the transaction.
<b>DAY</b>	int64	Day of the transaction.

<b>HOUR</b>	int64	Hour of the transaction.
<b>MINUTE</b>	int64	Minute of the transaction.
<b>SECOND</b>	int64	Second of the transaction.
<b>DAY_OF_WEEK</b>	int64	Day of the week of the transaction.

### 3.5.2. Feature Importance

The significance of each feature was not constant across all machine learning techniques for model development, with the exception of the Naive Bayes approach, where feature importance is not directly applicable. The goal was to predict the TX\_FRAUD column, so feature importance was assessed to determine which variables have a significant impact on the prediction of this target variable.

Using feature importance analysis across different machine learning models, we can ensure that our models are not just using redundant information and are focusing on genuinely significant variables that contribute to accurate fraud prediction. The variation in feature importance across models emphasizes the distinctness of each machine learning technique and how they interpret the significance of each variable in the dataset.

### 3.6. Model Training

We used two different methodologies to optimize machine learning techniques for the dataset.

First, we used a time-based data splitting strategy, taking advantage of the TX\_DATETIME column. This method was chosen in recognition of the time-series nature of transactional data. Temporal dependencies in such data may be critical, so models must be trained in a chronologically coherent manner.

Second, as an alternative data sampling strategy, we used the k-fold cross-validation approach with k=5. This is a well-known method for evaluating model performance that divides the data into k equal-sized folds and then uses each fold as a test set in turn.

Furthermore, hyperparameter optimization was rigorously used to determine the most advantageous parameter values for the machine learning techniques used. The deliberate decision to avoid oversampling was an important aspect of the methodology. Oversampling methods consistently harmed model generalization in multiple experimental iterations, manifesting as overfitting.

With these specific hyperparameters and columns below, several algorithms were trained and evaluated:

### 3.6.1. Logistic Regression

- Important Features

Table 2: Logistic Regression Features

Feature
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW'

'TX_AMOUNT'
'TERMINAL_ID_RISK_7DAY_WINDOW'
'MONTH'
'TERMINAL_ID_RISK_1DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW'
'TERMINAL_ID_NB_TX_1DAY_WINDOW'
'TRANSACTION_ID'
'TX_TIME_DAYS'
'TX_TIME_SECONDS'
'CUSTOMER_ID_NB_TX_7DAY_WINDOW'
'TERMINAL_ID_NB_TX_7DAY_WINDOW'

- HyperParameters

Table 3: Logistic Regression HyperParameters

HyperParameter
class_weight='balanced'
C=0.02091873512449284
penalty='l2'
max_iter=1000
solver='lbfgs'

### 3.6.2. Random Forest

- Important Features

Table 4: Random Forest Features

Feature
"TERMINAL_ID_RISK_30DAY_WINDOW"
"TERMINAL_ID_RISK_7DAY_WINDOW"
"TX_AMOUNT"
"CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW"
"TERMINAL_ID_RISK_1DAY_WINDOW"
"CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW"
"CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW"
"TRANSACTION_ID"
"TX_TIME_SECONDS"
"TERMINAL_ID_NB_TX_30DAY_WINDOW"
"TERMINAL_ID_NB_TX_1DAY_WINDOW"
"TX_TIME_DAYS"
"TERMINAL_ID"

- HyperParameters

Table 5: Random Forest HyperParameters

HyperParameter
n_estimators= 133,
max_depth = 7,
min_samples_split= 4,

min_samples_leaf= 6,
class_weight='balanced'

### 3.6.3. XGBoost Gradient Boosting

- Important Features

Table 6: XGBoost Features

Feature
'TERMINAL_ID_RISK_7DAY_WINDOW'
'TX_AMOUNT'
'TERMINAL_ID_RISK_1DAY_WINDOW'
'TERMINAL_ID_RISK_30DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW'
'TRANSACTION_ID'
'TERMINAL_ID_NB_TX_7DAY_WINDOW'
'TERMINAL_ID'
'CUSTOMER_ID'

- HyperParameters

Table 7: XGBoost HyperParameters

HyperParameter
scale_pos_weight=sum(y==0) / sum(y==1)
n_estimators = 148, max_depth = 31
learning_rate = 0.1486879513405262
subsample = 0.5728474806496494

colsample\_bytree = 0.9213946638322462

### 3.6.4. SGD Classifier

- Important Features

Table 8: SGD Classifier Features

Feature
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW'
'TX_AMOUNT'
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW'
'TERMINAL_ID_RISK_7DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW'
'TERMINAL_ID_RISK_1DAY_WINDOW'
'TERMINAL_ID_NB_TX_1DAY_WINDOW'
'DAY'
'TERMINAL_ID_NB_TX_7DAY_WINDOW'
'CUSTOMER_ID_NB_TX_1DAY_WINDOW'
'SECOND'
'CUSTOMER_ID_NB_TX_30DAY_WINDOW'
'HOUR'
'TERMINAL_ID'

