Credit Card Fraud Detection Using Machine Learning

-

Exemption Assignment, February 2025

Vezalis Petros
University of Macedonia
ics22106@uom.edu.gr

Abstract

Credit card fraud detection is a critical challenge in the financial sector, requiring robust and accurate methods to identify fraudulent transactions amidst highly imbalanced datasets. This project explores machine learning techniques to address this issue using the Credit Card Fraud Detection dataset. The study involves data preprocessing, feature selection, the implementation of various classification algorithms, including Logistic Regression, Random Forests, Support Vector Machines, Artificial Neural Networks and hyperparameter optimization. Performance evaluation is conducted using metrics such as precision, recall, F1-score, and AUC-ROC. The findings demonstrate the strengths and limitations of different models, offering practical recommendations for deploying effective fraud detection systems.

Contents

1 Introduction to Credit Card Fraud Detection

Credit card fraud detection has become a vital area of study as online transactions grow in volume and complexity. This section explores the problem statement and objectives of this study, highlighting the need for efficient, accurate systems to identify fraudulent transactions.

An overview of the dataset used for analysis will be provided, detailing its key characteristics and relevance. Additionally, the challenges associated with fraud detection, including class imbalance, real-time detection, and feature selection, will be discussed to frame the importance of machine learning techniques in tackling this critical issue.

1.1 Problem Statement and Objectives

Credit card fraud poses significant financial and reputational risks to businesses and individuals, necessitating robust detection systems. The primary challenge lies in identifying fraudulent transactions from legitimate ones in real-time, as fraudulent activity often accounts for a minute proportion of all transactions, creating a heavily imbalanced dataset. The objective of this study is to design and evaluate machine learning models that can accurately classify transactions as fraudulent or non-fraudulent. This includes preprocessing data to handle imbalances, implementing various classification algorithms, and optimizing their performance. The ultimate goal is to develop a system that achieves high precision and recall, ensuring both reliability and scalability for practical deployment.

1.2 Dataset Overview

The Credit Card Fraud Detection Dataset provides anonymized transaction data for the purpose of identifying fraudulent credit card transactions. This dataset, originally published by researchers *Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi*, contains credit card transactions made by European cardholders over a two-day period in September 2013. The dataset is accessible from Kaggle .

The dataset includes 284,807 transactions, with only 492 labeled as fraudulent, reflecting the inherent imbalance in fraud detection tasks. Each transaction is described by 31 features, with 28 anonymized numerical features (V1,..V28) , one time-based feature, the transaction amount and the class which is the target variable.

The dataset's transaction amount distribution as visualized in Figure 1 highlights the wide range of amounts, with the majority concentrated on smaller values.

This observation is important as fraudulent transactions may show distinct patterns compared to legitimate ones in this feature. The second plot illustrated in Figure 2 shows the distribution of the anonymized numerical features (V1 to V28), which vary widely in range and scale. These features, generated using PCA, do not have a straightforward interpretation but may hold valuable information for distinguishing between fraudulent and legitimate transactions.

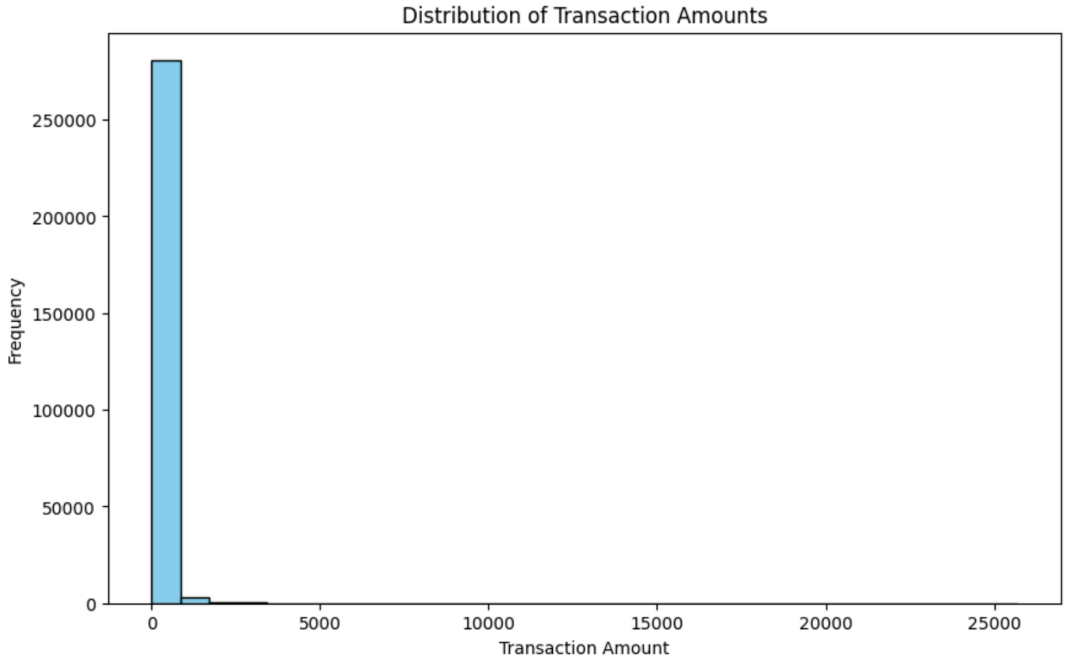

Figure 1: Distribution of Transaction Amounts upon the dataset

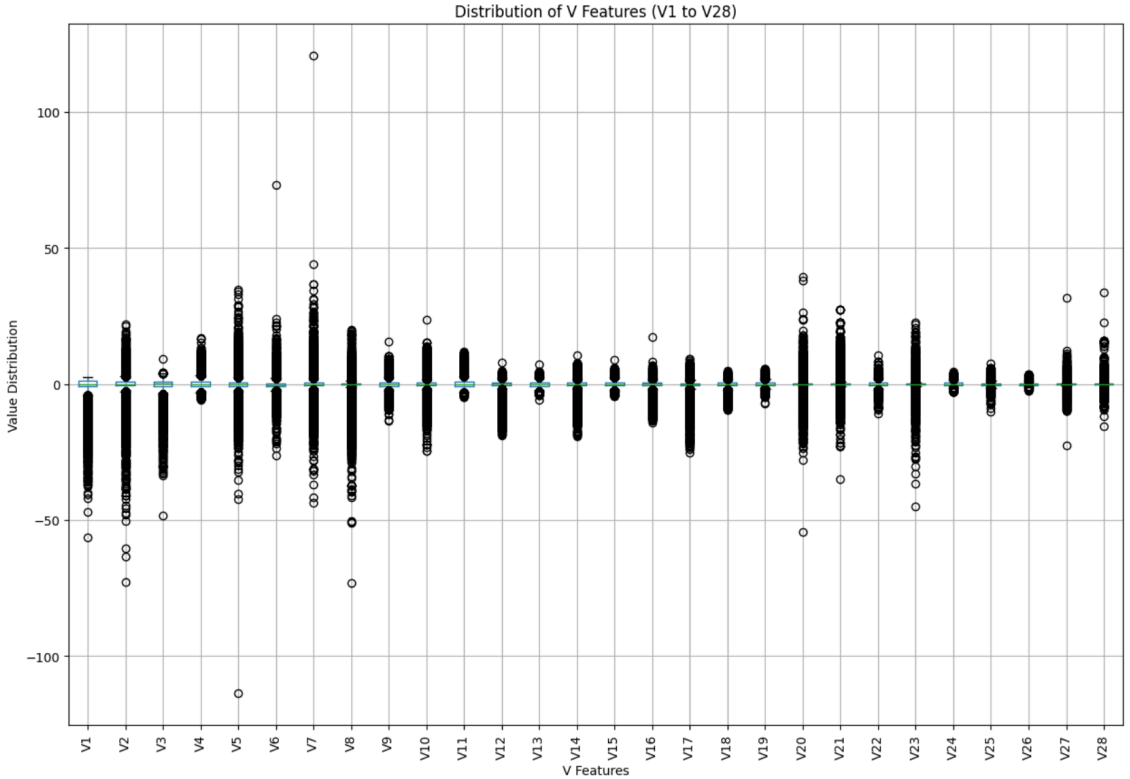

Figure 2: Reveals the diverse range and scale of the anonymized numerical features (V1 to V28)

To further understand the dataset, I analyzed the class distribution (fraudulent vs. non-fraudulent transactions) in Figure 3. The dataset is highly imbalanced, with fraudulent transactions comprising only a tiny fraction. This imbalance poses a challenge for machine learning models and requires careful handling to ensure effective fraud detection.

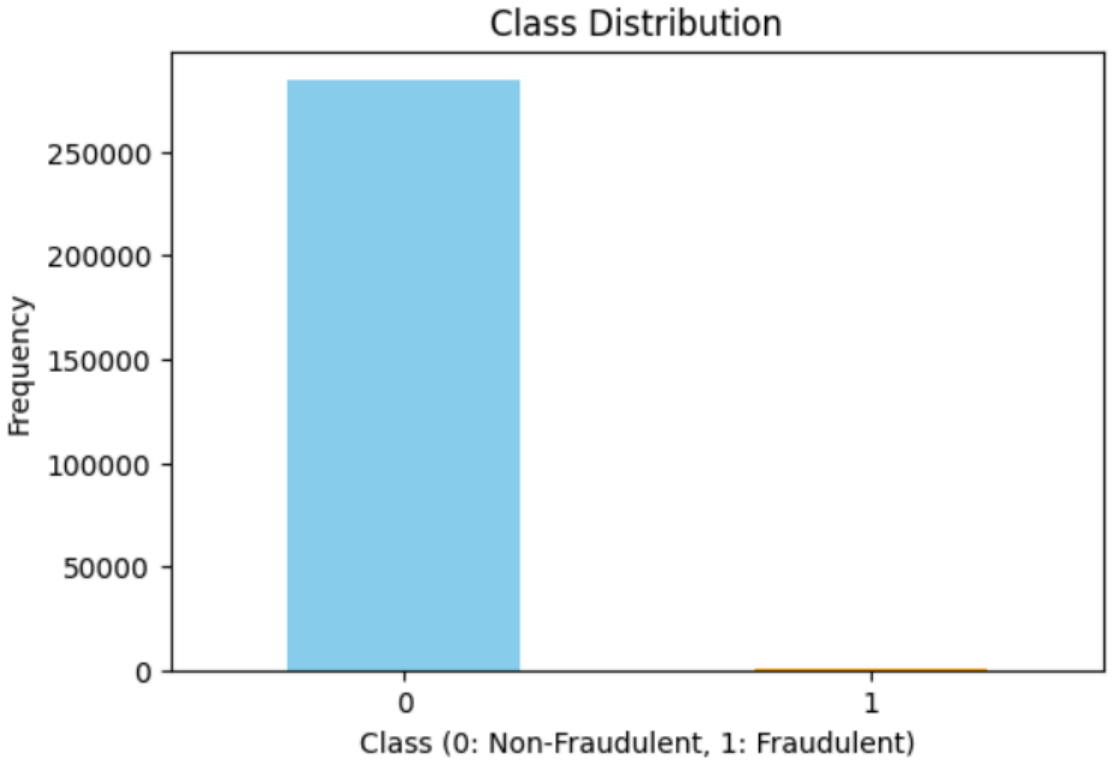

Figure 3: Distribution of Class (0: Non-Fraudulent , 1: Fraudulent)

As we see the dataset contains 284,807 transactions. Only 492 transactions (about 0.17%) are labeled as fraudulent, making this a highly imbalanced dataset.: The vast majority of records (284,315) represent legitimate transactions.With less than 1% of the data representing fraud, it presents an ideal setting for testing models designed to handle class imbalance

I also generated a correlation heatmap to examine relationships between features in Figure 4 . Strong correlations (positive or negative) can indicate redundancy or informative relationships, helping to refine feature selection and improve model performance. Most features have very weak correlations with each other (light blue), suggesting limited redundancy. However, a few notable correlations (e.g., between features such asV7, V20 , and the Amount variable) may indicate potential influence or association with fraudulent activity. These observations can guide further feature selection and preprocessing steps to enhance model performance.

