POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Credit Card Fraud Detection Report

Numerical Analysis for Machine Learning
Computer Science and Engineering

Daniele Laganà - Gianluigi Palmisano

Teacher:
Prof. Edie Miglio

Academic year:
2023-2024

Abstract: This report addresses the critical issue of credit card fraud detection by leveraging various machine learning methodologies, with a particular focus on ensemble learning models. The primary challenge in this domain is the significant imbalance between fraudulent and non-fraudulent transactions, which can skew the performance of detection models. To mitigate this, techniques such as under-sampling and Synthetic Minority Over-sampling Technique (SMOTE) were employed to balance the dataset. The study evaluates the effectiveness of multiple machine learning models, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Bagging, and Boosting. An ensemble model combining these classifiers within a voting framework demonstrated superior performance in terms of precision, recall, F1-score, ROC, and overall accuracy. The computational efficiency of these models was also examined, highlighting trade-offs between training and testing times. The results underscored the importance of balancing accuracy with computational efficiency to ensure the models' applicability in real-time fraud detection. Future research should explore the integration of deep learning techniques and dynamic data sampling strategies to enhance model resilience and scalability against evolving fraud patterns and adversarial attacks.

Key-words: Credit card fraud detection, machine learning, ensemble learning, data imbalance, SMOTE, computational efficiency.

1. Introduction

Various approaches, including Statistical, Machine Learning, and Deep Learning methods are utilised for credit card fraud detection. Statistical techniques such as Regression, hypothesis testing, and clustering are applied to identify and analyse anomalies in credit card transactions. In contrast, machine learning methods leverage algorithms to detect fraudulent activity in real time by analysing historical data. Deep learning methodologies utilise neural networks to autonomously identify intricate patterns and features within complex datasets, resulting in exceptionally accurate detection of fraudulent activities.
A significant issue with credit card fraud detection is data imbalance which stems from the uneven distribution of fraudulent and non-fraudulent transactions within the dataset. This imbalance poses a risk of biased model outcomes and sub-optimal fraud detection capabilities. Several studies have tackled this concern through the application of machine learning techniques like data balancing, oversampling, undersampling, and the utilisation of the synthetic minority oversampling technique (SMOTE) to manage imbalanced credit card fraud data. However, a comprehensive exploration of the effectiveness of these techniques remains lacking.

Ensemble learning models and techniques play a crucial role in credit card fraud detection. These approaches involve combining multiple individual models to create a more robust and accurate fraud detection system. Ensemble methods, such as bagging, boosting, and stacking, are commonly employed to address challenges like imbalanced datasets and to enhance overall predictive power. By leveraging the strengths of diverse base models, ensemble techniques contribute to improved fraud detection performance, reducing the risk of false positives and false negatives.

The adaptability and effectiveness of ensemble learning make it a valuable strategy in the continuous battle against credit card fraud. Despite their success, there exists a crucial need to delve into the computational efficiency of these ensemble models.

The demand for computational efficiency in ensemble machine learning models is paramount due to its implications for real-world applicability and scalability. As these models become increasingly prevalent in finance, the ability to process large datasets swiftly and make timely predictions is crucial. Computational efficiency ensures that ensemble models can handle the complexities of intricate algorithms, intricate feature engineering, and the integration of diverse base models without compromising speed or responsiveness. This metric is especially pertinent in scenarios such as credit card fraud detection, where rapid decision-making is essential for preventing financial losses. Moreover, efficient computation supports the deployment of ensemble models in resource constrained environments, making them accessible to a broader range of applications.

The following report seeks to assess the overall performance, including the computational efficiency of ensemble models implementing data balancing techniques in credit card fraud detection.

The main topics of this report are the following:

1. To propose an effective credit card fraud detection model that addresses the prevalent challenge of data imbalance, a major concern arising from the uneven distribution of fraudulent and non-fraudulent transactions within datasets.
2. To demonstrate the computational efficiency of the proposed ensemble models, ensuring that the ensemble models can effectively handle complex algorithms, intricate feature engineering, and the integration of diverse base models without compromising speed.
3. To compare the performance of various machine learning models in identifying credit card fraud, such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), Bagging, and Boosting.

2. Methodology

2.1. Dataset

2.1.1 Overview

For our model training and testing, the Credit Card Fraud Detection dataset was utilized. This dataset contains records of transactions made by European credit cardholders over a span of two days. It includes 284,807 transactions, with 492 identified as fraudulent. This results in the positive class (frauds) constituting only 0.172% of all transactions, indicating a significant imbalance.

2.1.2 Features

Each transaction in the dataset is characterized by 30 features:

- V1 to V28: These features were derived using Principal Component Analysis (PCA) to address confidentiality concerns.
- Time: This feature represents the elapsed time in seconds between each transaction and the first transaction in the dataset. It was not transformed by PCA.
- Amount: This feature represents the transaction amount and was also not transformed by PCA.

2.1.3 Addressing Class Imbalance

The pronounced skewness of the dataset posed challenges for effectively training and testing the model. To address this issue, two methods were employed:

Under-sampling: This approach involved randomly selecting 492 instances from the 284,315 legitimate transactions to create a balanced dataset with an equal distribution of legitimate and fraudulent transactions.

Synthetic Minority Over-sampling Technique (SMOTE): This technique involved generating synthetic samples to increase the number of fraudulent instances that originally were 492 to match the number of legitimate entries, resulting in a balanced representation of both classes.

