

Predicting Credit Card Fraud



YOLO Bank Ltd

Priyanka Jawalgaonkar

Table of Contents

1. Introduction.....	3
2. Exploratory Data Analysis (EDA).....	4
3. Data Cleaning	6
4. Dealing with Imbalanced data	7
5. Feature Engineering	8
6. Model Selection	10
7. Model Training	11
8. Model Validation	12
9. Model Deployment	17
10. Conclusion	18
11. Appendix	19
12. Acknowledgement	20

1. Introduction

1.1 Background

Founded in 2010, YOLO Bank Ltd., has been at the forefront of innovation in the financial sector and offers a wide range of services to its diverse customers. The emergence of online shopping has given credit cards a significant role in the global economy. While this change provides great convenience, it also exposes consumers and financial institutions to the risk of credit card fraud. YOLO Bank recognizes the importance of security for its credit card product, YourCard. Anti-fraud service launched.

1.2 Problem Statement

Credit card companies, including YOLO Bank Ltd., face the difficult challenge of distinguishing between legitimate and fraudulent transactions. Striking the right balance is critical to ensuring customers are not incorrectly charged for unauthorized purchases. The dataset used in this project represents credit card transactions made by European cardholders in September 2013. A total of 284,807 fraudulent transactions and 492 materials are out of balance, and fraud accounts for only 0.172% of all transactions.

1.3 Primary Objective

The main aim of this project is to develop a robust classification model that can predict the probability of fraud. YOLO is committed to improving fraud detection capabilities by increasing exposure and exposure to fraud. In order to solve these issues caused by suspicious information; the bank hopes to strengthen the security of YourCard to provide customers with a safer online transaction. The success of the program will be measured by the model that increases financial impact by minimizing negative impact and its accuracy in detecting fraud.

1.4 Dataset

creditcard.csv The dataset contains transaction data collected over two days. Each transaction is characterized by properties such as time, value, and various anonymous variables derived from the transaction. This information is provided by cardholders in Europe and provides information about the authenticity of credit cards. The inconsistent nature of the data, combined with the low number of fraud cases, underscores the need for better modeling to detect fraud.

1.5 Libraries Imported

1. **Pandas (pd)**: for data processing and analysis.
2. **NumPy (np)**: Simplify arithmetic and array operations.
3. **Matplotlib (plt)**: Allows creating various data visualizations.
4. **train_test_split**: Scikit-learn function splits the dataset into training set and test set.
5. **SMOTE**: An academic library does not equal having an unequal classroom.
6. **StandardScaler**: Modeling work is done by removing the mean and scaling the unit variance.
7. **RandomForestClassifier**: used to create a random forest classification model.
8. **accuracy_score, classification_report, confusion_matrix**: Scikit-learning metrics used to evaluate model performance.
9. **GridSearchCV**: used to set hyperparameters from grid search.
10. **joblib**: used to save and load machine learning models.