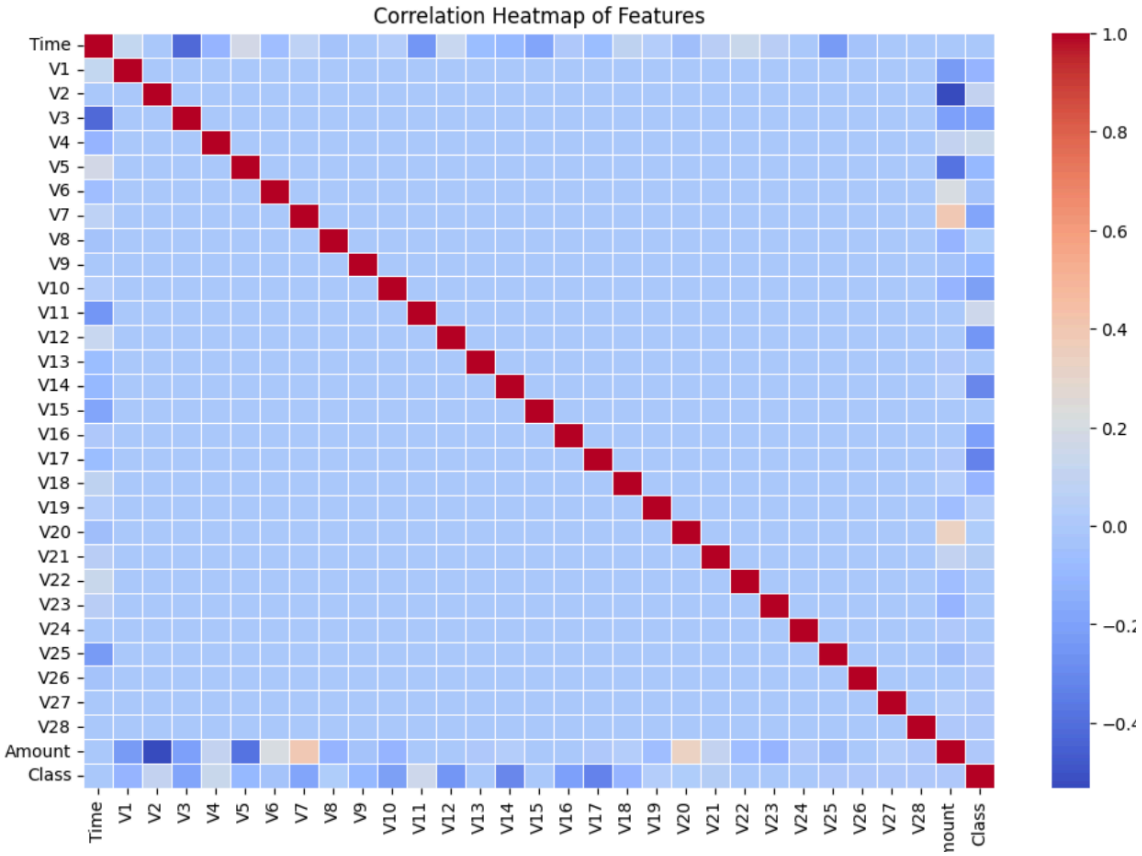

Figure 4: Correlation HeatMap of Features

1.3 Challenges in Fraud Detection

Fraud detection is a challenging task due to several key factors. First, the extreme class imbalance in datasets, where fraudulent transactions constitute a tiny fraction of the total, can cause machine learning models to focus on the majority class and overlook fraud cases. This necessitates the use of techniques like oversampling , undersampling or biasing the models to pay more attention to the minority class as I did in the next sections. Second, feature selection and engineering become difficult with anonymized datasets, as the lack of explicit meaning for features like V1 to V28 requires advanced methods to identify the most informative attributes. Third, the requirement for real-time detection adds another layer of complexity, as systems must not only be accurate but also computationally efficient to process large volumes of data instantly. Lastly, managing false positives is critical , flagging legitimate transactions as fraudulent can cause user dissatisfaction and operational inefficiencies, making it essential to strike a balance between precision and recall in model performance.

2 Data Preprocessing and Exploration

Effective data preprocessing and exploration are crucial steps in building a robust fraud detection system. In this section, we will begin with an exploratory data analysis (EDA) to gain valuable insights into the dataset's structure and characteristics. This includes visualizing feature distributions, detecting outliers, and examining correlations to guide feature selection and engineering. Next, we will address data cleaning, ensuring that the dataset is free of missing values and normalized to handle any inconsistencies. Finally, we will tackle the challenge of class imbalance by leveraging class weights during the training process, ensuring the model gives appropriate importance to the minority class (fraudulent transactions). Together, these steps will lay a solid foundation for effective model training and evaluation.

2.1 Exploratory Data Analysis (EDA)

For the Exploratory Data Analysis, we delve deeper into the characteristics of fraudulent versus non-fraudulent transactions. By comparing the minimum, maximum (Figure 5 and Figure 6) , and mean values across features for these two classes (Figure 7), clear distinctions emerge that may guide model training. Visualizations, such as line plots, showcase how certain features behave differently for fraudulent transactions, providing insights into feature importance.

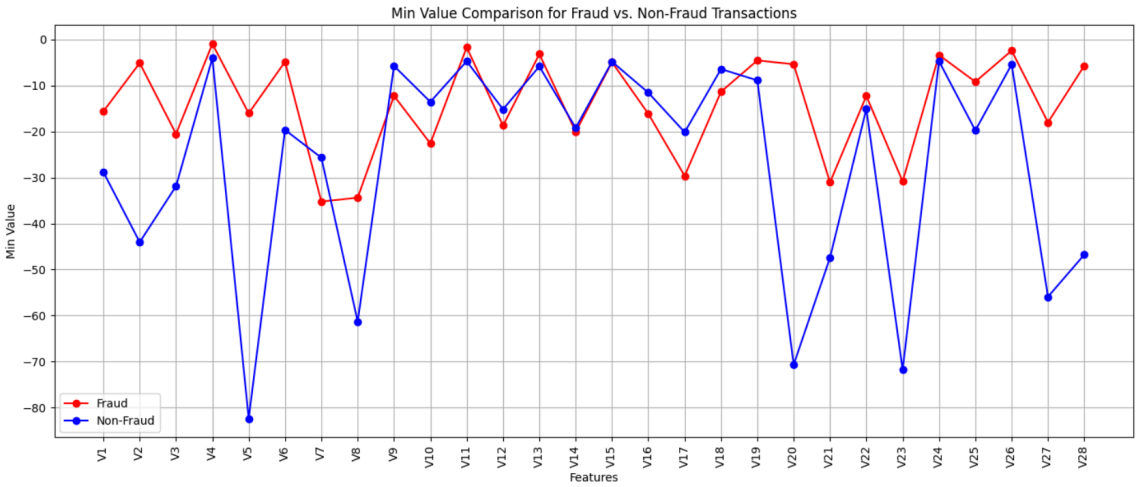

Figure 5: Min Value Comparison for Fraud vs Non-Fraud Transactions

As we see from Figure 5 significant differences are observed in some features (e.g., V5, V8, and V23), where the minimum values for non-fraud transactions are considerably lower than those for fraud. Such features may indicate potential discriminatory power for fraud detection. Other features, like V1, show minimal variation, suggesting less relevance for classification based on minimum values.

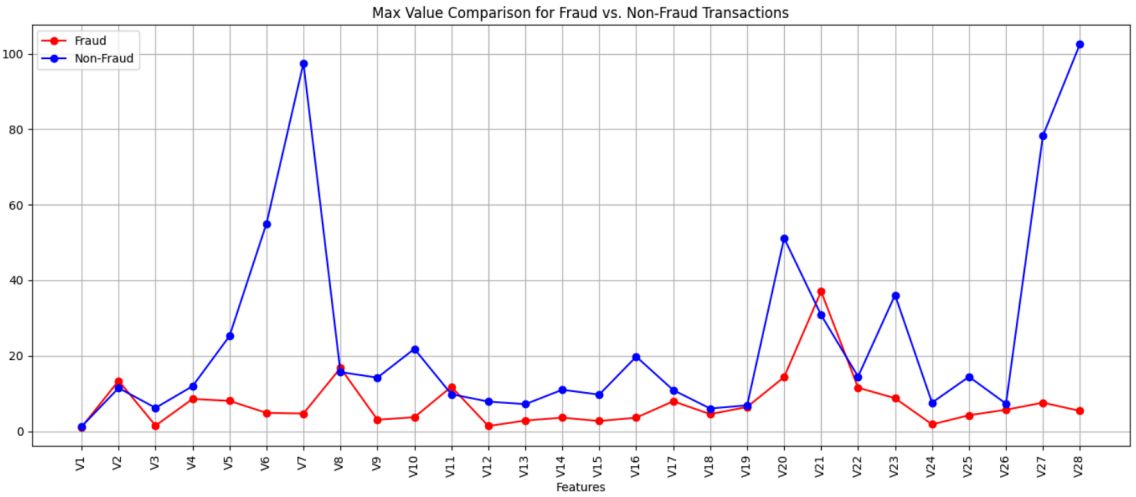

Figure 6: Max Value Comparison for Fraud vs Non-Fraud Transactions

Figure 6 diagram shows the maximum values of features for fraud and non-fraud transactions. Non-fraud transactions often have higher maximum values, especially in features like V27 and V28, while fraudulent transactions exhibit tighter ranges in most features. This pattern suggests that some features might provide critical thresholds for detecting fraudulent transactions.

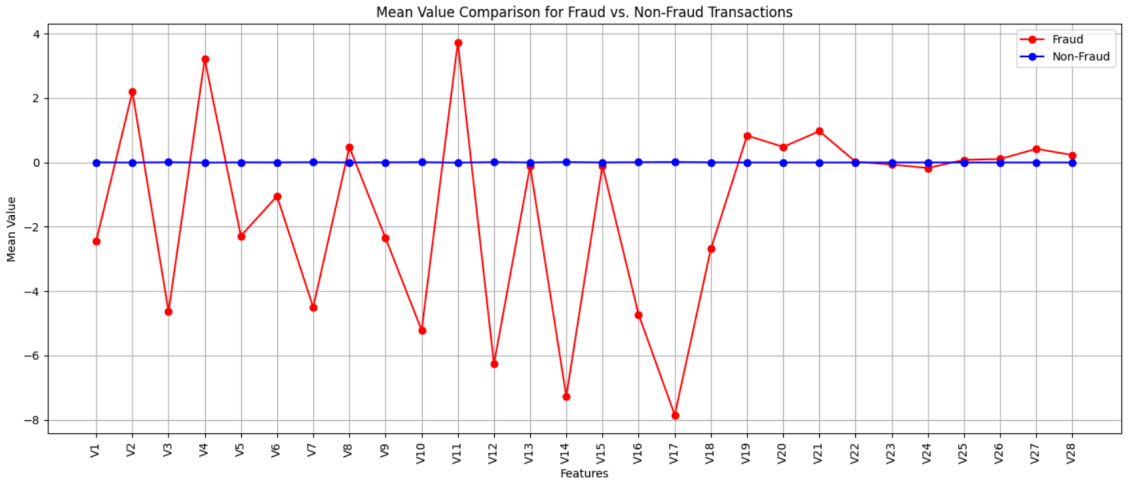

Figure 7: Mean Value Comparison for Fraud vs Non-Fraud Transactions

Fraudulent transactions have distinctly lower mean values in features like V14, V17, V12, and V16, which may serve as strong indicators for classification. Meanwhile, non-fraudulent transactions display a more uniform mean distribution across feature

Furthermore, density plots for the anonymized features (V1 to V28) highlight distributional differences between the classes (Figure 8). These plots reveal

patterns that could aid the model in distinguishing fraudulent from legitimate transactions. Features with starkly differing densities across classes are particularly promising for classification.