- HyperParameters

Table 9: SGD Classifier HyperParameters

Hyperparameter
alpha = 0.06807346849670307
penalty = 'l2'

loss = 'modified_huber'
class_weight='balanced'

### 3.6.5. Decision Tree

- Important Features

Table 10: Decision Tree Features

Feature
'TX_AMOUNT'
'TERMINAL_ID_RISK_7DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW'
'TERMINAL_ID_RISK_1DAY_WINDOW'
'TERMINAL_ID_RISK_30DAY_WINDOW'
'CUSTOMER_ID_NB_TX_7DAY_WINDOW'
'TERMINAL_ID'
'TRANSACTION_ID'
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW'
'TERMINAL_ID_NB_TX_7DAY_WINDOW'
'TERMINAL_ID_NB_TX_30DAY_WINDOW'
'CUSTOMER_ID_NB_TX_1DAY_WINDOW'
'TERMINAL_ID_NB_TX_1DAY_WINDOW'

- HyperParameters

Table 11: Decision Tree HyperParameters

HyperParameter
class_weight='balanced'

max_depth = 8
min_samples_split = 8
min_samples_leaf = 1
criterion = 'entropy'

### 3.6.6. Naive Bayes

- Important Features

Feature Engineering is not possible on Naïve Bayes because this technique assumption is all variable are independent.

- HyperParameters

var\_smoothing = 4.749613610827011e-10

### 3.6.7. AdaBoost

- Important Features

Table 12: AdaBoost Features

Feature
'TX_AMOUNT'
'CUSTOMER_ID_AVG_AMOUNT_30DAY_WINDOW'
'TERMINAL_ID_RISK_7DAY_WINDOW'
'TERMINAL_ID_RISK_30DAY_WINDOW'
'CUSTOMER_ID_AVG_AMOUNT_7DAY_WINDOW'
'TERMINAL_ID_RISK_1DAY_WINDOW'
'TX_TIME_SECONDS'
'CUSTOMER_ID_AVG_AMOUNT_1DAY_WINDOW'
'TRANSACTION_ID'

- HyperParameters

Table 13: AdaBoost HyperParameters

HyperParameter
estimator= DecisionTreeClassifier(max_depth=1)
n_estimators = 107
learning_rate = 0.1135846389723132

### 3.7. Model Evaluation and Comparison

The evaluation and comparison of trained models is a critical stage in the machine learning pipeline, ensuring that they not only fit the data well but also generalize effectively to unseen samples. The primary goal of the task at hand was to maximize the ROC AUC score.

In the context of fraud detection, the ROC AUC (Receiver Operating Characteristic - Area Under the Curve) score is especially important. This metric reflects the model's ability to distinguish between legitimate and fraudulent transactions. Across all classification thresholds, it quantifies the trade-off between the True Positive Rate (sensitivity) and the False Positive Rate (1-specificity). A model with perfect discrimination capability has a ROC AUC score of 1, whereas a random classifier has a score of 0.5.

Given the inherent class imbalance in fraud detection scenarios, where fraudulent transactions are typically much rarer than genuine ones, a high ROC AUC score indicates a model's ability to detect rare fraudulent cases among a sea of genuine transactions. In this context, such a score is more informative than mere accuracy, because high accuracy can be obtained by simply classifying most transactions as genuine.

A comprehensive suite of evaluation metrics was used to conduct a holistic assessment of each machine learning technique implemented:

**Accuracy:** This is a simple ratio of correctly predicted observations to total observations. While accuracy is important, it can be misleading in imbalanced datasets, making complementary metrics essential.

**Precision** is defined as the proportion of correctly predicted positive observations to total predicted positives. High precision is associated with a low false positive rate, which is critical in avoiding the misclassification of legitimate transactions as fraudulent.

**Recall** is the ratio of correctly predicted positive observations to all actual positives. It is also known as sensitivity or the true positive rate. A high recall ensures that the majority of fraud cases are detected, minimizing financial loss.

The **F1-score** metric strikes a balance between precision and recall. It's especially useful when the class distribution is uneven, because it takes into account both false positives and false negatives.

**ROC AUC Score:** As previously stated, this metric highlights the model's ability to distinguish between classes, which is especially important in fraud detection contexts.

In summary, the study aspires to deliver robust and dependable machine learning solutions for fraud detection through a meticulous process of model evaluation and comparison leveraging a plethora of metrics.

### 3.8. Environment Setup

To ensure replicability and consistency in the experiments and analyses carried out throughout this study, it is necessary to outline the computational environment used. The following summarizes the technical setup:

- **Programming Language:** Python, specifically version 3.9.13, was used as the primary implementation language. This version, tagged v3.9.13 and released on May 17, 2022, provides a strong and adaptable foundation, ensuring compatibility with a wide range of modern data analysis libraries.
- **Numerical Computation:** At version 1.23.5, NumPy served as the foundation for numerical operations. Its ability to handle multi-dimensional arrays and matrices, as well as a comprehensive set of mathematical functions, makes it indispensable for machine learning tasks.
- **Data Manipulation:** Pandas, version 1.5.3, was used to manage and analyze data structures efficiently. It greatly simplifies tasks like data cleaning, transformation, and aggregation due to its intuitive syntax and comprehensive functionalities.
- **Visualization:** Matplotlib 3.7.1 enabled the creation of a plethora of plots and visualizations. These graphics are critical for comprehending underlying data patterns, model behavior, and result interpretation.
- **Machine Learning Library:** Scikit-learn (sklearn) version 1.2.1 was the primary library used in this study for most machine learning algorithms and tools. Its comprehensive set of data mining and data analysis tools is ideal for a wide range of tasks, from pre-processing to model evaluation.