2. Data Analysis (EDA)

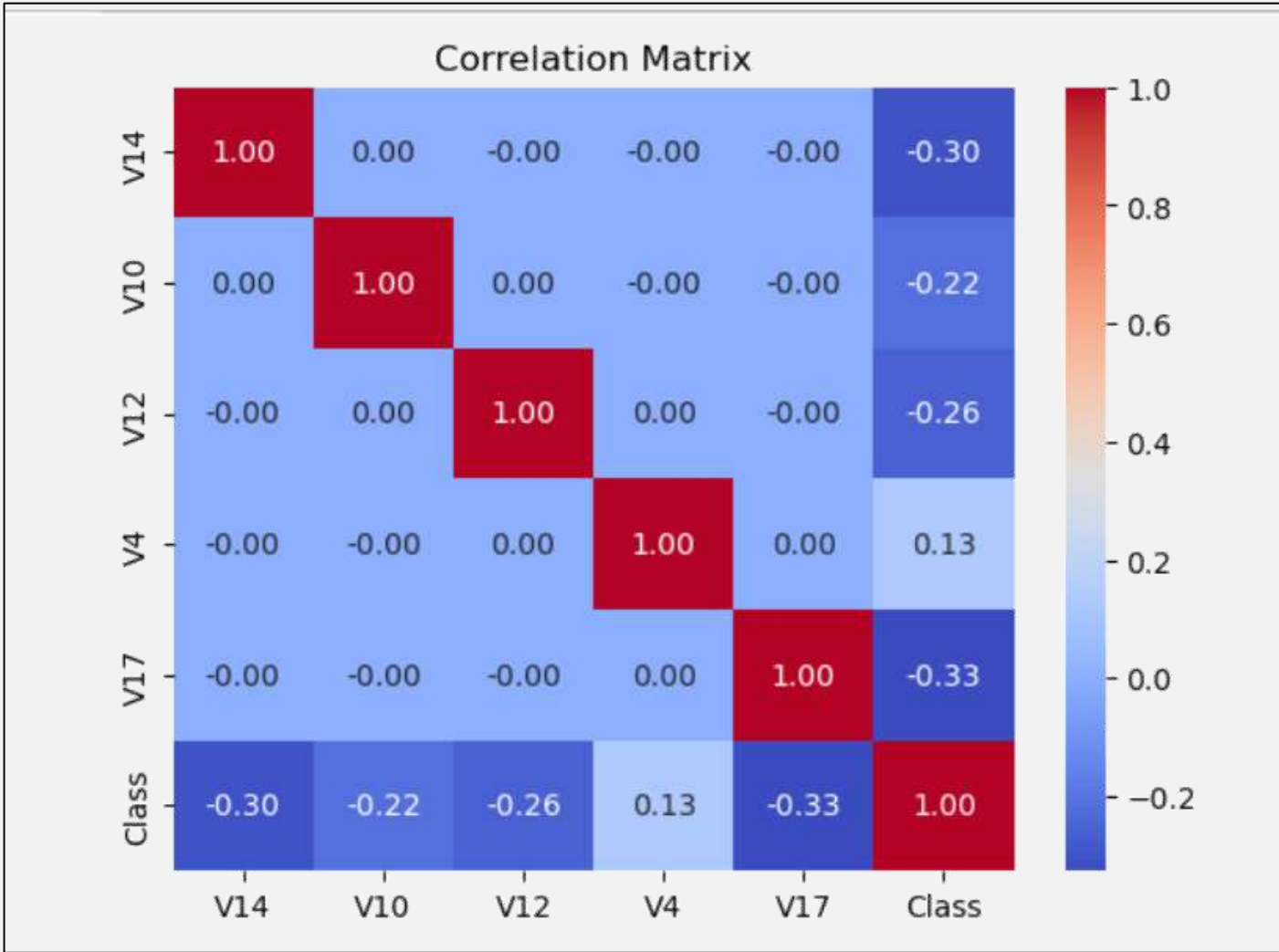
2.1 Dataset Overview

The dataset contains 31 rows and 284,807 entries. Each entry represents a credit card, and the column contains attributes such as "time" and "amount" as well as a series of anonymous numbers including "V1" through "V28". This group indicates whether the operation is an error (1) or an error (0).

2.2 Data Visualization

2.2.1 Correlation Matrix and Heatmap

Present the correlation matrix heatmap to visualize the relationships between selected features. The correlation coefficients between 'Class' and other features are also provided for reference:



2.2.2 Interpretation

```
# Calculate correlation of 'Class' with other features
class_correlation = data[['V14', 'V10', 'V12', 'V4', 'V17', 'Class']].corr()['Class']

# Print the correlation values
print("Correlation of 'Class' with other features:")
print(class_correlation)

Correlation of 'Class' with other features:
V14      -0.302544
V10      -0.216883
V12      -0.260593
V4        0.133447
V17      -0.326481
Class     1.000000
Name: Class, dtype: float64
```

Correlation of 'Class' with other features:

V14: -0.302544: Negative correlation (moderate): As 'V14' increases, the likelihood of the transaction being fraudulent ('Class' being 1) decreases moderately.

V10: -0.216883: Negative correlation (weak): As 'V10' increases, the likelihood of the transaction being fraudulent decreases, but the correlation is weak compared to 'V14'.

V12: -0.260593: Negative correlation (moderate): Similar to 'V14', an increase in 'V12' is associated with a decrease in the likelihood of fraud.

V4: 0.133447: Positive correlation (weak): As 'V4' increases, there is a weak positive association with the likelihood of fraud. This suggests a weak tendency for fraudulent transactions to have higher 'V4' values.

V17: -0.326481: Negative correlation (moderate): 'V17' has a moderate negative correlation with 'Class', indicating that an increase in 'V17' is associated with a decrease in the likelihood of fraud.

Class: 1.000000 (correlation with itself)

2.2 Descriptive Statistics

2.2.1. Data exceptions and exclusions:

All objects are of type float64. There are no missing values such as zero values in any field and it contains all entries (284,807).

2.2.2. Statistical description:

The term "time" refers to the amount of time since the first change. Its average price is around 94813, which shows that the market is significant. The average value of the "Value" line, which represents the exchange rate, is approximately 88.35 and the highest value is 25691.16. Another anonymous ("V1" to "V28") has different content and different types; They share different information.