2.2. The Proposed Model

The experimental design of the proposed model was selected to test and assess fraud detection strategies in a controlled environment. This approach allowed for variable manipulation and cause-and-effect analysis, which is vital for assessing ensemble credit card fraud detection solutions.
The ensemble machine learning approach combines various classifiers, each selected for its unique strengths. The classifiers used include:

- **Support Vector Machines (SVM)**: Effective in finding hyperplanes for class separation.
- **Logistic Regression (LR)**: Models event probabilities.
- **Random Forest (RF)**: Constructs robust decision trees.
- **K-Nearest Neighbors (KNN)**: Classifies based on the majority class among its nearest neighbors.
- **Bagging**: Uses KNN as its base classifier.
- **Boosting**: Uses RF as its base classifier.
- **Voting Classifier**: Integrates the predictions of multiple classifiers to improve overall prediction power.

This comprehensive ensemble of classifiers is designed to enhance the predictive capabilities of the proposed model.

2.2.1 Addressing Class Imbalance

Credit card fraud datasets typically show a significant imbalance, with fraudulent transactions representing only a small portion of the total data. Ensemble methods are particularly effective in handling such class imbalances, showing strong performance in identifying minority classes. These models also enhance interpretability and transparency in decision-making processes, which are critical in financial contexts where understanding the reasoning behind model predictions is vital. By combining multiple weak learners, ensembles improve the model's overall predictive accuracy. Furthermore, compared to deep learning architectures, ensemble methods are often less demanding computationally, making them more suitable for environments with limited computational resources.

2.2.2 Model Development Process

The development of the proposed ensemble model involved a thorough examination of different base classifiers and weighting schemes. The ensemble combined SVM, KNN, RF, Bagging, Adaboost, and a voting classifier. The SVM model was evaluated across multiple configurations, including different regularization values ranging from 1.0 to 10 and various kernels (linear, poly, and rbf). Interestingly, the highest efficiency was achieved with the default setup parameters during both the training and testing phases. Similarly, KNN and RF were examined using various values for n-neighbors (ranging from 2 to 10) and n-estimators (ranging from 10 to 100). The analysis showed that the default parameters produced the best outcomes. Bagging and Boosting algorithms, which use KNN and RF as base classifiers respectively, also performed best with their default settings. These configurations were tested on a smaller dataset, resulting in faster training and evaluation processes compared to the SMOTE dataset.

2.3. Data Pre-Processing

After selecting the dataset, the following steps were taken to prepare the data for model training and testing:

- **Handling Missing Values**: Identified and filled or removed any null values.
- **Standardizing 'Amount' Column**: Standardized the 'Amount' column to facilitate analysis.
- **Removing 'Time' Column**: Removed the 'Time' column as it did not significantly contribute to training and evaluation.
- **Removing Duplicates**: Checked for and removed duplicate entries.

The dataset had no missing values. Outliers were not explicitly handled, as the chosen machine learning model was deemed naturally resistant to them. Including outliers was considered beneficial for aligning the model with real-world scenarios.
The 'Amount' column was standardized rather than normalized because all other features (except 'Time') were results of Principal Component Analysis (PCA) and differed significantly in scale. Feature selection techniques were not used due to the lack of detailed feature information, and the 'Time' column was excluded as it only provided a sequential count of entries without temporal significance. This approach ensured the retention of all potentially relevant features.

2.4. Data Sampling

Following pre-processing, the next step involved addressing the data imbalance issue through data sampling. The dataset comprised 275,190 legitimate entries and 473 fraud-labelled entries, indicating significant skewness. Two sampling techniques were employed: under-sampling and SMOTE. The sampling process involved:

1. Separating data entries based on labels (Legit/Fraud).
2. Applying the required sampling technique to specific data.
3. Concatenating all data to create a single dataset.

2.4.1 Under-Sampling

A random sample was picked from the major class (legitimate transactions, labelled as 0). The number of random samples was made equal to the number of minority class entries (fraudulent transactions, labelled as 1). This ensured both classes had an equal number of entries, which were then concatenated into one dataset.

2.4.2 SMOTE (Synthetic Minority Over-Sampling Technique)

SMOTE is a method for increasing the number of minority class instances in a balanced manner. It generates new instances from existing minority cases. In this approach, the fraud class was oversampled to match the number of legitimate entries. The oversampled fraud class and the legitimate class were then merged to create a balanced dataset for optimal model training.

2.5. Data Splitting and Model Training

After the sampling process, the subsequent step involved splitting the data into training and testing samples. The training samples were utilized for model training and result assessment, while the testing samples were employed to evaluate how the model performs on unseen data. The 'Class' column, containing the label of each entry, was separated before the data split. The dataset was then divided into training (80% of the dataset) and testing samples (20% of the dataset).

2.5.1 Model Training

The training sample was employed for model training. Once the models were trained, evaluations were conducted on both the training and testing samples, with the results discussed in the next section.

After carefully dividing the dataset into training and testing samples, the model training phase focused on identifying patterns and relationships in the data. The chosen machine learning algorithms utilized the training samples to undertake a detailed process of learning and adjusting to the complexities of credit card transaction characteristics. The model-refining method involved iteratively modifying the intrinsic parameters of the algorithms to improve prediction accuracy.

It is crucial to emphasize that this step goes beyond simple algorithmic integration; it involves a dynamic interaction between the algorithms and the subtle details of the dataset. This mutually beneficial interaction not only enabled the extraction of complex patterns but also guaranteed the model's ability to withstand real-world challenges.

