SHIELDING FINANCES: MACHINE LEARNING FOR CREDIT CARD FRAUD DETECTION

A Report

by

RAHUL GAUTAM - 334003785

PRINCE TIBADIYA - 334002581

RITHVIK SRINIVASAIYA - 834006823

Submitted
to
PROFESSOR SHARMISTHA GUHA
Assistant Professor
Department of Statistics
Texas A&M University

in partial fulfillment of the requirements for the project part of

STAT654: STATISTICAL COMPUTING WITH R AND PYTHON

Spring 2024

ABSTRACT

Credit card fraud remains a persistent challenge, demanding effective fraud detection mechanisms amidst data imbalance. This study employs various machine learning algorithms, including logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks, to detect fraudulent transactions, addressing data imbalance through undersampling, oversampling, ADASYN, and SMOTE techniques. Through rigorous evaluation on real-world data, we analyze algorithm performance in terms of accuracy, precision, recall, and F1-score, revealing nuanced insights into the effectiveness of each approach and the associated trade-offs. Our findings provide practical guidance for optimizing fraud detection systems, highlighting the importance of algorithm selection and resampling strategy tailored to specific operational needs, thus advancing the efficacy and integrity of credit card fraud detection.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This research project was conducted by a team of three Master's students as part of the Stat654 course during the Spring 2024 semester. The initial problem statement was formulated by Prince, who conducted a thorough analysis of multiple use-cases and presented the findings to the group. The final research topic, along with dataset details, was approved by Professor Guha. Subsequently, Rahul devised a comprehensive plan outlining the workflow of the project, including the selection of suitable models and feature transformation methods. After careful consideration, Rithvik distributed tasks related to data transformation and machine learning models among the team members. Prince focused on implementing Logistic Regression and Naive Bayes classifier, while Rithvik experimented with Support Vector Machines (SVM) and Decision Trees. Rahul conducted experiments with the remaining tree-based methods. Finally, the team compiled and analyzed the results, ultimately selecting the best-performing model based on predefined metrics.

TABLE OF CONTENTS

Page

1. INTRODUCTION

Credit card fraud poses a significant threat to financial institutions and consumers worldwide, demanding effective detection mechanisms to mitigate risks and maintain trust in the financial system. Traditional rule-based systems and manual interventions have proven inadequate against evolving fraudulent schemes. However, the advent of machine learning (ML) techniques offers promising solutions by leveraging advanced analytics to analyze transaction data and detect fraudulent activities with greater accuracy and speed. ML models excel at identifying subtle anomalies indicative of fraud, enabling early detection and prevention of fraudulent transactions.

ML algorithms also offer adaptability and scalability, continuously learning and evolving to combat emerging fraud trends. They can incorporate diverse data sources, including transaction metadata and user behavior patterns, enhancing fraud detection capabilities. Despite these advantages, challenges persist, particularly concerning the class imbalance present in credit card transaction datasets. Addressing this imbalance is crucial to ensuring the reliability of ML-based fraud detection systems.

This research aims to bridge the gap between traditional fraud detection methods and ML techniques by evaluating the efficacy of various ML algorithms, such as Logistic Regression, Naive Bayes Classifier, and Support Vector Machines. Additionally, different resampling techniques, including undersampling and oversampling, are investigated to enhance fraud detection accuracy.

Through rigorous experimentation and analysis on authentic credit card transaction data, this research seeks to provide insights into optimal strategies for combating credit card fraud using ML-based approaches. By enhancing the efficiency and accuracy of fraud detection systems, the study aims to bolster the security of financial transactions and safeguard the interests of both financial institutions and consumers.

2. DATASET

2.1 Dataset details

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) accounts for 0.172% of all transactions. It contains only numeric input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, more information is not provided on the features. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction amount; this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

2.2 Pre-processing methods

The dataset contains a total of 28 variables which were a result of PCA reduction. Apart from that only the "Amount" feature had been scaled by using log normalization. Different kinds of methods were tried like scaling by log, standardization, and normalization. However, it was

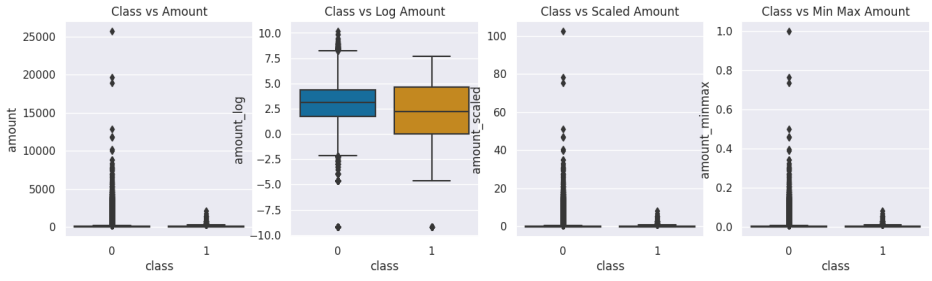


Figure 2.1: Figure showing results scaled according to different methods and compared with class feature