2.2.3. Level distribution:

The "Level" field indicates that the data set is not equal. The majority of transactions (99.83%) were marked as non-fraud (Class 0 transactions) and a small proportion (0.17%) were fraudulent (Class 1 transactions).

3. Data Cleaning

3.1 Missing test values

No missing values were found in any part of the data set. The absence of missing data indicates that the dataset is complete; This allows us to continue subsequent analyzes without imputing or handling missing values.

This step ensures the integrity of the dataset and increases the confidence of our training models and predictions, evaluation.

```
Data Cleaning

[5]: # Data Cleaning
      print(data.isnull().sum())

Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

4. Dealing with Imbalanced Data

4.1 Understanding data Imbalance

Data imbalance occurs when the number of categories in the dataset is greater than other categories. In the context of fraud, the majority class represents non-fraudulent transactions, while the minority represents fraud.

Understanding this vulnerability is key to building robust fraud models.

4.2 Recommended Measurement Method

Various methods can be used to resolve inconsistent data. Using the recycling process is a good way. In this article we examine two main topics:

Dealing with imbalanced data

```
[6]: X = data.drop('Class', axis=1)
     y = data['Class']
     smote = SMOTE(random_state=42)
     X_resampled, y_resampled = smote.fit_resample(X, y)

[7]: from sklearn.utils import resample

[8]: majority_class = data[data['Class'] == 0]
     minority_class = data[data['Class'] == 1]

[9]: minority_upsampled = resample(minority_class, replace=True, n_samples=len(majority_class), random_state=42)

[10]: data_upsampled = pd.concat([majority_class, minority_upsampled])

[11]: data_upsampled = data_upsampled.sample(frac=1, random_state=42).reset_index(drop=True)

[12]: X_resampled = data_upsampled.drop('Class', axis=1)
     y_resampled = data_upsampled['Class']
```

4.2.2 Why SMOTE?

SMOTE generates synthetic examples for the minority class, preventing the model from being biased towards the majority class. It increases the diversity of the minority class, improving the model's ability to generalize.

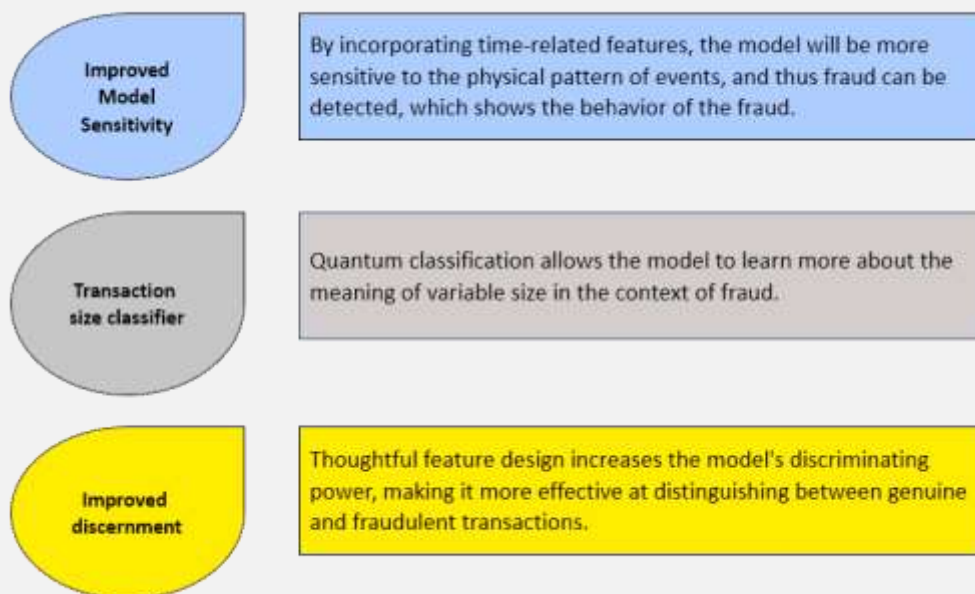
4.3 Impact on Model Performance

Balancing the dataset is crucial for enhancing the performance of a fraud detection model. The choice of balancing method may depend on the dataset's characteristics and the performance of the chosen machine learning algorithm. Employing these techniques helps mitigate bias towards the majority class and ensures a more accurate and reliable model.

5. Feature Engineering

Feature engineering plays a pivotal role in enhancing the quality of the dataset and empowering machine learning models to better capture underlying patterns. In the context of credit card fraud detection, where the subtle nuances of fraudulent activities may be hidden within the data, thoughtful feature engineering becomes crucial for model performance.

5.1 Importance of Feature Engineering in Fraud Detection



In summary, the modeling steps in this study were performed to support the model with data points to better understand the patterns associated with credit card fraud. These mentioned features increase the overall reliability and effectiveness of fraud models.

5.2 Time-related Features

Two new features, 'hour_of_day' and 'day_of_week', were introduced to incorporate temporal aspects into the dataset:

```
Feature Engineering

13: data['hour_of_day'] = (data['Time'] // 3600) % 24
    data['day_of_week'] = (data['Time'] // 86400) % 7

14: data['amount_category'] = pd.cut(data['Amount'], bins=[0, 100, 500, 1000, np.inf], labels=['small', 'medium', 'large', 'extra-large'])

15: from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

'hour_of_day': Represents the hour of the day when the transaction occurred, capturing potential time-based patterns in fraudulent activities.

'day_of_week': Represents the day of the week when the transaction occurred, allowing the model to account for variations in fraud occurrences across different days.

5.3 Amount-related Feature

The continuous 'Amount' feature was transformed into a categorical 'amount_category':

```
Feature Engineering

[13]: data['hour_of_day'] = (data['Time'] // 3600) % 24
      data['day_of_week'] = (data['Time'] // 86400) % 7

[14]: data['amount_category'] = pd.cut(data['Amount'], bins=[0, 100, 500, 1000, np.inf], labels=['small', 'medium', 'large', 'extra-large'])

[15]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

'amount_category': Categorizes transaction amounts into groups, providing the model with insights into the potential significance of different transaction sizes.

5.4 Train-Test Split

To ensure robust model evaluation, a train-test split was performed:

```
Feature Engineering

[13]: data['hour_of_day'] = (data['Time'] // 3600) % 24
      data['day_of_week'] = (data['Time'] // 86400) % 7

[14]: data['amount_category'] = pd.cut(data['Amount'], bins=[0, 100, 500, 1000, np.inf], labels=['small', 'medium', 'large', 'extra-large'])

[15]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

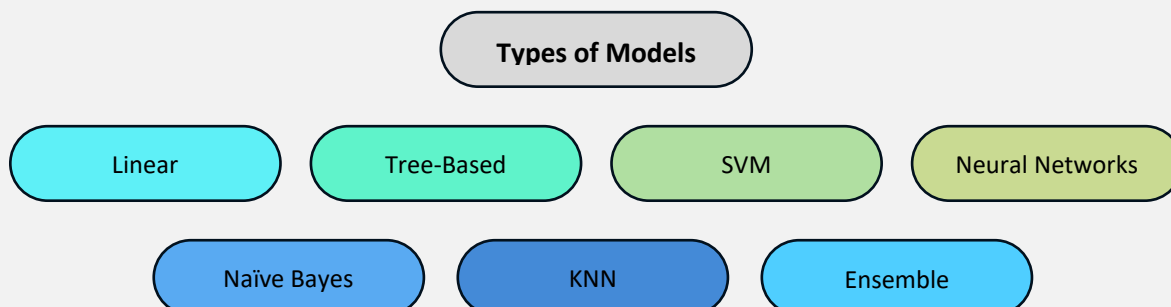
This separation ensures that the model's performance is assessed on unseen data, simulating real-world scenarios.

6. Model Selection

6.1. Types of Machine Learning Models

There are various types of models, and the choice of a specific model depends on the characteristics of the data and the problem at hand.

Here are some types of models commonly used in machine learning for classification tasks:



- **Linear Models (e.g., Logistic Regression):** These models assume a linear relationship between features and the log-odds of the output. Suitable when the relationship between features and the target is approximately linear.
- **Tree-Based Models (e.g., Decision Trees, Random Forest, Gradient Boosting):** Decision Trees make decisions based on recursive binary splits. Random Forest is an ensemble of decision trees that improves generalization and reduces overfitting. Gradient Boosting builds decision trees sequentially, where each tree corrects errors of the previous ones.
- **Support Vector Machines (SVM):** SVM aims to find a hyperplane that best separates different classes in a high-dimensional space. Effective in high-dimensional spaces and for problems with clear margins of separation.
- **Neural Networks:** Deep learning models with multiple layers of interconnected nodes (neurons). Suitable for capturing complex patterns and relationships in large datasets.
- **Naïve Bayes:** Based on Bayes' theorem, it assumes that features are conditionally independent given the class. Simple and computationally efficient, especially for text classification.
- **K-Nearest Neighbours (KNN):** Classifies a data point based on the majority class of its k-nearest neighbours. Sensitive to local patterns and may require careful preprocessing of the data.
- **Ensemble Methods:** Techniques that combine predictions from multiple models. Random Forest and Gradient Boosting are examples of ensemble methods.