- **Gradient Boosting Framework:** For its gradient boosting capabilities, XGBoost, version 1.7.6, was chosen. XGBoost, which is known for its performance and efficiency, provides an advanced implementation of gradient boosting algorithms, which is essential for tasks requiring optimized results.
- **Hyperparameter Optimization:** Optuna 3.3.0 was used for hyperparameter optimization. This library simplifies the process of determining the best hyperparameters for machine learning models, making it critical for ensuring that the algorithms used perform optimally.

In summary, this environment setup, comprised of a mix of cutting-edge tools and libraries, provides a comprehensive and robust framework. Such a setup is critical for ensuring the study's rigor and ensuring that the results obtained can be reliably reproduced by peers and researchers in similar contexts.

### 3.9. Methodology Summary

This section provides a consolidated overview of the research methodologies used in this study to provide clarity and structure. Each step has been meticulously planned and carried out to ensure the highest levels of rigor and dependability.

**Variables Analysis:** The variables in the dataset were thoroughly explored at the start. The main goal was to correctly predict the TX\_FRAUD column. With the exception of Naive Bayes, nearly every machine learning technique used a feature importance code. This revealed that important characteristics varied across techniques, emphasizing the unique nature of each algorithm.

**Model Training:** To compare various machine learning techniques, two distinct methods were used:

**Temporal Splitting:** The dataset was divided into sections based on the TX\_DATETIME variable. This approach is appropriate because transactional data has a time series structure by definition.

**K-Fold Cross-Validation:** A k-fold technique was used, specifically with k=5. This method ensures that every data point is used in both training and validation, increasing the generalizability of the model.

Importantly, hyperparameter optimization was performed for each algorithm to determine the most effective parameter values. Oversampling methods were avoided consciously because they frequently resulted in overfitting in preliminary experiments.

**Model Assessment and Comparison:** The models' performance was rigorously evaluated after training. The ROC AUC score was a critical criterion, especially in the realm of Fraud Detection. This metric measures a model's ability to distinguish between legitimate and fraudulent transactions, with higher scores indicating superior discernment abilities. To provide a comprehensive evaluation landscape, several other metrics were used, including accuracy, precision, recall, F1-score, and the confusion matrix.

**Environment Configuration:** The computational environment was explicitly defined to ensure the study's reproducibility. The study, which was built on Python 3.9.13, used libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, and XGBoost. This technical

configuration ensured both robustness and versatility, allowing for the smooth execution of various data analysis and machine learning tasks.

Finally, the methodology of this study was founded on a synthesis of traditional machine learning practices and contemporary innovations. When combined with meticulous execution and rigorous evaluation, such an approach aimed to yield profound and actionable insights in the domain of Fraud Detection.

## 4. Results, Discussion and Future Work

The primary goal of this study was to compare the effectiveness of various machine learning algorithms in detecting fraud. Given the gravity of fraudulent transactions and their consequences, it is critical to assess the algorithms' performance using a variety of metrics, each of which reveals a different aspect of the algorithms' capabilities.

The following metrics were taken into account:

- **Accuracy:** Accuracy measures the proportion of correct predictions made in comparison to total predictions made. While high accuracy may sound appealing, it can be misleading in the context of fraud detection, especially when fraudulent activities are uncommon. In imbalanced datasets, a model with high accuracy may simply predict the majority class, rendering it unreliable.
- **Precision:** Precision measures the proportion of correct positive predictions among all positive predictions. A high precision score in fraud detection indicates that more transactions flagged as fraudulent are genuinely fraudulent. This reduces false alarms but may miss some legitimate fraudulent transactions.
- **Recall:** The proportion of correct positive predictions in the real positives is quantified by recall. A high recall in fraud detection ensures that the majority of fraudulent activities are detected, even if this means flagging some legitimate transactions as fraudulent.
- **F1 Score:** The harmonic mean of Precision and Recall is the F1 Score. It becomes critical when both false positives and false negatives are costly. An ideal F1 score denotes a harmonious balance of precision and recall.
- **ROC AUC Score:** The Area Under the Receiver Operating Characteristic Curve is referred to as the ROC AUC Score. In an ideal world, a model would have a ROC

AUC score of 1, while a model with no discernment capability would have a score of 0.5. A higher ROC AUC score indicates a better ability to distinguish between fraudulent and legitimate transactions.