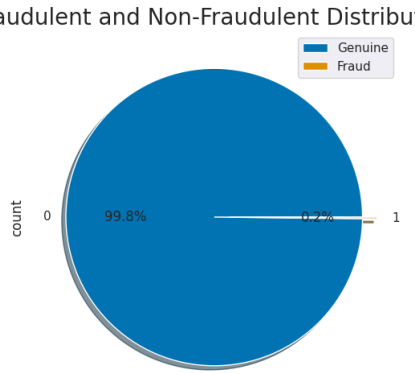
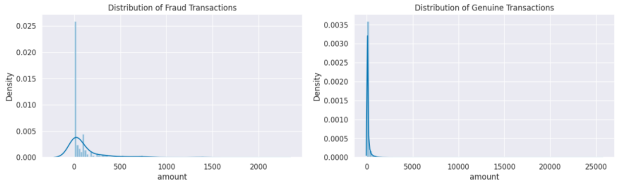


Figure 2.2: Figure showing Class Distribution among genuine and Fraud Transactions



(a) Distribution of transaction amounts

(b) Transaction amounts against class

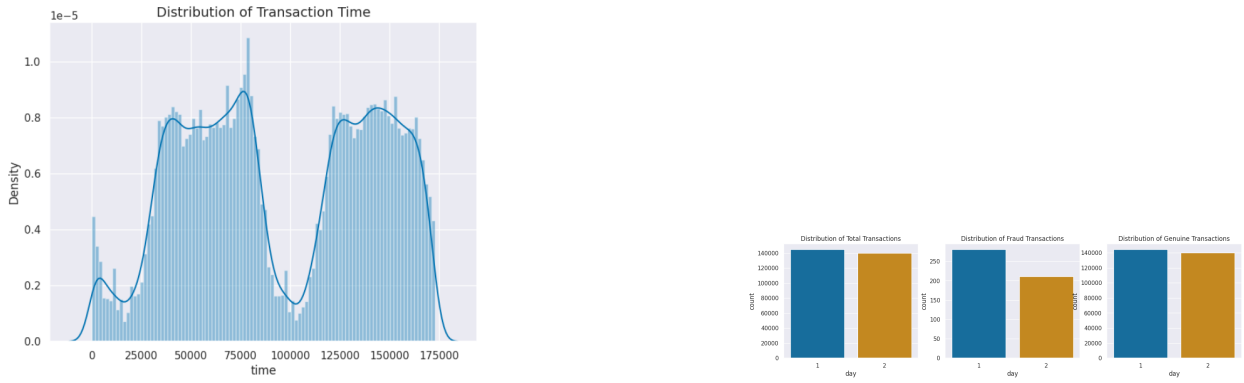
Figure 2.3: Comparison of transaction amount distributions

concluded that Scaling by log yielded the best results. The time feature is also changed to create a day and hour feature as the record shows transactions for two days.

2.3 Exploratory Data Analysis

Some plots were generated to get an in-detail view of certain features. It was observed that the dataset consisted of 492 fraudulent transactions whereas the rest were genuine transactions. This meant that 99.8% of the data consisted of a genuine transaction

Furthermore, a distribution was calculated to show around what amount did fraud transactions generally occurred. As seen from the figure, most of the fraudulent transactions are made in



(a) Distribution of Transaction Time

(b) Distribution of Transaction with Day

Figure 2.4: Comparison of transaction time and day distributions

the 0 to 500 USD range. That is evident as most of companies and consumers must have an authentication system for transactions with a higher amount. On further inspection, we also see that many of the transactions had an amount of 0 or 1 USD. On further research, it was found that this was a "Zero Authentication" fraud. This zero-authentication transaction allows companies and vendors to charge customers a 0-dollar transaction which usually is used to validate a payment method. Recurrent transactions can also be incurred further on after the zero authorization is granted. Scammers use this method to enable them to further charge consumers for unauthorized payments.

A distribution of time was also made to check the time for most of these transactions. As we see from the figure, we understand that there were only a few peaks in the distribution with 2 major peaks. This may indicate the time of the day. The peaks signify day and the crests signify night. Since the data consists of transactions made over 2 days, this checked out as expected.

Lastly, a correlation heatmap was generated to check for underlying correlations between different features. We see that some features like V2, V4, V11, and V19 have a high correlation which means that as the value of these features increases, there is a higher chance that a transaction will be fraudulent. Some features like V3, V10, V12, V14, and V16 have a negative correlation, which means that as these value decreases there is a higher chance that a transaction will be a fraudulent one.