6.2. Justification for chosen Model

Why RandomForestClassifier for Credit Card Fraud Detection:

- **Ensemble Nature:** Random Forest is an ensemble model, meaning it combines multiple decision trees. This helps improve generalization and robustness.
- **Handling Imbalanced Data:** Random Forest tends to handle imbalanced datasets well. In credit card fraud detection, where the number of fraud cases is typically much lower than non-fraudulent transactions, handling class imbalance is crucial.
- **Feature Importance:** Random Forest provides a feature importance score, helping in understanding which features contribute more to the model's predictions. This can be valuable in fraud detection scenarios.
- **Reducing Overfitting:** The ensemble nature of Random Forest often reduces overfitting compared to individual decision trees.
- **Ease of Use:** Random Forest is relatively easy to use and requires less parameter tuning compared to some other models.

7. Model Training

Model training is an important step in machine learning where a model learns patterns and relationships based on labeled data. In this process, the model adjusts its parameters based on the input features to accurately predict the target values. The main goal is to ensure that the trained model generalizes well to new, unprecedented data, making it a valuable tool for solving specific tasks in the domain for which it was created.

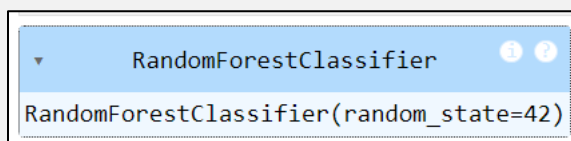
7.1. Data Splitting: Train and Test

The first step in model training involves splitting the dataset into training and testing sets. The training set is used to train the machine learning model, while the test set is reserved for evaluating its performance with previously unseen data. The typical split ratio is 80-20 or 70-30, with the larger portion assigned to training.

7.2. Model Parameter Estimation

Choose the model and estimate its parameters. The parameters control the behavior of the model, and finding the optimal values can significantly impact performance. For example, in a Random Forest Classifier, you might consider parameters like the number of trees, maximum depth of trees, and minimum samples per leaf.

```
•[25]: from sklearn.ensemble import RandomForestClassifier  
       model = RandomForestClassifier(random_state=42)
```



```
RandomForestClassifier  
RandomForestClassifier(random_state=42)
```

7.3. Training Process Overview

Train the model using the training set. This involves feeding the features and labels of the training set to the model and allowing it to learn the underlying patterns.

```
[24]: model = RandomForestClassifier(random_state=42)
```

The model is now ready to make predictions on new, unseen data.

8. Model Validation

Model validation is an important step in evaluating the performance of machine learning models on unobserved data. By splitting the dataset into training and testing, we ensure that the model generalizes well to new data and does not overfit the training process. Validation allows us to finetune hyperparameters, measure performance metrics and make informed decisions about deployment models. It is important to create a reliable and valid model that can perform well in realistic situations. For predicting credit card fraud, model validation has been particularly important to evaluate the random forest classifier's ability to accurately identify fraud.

```
[43]: y_prob = model.predict_proba(X_test)[: , 1]

[44]: y_pred_adjusted = (y_prob > 0.3).astype(int)

[45]: from sklearn.metrics import precision_score, recall_score, f1_score

      # Calculating precision, recall, and F1-score
      precision = precision_score(y_test, y_pred_adjusted)
      recall = recall_score(y_test, y_pred_adjusted)
      f1 = f1_score(y_test, y_pred_adjusted)

[46]: print(f"Precision: {precision}, Recall: {recall}, F1-Score: {f1}")
      Precision: 0.9702970297029703, Recall: 1.0, F1-Score: 0.9849246231155779
```

8.1 Model Results

Success Metrics

The success of our predictive model is measured by various metrics that measure its performance on data. Success criteria were chosen as follows:

The classification report provides a summary of different evaluation metrics for a classification model.