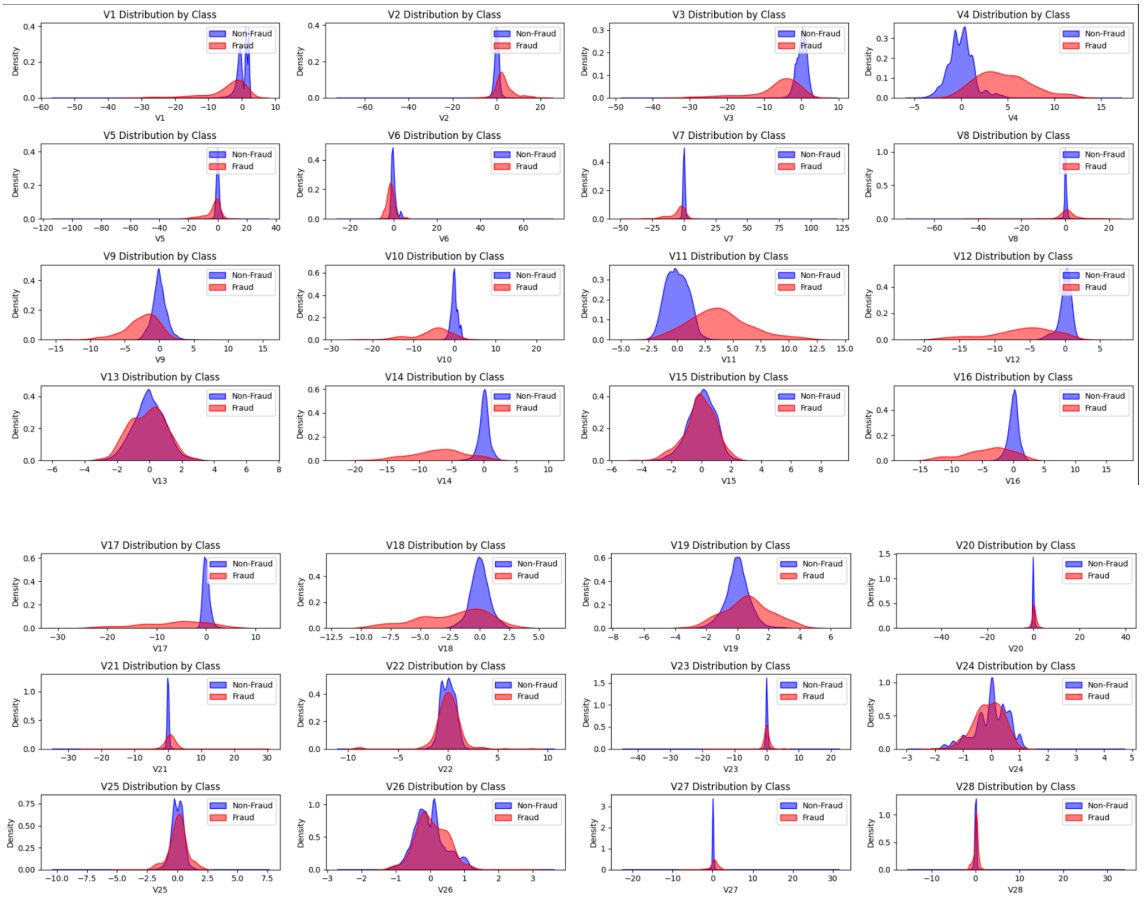

Figure 8: Destiny plots for features (V1-V28)

The density plots from Figure 8 provide a comparative view of feature distributions between fraudulent and non-fraudulent transactions, revealing valuable insights into the dataset's structure. Certain features, such as V4, V10, V12, and V14, V17 exhibit clear distinctions between the two classes, making them strong candidates for identifying fraudulent transactions. Conversely, features like V5, V9, and V13 show significant overlap, indicating limited discriminatory power when used individually. Additionally, features such as V7, V8, and V28 are highly concentrated, particularly for non-fraudulent transactions, while skewed features like V20 and V23 highlight specific patterns unique to fraud. These observations suggest that features with distinct class separation should be prioritized during feature selection, while overlapping features might require the use of advanced modeling techniques to capture non-linear relationships. Overall, these density plots emphasize the importance of feature analysis in building an effective fraud detection model.

2.2 Cleaning and Handling Missing Data

To ensure data quality and consistency, we first checked the dataset for missing values. Fortunately, no missing values were found, allowing us to proceed directly to feature standardization. To improve the model's performance, all numerical features (excluding the Time variable) were standardized by removing the mean and scaling to unit variance. This preprocessing step ensures that all features are on the same scale, preventing certain features from dominating the learning process due to their larger magnitude. By excluding the Time variable from standardization, we preserve its original temporal meaning while focusing on the other features for machine learning tasks.

2.3 Addressing Imbalanced Classes

Handling the severe class imbalance in the dataset is critical for building effective fraud detection models. In this project, I employ K-Fold Cross-Validation, a robust method for splitting the data into multiple training and testing subsets to ensure reliable performance evaluation across all data samples. During training, we compute sample weights using the *compute_sample_weight* function from scikit-learn. These weights help balance the contributions of the minority class (fraudulent transactions) and majority class (non-fraudulent transactions) during the model's learning process. By combining K-Fold Cross-Validation with class weights, we address the imbalance issue while also obtaining a more generalized evaluation of model performance. This approach ensures that the model learns to identify fraudulent transactions without being biased toward the majority class.

3 Dimensionality Reduction and Feature Selection

Dimensionality reduction and feature selection are essential steps for improving the efficiency and interpretability of machine learning models. High-dimensional data can lead to increased computational costs, overfitting, and challenges in model interpretation. In this section, we explore techniques to reduce the feature set while retaining the most informative attributes

3.1 Feature Importance and Correlation Analysis

In this analysis, we used a Random Forest model to determine feature importance, identifying which attributes of the dataset contribute the most to

classifying transactions as fraudulent or non-fraudulent. The feature importance scores were calculated based on the reduction in impurity achieved by splitting on each feature during training. This method highlights the most critical predictors as we see in Figure 9, such as V17, V14, and V12, which strongly influence the model's decisions. Features with minimal importance, such as V28 and V25, may be excluded in subsequent steps to streamline the model and reduce computational overhead. This step provides a clear roadmap for refining the feature set and improving the model's efficiency and interpretability.

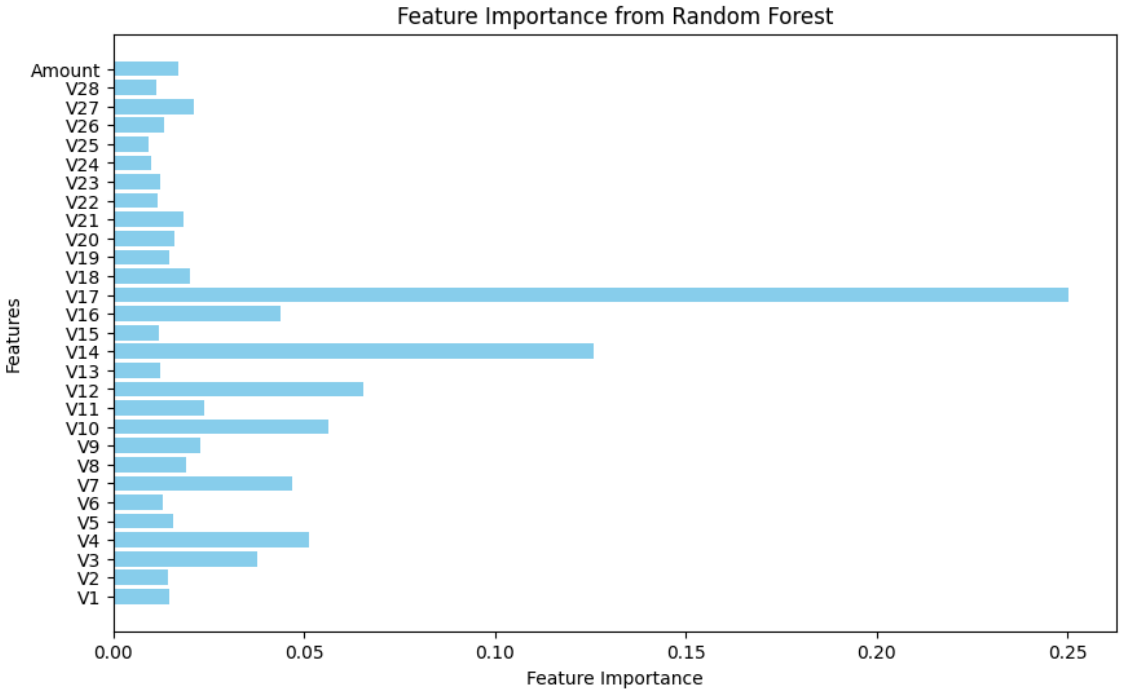

Figure 9: Feature Importance from Random Forest for features (V1-V28)

A correlation analysis was also conducted using the heatmap we built in the Dataset Overview section before (Figure 4) , to identify highly correlated features that could contribute to redundancy in the dataset. This analysis aimed to detect pairs of features with a strong linear relationship (|correlation| > 0.9) to consider removing one of the pair. However, no highly correlated feature pairs were identified, indicating minimal redundancy among the features. This result suggests that the dataset's features are largely independent, reducing the risk of overfitting and ensuring that all features provide unique information for the model. As a result, no features were removed based on correlation analysis.

3.2 Principal Component Analysis (PCA)

In this analysis, we applied Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while retaining as much variance as possible. The

Cumulative Explained Variance Plot illustrates the proportion of variance captured by the principal components as shown in Figure 10. This step was critical for determining how many components are sufficient to explain the majority of the dataset's variability. The plot shows that retaining the first 21 components captures approximately 95% of the total variance, significantly reducing the dimensionality from the original 29 features. This reduction simplifies the dataset, reduces computational complexity, and can help prevent overfitting, particularly in high-dimensional data.