Afterward, the trained models were subjected to a thorough evaluation using both the training and the previously unused testing data. This provided a reliable assessment of their ability to generalize model outputs. The results of this phase, explained in the next section, reveal the effectiveness and flexibility of the developed models in successfully navigating the complex field of credit card fraud detection.

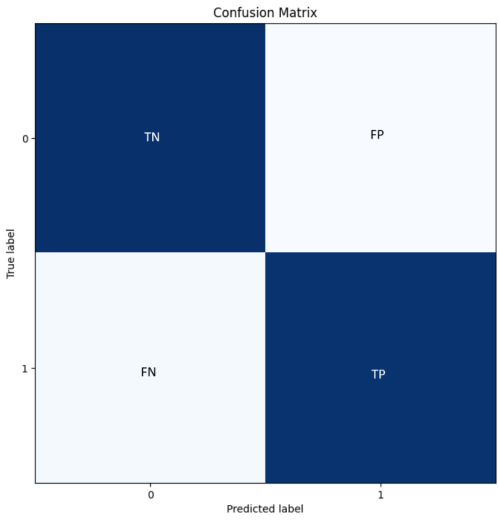

Figure 1: Visual Representation of the Confusion Matrix

Accuracy (ACC) is the ratio of all correct predictions (TP + TN) to the total number of predictions or entries in the sample (TP + TN + FN + FP). Equations 1–4 show the mathematical representation of how the Accuracy, Precision, Recall, and F1-score of a model are calculated.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision is the ratio of TP to all positive predictions (TP + FP) made by a model. In other words, it is the accuracy of the positive predictions made by the model.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall is a metric used to measure the ability of the machine learning model to identify all relevant instances of the positive class. It is the ratio of correctly predicted positive observations to the total actual positive observations.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-score is a metric used to combine the results of precision and recall into a single value. The formula of the F1-score is as follows.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

2.5.2 Performance Evaluation

The performance evaluation for both Under-sampling and SMOTE samples for each model are divulged in the sections below.

2.5.3 Undersample Results

The results of models are shown in table 1

	SVM	KNN	RF	Bagging	Boosting	P_M_1	LR	P_M_2
True Positive	336	345	378	375	378	372	350	372
True Negative	375	371	378	378	378	377	374	377
False Positive	3	7	0	0	0	1	4	1
False Negative	42	33	0	3	0	6	28	6

Table 1: Undersample training results

Concerning the results obtained in the training phase of the models, it can be observed that RF and Boosting are the best models with zero errors. At the third place, Bagging shows only 3 samples Negative classified. After that, both the proposed models (the one with SVM, P_M_1 and the one with LR, P_M_2) show the same results with only 1 False Positive and 6 False Negative. Overall those are good results but a new fitting on unseen data is necessary to avoid overfitting and underfitting and to ensure that models are correctly fitted. To assess how the models respond to unseen data, the evaluation was conducted on a testing sample of undersampled data.

	SVM	KNN	RF	Bagging	Boosting	P_M_1	LR	P_M_2
True Positive	80	82	81	81	81	81	83	81
True Negative	95	91	93	93	94	93	95	93
False Positive	0	4	2	2	1	2	0	2
False Negative	15	13	14	14	14	14	12	14

Table 2: Undersample testing results

The comparison of the results indicates that the models are optimally fitted, as the outcomes from both samples align well (further detailed in the discussion of other parameters). These findings highlight that Boosting is still a good model and regarding proposed models they are among the best models both with only 2 False Positives and 14 False Negatives. As previously noted, accuracy is defined as the ratio of correct predictions to the total number of predictions or entries in the sample. The following table presents the accuracy values of all models on both the training and testing samples.

Training Sample of Under-Sample Dataset								
	SVM	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M_1**	**LR**	**P_M_2**
Accuracy	0.9405	0.9471	1.000	0.9960	1.0000	0.9907	0.9577	0.9907
Testing sample of Under-Sample Dataset								
Accuracy	0.9210	0.9105	0.9158	0.9158	0.9210	0.9158	0.9368	0.9158

Table 3: Undersample training accuracy

The results of the confusion matrix for the training samples indicate that the accuracy (ACC) of the Random Forest (RF) and Boosting models is 100%. The accuracy of the other models, including P_M_1, P_M_2, Logistic Regression (LR), Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and bagging, are 99.07%, 99.07%, 95.77%, 94.05%, 94.71%, and 99.60%, respectively. These values are illustrated in Figure 2, which visually represents them through a bar chart. The chart uses different colors for each model, with a legend box in the bottom right corner specifying the colors and corresponding models.