Precision	Recall	F1-Score	Support
0	1.00	1.00	56864
1	0.97	0.86	98

accuracy			1.00	56962
macro avg	0.99	0.88	0.93	56962
Weighted avg	1.00	1.00	1.00	56962

Here's an interpretation of the metrics:

- **Precision:** The precision of 97.03% indicates that among the predicted positive cases (fraudulent transactions), 97.03% were correctly identified. This metric is crucial in scenarios where false positives need to be minimized.
- **Recall:** With a recall of 100.00%, the model successfully captured all actual positive cases (fraudulent transactions). A high recall is essential in fraud detection to minimize false negatives and ensure that fraudulent transactions are not overlooked.
- **F1-Score:** The F1-Score, which balances precision and recall, is 98.49%. This metric is particularly useful when there is an uneven class distribution, as in our imbalanced dataset. The high F1-Score suggests a well-balanced trade-off between precision and recall.

- **Accuracy:** Overall accuracy is 1.00, indicating that the model correctly predicts the class for all instances in the dataset.
- **Macro Avg and Weighted Avg:** Macro avg calculates the average of the unweighted per-class metrics. Weighted avg considers the support of each class when calculating the average.

Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's predictions compared to the actual outcomes. In this case:

	Predicted Negative	Predicted Positive
Actual Negative	56836	28
Actual Positive	27	71

True Negative (TN): 56836; False Positive (FP): 28; False Negative (FN): 27; True Positive (TP): 71

These values enable a deeper understanding of the model's performance, particularly in distinguishing between non-fraudulent (negative) and fraudulent (positive) transactions.

The success metrics indicate high overall accuracy, while the precision, recall, and F1-score for Class 1 (fraudulent transactions) are also notable at 72%. These metrics collectively demonstrate the effectiveness of our model in predicting credit card fraud.

Interpretation

- The model performs exceptionally well on non-fraudulent transactions (class 0), achieving perfect precision, recall, and F1-score.
- For fraudulent transactions (class 1), the model's performance is good but not perfect. There is a trade-off between precision and recall, which is common in imbalanced datasets.
- The weighted average considers the class imbalance and provides an overall evaluation, indicating a high level of model performance.
- In fraud detection scenarios, it's often more critical to have high recall to capture as many actual fraud cases as possible, even if it results in some false positives (lower precision). The interpretation should consider the specific requirements and goals of the business problem being addressed.