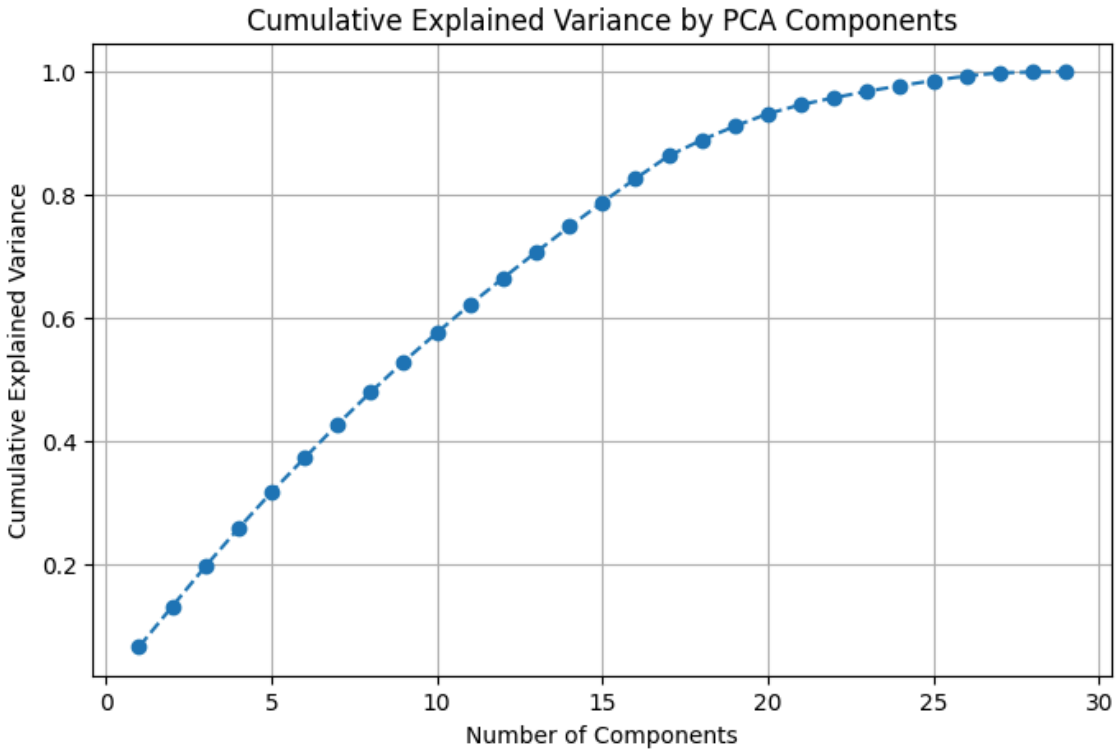

Figure 10: Cumulative Explained Variance by PCA Components

From the cumulative variance plot, we observe that the initial components contribute more to the total variance, while the later components add diminishing amounts. This suggests that the dataset contains underlying patterns that can be captured using fewer dimensions, which is the goal of PCA.

Following this, we visualized the PCA-transformed dataset in both 2D and 3D spaces using the first two and three principal components, respectively. These visualizations help to understand the data's structure and any inherent separation between fraudulent and non-fraudulent transactions.

- 2D PCA Projection: The 2D plot illustrated in Figure 11 projects the dataset onto the first two principal components. While most data points cluster densely, representing non-fraudulent transactions, a few outliers

corresponding to fraudulent transactions can be observed. However, the overlap between the two classes indicates the challenge of distinguishing fraudulent transactions based solely on these two components.

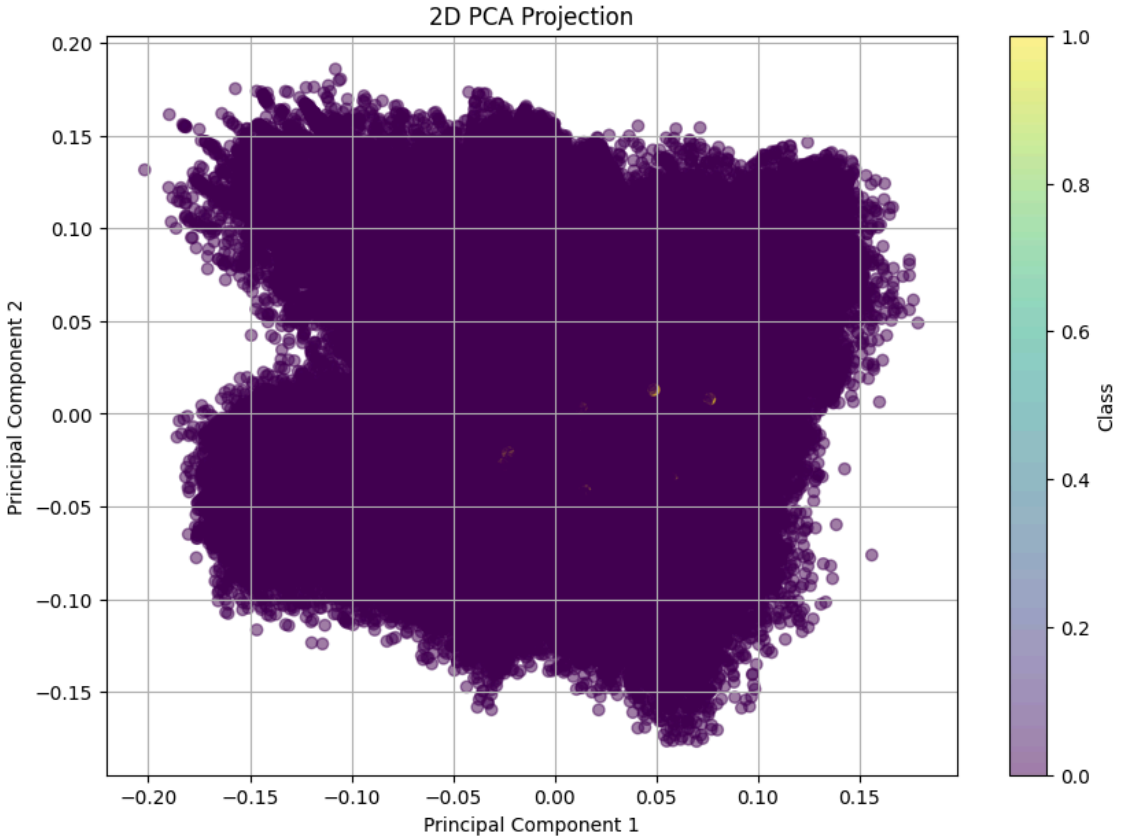

Figure 11: 2D PCA Projection

- 3D PCA Projection: The 3D plot illustrated in Figure 12 expands the analysis to include the third principal component. This additional dimension provides slightly more insight into the dataset's spread, but the general structure remains similar to the 2D projection. The dense clustering of the majority class (non-fraudulent transactions) and the dispersed outliers (fraudulent transactions) further highlight the difficulty of separating these classes.

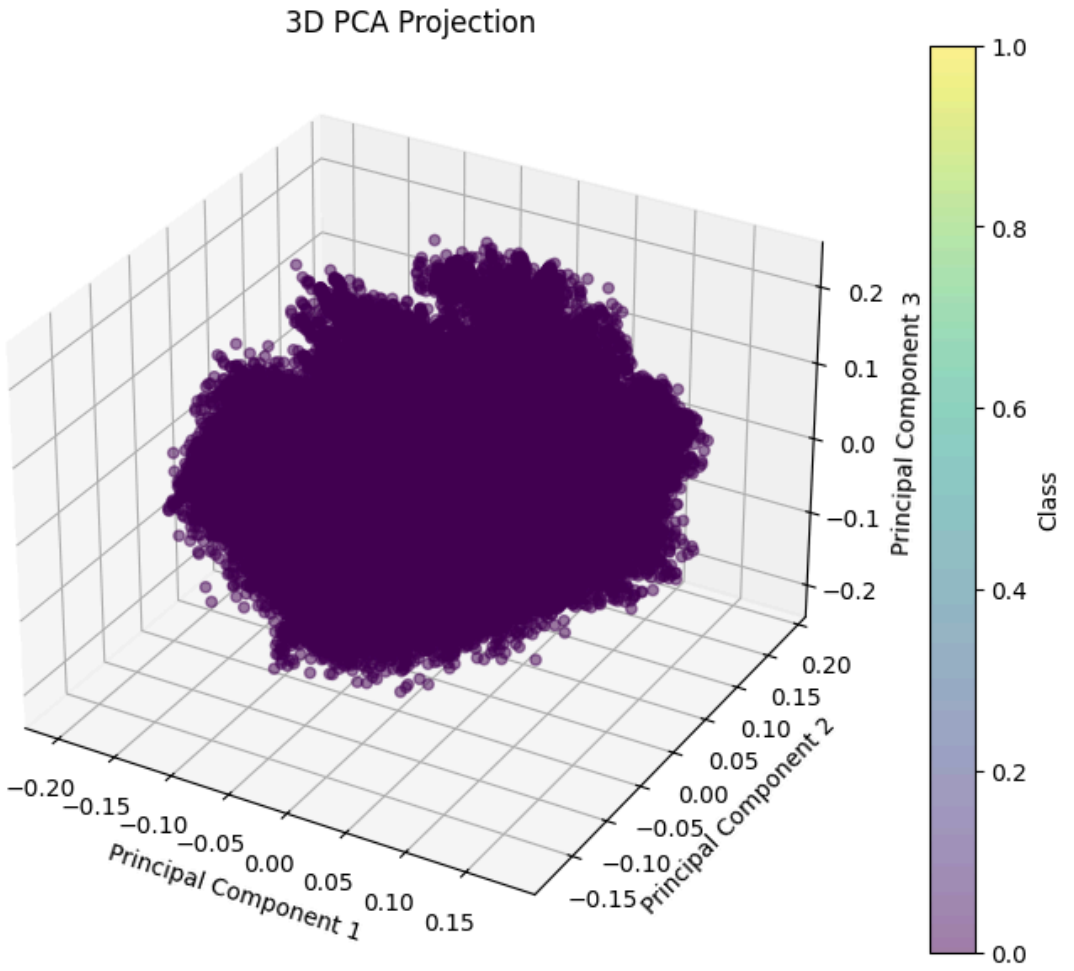

Figure 12: 3D PCA Projection

We also conducted an experiment to compare the performance of a Random Forest Classifier on the original dataset and the PCA-transformed dataset, aiming to evaluate the impact of dimensionality reduction on classification performance. Using 5-fold Stratified Cross-Validation and applying class weights to handle the imbalance, we observed based on Table 1 that the classifier achieved excellent results on the original dataset, with accuracies consistently around 99.94% to 99.95% and strong minority class metrics, including precision ranging from 0.93 to 0.99 and recall between 0.72 to 0.77. On the PCA-transformed dataset, which reduced the dimensionality from 29 features to 21 components while retaining 95% of the variance, the accuracy dropped slightly to 99.92% to 99.93%. Precision for the minority class remained high, ranging from 0.94 to 1.00, but recall decreased to 0.55 to 0.60, leading to lower F1-scores for the minority class.

This experiment highlights the trade-off introduced by PCA: while it simplifies the dataset and reduces computational complexity, it can result in a slight loss of critical information affecting minority class performance. Overall, the results underline the importance of balancing dimensionality reduction with the

preservation of key patterns in the dataset, especially for challenging tasks like fraud detection.

Fold	Dataset	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	Original Dataset	0.9994	0.94	0.73	0.82
2	Original Dataset	0.9995	0.99	0.74	0.84
3	Original Dataset	0.9995	0.97	0.71	0.82
4	Original Dataset	0.9994	0.93	0.72	0.82
5	Original Dataset	0.9995	0.94	0.77	0.84
1	PCA-Transformed Dataset	0.9992	0.98	0.55	0.70
2	PCA-Transformed Dataset	0.9992	0.98	0.58	0.73
3	PCA-Transformed Dataset	0.9992	1.00	0.55	0.71
4	PCA-Transformed Dataset	0.9993	0.97	0.60	0.74
5	PCA-Transformed Dataset	0.9992	0.94	0.60	0.73

Table 1: Original Dataset - PCA Transformed Dataset metrics comparison with Random Forest Classifier

3.3 Recursive Feature Elimination (RFE)

In this analysis, we implemented Recursive Feature Elimination (RFE) to identify the most important features that contribute to fraud detection. RFE iteratively removes the least important features and fits the model multiple times, ranking the features based on their importance to the classifier's performance. Using a Random Forest classifier, we selected the top 20 features out of 29, reducing the dimensionality of the dataset while retaining the most informative features. After selecting these features, the Random Forest model was retrained on the reduced feature set, and the feature importances of the selected features were visualized. This process ensures that only the most relevant features are used, potentially improving model performance and reducing computational complexity. Additionally, the rankings of all features were plotted in Figure 13, providing insights into which features were deemed less impactful during the selection process.