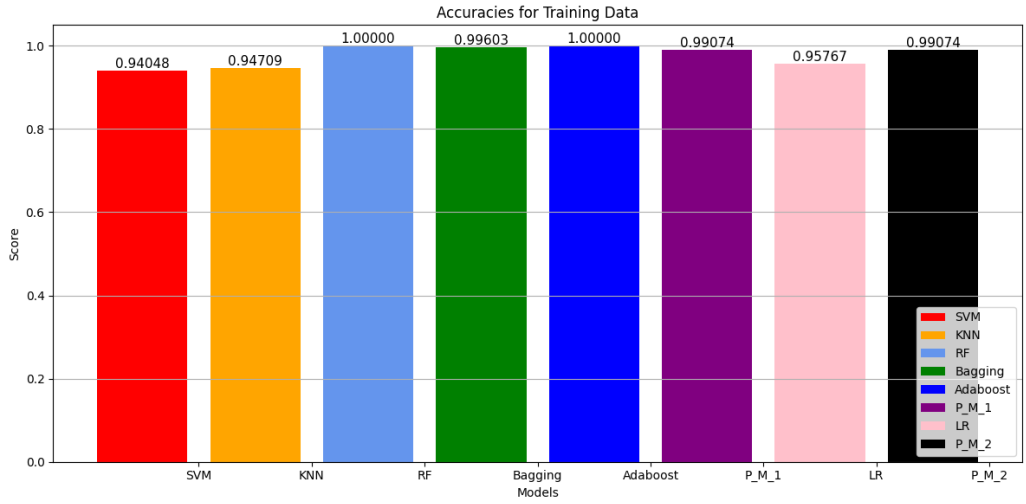

Figure 2: Comparison of accuracy in Undersample Training

In the results obtained from the testing sample, the accuracy (ACC) of LR, SVM and Boosting classifiers were the highest among all models, reaching 93.68%. Following closely were the ACC values of the P_M_1, Bagging and Random Forest (RF) classifiers, both at 91.58%. The ACC of the k-Nearest Neighbors (KNN) was the worst with 91.05%. These results are visually represented in Figure 3, which shows how the models respond to unseen data.
Precision, recall, and F1-score provide a deeper understanding of a classifier's performance beyond overall accuracy. The results of these parameters for the training sample are listed in Table 4.

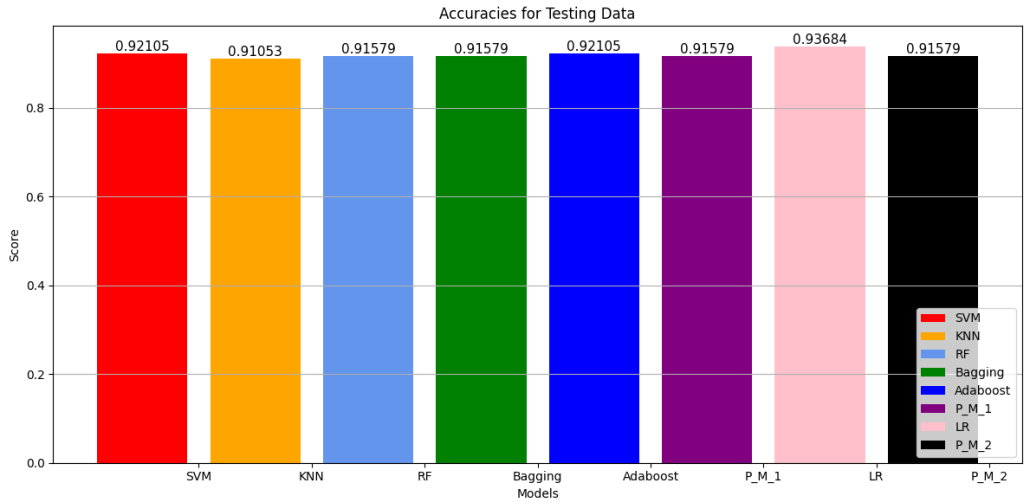

Figure 3: Comparison of accuracy in Undersample Testing

	SVM	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M_1**	**LR**	**P_M_2**
Precision	0.9912	0.9801	1.0000	1.0000	1.0000	0.9973	0.9887	0.9973
Recall	0.8889	0.9127	1.0000	0.9921	1.0000	0.9841	0.9259	0.9841
F1-score	0.9372	0.9452	1.0000	0.9960	1.0000	0.9907	0.9563	0.9907

Table 4: Undersample training performances

In line with the confusion matrix and ACC values, the RF and Boosting classifiers demonstrated 100% precision (accuracy of positive prediction) and 100% recall (ability to identify the positive class correctly). Following the best-performing classifiers are P_M_2 and P_M_1, both achieving an accuracy of 99%. Subsequently, SVM, KNN, bagging, and LR each attained results of 94-95%. The representation of these parameters is visually presented in Figure 4.

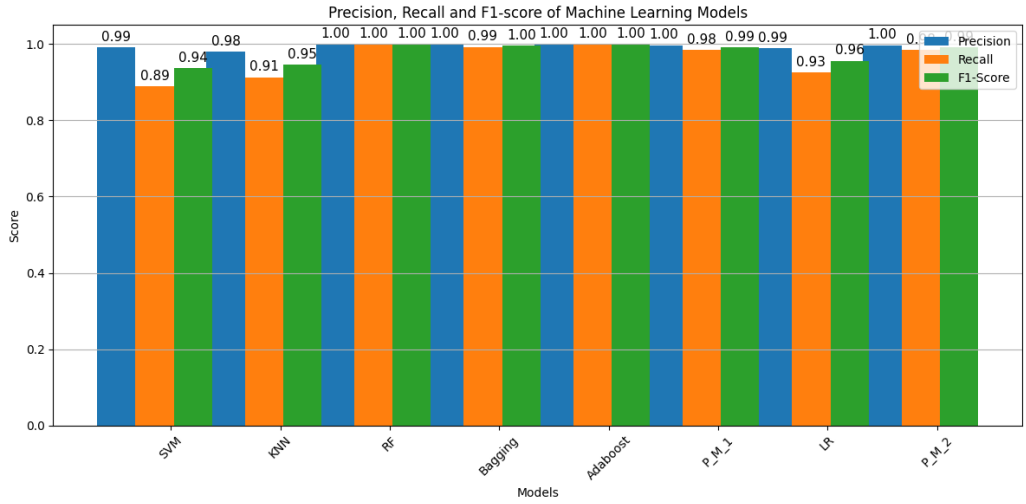

Figure 4: Comparison of performances in Undersample Training

Likewise, Table 5 presents the results for precision, recall, and F1-score for all models when applied to unseen samples of the under-sampled data.