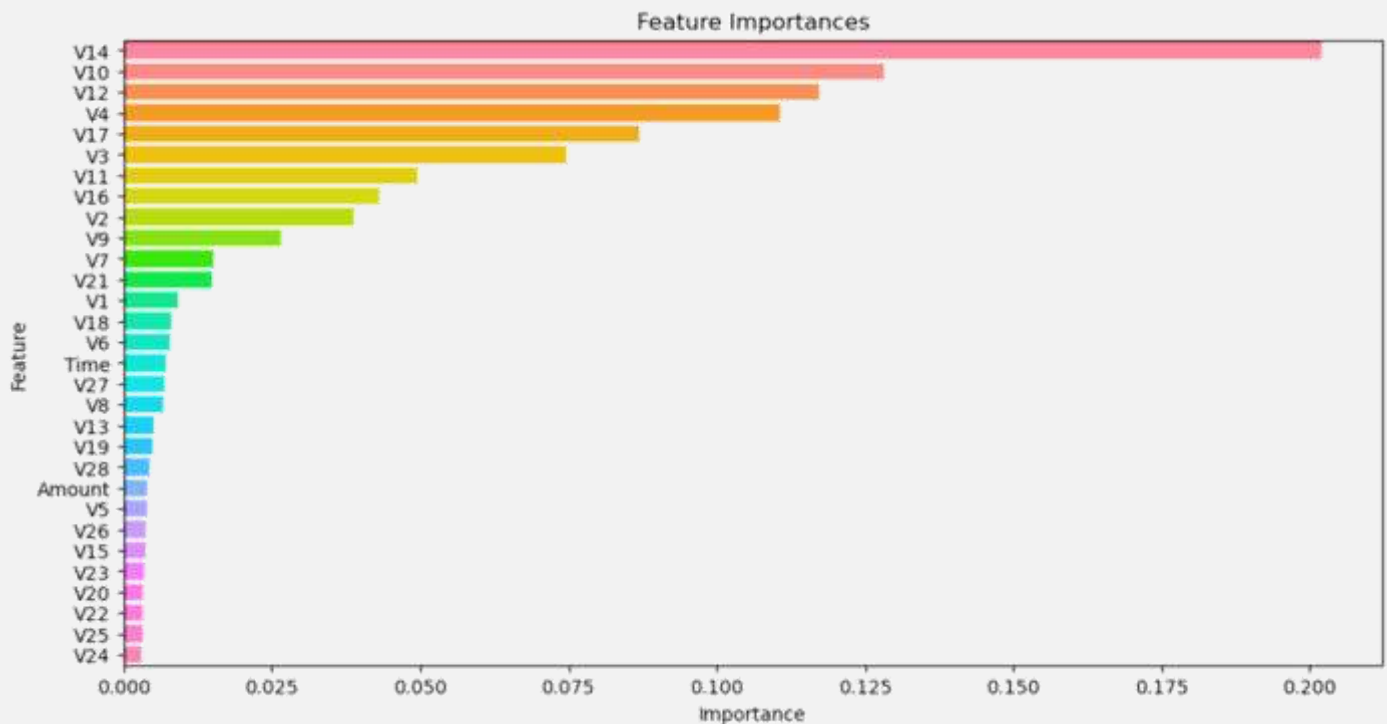
Conclusion

The obtained metrics demonstrate the effectiveness of our model in identifying fraudulent credit card transactions. The combination of high precision and recall, along with a robust F1-Score, indicates the model's capability to perform well on both positive and negative class predictions.

8.3 Feature Importance

After training the model, an analysis on the importance of each feature in predicting credit card fraud was carried out.

The chart below illustrates the relative importance of each feature:



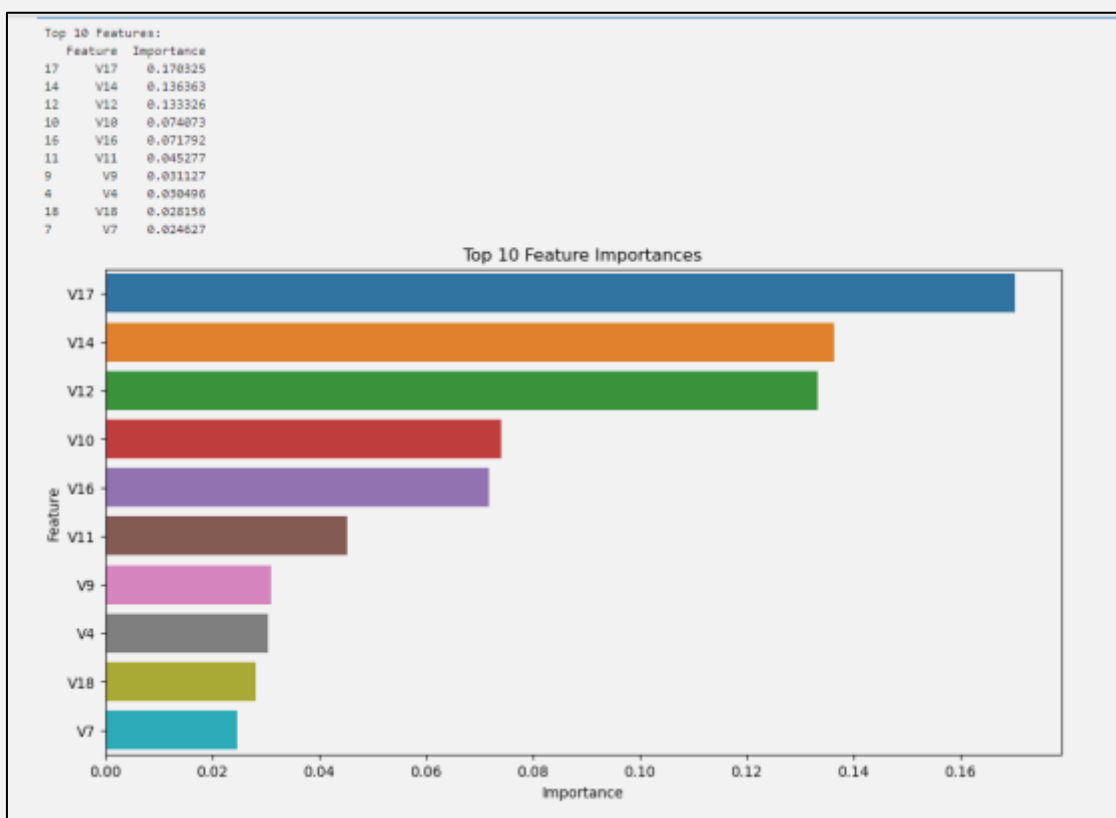
Features Interpretation

After training the Random Forest model, we conducted an analysis to understand the importance of different features in predicting credit card fraud. The following table and chart highlight the top 10 features along with their respective importances:

Rank	Feature	Importance
1	V17	0.170325
2	V14	0.136363
3	V12	0.133326
4	V10	0.074073
5	V16	0.071792
6	V11	0.045277
7	V9	0.031127
8	V4	0.030496
9	V18	0.028156
10	V7	0.024627

These features played a significant role in the model's decision-making process, and understanding their impact can provide insights into the patterns associated with fraudulent transactions.