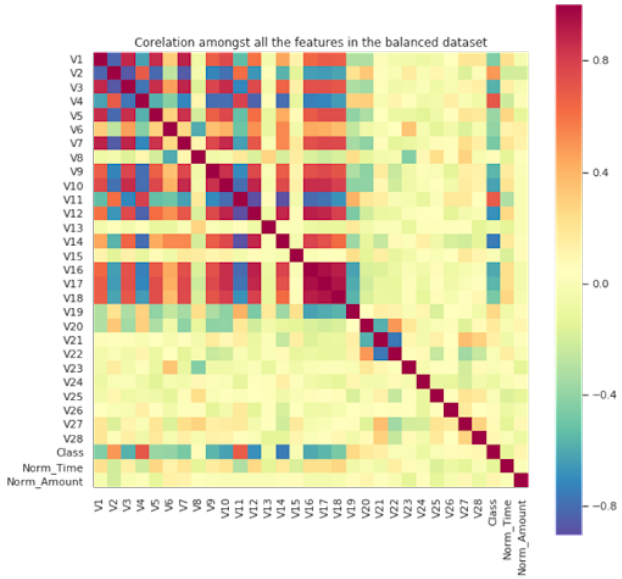


Figure 2.5: Feature Correlation-matrix

2.4 Sampling Methodologies

Sampling methodologies were crucial in the analysis of imbalanced datasets, particularly in our scenario, where fraudulent transactions were rare compared to legitimate ones. In our dataset, there were 492 frauds out of 284,807 transactions. This severe class imbalance posed a significant challenge for fraud detection algorithms, potentially leading to biased predictions towards the majority class and inaccurate detection of fraudulent transactions.

2.4.1 Random Under-sampling

Random under sampling involves randomly selecting a subset of the majority class to create a more balanced dataset. Hence, randomly selecting 492 transactions from the majority class resulted in a balanced dataset with 492 frauds and 492 non-frauds.

However, random under sampling had its limitations, as it could lead to a loss of important information present in the majority of class instances that were removed, potentially reducing the overall performance of the algorithm.

2.4.2 Random Oversampling

Random oversampling involves randomly duplicating instances of the minority class to increase its representation in the dataset. Hence, randomly duplicating the 492 fraud transactions resulted in a balanced dataset with 284,315 frauds and 284,315 non-frauds.

One drawback of random oversampling was that it could lead to over-fitting, as the algorithm might become too focused on the duplicated instances of the minority class, reducing generalization performance on unseen data.

2.4.3 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generated synthetic instances of the minority class by interpolating between existing instances. Hence, using SMOTE to generate synthetic instances for the 492 fraud transactions resulted in a balanced dataset with 284,315 frauds and 284,315 non-frauds.

SMOTE was often more effective than simple random oversampling, as it addressed the issue of over-fitting by generating synthetic instances that were close to, but not identical to, the original minority class instances.

The SMOTE generates synthetic samples for the minority class by the following steps:

1. **Selecting Samples:** For each instance in the minority class, select k nearest neighbors from the same class. Let x_i be a sample from the minority class.

2. **Synthetic Sample Generation:** For each nearest neighbor x_{nn}, generate synthetic instances by interpolating between x_i and x_{nn}. The formula for creating a new synthetic data point x_{new} is:

$$x_{new} = x_i + \lambda \cdot (x_{nn} - x_i)$$

   where λ is a random number between 0 and 1.

3. **Repeating the Process:** Repeat the process until the class distribution is balanced. In this case, generate enough samples to match the number of instances in the majority class.

This method helps in creating a more diverse set of examples by interpolating between multiple pairs of closely situated samples, reducing the risk of overfitting compared to simple random oversampling.

2.4.4 ADASYN (Adaptive Synthetic Sampling)

ADASYN was an extension of the SMOTE algorithm that adaptively generated synthetic instances based on the density of the minority class. It focused more on generating instances for minority class instances that were harder to classify. Hence, using ADASYN to generate synthetic instances for the 492 fraud transactions resulted in a balanced dataset with 284,315 frauds and 284,325 non-frauds.

ADASYN was particularly useful in situations where the distribution of the minority class was highly skewed or when there were overlapping regions between the minority and majority classes.

ADASYN algorithm aims to generate synthetic samples according to the data distribution: it adaptively changes the number of synthetic samples for different minority class examples. The steps are as follows:

1. **Density Distribution:** For each minority class sample x_i, calculate the ratio γ_i of the number of majority class neighbors to the k nearest neighbors.

2. **Generation of Synthetic Samples:** The number of synthetic samples to generate for each minority class sample x_i is determined by:

$$G_i = G \cdot \frac{\gamma_i}{\sum_{j=1}^{n} \gamma_j}$$

   where G is the total number of synthetic samples to generate, and n is the total number of minority class samples.