	SVM	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M_1**	**LR**	**P_M_2**
Precision	1.0000	0.9535	0.9759	0.9759	0.0878	0.9759	1.0000	0.9759
Recall	0.8421	0.8632	0.8526	0.8526	0.8526	0.8526	0.8737	0.8526
F1-score	0.9143	0.9061	0.9101	0.9101	0.9153	0.9101	0.9326	0.9101

Table 5: Undersample testing performances

Regarding the outcomes from the testing samples, both the proposed models, SVM, RF, emerge achieving precision, recall, and an F1-score of respectively 97%, 85% and 91%. This denotes that likely all models are bad on predicting true negatives, ending in high numbers of false negatives. In comparison, SVM and LR are perfect on predicting True Positive, and have recall and F1-score nearly at 95% and 92%. At the lowest place KNN shows values of 95%, 86% and 90% for the three measures.

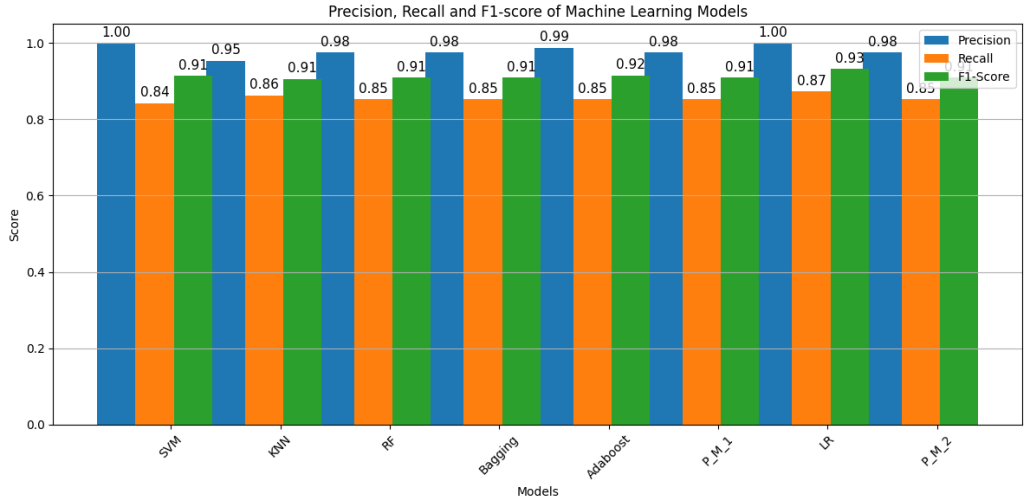

Figure 5: Comparison of performances in Undersample Testing

The ROC curve presented in Figure 6 illustrates the trade-off between the true positive rate (sensitivity or recall) and the false positive rate as the classifier's decision threshold varies. The ROC curve is generated by plotting the true positive rate (TPR) on the y-axis against the false positive rate (FPR) on the x-axis at different classification thresholds. Figure 6 displays the ROC curve for all models, with the corresponding AUC-ROC values of each model indicated in the bottom right corner of the image.

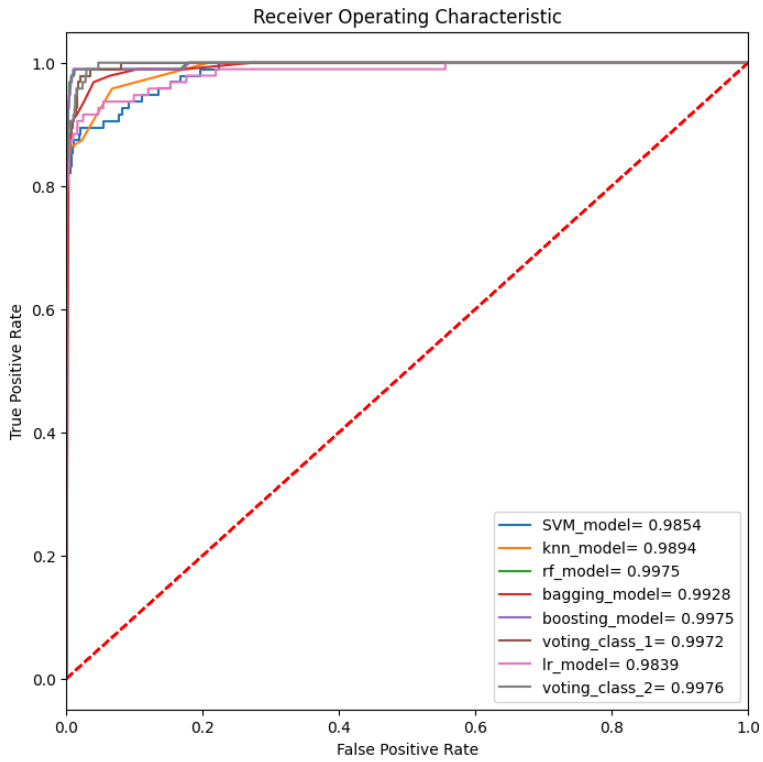

Figure 6: ROC Curve comparison

These ROC curves highlight distinct trade-offs between sensitivity and specificity across our models. AUC-ROC values provide a concise overview of the overall ability of the models to distinguish between positive and negative examples, with larger values indicating superior performance. Notably, the P_M_2 model exhibited the highest Area Under the Receiver Operating Characteristic Curve (AUC-ROC) value at 0.9976, signifying robust discriminatory capabilities. Furthermore, the Random Forest (RF), Boosting, and our other proposed model (P_M_1) emerge as strong contenders, with AUC-ROC values surpassing 0.997.