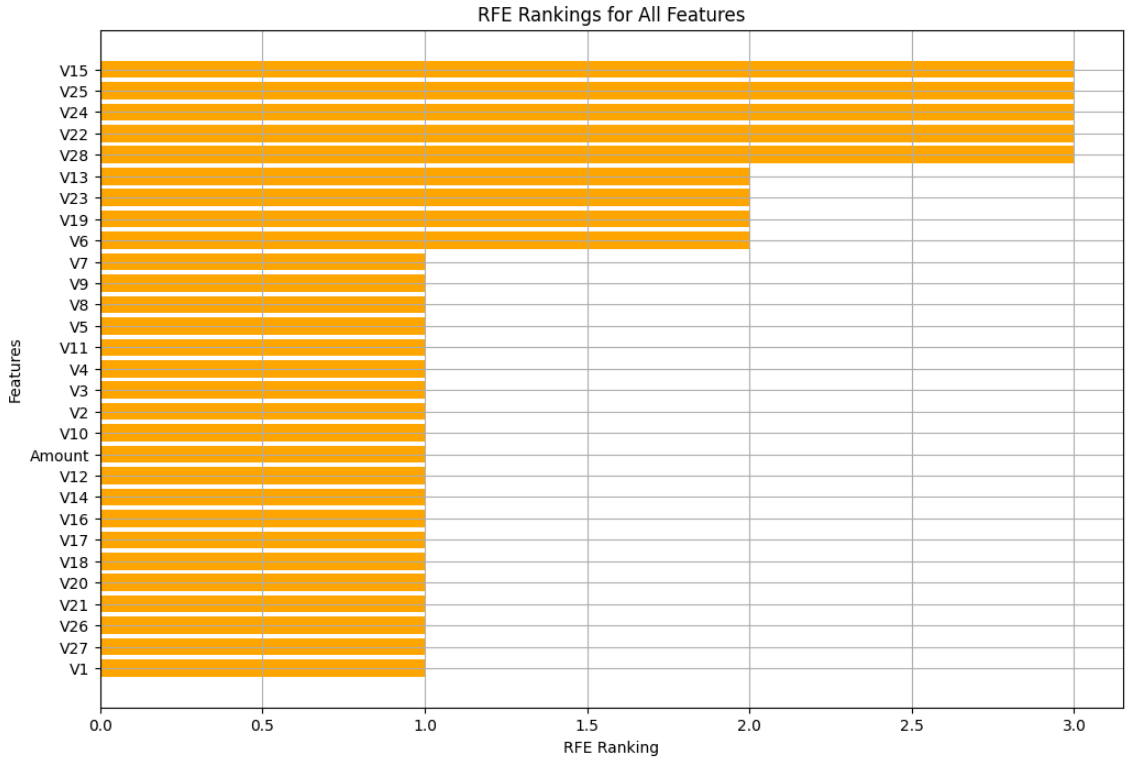

Figure 13: RFE Rankings of all Features

fraud detection.The bar plot of RFE Rankings for All Features shows the ranking of all 29 features in the dataset, where a ranking of 1 indicates a selected feature that was retained for model training. The features ranked as 1 (e.g., V17, V27) were determined to be the most important by RFE and are retained in the reducedfeature set. Features with higher rankings (e.g., V15, V25, V24) were excluded as they were deemed less influential in improving the classifier's performance.

We also conducted an experiment, to compare the performance of the Random Forest classifier on the original dataset and the dataset reduced using Recursive Feature Elimination (RFE). The goal was to evaluate the impact of feature selection on model accuracy and the key performance metrics: accuracy, precision, recall, and F1-score. The results showed that, for the original dataset, the classifier achieved an average precision of 0.94, recall of 0.73, and F1-score of 0.82 for the minority class (fraudulent transactions) in the first fold. After applying RFE, which reduced the dataset to 20 selected features, the precision for the minority class was slightly lower in some folds (e.g., 0.91 in Fold 1) but improved recall, achieving up to 0.76 in several folds. This led to a balanced increase in F1-scores across folds, reaching values of up to 0.86 in Fold 2.

While the overall accuracy remained consistently high (0.9994–0.9996), the reduction in features via RFE appears to improve the model's recall slightly, which is critical in fraud detection since it measures the proportion of actual frauds correctly identified. Precision, on the other hand, had similar results between the two datasets. This experiment highlights that feature selection can optimize the model's ability to generalize while maintaining comparable or improved performance, particularly for challenging cases like fraud detection.

Fold	Dataset	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	RFE-Reduced Dataset	0.9994	0.91	0.75	0.82
2	RFE-Reduced Dataset	0.9996	1.00	0.76	0.86
3	RFE-Reduced Dataset	0.9995	0.97	0.74	0.84
4	RFE-Reduced Dataset	0.9995	0.94	0.76	0.84
5	RFE-Reduced Dataset	0.9995	0.93	0.76	0.83

Average	RFE-Reduced Dataset	0.9995	0.95	0.754	0.838
Average	Original Dataset	0.99946	0.95	0.734	0.828

Table 2: RFE -Reduced Dataset metrics evaluation with Random Forest classifier and Average metrics for each Dataset

4 Performance Assessment

Performance assessment is a crucial aspect of evaluating the effectiveness of machine learning models, especially in critical applications like fraud detection. In this section, we will focus on assessing the models using various evaluation metrics and validation techniques to ensure their robustness and reliability. The goal is to provide a comprehensive understanding of how the models perform in detecting fraudulent transactions while balancing trade-offs between different performance metrics. Specifically, the metrics we will use in the next sections include Accuracy, Precision, Recall, F1-score, and AUC-ROC. Each of these metrics will be thoroughly explained, providing insights into their significance and application. Additionally, we will employ k-Fold Cross-Validation for splitting the train and test sets, ensuring robust evaluation and reducing the risk of overfitting.

4.1 Evaluation Metrics:Accuracy, Precision, Recall, F1-score, AUC-ROC

We will explore the key metrics used to evaluate the performance of classification models. Each metric provides unique insights into the model's ability to identify fraudulent transactions and handle class imbalance effectively:

- Accuracy: Accuracy is the proportion of correctly predicted instances (both fraudulent and non-fraudulent) out of the total number of instances. While accuracy is a straightforward metric, it can be misleading in imbalanced datasets, such as fraud detection, where the majority class dominates.
- Precision: Precision measures the proportion of true positive predictions (correctly identified fraudulent transactions) out of all predicted positives. It focuses on the correctness of fraud predictions and is critical in scenarios where false positives need to be minimized, such as avoiding unnecessary interventions for legitimate transactions.
- Recall: Recall, also known as sensitivity or true positive rate, calculates the proportion of true positive predictions out of all actual positives. It measures the model's ability to identify all fraudulent transactions. High recall is crucial when missing a fraudulent transaction is costly.
- F1-Score: The F1-score is the harmonic mean of precision and recall, providing a single metric to balance their trade-offs. It is particularly useful in imbalanced datasets as it considers both false positives and false negatives.
- AUC-ROC: The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) evaluates the model's ability to distinguish between classes

across various thresholds. AUC-ROC is an essential metric for understanding the overall classification performance, especially when class distributions are imbalanced.

Each of these metrics will be computed for our models and analyzed to ensure robust evaluation.

4.2 Cross-Validation Techniques: k-Fold and LOOCV

In this subsection, we will discuss cross-validation techniques used to assess the generalizability of machine learning models. Cross-validation is a vital strategy to evaluate the performance of models on unseen data and avoid overfitting.

In this paper, we will primarily use k-Fold Cross-Validation for my experiments. This technique involves dividing the dataset into k equal-sized folds. The model is trained on $k-1$ folds and tested on the remaining fold. This process is repeated k times, with each fold serving as the test set exactly once. The results are averaged to provide an overall performance estimate. k-Fold Cross-Validation ensures that every observation is used for both training and validation, providing a robust estimate of model performance.LOOCV on the other hand is an extreme case of k-Fold Cross-Validation where k equals the total number of observations. Each observation serves as the test set once, while the remaining observations form the training set. While LOOCV is computationally expensive for large datasets, it provides an unbiased estimate of model performance.

In this study, we will rely on k-Fold Cross-Validation due to its balance between computational efficiency and reliable performance estimation. The use of cross-validation ensures that our models are evaluated rigorously and are less prone to overfitting or underfitting.

5 Classification Techniques

In this section, we will explore a variety of classification techniques to determine the most effective model for detecting fraudulent transactions. Specifically, we will evaluate the performance of Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs). Each technique will be tested using the dataset with the reduced feature set obtained through Recursive Feature Elimination (RFE), as it demonstrated superior performance in terms of Accuracy, Precision, Recall, and F1-Score in our previous experiments with Random Forests. By leveraging this optimized

feature set, we aim to ensure a fair comparison across different classification models while maintaining computational efficiency and robust evaluation metrics. Through this comprehensive analysis, we aim to identify the classification technique that provides the best balance between performance and practicality for fraud detection.

5.1 Logistic Regression

In this experiment, we evaluated the performance of Logistic Regression on the RFE-reduced dataset using the hyperparameters max_iter=1000 and the default regularization parameter C=1.0. While the model achieved consistently high accuracy across all folds (averaging 96.94%), its precision for predicting fraudulent transactions (Class 1) was extremely low, averaging just 0.05. This indicates that the model produced a large number of false positives, predicting many non-fraudulent transactions as fraudulent. One reason for this behavior is that Logistic Regression assumes a linear decision boundary, which may not effectively separate the two classes in a dataset with complex patterns or non-linear relationships. Additionally, the high class imbalance in the dataset exacerbates this issue, as the minority class (fraud) has significantly fewer examples.

Fold	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	0.9720	0.05	0.91	0.10
2	0.9665	0.05	0.94	0.09
3	0.9705	0.05	0.93	0.10
4	0.9690	0.05	0.92	0.09
5	0.9688	0.05	0.89	0.09
Average	0.9693	0.05	0.91	0.09

Table 3: Logistic Regression performance across folds

To address these challenges, we will explore hyperparameter optimization in the next section. Specifically, tuning the regularization strength (C) could help balance the trade-off between underfitting and overfitting, potentially improving precision. Additionally, experimenting with class weighting or feature

engineering may help enhance the model's ability to distinguish between fraudulent and non-fraudulent transactions. Adjusting the decision threshold for classification is another approach we plan to investigate to strike a better balance between precision and recall. These adjustments aim to optimize the model's performance, particularly for the minority class.

5.2 Decision Trees and Random Forests

For the Decision Tree classifier, we evaluated its performance on the RFE-reduced dataset using default hyperparameters, including random_state=42. The model achieved an impressive average accuracy of 99.93% across all folds. For the minority class (fraudulent transactions), it achieved an average precision of 0.81, recall of 0.76, and F1-score of 0.78. These results indicate a strong ability to identify fraudulent transactions while maintaining a balance between precision and recall. Decision Trees inherently handle class imbalance better than simpler models due to their ability to split the data iteratively based on feature importance. However, fine-tuning hyperparameters such as max_depth, min_samples_split, and min_samples_leaf could potentially improve performance further. In the next section, we aim to explore these adjustments as part of a comprehensive hyperparameter optimization strategy.

Fold	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	0.9992	0.79	0.73	0.76
2	0.9994	0.89	0.75	0.81
3	0.9994	0.84	0.81	0.82
4	0.9992	0.78	0.73	0.76
5	0.9991	0.74	0.77	0.75
Average	0.9992	0.808	0.75	0.78

Table 4: Decision Trees performance across folds

For the Random Forest classifier, we previously evaluated its performance in the section discussing Recursive Feature Elimination (RFE). The results demonstrated that Random Forest achieved consistently high performance across all metrics, with an average accuracy of 99.95%, precision of 0.95, recall of 0.75, and F1-score of 0.84 for the minority class (fraudulent transactions). These results, presented in Table 2, highlight Random Forest's robustness and its ability

to handle class imbalance effectively, making it a strong candidate for fraud detection in this study.