3. **Synthetic Sample Creation:** For each x_i, generate G_i synthetic samples by interpolating

between x_i and one of its k nearest neighbors x_{nn} in the minority class:

$$x_{new} = x_i + \lambda \cdot (x_{nn} - x_i)$$

where λ is a random number between 0 and 1.

4. **Adjustment of Samples:** The algorithm focuses more on those minority class samples that are harder to learn (i.e., those that are more surrounded by majority class samples), leading to better adaptability of the synthetic sample generation.

ADASYN enhances learning about the minority class in areas where the classifier is underperforming, leading to a more sophisticated model behavior and better generalization capabilities.

3. METHODOLOGY

In this study, we aim to evaluate various machine learning algorithms' performance in handling imbalanced datasets for credit card fraud detection. Our goal is to identify effective approaches to improve fraudulent transaction detection despite significant disparities between genuine and fraudulent instances. We assessed popular algorithms like logistic regression, naive Bayes, support vector machines (SVM), decision trees, random forests, gradient boosting, and XGBoost. Utilizing four datasets created using oversampling, undersampling, SMOTE, and the original imbalanced dataset, we comprehensively evaluated the algorithms' ability to handle class imbalance and compared their performance across different data distributions.

3.1 Logistic Regression

Logistic regression estimates the probability of a binary outcome based on one or more predictor variables and is widely used for its simplicity and effectiveness in binary classification tasks. However, the performance of logistic regression can be adversely affected by class imbalance, which is typical in fraud detection where fraudulent transactions are much rarer than legitimate transactions. It is modeled as,

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)}}$$

Where:

- y is the binary target variable (0 for non-fraud, 1 for fraud).
- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are the predictors which could include variables like transaction amount, merchant type, time since the last transaction, etc.
- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients of the model, which are learned from the training data.
- e is the base of the natural logarithm.

3.2 Naive Bayes Classifier

Naive Bayes classifies data based on the probabilistic relationships dictated by Bayes' Theorem. It estimates the probability of a class given a set of independent predictor variables and is particularly favored for its efficiency and straightforward application in multi-class classification scenarios. Despite its efficacy, Naive Bayes can sometimes struggle with highly correlated features due to its assumption of predictor independence. It is modeled as,

$$P(y = c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid y = c) \times P(y = c)}{P(\mathbf{x})}$$

Where:

- $y = c$ is the class variable (e.g., 'fraud' or 'non-fraud').
- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are the predictors, which could include variables such as transaction amount, merchant type, time of transaction, etc.
- $P(\mathbf{x} \mid y = c)$ is the likelihood of observing the predictors given the class.
- $P(y = c)$ is the prior probability of the class.
- $P(\mathbf{x})$ is the marginal probability of the predictors, often calculated as the sum of the probabilities of the predictors given each class, weighted by the class probabilities.

3.3 Support Vector Machine

Support Vector Machines are employed in credit card fraud detection by constructing a hyperplane that effectively separates fraudulent transactions from legitimate ones in a high-dimensional space. Given the complex nature of transaction data and the subtlety of fraud patterns, SVM's capability to operate effectively in high-dimensional spaces makes it particularly suitable for this application. The model focuses on maximizing the margin between the classes, which is critical in ensuring robustness, especially when dealing with highly imbalanced datasets typical in fraud detection scenarios.

The SVM model for credit card fraud detection is generally formulated as:

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \right\}$$

Subject to,

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

Where:

- $\mathbf{w}$ is the weight vector of the hyperplane.
- b is the bias term, offsetting the hyperplane from the origin.
- $\phi(\mathbf{x}_i)$ may represent a transformation applied to the input features $\mathbf{x}_i$, such as transaction amount, time of transaction, cardholder behavior patterns, and other relevant data.
- $y_i \in -1, 1$ indicates the class labels, where 1 could represent a fraudulent transaction and -1 a non-fraudulent transaction.
- ξ_i are the slack variables, allowing for some misclassifications, essential in dealing with noisy data or outliers.
- C is the regularization parameter, providing a trade-off between the classifier's complexity and the degree to which deviations larger than the margin are penalized, thus controlling over-fitting and allowing the model to better generalize.