2.5.4 SMOTE Results

All training and testing results in this section are based on the SMOTE-sampled dataset. As in the previous section, we start by discussing the confusion matrix results for all models, since key parameters such as TP, FP, TN, and FN are derived from it.
Table 6 presents the confusion matrix for all models trained on the oversampled dataset. Table 7 represents instead the one related to the testing set. These confusion matrices show the prediction results for the 440,304 entries in the training sample.

	LR	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M**
True Positive	200897	220152	220152	220145	213439	220152
True Negative	214635	219866	220152	220149	217155	220136
False Positive	5517	286	0	3	2997	16
False Negative	19255	0	0	7	6713	0

Table 6: Confusion matrix for all models trained on the oversampled dataset

	LR	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M**
True Positive	50224	55038	55038	55018	53337	55038
True Negative	53683	54940	55028	54982	54262	55012
False Positive	1355	98	10	56	776	26
False Negative	4814	0	0	20	1701	0

Table 7: Confusion matrix for all models tested on the oversampled dataset

Table 8 represents the accuracies of both the training and testing set of all proposed models after applying SMOTE.

Training Sample of SMOTE Dataset						
	LR	**KNN**	**RF**	**Bagging**	**Boosting**	**P_M**
Accuracy	0.9437	0.9993	1.0000	0.9999	0.9779	0.9999
Testing sample of Under-Sample Dataset						
Accuracy	0.9440	0.9991	0.9999	0.9993	0.9775	0.9997

Table 8: Table with accuracies of testing and training sets

Figures 7 and 8 demonstrate how all the models perform on the training and testing set. Results on unseen data highlight how the proposed model (PM) alongside with the Random Forest demonstrate high accuracy with both converging at nearly 100% of accuracy. This outcome demonstrates the models' proficiency in understanding the underlying patterns within the training data, leading to predictions that closely match the true labels. Logistic regression performes worst by obtaining an accuracy of 94% both with the training and testing sets. This indicates a comparatively higher rate of misclassification than other models. However, the consistent presence of these patterns in the training sample underscores the stability and generalizability of our proposed model.