#### 4.1. Performance of Machine Learning Algorithms

The following results were obtained from the models:

##### 4.1.1. Key Results

Following a review of the performance of various machine learning algorithms on the provided dataset:

- **Accuracy:** In terms of accuracy, the top-performing models were Xgboost (99.69%), AdaBoost (99.58%), and Naive Bayes (98.51%).
- **Precision:** AdaBoost took first place with a precision score of 93.64%, followed by Xgboost at 92.97%, and, surprisingly, Naive Bayes took third place with 32.78%.
- **Recall:** With a recall score of 78.07%, Logistic Regression outperformed other models. The Decision Tree and SGD Classifier came in second and third, with recall scores of 75.89% and 75.74%, respectively.
- **F1 Score:** Xgboost received the highest F1 score of 79.48%, followed by AdaBoost (69.83%) and Naive Bayes (43.80%).
- **ROC AUC Score:** With a ROC AUC score of 87.06%, the Decision Tree model outperformed Logistic Regression (86.96%) and the SGD Classifier (86.27%).

### 4.1.2. Execution Time

The execution times of the models varied significantly:

- Naive Bayes (2.1 seconds) and its Kfold variant (15.9 seconds) were the fastest models.
- The Random Forest Kfold took the longest, at 66 minutes and 36.8 seconds.
- Xgboost was 4 minutes and 1.9 seconds faster, while its Kfold variant took about 20 minutes.

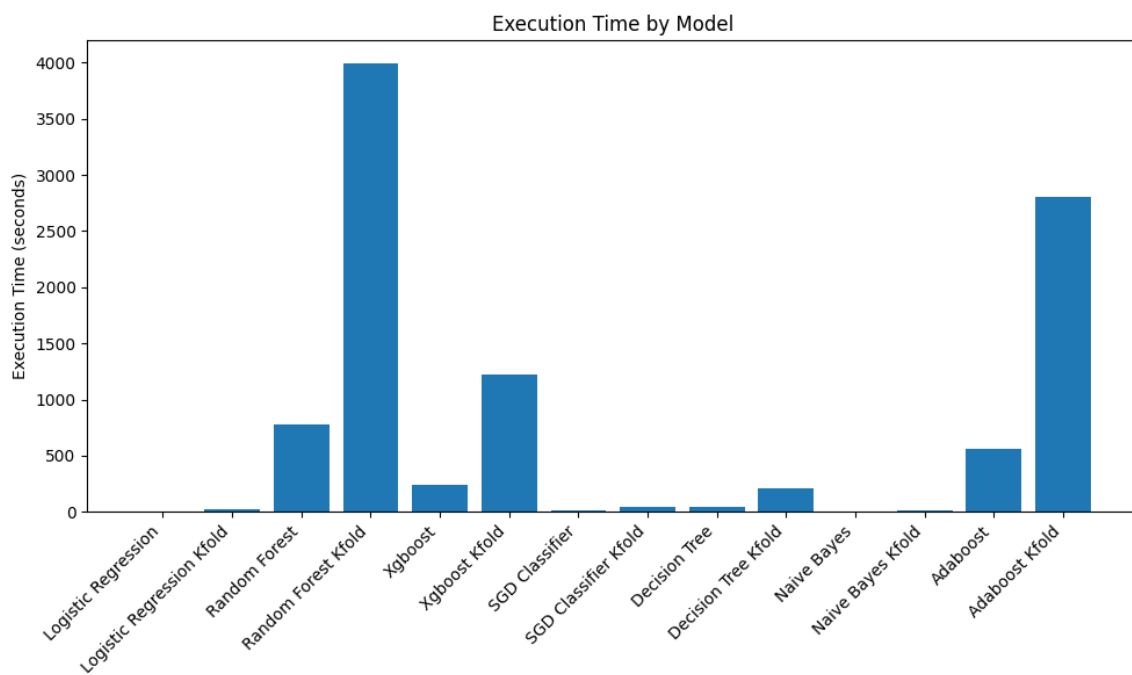


Figure 2: Execution Time of Models

### 4.1.3. Results for Time Based Split

**Figure 3** shows a comprehensive comparison of all the algorithms on the raw dataset based on each of the metrics. In this plot, the performance of each algorithm can be compared across all metrics.