5.3 Support Vector Machines (SVMs)

In this experiment, we evaluated the performance of the Support Vector Machine (SVM) classifier on the RFE-reduced dataset using the default hyperparameters: a linear kernel, C=1.0 (default regularization), and class_weight="balanced" to account for class imbalance. The model achieved an average accuracy of 99.64%, reflecting its strong ability to classify the majority class (non-fraudulent transactions). However, the precision for fraudulent transactions (Class 1) was significantly lower, averaging only 0.31. This indicates that the model frequently misclassified non-fraudulent transactions as fraudulent, resulting in a high number of false positives. While the recall was notably high at 0.87, capturing most actual fraudulent transactions, the low precision highlights a critical trade-off inherent in imbalanced datasets. This behavior could stem from the overlapping feature distributions between classes, the complexity of the dataset, and the default linear decision boundary, which may not adequately separate the two classes

Fold	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	0.9968	0.33	0.85	0.48
2	0.9969	0.34	0.89	0.50
3	0.9956	0.27	0.88	0.41
4	0.9960	0.29	0.87	0.43
5	0.9965	0.31	0.84	0.45
Average	0.99636	0.308	0.866	0.454

Table 5: Support Vector Machines (SVMs) performance across folds

In the next sections, we will explore hyperparameter optimization to improve SVM's performance, particularly its precision. We plan to experiment with non-linear kernels such as rbf or poly, which can better handle complex feature relationships. Additionally, tuning the C parameter (regularization) can help balance the trade-off between maximizing the margin and minimizing misclassifications. For non-linear kernels, optimizing the gamma parameter will allow us to adjust the influence of individual data points on the decision

boundary. Furthermore, we may refine the class weights beyond the default balanced setting to improve the model's focus on correctly identifying fraudulent transactions. These adjustments aim to enhance SVM's ability to strike a better balance between precision and recall for effective fraud detection.

5.4 Artificial Neural Networks (ANNs)

The Artificial Neural Network (ANN) architecture used in this experiment consisted of a single hidden layer with 100 neurons. The input layer size corresponded to the number of features in the RFE-reduced dataset, which was 20, ensuring that each input feature was represented. The output layer consisted of a single neuron using a sigmoid activation function, suitable for binary classification tasks (fraudulent vs. non-fraudulent transactions). The network employed the ReLU activation function for the hidden layer to introduce non-linearity and the Adam optimizer, which combines adaptive learning rate techniques for efficient gradient-based optimization. Training was performed for up to 1,000 iterations or until convergence.

The ANN achieved an average accuracy of 99.44%, showcasing its capability to classify the majority class (non-fraudulent transactions) effectively. However, precision for fraudulent transactions (Class 1) was relatively low, averaging 0.23, indicating a high number of false positives. Recall for Class 1 was significantly higher at 0.86, demonstrating the model's ability to identify most fraudulent transactions, though at the cost of misclassifying non-fraudulent transactions as fraudulent. This imbalance resulted in an average F1-score of 0.36 for Class 1, highlighting the need to improve precision without sacrificing recall.

Fold	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
1	0.9905	0.14	0.86	0.24
2	0.9953	0.26	0.91	0.40
3	0.9957	0.27	0.86	0.41
4	0.9947	0.23	0.86	0.36
5	0.9958	0.26	0.82	0.40
Average	0.9944	0.23	0.86	0.36

Table 6: Artificial Neural Networks (ANNs) performance across folds

To handle the imbalance in the dataset, we utilized oversampling during training. For each fold in the Stratified K-Fold Cross-Validation, the minority class (fraudulent transactions) was oversampled using the RandomOverSampler technique to balance the training data. This ensured that the network had sufficient examples of fraudulent transactions to learn meaningful patterns. Despite these efforts, the low precision suggests potential improvements in the network's architecture and hyperparameters, such as experimenting with deeper networks, more neurons, or alternative optimization strategies, which will be explored in subsequent sections.

6 Hyperparameter Tuning and Optimization

In this section, we will focus on optimizing the performance of our classification models through hyperparameter tuning and optimization techniques. Our primary goal is to address the low precision observed in models like Logistic Regression, Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs). These models, while achieving good recall, struggled with precision, indicating a tendency to misclassify non-fraudulent transactions as fraudulent. We will begin with an overview of hyperparameters relevant to these models, understanding their impact on performance and behavior. Next, we will explore Grid Search and Randomized Search to systematically evaluate different hyperparameter combinations. Finally, we will delve into Bayesian Optimization, a more advanced technique that balances exploration and exploitation to efficiently identify optimal configurations. By tailoring these methods to Logistic Regression, SVMs, and ANNs, we aim to improve their precision while maintaining a strong balance with recall, ultimately enhancing their suitability for fraud detection.

6.1 Overview of Hyperparameters

In this section, we focus on exploring and optimizing the key hyperparameters of Logistic Regression, Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), Decision Trees, and Random Forests to enhance their performance, particularly in predicting fraudulent transactions. For Logistic Regression, the primary hyperparameter is the regularization strength (**C**), which controls the trade-off between underfitting and overfitting. Fine-tuning this parameter can improve the model's ability to handle complex relationships between features and enhance precision.For SVMs, we will explore hyperparameters such as the kernel type (e.g., linear, rbf, poly), regularization strength (**C**), and gamma for non-linear kernels, which determine the decision

boundary and the influence of individual data points. These adjustments aim to improve the balance between precision and recall, especially given the class imbalance in the dataset. For Artificial Neural Networks, we will experiment with the number of hidden layers, neurons per layer, learning rate, and optimization algorithms (e.g., Adam, SGD). Additionally, we will explore techniques like weighted loss functions and dropout to improve generalization and address the imbalance in the dataset.For Decision Trees, we will tune key hyperparameters such as the maximum tree depth, minimum samples required to split an internal node, and minimum samples required to form a leaf node. These adjustments help control overfitting and ensure the tree generalizes well to unseen data. For Random Forests, we will focus on the number of estimators, tree depth, and split criteria to leverage the ensemble learning mechanism effectively. By systematically tuning these hyperparameters, we aim to reduce false positives, improve precision, and maintain high recall, ultimately enhancing the classifiers' ability to effectively detect fraudulent transactions. This comprehensive approach ensures that each model is optimized to handle the unique challenges posed by the dataset's imbalance and complex patterns.

6.2 Grid Search and Randomized Search

In this section, we explore Grid Search and Randomized Search, two widely used techniques for hyperparameter optimization. Grid Search systematically evaluates all possible combinations of predefined hyperparameter values, ensuring that the global optimal configuration is found within the search space. However, this exhaustive approach can be computationally expensive, especially when the number of hyperparameters and their values increases. To address this, we will instead focus on Randomized Search, which randomly samples a subset of hyperparameter combinations, significantly reducing computation time while still providing competitive results.Randomized Search will be performed with 5 iterations and a k=3 fold cross-validation to balance efficiency and robust evaluation. This approach ensures a systematic and computationally efficient exploration of hyperparameters across all classifiers.

The hyperparameter tuning with Randomized Search significantly improved the performance of some classifiers, as illustrated in Figures 14, 15, and 16, while showing limited improvements or declines for others. Logistic Regression (Figure 14) achieved a high F1-score of 0.7637, reflecting a dramatic improvement in precision (from 0.05 to 0.8658) while maintaining a balanced recall (0.6931). This indicates that tuning the C parameter and max_iter allowed the model to better handle class imbalance. SVMs (Figure 15), which previously struggled with precision and achieved an F1-score of 0.45, saw significant gains

in both precision (0.8527) and recall (0.7642), resulting in an F1-score of 0.8022. Finally, Artificial Neural Networks (ANNs) (Figure 16) demonstrated the most substantial improvement, achieving the highest F1-score of 0.8078, with precision increasing from 0.23 to 0.8697 and recall remaining strong at 0.7581. These results highlight the effectiveness of hyperparameter tuning in improving the balance between precision and recall, especially for challenging fraud detection tasks.

Classifier	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
Logistic Regression	0.9993	0.8658	0.6931	0.7637
SVM	0.9993	0.8527	0.7642	0.8022
ANN	0.9994	0.8697	0.7581	0.8078
Decision Tree	0.9990	0.8137	0.5894	0.6698
Random Forest	0.9992	0.8754	0.6768	0.7566

Table 6: Metrics for each Classifier based on the Random Search results

However, the results for Decision Tree and Random Forest showed that hyperparameter tuning did not yield improvements over their default configurations. For the Decision Tree, the best F1-score after Randomized Search was 0.6698, which was lower than the 0.6863 achieved with default settings. Similarly, the Random Forest achieved an F1-score of 0.7566 after tuning, a decline compared to the 0.838 observed with default hyperparameters. These declines suggest that the default hyperparameters for these models may already have been well-suited to the dataset or that the hyperparameter search space was not sufficiently broad to include configurations better than the defaults.

The improvements across Logistic Regression, SVMs, and ANNs, as shown in Figures 14, 15, and 16, are notable. Logistic Regression (Figure 14) showed the most significant relative improvement in precision (from 0.05 to 0.8658) and F1-score (from 0.09 to 0.7637), making it a much more viable option for fraud detection. SVMs (Figure 15) also saw substantial gains in F1-score (from 0.45 to 0.8022), indicating better handling of false positives and overall predictive power. ANNs (Figure 16), which already had the highest recall in the previous evaluation, now excel in precision and F1-score after optimizing the number of neurons, hidden layers, and learning rate. However, the decline in performance

for Decision Tree and Random Forest underscores the importance of carefully selecting hyperparameter tuning strategies to avoid over-tuning or narrowing the search space excessively (Figure 17, Figure 18).