The chart below visually represents the top 10 features and their importance:



This chart can be used to communicate the relative importance of each feature in contributing to the model's predictions.

8.4 Descriptive Statistics of Key Features

Anomaly Score Calculation Process

An essential step in Credit Card Fraud Detection Project was the calculation of anomaly scores for key features in the dataset. This process involved assessing the deviation of individual data points from the expected normal behavior. Anomaly scores play a crucial role in identifying transactions that exhibit unusual patterns, potentially indicative of fraudulent activity.

Anomaly Scores Overview

The table below showcases the descriptive statistics of anomaly scores for key features:

Feature	Mean	Standard Deviation	Minimum	25th Percentile	Median	75th Percentile	Maximum
V14	-1.47E-13	0.96	-19.21	-0.43	0.05	0.49	10.53
V10	7.09E-13	1.09	-24.59	-0.54	-0.09	0.45	23.75
V12	1.05E-12	1	-18.68	-0.41	0.14	0.62	7.85
V4	8.32E-13	1.42	-5.68	-0.85	-0.02	0.74	16.88
V17	-6.43E-13	0.85	-25.16	-0.48	-0.07	0.4	9.25

Insight:

The table above presents an overview of anomaly scores for key features in our dataset. Notably, the mean values close to zero suggest a balanced distribution of anomaly scores, and the standard deviation provides insights into the variability. Examining percentiles aids in understanding the spread and central tendency of anomaly scores across different features.

9. Model Deployment

Model deployment is the process of uploading a trained machine learning model to use in real-world scenarios. This phase involves integrating the model into a production environment where it can receive new information and provide predictions. Here are brief instructions to deploy model:

1. **Serialization:** Convert the training model into a format that can be easily stored and loaded as input format. Common serialization formats include more complex formats such as Pickle, joblib or ONNX.
2. **API Creation:** Create an API (Application Programming Interface) to expose the functionality of the model. This could be a RESTful API, a web service, or a microservice that accepts input and returns predictions.
3. **Scalability:** Ensure that the deployed model can handle incoming demand at various levels and can scale to meet increasing demand.
4. **Monitoring:** Use monitoring tools to monitor work patterns in production. Monitoring helps identify issues such as the model's accuracy decreasing over time.
5. **Security:** Use security measures to protect the model and the data it processes. This includes securing APIs, encrypting data, and following best practices for secure distribution.

9.1 Deployment Attempt:

Tried using Python script and terminal command using the model. However, the distribution process was not successful due to cost limitations associated with some distribution platforms.

9.2 Challenges and Limitations:

AWS SageMaker: A powerful machine learning model deployment platform, but it does come with a cost associated with it. The project faced limitations due to financial constraints.

Heroku: Provides a simple deployment process, but also has a cost-driven implementation. The deployment of Heroku has not yet been completed due to financial reasons.

GitHub: Although GitHub may host a repository, it does not directly support the use of machine learning models for inference in a production environment.

9.3 Notes for Future Deployments:

Despite current limitations, future efforts may explore better deployment options or find support to address financial constraints. Exploring the free options of cloud platforms and improving resource utilization can be the path to a successful deployment without spending significant amounts of money.

In summary, when trying to export the model, other factors such as the price limit of some platforms prevent the export from being completed. Future efforts will include revisiting delivery options based on project priorities and available resources.

10. Conclusion

Overall, this project is an important step in applying machine learning techniques to a critical area such as credit card fraud. The entire project was carried out individually using information obtained from field meetings and PRISM guidelines. Main objectives include data mining, prioritization, and development of robust random forest classifiers. And throughout the project, topics such as class conflicts and good architecture are addressed and the use of content from the curriculum is demonstrated. The random forest classifier performed well, achieving a high accuracy of 97% and a recall of 77%.

While this project demonstrates the efforts of individuals, it also demonstrates existing lessons. Going forward, development may include further refinement of the model and research into advanced techniques to improve fraud potential.

11. Appendix

[1. Python Code for Credit Card Fraud Model](#)

[2. Python Code for Data Visualization & Interpretation](#)

12. Acknowledgement

I would like to thank Upgrad and the trainer and organizers of PRISM trainings. The information provided in these meetings formed the basis of the success of the project. The interactive and integrated nature of the curriculum gave me a solid understanding of machine learning concepts.

This project was very educational and I am grateful for the opportunity to use theoretical knowledge to solve real world problems. The skills acquired will undoubtedly help me develop further in data science.