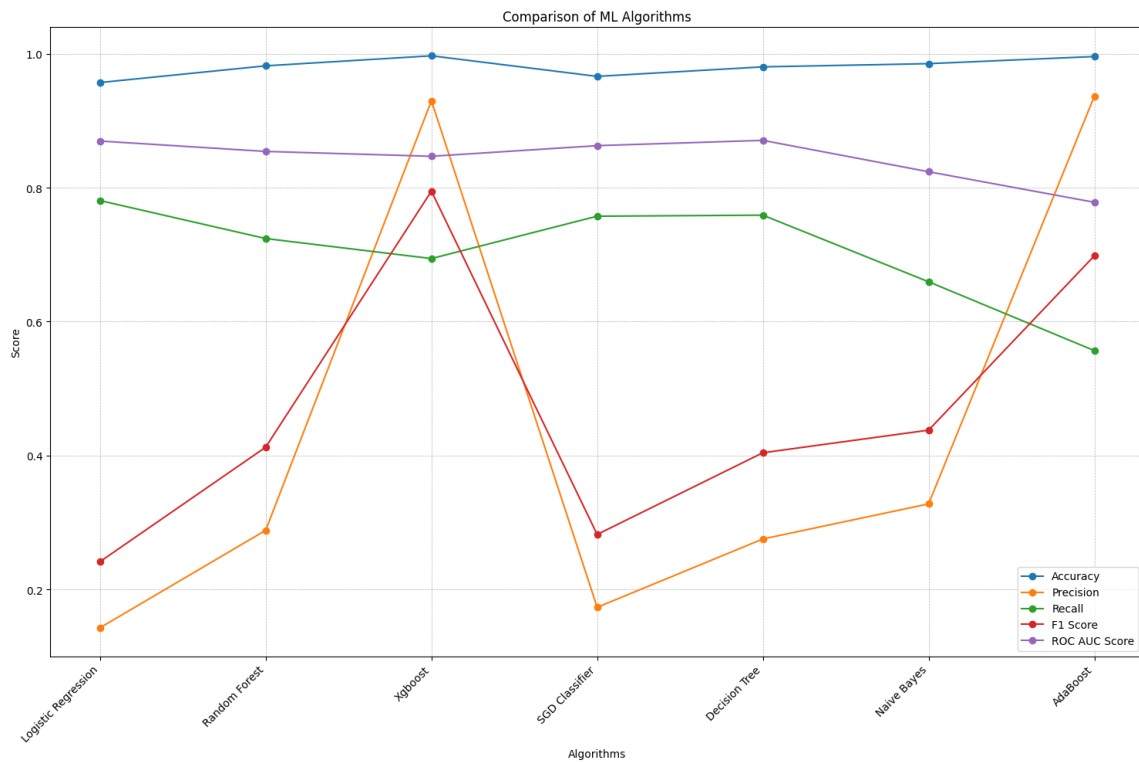


Figure 3: Combined Comparison of All Models and Metrics

**Figure 4** delves deeper into each metric, providing individual plots for each metric, which can be useful for a more granular performance analysis.