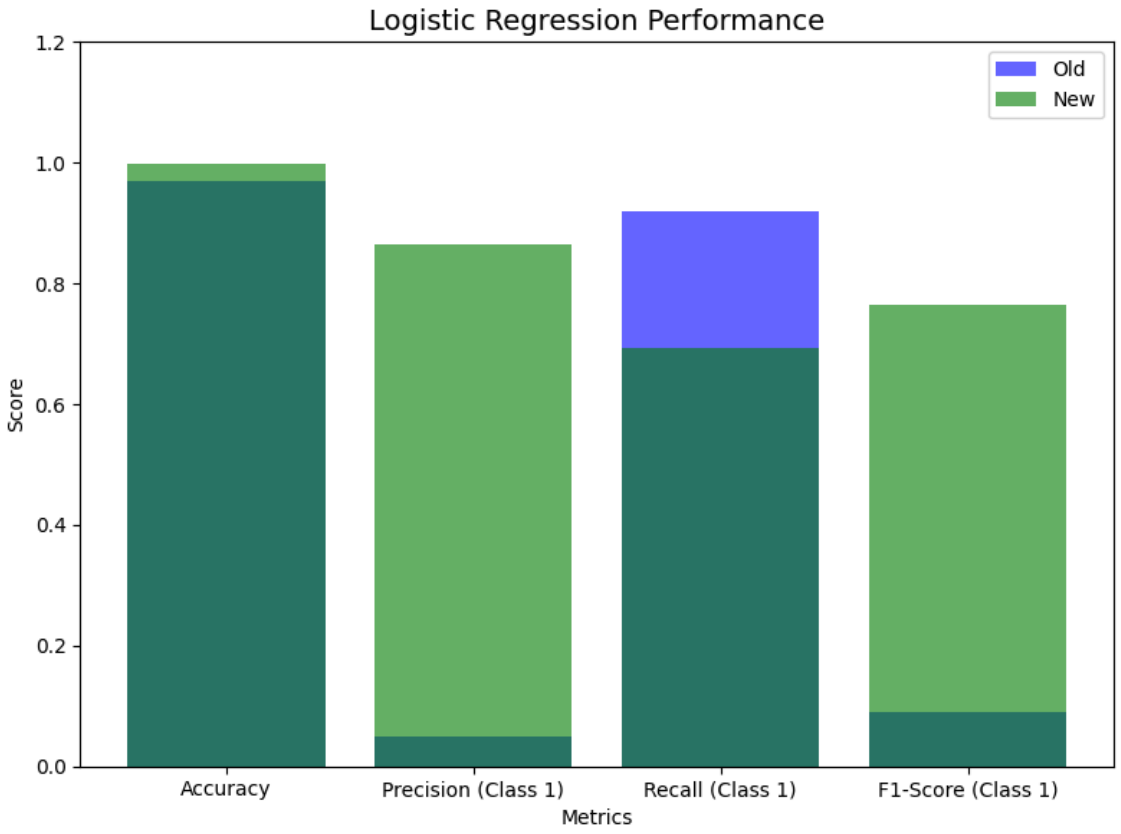

Figure 14: Logistic Regression Performance after Random Search

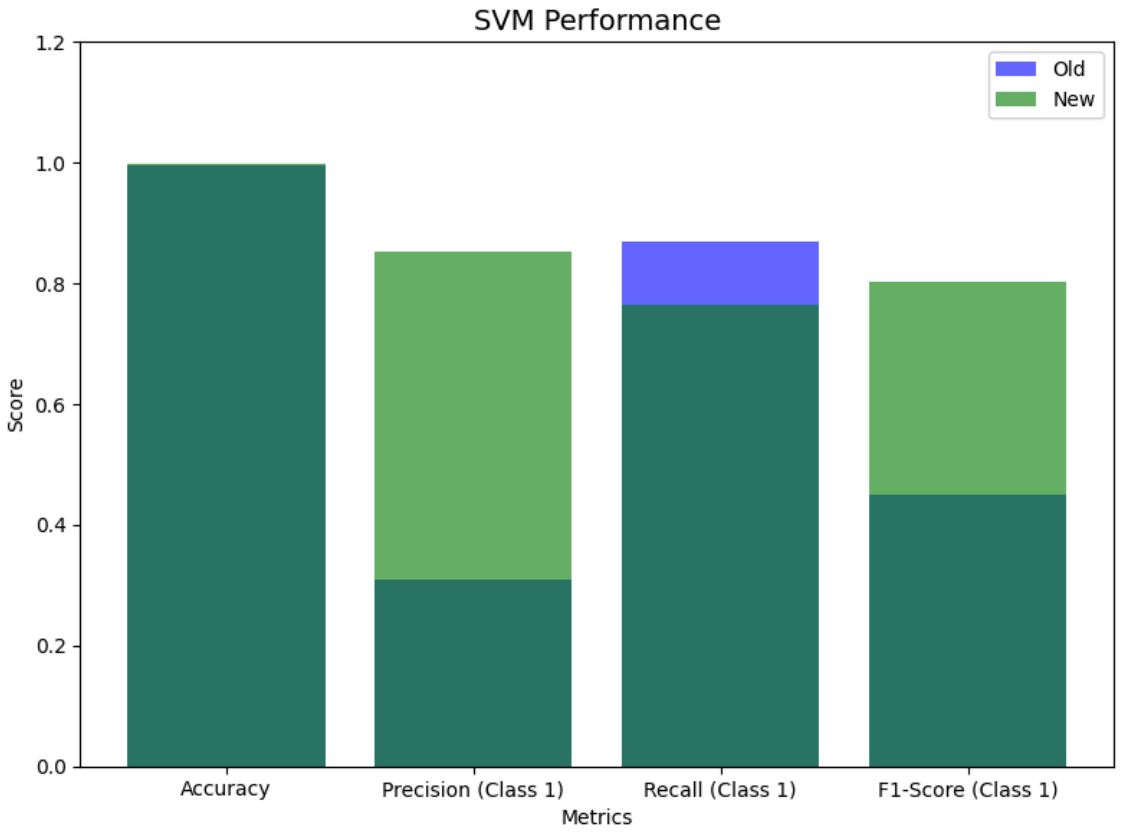

Figure 15: SVM Performance after Random Search

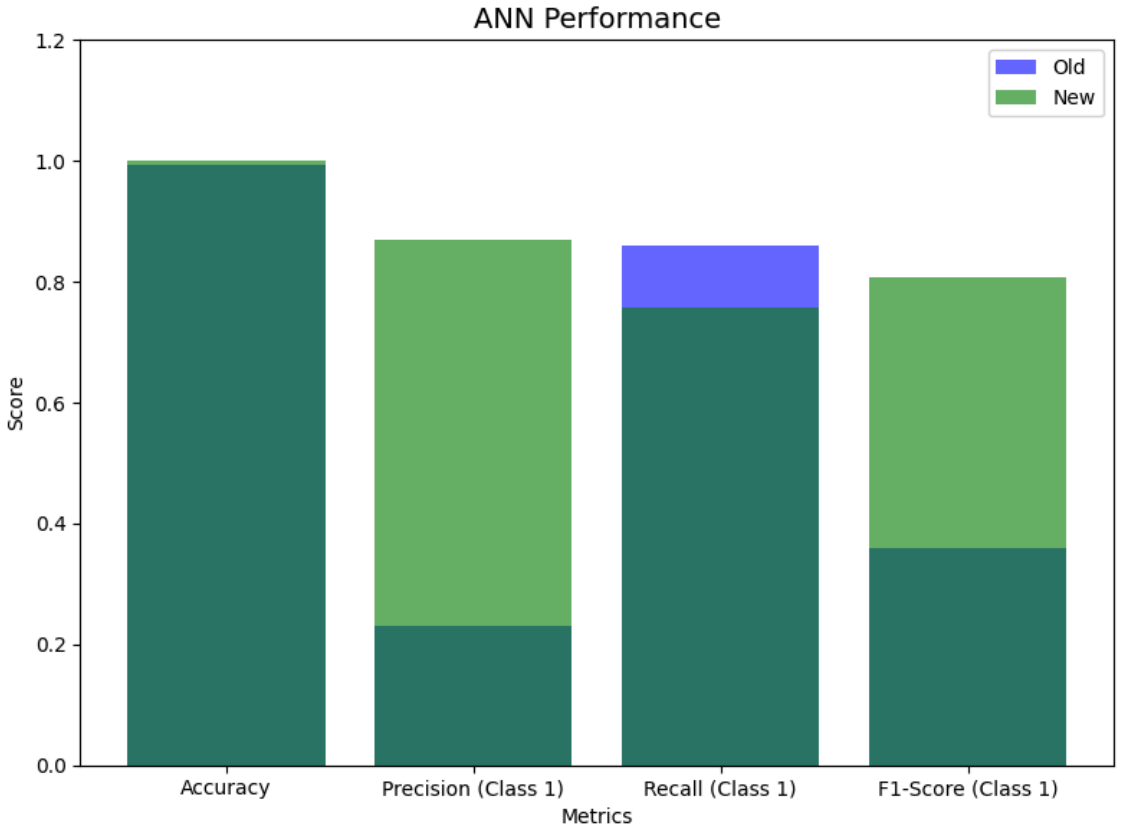

Figure 16: ANN Performance after Random Search

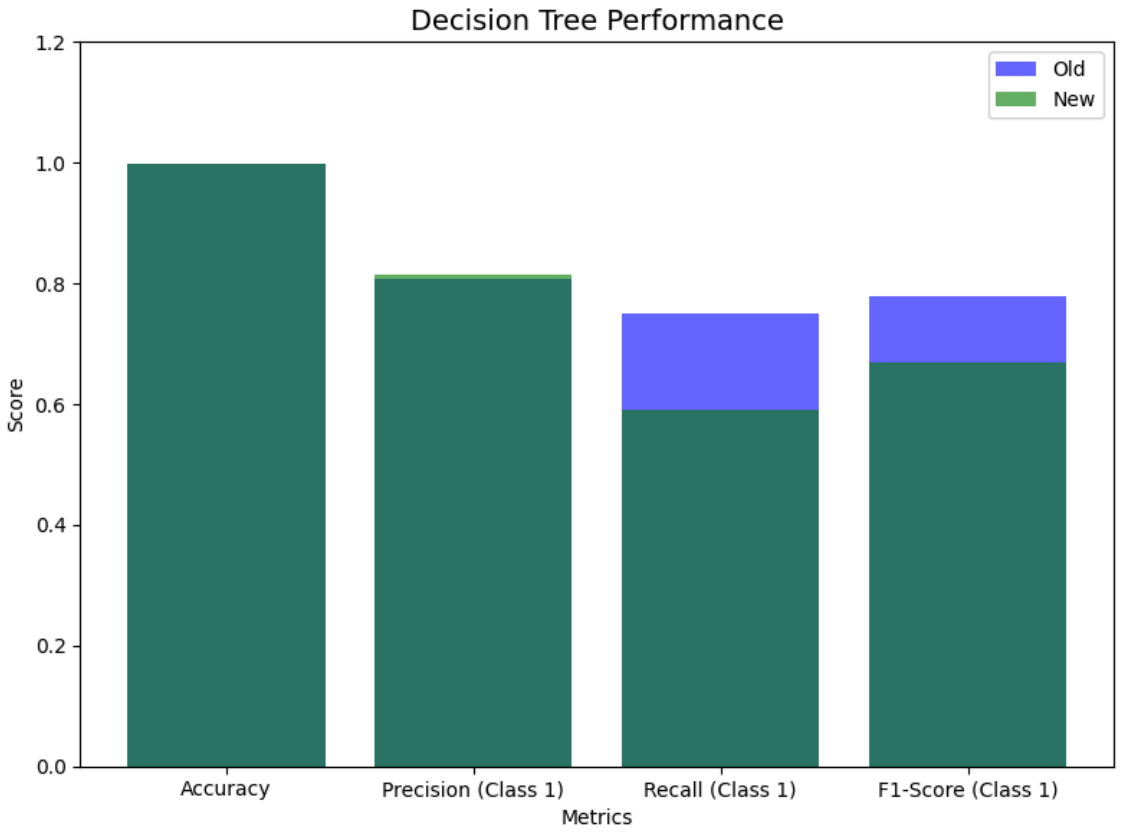

Figure 17: Decision Tree Performance after Random Search

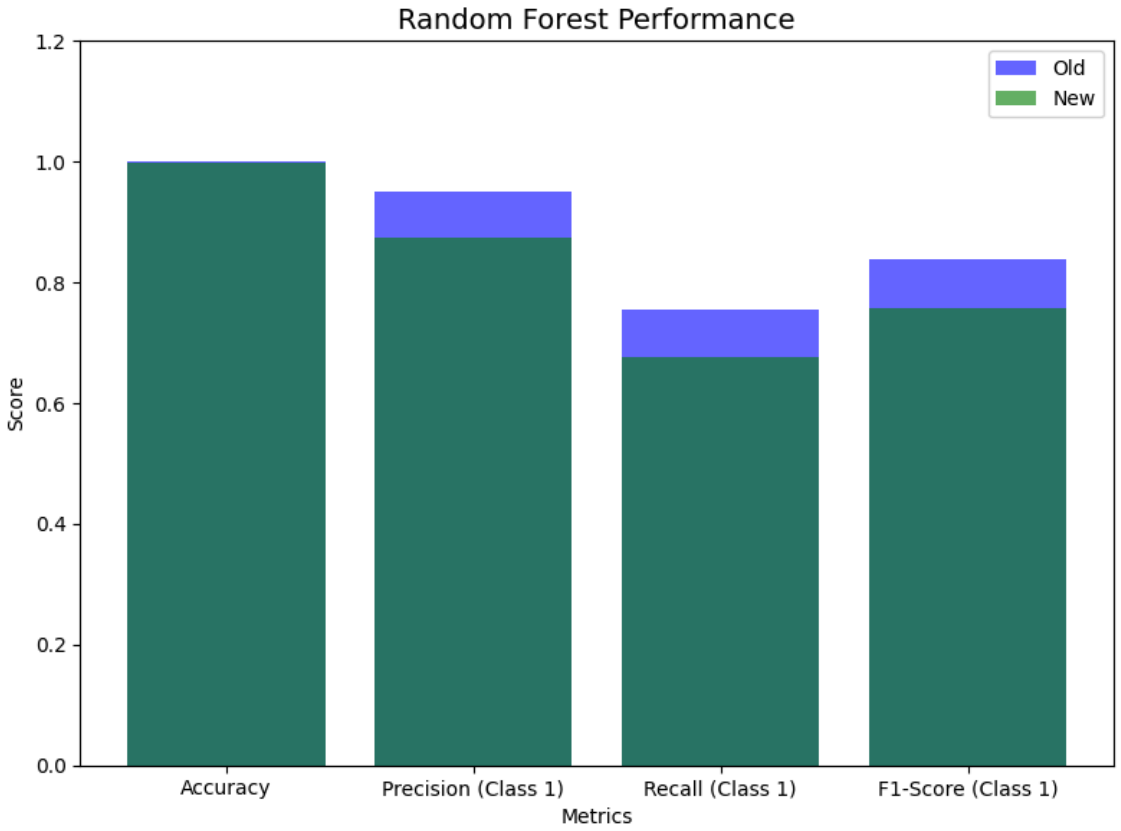

Figure 18: Random ForestPerformance after Random Search

6.3 Bayesian Optimization

In this section, we utilized Bayesian Optimization, implemented using the optuna library, to fine-tune the hyperparameters of five classifiers: Logistic Regression, SVM, Artificial Neural Networks (ANNs), Decision Tree, and Random Forest. Bayesian Optimization intelligently explored the hyperparameter space, balancing exploration and exploitation, to maximize the F1-score for predicting fraudulent transactions. This approach allowed us to identify optimal configurations for each classifier efficiently.For Logistic Regression, hyperparameter tuning with Bayesian Optimization resulted in a significant improvement, achieving an F1-score of 0.7767, reflecting better handling of class imbalance compared to earlier experiments including Randomized Search. Similarly, the SVM achieved the highest F1-score of 0.8156, demonstrating superior performance in balancing precision and recall. The ANN, after optimization, achieved an F1-score of 0.8119, showcasing a significant improvement in precision while maintaining strong recall. However, for the Decision Tree and Random Forest, there was no improvement observed from hyperparameter tuning, and the default configurations of these classifiers outperformed the optimized versions.

As we saw, the default Random Forest achieved a higher F1-score of 0.838 compared to the tuned version, while the Decision Tree also performed better without hyperparameter optimization. These results highlight the effectiveness of Bayesian Optimization in improving the performance of some classifiers, while also emphasizing the importance of carefully evaluating default configurations, which can sometimes offer better generalization. This process ensures a balanced approach to precision and recall, critical for effective fraud detection.

Classifier	Best Parametres
Logistic Regression	{'C': 54.205022100367316, 'max_iter': 677}
SVM	{'C': 95.81768397133132, 'kernel': 'rbf', 'gamma': 'auto'}
ANN	{'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'solver': 'adam', 'max_iter': 472}
Decision Trees	{'max_depth': 12, 'min_samples_split': 8}
Random Forest	{'n_estimators': 52, 'max_depth': 16, 'min_samples_split': 3}

Table 7: Best Parameters for each Classifier based on the Random Search results

Metric	Logistic Regression	SVM	ANN	Decision Tree	Random Forest
Accuracy	0.999	0.999	0.999	0.999	0.999
Precision (Class 1)	0.864	0.838	0.859	0.805	0.87
Recall (Class 1)	0.713	0.800	0.774	0.621	0.668
F1-Score (Class 1)	0.776	0.815	0.811	0.691	0.753

Table 8: Metrics for each Classifier based on the Random Search results

These results, as summarized in Table 8, confirm that Bayesian Optimization significantly improved the performance of most classifiers. However, for the Random Forest and Decision Tree, the default hyperparameters demonstrated better generalization, emphasizing the importance of evaluating both optimized and default configurations. This optimization process ensures a better balance between precision and recall, crucial for effective fraud detection.

7 Comparative Analysis of Models

In this section, we will summarize the best performance metrics achieved for each classifier Logistic Regression, SVMs, ANNs, Decision Trees, and Random Forests after applying various hyperparameter optimization techniques and dimensionality reduction methods like RFE. The focus will be on identifying the most effective model for fraud detection based on metrics like F1-score, precision, and recall.We will also discuss Practical Considerations for Real-World Deployment, including model interpretability, computational efficiency, and compliance with regulatory requirements. This section will provide insights into how these models can be integrated into operational systems effectively while balancing complexity, performance, and transparency.

7.1 Summary of Model Performance

The journey began with data preprocessing, during which missing values were handled, features were standardized, and class imbalance was addressed to ensure

the dataset was prepared for effective modeling. Dimensionality reduction techniques, such as PCA and Recursive Feature Elimination (RFE), were explored to reduce complexity and enhance performance. RFE was selected due to its superior ability to retain key features essential for classification tasks.