3.4 Decision Trees

Decision Trees are simple and interpretable machine learning algorithms used for classification and regression tasks. They partition the feature space based on feature values, recursively splitting the data into subsets until certain stopping criteria are met. At each split, the algorithm selects the feature that best separates the data, aiming to minimize impurity or maximize information gain. Prediction is done by traversing the tree from the root to a leaf node, where the majority class

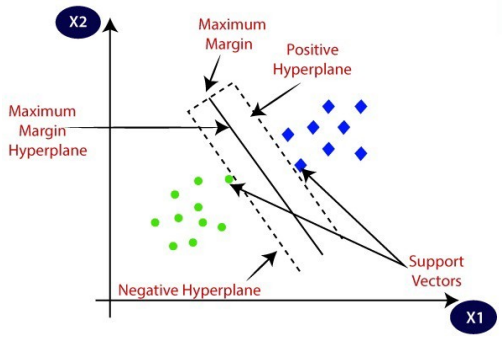
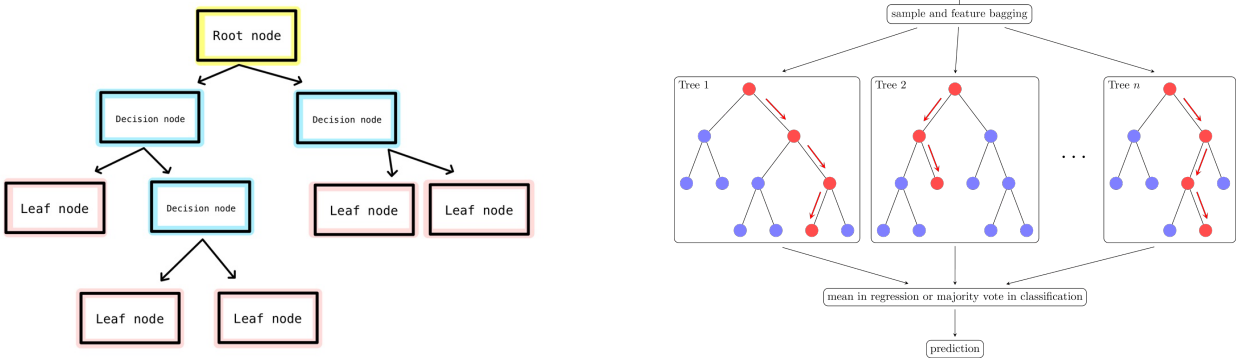


Figure 3.1: Support Vector Machine



(a) Decision Tree Classifier

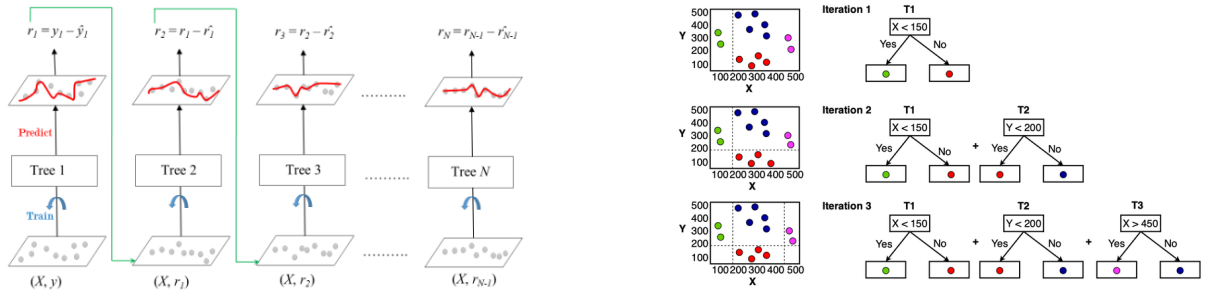
(b) Random Forest Classifier

Figure 3.2: Decision Tree vs Random Forest

or average target value determines the prediction. Decision Trees are easy to interpret and find applications in various domains, but they can over-fit if not properly regularized.

3.5 Random Forest

The Random Forest Classifier is a powerful machine learning algorithm that constructs multiple decision trees during training and combines their predictions to produce accurate results. By training each tree on a random subset of the data and a random subset of features, it promotes diversity and reduces overfitting. The final prediction is made through a voting mechanism, where each tree's output contributes to the overall decision. With its simplicity, effectiveness, and ability to handle various types of data, the Random Forest Classifier is widely used for classification tasks



(a) Gradient Boosted Trees

(b) XGBoost Tree Classifier

Figure 3.3: GBDT vs XGBoost

across different domains.

3.6 Gradient Boost

Gradient Boosting is a machine learning technique used for regression and classification problems. Unlike Random Forest, which builds multiple decision trees independently, Gradient Boosting builds trees sequentially, each tree learning from the mistakes of its predecessor. In each iteration, the model focuses on the residuals (the difference between predicted and actual values), gradually reducing errors and improving accuracy. It's a powerful algorithm known for its high predictive performance and flexibility in handling different types of data. However, it can be computationally intensive and requires careful tuning of parameters to prevent overfitting. Overall, Gradient Boosting is widely used in practice due to its ability to produce highly accurate predictions in a variety of domains.