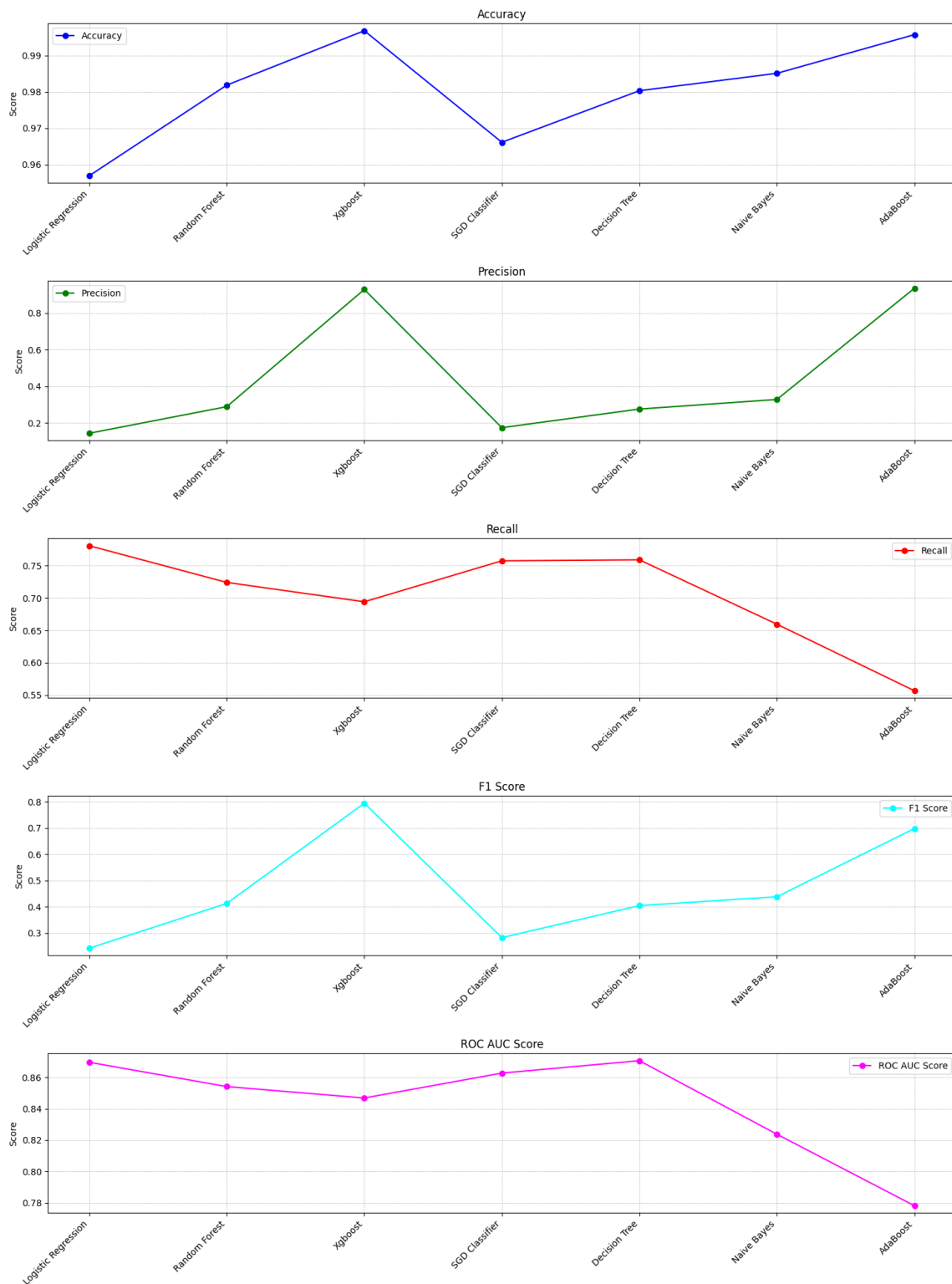


Figure 4: Separate Comparison of All Models and Metrics

#### 4.1.4. Results for Kfold Split

Similarly, **Figure 5** shows the K-Fold Cross-Validation results for all algorithms in a single view.

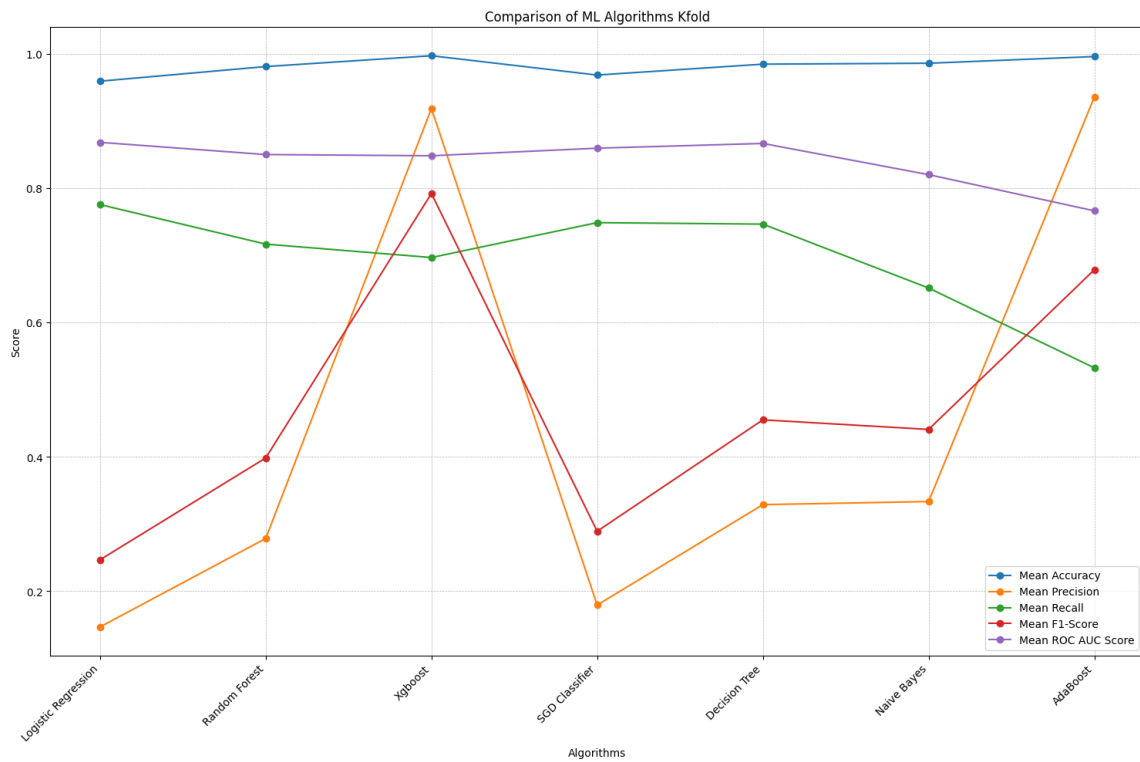


Figure 5: Combined Comparison of All Models and Metrics(Kfold)

Figure 6 shows individual K-Fold metric plots.