Following dimensionality reduction, Randomized Search and Bayesian Optimization were applied as hyperparameter optimization techniques. While significant improvements were observed for most classifiers, including Logistic Regression, SVMs, and ANNs, through Bayesian Optimization, Decision Trees and Random Forests were found to perform better with their default parameters. Figure 19 were created to illustrate the improvements in Accuracy, Precision, Recall and F1-score for each classifier, making comparisons more intuitive. This section summarizes these findings, presenting the final performance of each model and its suitability for fraud detection.

Metric	Logistic Regression	SVM	ANN	Decision Tree	Random Forest
F1-Score (Class 1)	0.776	0.815	0.811	0.78	0.838

Table 8: The best F1-Score for each Classifier

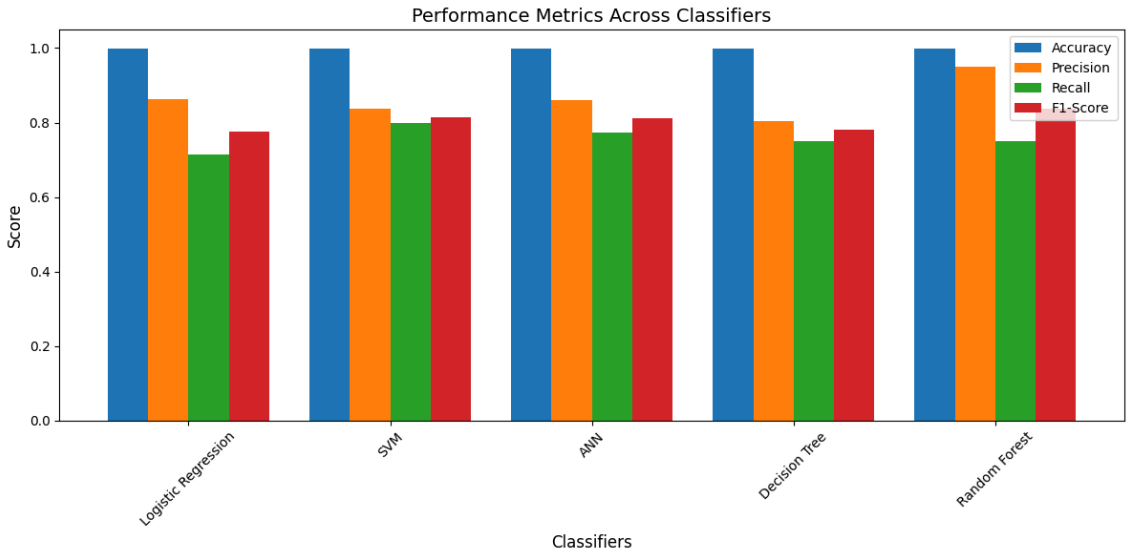

Figure 19: Best Performance Metrics Across all Classifiers

While all classifiers achieved identical Accuracy values of 0.999, their differences become evident in other metrics. Random Forest demonstrated the

highest Precision (0.95), excelling at minimizing false positives, while Logistic Regression (0.864) and ANN (0.859) also performed well in this area. SVM achieved the highest Recall (0.800), reflecting its ability to capture fraudulent transactions effectively, followed closely by ANN (0.774) and Random Forest (0.75). In terms of F1-Score, which balances Precision and Recall, Random Forest (0.838) and SVM (0.815) emerged as the strongest performers. Logistic Regression (0.776) and Decision Tree (0.78), while slightly lower, still maintained competitive F1-Scores. Overall, Random Forest , SVM and ANN displayed balanced performance, with Random Forest excelling in Precision, making these models particularly suitable for fraud detection tasks.

7.2 Practical Considerations for Real-World Deployment

When deploying machine learning models for fraud detection in real-world systems, several practical considerations must be addressed. First, model interpretability is critical, as decisions need to be transparent and explainable to gain trust from stakeholders and comply with regulatory requirements, especially in industries like finance and healthcare. Tools such as SHAP and LIME can be integrated to provide insights into the decision-making process. Second, scalability and computational efficiency are important to ensure the model can handle large volumes of transactions in real-time, particularly for high-frequency systems. This involves selecting models with manageable complexity, such as Logistic Regression or Decision Trees, for deployment when computational resources are limited. Additionally, strategies to handle class imbalance effectively, such as rebalancing techniques or adjusting decision thresholds, must be integrated to maintain performance in highly skewed datasets. Lastly, continuous monitoring and retraining are essential to ensure the model remains robust over time as fraud patterns evolve, requiring automated pipelines for updating the model with new data and regular performance evaluation to identify potential drift. These considerations ensure that the deployed system is both effective and sustainable in a dynamic, real-world environment.

8 Conclusion and Recommendations

Based on the final results, the Random Forest classifier emerges as the most suitable model for our fraud detection problem. While all classifiers achieved identical Accuracy (0.999), Random Forest demonstrated the highest Precision (0.95) and the best overall F1-Score (0.838), indicating its ability to balance false positives and false negatives effectively. This is critical in fraud detection, where minimizing false positives (to avoid unnecessary alerts) while still capturing fraudulent transactions is paramount. Although the SVM and ANN classifiers performed slightly better in Recall (0.800 and 0.774, respectively), Random Forest's superior Precision ensures that flagged transactions are more likely to be genuine fraud cases, reducing the burden on investigators. Additionally, Random Forest's ensemble nature provides robustness against overfitting and better generalization, making it ideal for real-world deployment in imbalanced datasets like ours. This conclusion was reached by thoroughly analyzing the metrics and weighing the trade-offs between Precision, Recall, and F1-Score to identify the model that aligns best with the practical needs of fraud detection.

9 References

[1] Credit Card Fraud Detection Database, Anonymized credit card transactions labeled as fraudulent or genuine,
https://www.kaggle.com/mlg-ulb/creditcardfraud

[2]Feature Selection, Wikipedia Page,
https://en.wikipedia.org/wiki/Feature_selection

[2] Principal Component Analysis, Wikipedia Page,
https://en.wikipedia.org/wiki/Principal_component_analysis

[4] Hyperparameter Optimization, Wikipedia Page,
https://en.wikipedia.org/wiki/Hyperparameter_optimization

[5] RandomForrestClassifier,
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[6] Πρωτοπαπαδάκης, Ε. (2025). *Lecture Notes on Machine Learning*, University of Makedonia.