3.7 XGBoost

XGBoost, short for eXtreme Gradient Boosting, is an optimized and scalable implementation of Gradient Boosting. It's renowned for its speed and performance, making it a popular choice for various machine learning tasks, including classification, regression, and ranking problems. XGBoost enhances the original Gradient Boosting algorithm by introducing regularization techniques to control overfitting and handling missing data. It also incorporates advanced features like parallel computing and tree pruning to improve efficiency and scalability. With its robustness, speed, and

versatility, XGBoost has become a go-to algorithm for many data scientists and machine learning practitioners, winning numerous competitions and earning widespread adoption in both academia and industry.

4. RESULTS AND CONCLUSION

4.1 Results

In the development and evaluation of machine learning models, particularly in applications like fraud detection, utilizing a variety of performance metrics is crucial for ensuring optimal performance. Key metrics include Accuracy, which measures overall correctness, AUC (Area Under the Curve), representing the model's discrimination ability, Precision for accuracy of positive predictions, Recall for the model's ability to identify all relevant instances, and F1 Score, which balances precision and recall. Given the focus on robust identification of positive cases while maintaining a reasonable precision threshold, the F1 Score is particularly important, offering a balanced measure critical in minimizing false negatives and effectively managing the rate of false positives in fraud detection.

4.1.1 Logistic Regression

The performance metrics for a logistic regression model applied to our credit card fraud detection dataset with different sampling strategies reveal varying outcomes in terms of accuracy, AUC, precision, recall, and F1 score. When the dataset is imbalanced, the model achieves an exceptionally high accuracy of 99.92%, which might be misleading as it reflects the majority class performance predominantly. The AUC score of 80.549% is moderate, suggesting an average ability to differentiate between the classes. The precision here is quite high at 88%, but the recall is much lower at 61.111%, leading to a decent F1 score of 72.131%, indicating a skewed performance towards the majority class.

In contrast, methods like under-sampling, oversampling, SMOTE, and ADASYN, which aim to balance the dataset, show a significant trade-off between accuracy and the ability to correctly classify fraud cases (recall) but showed very poor results when evaluation is done on F1 score. From the confusion matrix, we can infer that out of the 144 fraudulent cases in the test dataset, only 88 were predicted to be fraudulent, while there is a significant number of false positives and

Logistic Regression	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99920	0.80549	0.88000	0.61111	0.72131
Undersampling	0.97252	0.94117	0.05312	0.90972	0.10038
Oversampling	0.97799	0.93698	0.06486	0.89583	0.12062
SMOTE	0.97604	0.93601	0.05969	0.89583	0.11193
ADASYN	0.91866	0.93153	0.01921	0.94444	0.03766

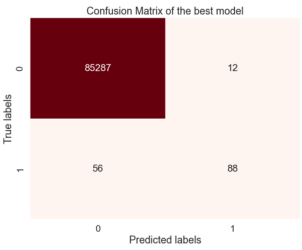


Figure 4.1: Logistic regression Results and best model Confusion matrix

Naive Bayes	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.97788	0.90920	0.06087	0.84028	0.11351
Undersampling	0.95951	0.91386	0.03505	0.86806	0.06739
Oversampling	0.97411	0.91424	0.05315	0.85417	0.10008
SMOTE	0.97451	0.91444	0.05395	0.85417	0.10149
ADASYN	0.95956	0.92082	0.03562	0.88194	0.06848

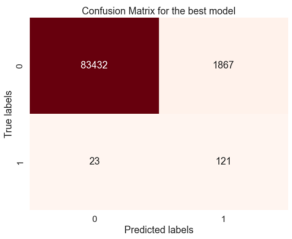


Figure 4.2: Naive Bayes Results and best model Confusion matrix

false negatives. This led to high recall, but poor precision for the sampling techniques.

4.1.2 Naive Bayes Classifier

The analysis of F1 scores across various sampling techniques for the Naive Bayes model in fraud detection highlights the challenge of balancing precision and recall in highly skewed datasets. Despite efforts with different techniques, including oversampling, undersampling, and synthetic minority oversampling techniques (SMOTE), the model consistently struggled to achieve a balance between correctly identifying fraud cases and minimizing false positives. This underscores the inherent difficulty of using Naive Bayes for imbalanced datasets in fraud detection. Future research may explore alternative sampling methods or modeling approaches to address this issue and improve the model's predictive performance without compromising either precision or recall. From the confusion matrix, we can infer that out of the 144 fraudulent cases in the test dataset, only 121 were predicted actually fraudulent, while there is a very large number of false negatives. This led to high recall, but poor precision for the sampling techniques.