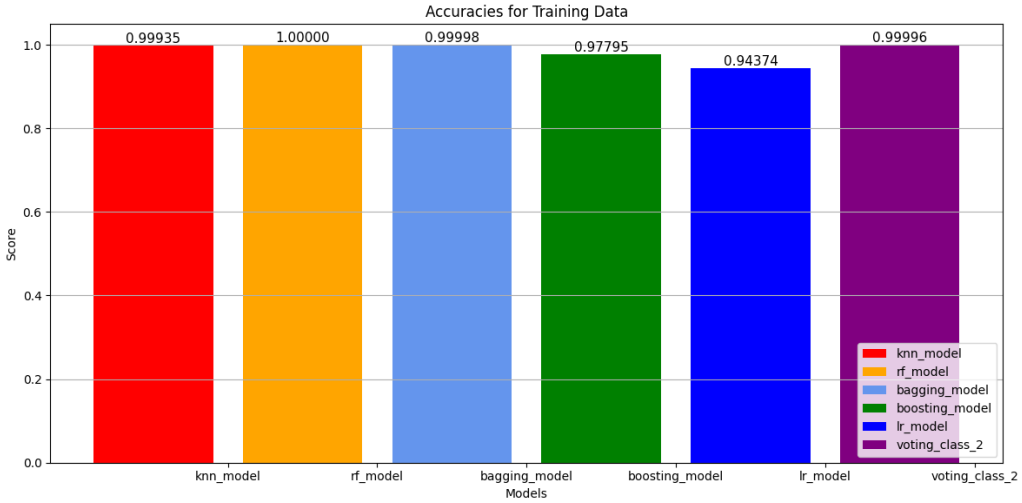

Figure 7: Comparison of training accuracy with SMOTE

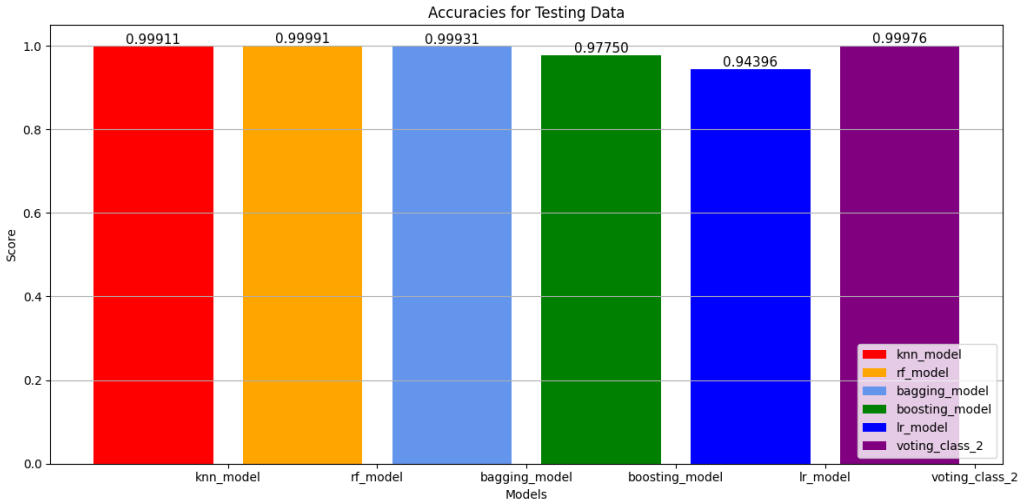

Figure 8: Comparison of testing accuracy with SMOTE

In a comparative analysis of the confusion matrix results, we observe a consistent trend in precision, recall, and accuracy metrics across our models for both the training and testing samples. Notably, most models achieve precision, recall, and accuracy scores close to 100 percent, showcasing their effectiveness in accurately classifying positive instances.
The results and visual representations clearly show that these models effectively capture relevant data, resulting in precise predictions and minimal false negatives. However, the Logistic Regression (LR) model, while still performing well, exhibits slightly lower precision, recall, and accuracy scores from 91.0% to 97.0%. This slight difference underscores the LR classifier's sensitivity to specific data complexities, while also affirming the overall robustness and high performance of the other classifiers. These results are visible in figure 9 for the testing set.

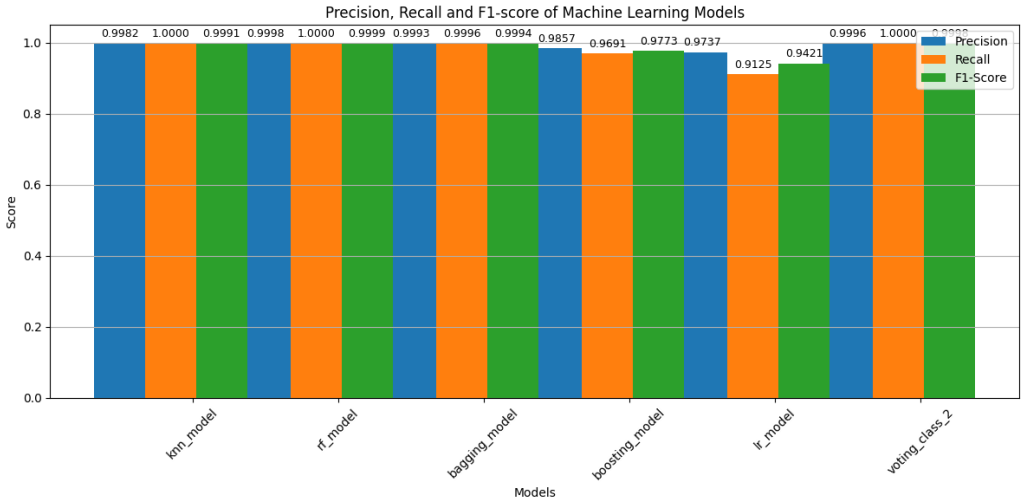

Figure 9: Precision, recall and f1-score with SMOTE (testing)

2.6. Performance

Computational efficiency in Machine Learning pertains to the time an algorithm requires for training and evaluation. In order to measure the training and evaluation durations, the Python-auto time extension was employed. The collected data revealed variations in training and testing times among the different algorithms. Notably, some algorithms, such as LR and KNN, exhibited longer training times but shorter testing times, while others demonstrated the opposite trend. Time results for undersampling can be seen in Figure 10

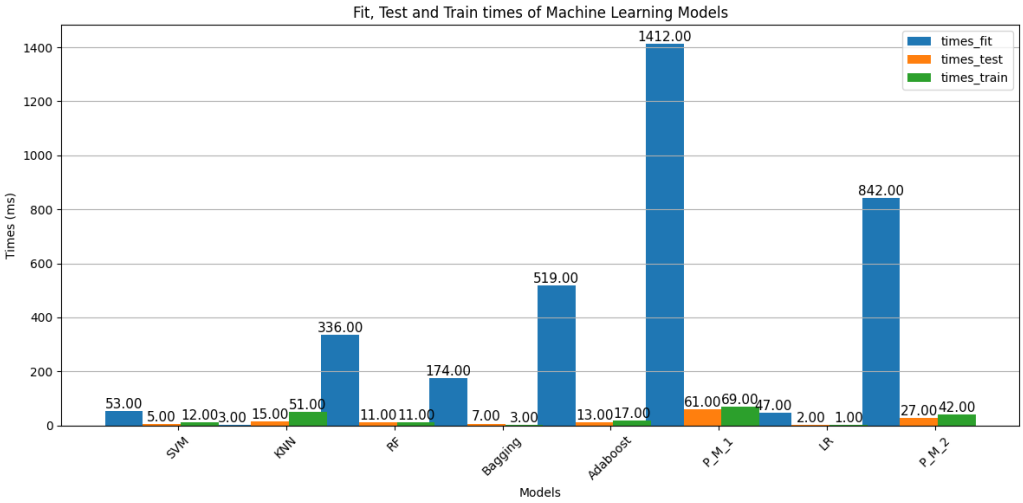

Figure 10: Time Performance comparison (undersampling)

As shown in Figure 11, there is a huge difference in the amount of time to train and test the models due to the number of samples in SMOTE. Some algorithms such as Random Forest, Bagging and the Proposed Model present higher training times in comparison with KNN that shows striking times in both testing on training and testing samples. On the other hand, all the times in SMOTE are quite relevant, highlighting training times for the Proposed Model with 861332ms and Boosting (Adaboost) with 602825ms.