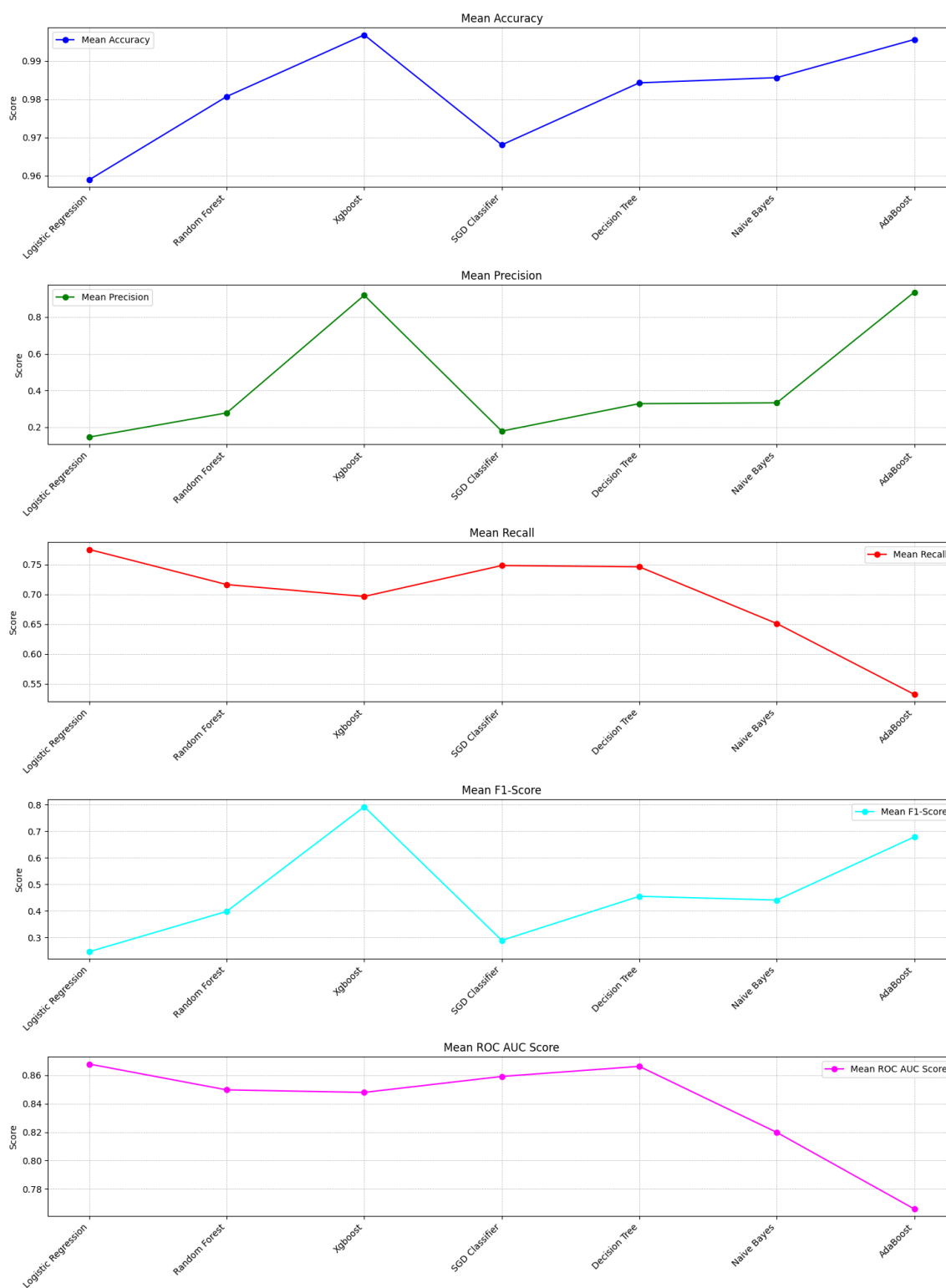


Figure 6: Separate Comparison of All Models and Metrics(Kfold)

## 4.2. Discussion

- When compared to other models, the **Xgboost** and **AdaBoost** algorithms have significantly higher precision. This indicates that they have fewer false alarms.
- Most models performed comparably in terms of recall, with Logistic Regression and Decision Tree having slightly higher recall than others, implying that they could identify a larger proportion of actual fraud cases.
- When considering the F1 score, which is a balance of precision and recall, Xgboost came out on top, closely followed by AdaBoost.
- Almost all models have a high ROC AUC Score, with Decision Tree and Logistic Regression taking a slight lead. This indicates a strong ability to distinguish between fraudulent and legitimate transactions.

## 4.3. Consideration

A weighted score system was established to prioritize the detection of fraudulent activities, with a greater emphasis on Recall (x4) and ROC AUC Score (x4). Xgboost was identified as the most capable model for detecting the majority of fraudulent activities based on the cumulative weighted metrics.

## 5. Conclusion and Future Work

### 5.1. Conclusions

Balancing various metrics to achieve a model that is both precise and recall-efficient is critical in the field of fraud detection. Because of its robust performance across multiple metrics and reasonable execution time, the provided results point to Xgboost as a potential frontrunner. However, the model should be chosen with the business impact of false positives and false negatives in mind. Efficient and timely detection can significantly reduce financial losses while also improving customer trust, emphasizing the importance of such analyses.

### 5.2. Future Work

Because our analysis of fraud detection models yielded insightful results, we can improve the efficacy of our approach by focusing on the following avenues:

#### 1. Data Augmentation:

- **Feature Engineering:** New features can be created from existing data to provide additional context to the models and improve predictive performance.