SVM	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99943	0.87146	0.89916	0.74306	0.81369
Undersampling	0.97446	0.94215	0.05696	0.90972	0.10720
Oversampling	0.97885	0.93741	0.06715	0.89583	0.12494
SMOTE	0.97848	0.93722	0.06605	0.89583	0.12303
ADASYN	0.92362	0.93402	0.02044	0.94444	0.04001

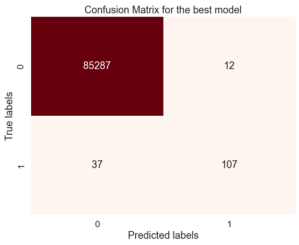


Figure 4.3: SVM Results and best model Confusion matrix

4.1.3 Support Vector Machine

For the imbalanced dataset, the SVM classifier showcases the highest accuracy of 99.943%, along with a robust AUC of 87.146%. This is complemented by a high precision of 89.916% and a reasonably good recall of 74.306%, resulting in the highest F1 score of 81.369% among all tested scenarios. This indicates that while the model performs well on an imbalanced set, it might still be somewhat influenced by the majority class.

When balancing techniques like undersampling, oversampling, SMOTE, and ADASYN are employed, there is a noticeable drop in accuracy, but improvements are seen in the AUC, which is critical for distinguishing between the fraud and non-fraud classes. From the confusion matrix, we can infer that out of the 144 fraudulent cases in the test dataset, 107 were predicted fraudulent, while there is still a significant number of false negatives and false positives combined.

4.1.4 Decision Trees

For the imbalanced dataset, the Decision Tree (DT) classifier achieves an impressive accuracy of 99.912% and an AUC of 88.864%, indicating its capability to discern between fraudulent and non-fraudulent transactions. However, the precision of 72.258% and recall of 77.778% suggest a balanced performance in identifying both classes, resulting in an F1 score of 74.916

Overall, the Decision Tree classifier showcases promising performance on the imbalanced dataset, with oversampling presenting the most effective approach in improving fraud detection accuracy while maintaining a balanced performance across precision and recall metrics.

DT	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99912	0.88864	0.72258	0.77778	0.74916
Undersampling	0.89660	0.89622	0.01442	0.89583	0.02837
Oversampling	0.99927	0.85058	0.84167	0.70139	0.76515
SMOTE	0.99792	0.88803	0.43411	0.77778	0.55721
ADASYN	0.99787	0.89148	0.42803	0.78472	0.55392

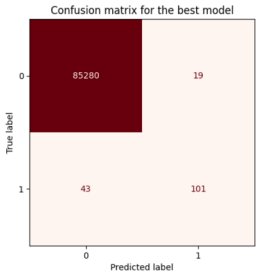


Figure 4.4: Decision Tree Results and best model Confusion matrix

4.1.5 Random Forest

For the imbalanced dataset, the Random Forest (RF) classifier achieves an accuracy of 99.958% and an AUC of 89.926%, indicating its robustness in distinguishing between fraudulent and non-fraudulent transactions. The precision of 94.262% and recall of 79.861% lead to a high F1 score of 86.466%, demonstrating the model's effectiveness in identifying both classes while minimizing false positives. However, oversampling leads to a slight decrease in accuracy to 99.957% but maintains a balanced performance with a precision of 94.215% and recall of 79.167%. The F1 score remains high at 86.038%, indicating a robust performance in fraud detection.

Overall, the Random Forest classifier exhibits strong performance on the imbalanced dataset, with oversampling presenting the most effective approach in improving fraud detection accuracy while maintaining a balanced performance across precision and recall metrics. From the confusion matrix, we can infer that out of the 144 fraudulent cases in the test dataset, 115 were predicted to actually be fraudulent, while there is still a significant number of false negatives and false positives combined.

4.1.6 Gradient Boost

The Gradient Boosting Machine (GBM) classifier performed best on the imbalanced dataset, achieving a high overall accuracy (99.910%) and strong ability to distinguish classes (AUC of 79.16%). It was also precise in its predictions (83.168%) and reasonably good at finding all relevant instances (recall of 58.333%), resulting in the highest overall effectiveness (F1 score of 68.571%).

RF	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99958	0.89926	0.94262	0.79861	0.86466
Undersampling	0.96915	0.92215	0.04592	0.87500	0.08726
Oversampling	0.99957	0.89579	0.94215	0.79167	0.86038
SMOTE	0.99954	0.91658	0.88889	0.83333	0.86022
ADASYN	0.99952	0.92003	0.87050	0.84028	0.85512

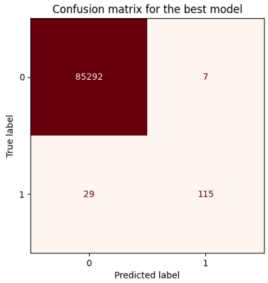


Figure 4.5: Random Forest Results and best model Confusion matrix

Gradient Boost	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99910	0.79157	0.83168	0.58333	0.68571
Undersampling	0.96492	0.94084	0.04235	0.91667	0.08096
Oversampling	0.99357	0.93439	0.19178	0.87500	0.31461
SMOTE	0.98770	0.93144	0.10871	0.87500	0.19340
ADASYN	0.96957	0.93623	0.04786	0.90278	0.09091