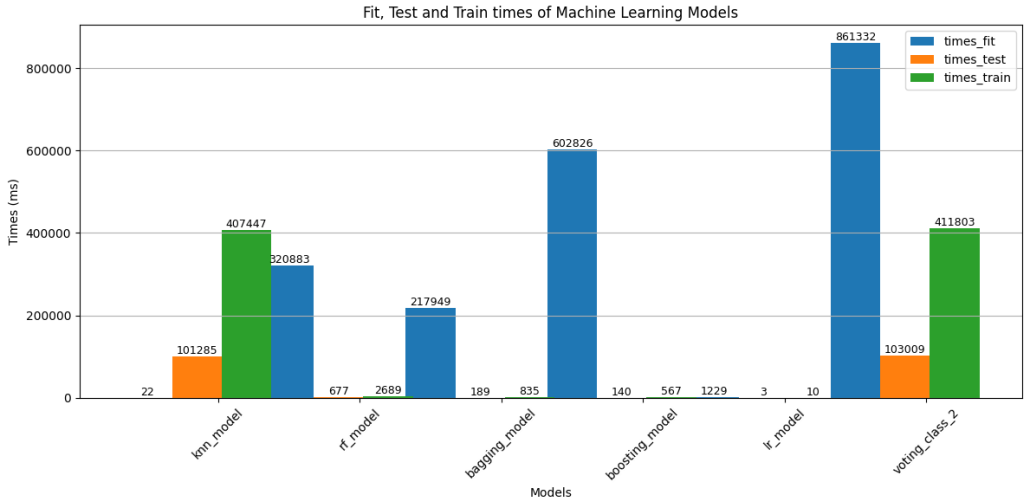

Figure 11: Time Performance comparison (SMOTE)

2.7. Limitations and challenges

While Google Colab provided a convenient implementation environment, it also presented unique challenges. Operating on a free cloud platform required careful monitoring due to its sensitivity to extended usage periods, network outages, and minor lapses in active engagement. It was crucial to manage these issues to prevent accidental disconnections and process restarts. Another challenge related to Colab was the execution time. By using the free platform the resource were quite limited and it took some hours to execute the whole code. So we had to be careful not to disconnect from internet otherwise we had to restart the whole execution from the beginning.
For this reason, we ended up not using Colab, which was suggested by the paper, anymore, Since it took more or less three hours for the whole execution. We decided to use other tools like github Codespaces that provided us a free cloud platform with more resources than Google colab. Enabling us to finish the execution of the whole code in less than an hour.

3. Conclusions

The increase in identity theft, especially through credit card fraud, has caused significant financial losses and emotional suffering for numerous victims. Statistics from organizations like the Federal Trade Commission (FTC) paint a troubling picture of the constantly evolving fraud landscape. To address these challenges, we examined various fraud detection techniques, including Statistical Analysis, Machine Learning, and Deep Learning Methods, to identify suspicious patterns in transaction data.
For classification tasks, a range of machine learning models, such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF), Bagging, and Boosting, emerged as effective tools. In this report, we thoroughly evaluated these models' effectiveness using a real-world dataset of European credit card transactions. This evaluation led to the proposal of an ensemble model that combines SVM, KNN, RF, Bagging, and Boosting classifiers within a voting framework. This ensemble demonstrated strong performance and highlighted the benefits of combining multiple classifiers to improve fraud detection accuracy. During the evaluation process, the models underwent rigorous testing, with their performance assessed using various metrics, including precision, recall, F1-score, ROC, and accuracy.
The results confirmed the effectiveness of our ensemble model in reducing false positives and false negatives, which are two critical challenges in credit card fraud detection. Balancing accuracy and computational efficiency emerged as a crucial consideration. Our analysis of computational efficiency revealed various algorithms' trade-offs between training and testing times. The investigation into computational efficiency also emphasized our model's performance, measured through training and testing time and memory usage.

Future research should focus on improving the model's efficiency. Despite the positive results from both our ensemble model and individual predictors, there is a strong interest in refining their training and testing durations. Reducing computational overhead could develop fraud detection systems capable of real-time operation, providing rapid responses to evolving fraud trends. Even if it's not the topic of the study, exploring the integration of deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), with traditional machine learning methods, may result in more accurate and adaptable fraud detection solutions.

Additionally, there is a need to explore dynamic data sampling strategies that adapt to changes in data distribution over time. This analysis is crucial for credit card fraud detection, as fraudulent activity patterns may change, and a model that can adapt to these changes is more likely to remain effective. The paper we refereed to also suggests further investigation into methods aimed at enhancing the proposed model's resilience against novel or adversarial attacks. Adversarial attacks can exploit vulnerabilities in machine learning models, and exploring techniques to mitigate these risks would be highly beneficial. Finally, future research should assess the model's scalability in handling larger datasets and meeting increasing computational demands. This could involve using parallel processing or distributed computing approaches to ensure efficient processing as dataset sizes grow.