#### 2. Model Enhancements:

- **Ensemble Techniques:** Boosting, bagging, or stacking multiple models can provide a unified approach, potentially improving accuracy.
- **Deep Learning Approaches:** The potential of neural networks, particularly autoencoders and LSTM networks, in fraud detection can be investigated.

### **3. Real-time Fraud Detection:**

- Putting in place a real-time fraud detection system could drastically reduce potential losses. For real-time predictions, future research could look into stream processing systems, edge computing, or cloud deployments.

### **4. Feedback Loop:**

- Create a mechanism for continuous feedback from on-the-ground prediction validation. This feedback can be used to retrain models, ensuring that they adapt and improve as new data is introduced.

### **5. Evaluation Metrics Refinement:**

- While we have evaluated models using common metrics, future work may include the development of custom metrics that are specifically tailored to the business needs and costs associated with false positives and false negatives.

### **6. External Datasets and Transfer Learning:**

- Using data from related domains or tasks can provide useful context. Investigating the use of external datasets or employing transfer learning techniques may result in improved detection capabilities.

### **7. Cybersecurity Measures:**

- As models are deployed, it is critical to ensure their security from adversarial attacks or data breaches. Future strategies must include strong cybersecurity practices.

## 6. References

- Adepoju, O., Wosowei, J. and Jaiman, H., 2019, October. Comparative evaluation of credit card fraud detection using machine learning techniques. In 2019 Global Conference for Advancement in Technology (GCAT) (pp. 1-6). IEEE.
- Awoyemi, J.O., Adetunmbi, A.O. & Oluwadare, S.A., 2017. Credit card fraud detection using machine learning techniques: A comparative analysis. In 2017 international conference on computing networking and informatics (ICCNi) (pp. 1-9). IEEE.
- European Central Bank (2023) 'Report on card fraud in 2020 and 2021', European Central Bank. Available at: <https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202305~5d832d6515.en.html#toc6> (Accessed: [13/08/2023]).
- Khatri, S., Arora, A. and Agrawal, A.P., 2020, January. Supervised machine learning algorithms for credit card fraud detection: a comparison. In 2020 10th international conference on cloud computing, data science & engineering (confluence) (pp. 680-683). IEEE.
- Lakshmi, S.V.S.S. and Kavilla, S.D., 2018. Machine learning for credit card fraud detection system. International Journal of Applied Engineering Research, 13(24), pp.16819-16824.
- Le Borgne, Y.-A., Sibli, W., Lebichot, B. and Bontempi, G. (2022). Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook. Université Libre de Bruxelles. Available at: <https://github.com/Fraud-Detection-Handbook/fraud-detection-handbook> [Accessed 20 July 2023].
- Maniraj, S.P., Saini, A., Ahmed, S. and Sarkar, S., 2019. Credit card fraud detection using machine learning and data science. International Journal of Engineering Research, 8(9), pp.110-115.
- Minastireanu, E.A. and Mesnita, G., 2019. Light gbm machine learning algorithm to online click fraud detection. J. Inform. Assur. Cybersecur, 2019, p.263928.
- Mittal, S. and Tyagi, S., 2019, January. Performance evaluation of machine learning algorithms for credit card fraud detection. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 320-324). IEEE.
- Raghavan, P. & El Gayar, N., 2019. Fraud detection using machine learning and deep learning. In 2019 international conference on computational intelligence and knowledge economy (ICCIKE) (pp. 334-339). IEEE.
- Sailusha, R., Gnaneswar, V., Ramesh, R. & Rao, G.R., 2020. Credit card fraud detection using machine learning. In: 2020 4th international conference on intelligent computing and control systems (ICICCS) (pp. 1264-1270). IEEE.
- Tanouz, D., Subramanian, R.R., Eswar, D., Reddy, G.P., Kumar, A.R. and Praneeth, C.V., 2021, May. Credit card fraud detection using machine learning. In 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 967-972). IEEE.
- Thennakoon, A., Bhagyani, C., Premadasa, S., Mihiranga, S. & Kuruwitaarachchi, N. 2019, 'Real-time credit card fraud detection using machine learning', In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, pp. 488-493.
- The Nilson Report. (2015). Global fraud losses reach \$16.31 Billion. *The Nilson Report*, Issue 1068, July 2015. (Accessed: 26 August 2023).
- Trivedi, N.K., Simaiya, S., Lilhore, U.K. and Sharma, S.K., 2020. An efficient credit card fraud detection model based on machine learning methods. International Journal of Advanced Science and Technology, 29(5), pp.3414-3424.
- Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M. & Anderla, A. (2019) 'Credit card fraud detection-machine learning methods', in 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), IEEE, pp. 1-5.
- Yee, O.S., Sagadevan, S. and Malim, N.H.A.H., 2018. Credit card fraud detection using machine learning as data mining technique. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 10(1-4), pp.23-27.