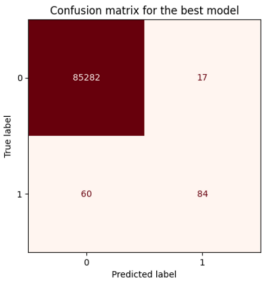


Figure 4.6: Gradient Boosting Results and best model Confusion matrix

However, this suggests a slight bias towards the majority class. Balancing techniques improved the model's ability to distinguish classes (AUC) but came at a cost. Undersampling, for example, significantly reduced accuracy (96.492%) and precision (4.235%) while boosting recall (91.667%). Oversampling and SMOTE showed similar trade-offs, achieving slightly better precision but still falling short of the imbalanced SVM's F1 score. ADASYN prioritized recall (90.278%) at the expense of precision and F1 score, indicating it might overemphasize the minority class.

4.1.7 XGBoost

The Extreme Gradient Boosting Machine (XGBoost) classifier performed best on the oversampled dataset, achieving a high overall accuracy (99.961%) and strong ability to distinguish classes (AUC of 90.968%). It was also precise in its predictions (94.4%) and reasonably good at finding all relevant instances (recall of 81.944%), resulting in the highest overall effectiveness (F1 score of 87.732%). However, this suggests a slight bias towards the majority class. This method gave the

XGBoost	Accuracy	AUC	Precision	Recall	F1
Imbalanced	0.99956	0.89232	0.94167	0.78472	0.85606
Undersampling	0.96324	0.94692	0.04104	0.93056	0.07862
Oversampling	0.99961	0.90968	0.94400	0.81944	0.87732
SMOTE	0.99939	0.92344	0.80263	0.84722	0.82432
ADASYN	0.99920	0.91641	0.73171	0.83333	0.77922

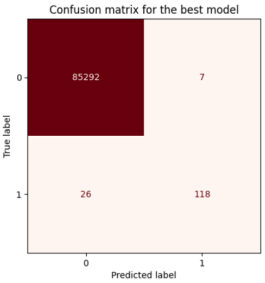


Figure 4.7: XGboost Results and best model Confusion matrix

Model	Accuracy_Test	AUC_Test	PrecisionScore_Test	RecallScore_Test	F1Score_Test
XGB Oversampling	0.999614	0.909681	0.944000	0.819444	0.877323
RF imbalance	0.999579	0.899265	0.942623	0.798611	0.864662
RF Oversampling	0.999567	0.895792	0.942149	0.791667	0.860377
RF SMOTE	0.999544	0.916579	0.888889	0.833333	0.860215
XGB imbalance	0.999555	0.892320	0.941667	0.784722	0.856061
RF ADASYN	0.999520	0.920033	0.870504	0.840278	0.855124
XGB SMOTE	0.999391	0.923435	0.802632	0.847222	0.824324
SVM imbalance	0.999427	0.871457	0.899160	0.743056	0.813688
XGB ADASYN	0.999204	0.916409	0.731707	0.833333	0.779221
DT Oversampling	0.999274	0.850583	0.841667	0.701389	0.765152

Figure 4.8: Top-10 results based on F1-score

best possible balance between Precision, Recall and F1 Score. The model fails to minimize false positives and false negatives while maximizing True positive and true negatives. Oversampling does avert class imbalance and thus not compromising on a biased model. The overall results can be seen in Figure 4.8 sorted on the basis of F1-score.

4.2 Conclusion

Based on the results, it can be concluded that the XGBoost model with the Oversampling technique is the most effective in detecting credit card fraud, achieving a high Model Accuracy and AUC. The Oversampling technique helps to improve the performance of the XGBoost model by increasing the number of minority class samples, which is essential in fraud detection where the number of fraudulent transactions is typically low. The results also suggest that different models and techniques have varying strengths and weaknesses. For instance, the Random Forest model with imbalance technique is good at identifying true positives (high Precision), while the XGBoost model with SMOTE technique is good at identifying true negatives (high Recall). Overall, the study highlights the importance of selecting the right machine learning model and technique to handle class imbalance in credit card fraud detection. By choosing the best approach, financial institutions can improve the accuracy of their fraud detection systems and reduce the risk of financial losses. For future work, deep learning approaches like generative encoders and adversarial training can be utilized to generate fraud samples with an identical distribution instead of using sampling techniques which might mitigate the imbalanced problems.

Code can be found *here*