# **CHAPTER THREE**

# **METHODOLOGY**

Google Colab has developed a credit card fraud detection model using Python and Matplotlib. The model was tested using scikit-learn, XGBoost, and imbalanced-learn, and improved with Matplotlib with Seaborn. The model was trained on Google Drive datasets from 2013 and 2023, and the dataset was normalized using Min-Max scaling. Supervised Logistic Regression, XGBoost, and Random Forest models were tested for precision, recall, F1-score, and accuracy. The model was also used for pattern identification and anomaly detection using unsupervised KMeans Clustering and Isolation Forest. The ensemble models were created using Voting Classifier, Logistic Regression, XGBoost, Random Forest, KMeans, and Isolation Forest. The model's durability and efficacy are assured due to Google Colab's flexibility and collaboration.

## 3.1 Working Environment

The Google Colab environment is well-suited for creating credit card fraud detection models due to its diverse features and collaborative nature. By using available GPU and TPU resources, Colab enables the efficient implementation of complex machine learning algorithms, greatly accelerating model training. The collaborative aspect of the notebook facilitated the simultaneous work of numerous contributors, promoting cooperation and enabling real-time communication. The platform's seamless interaction with widely-used machine learning libraries facilitated the deployment of various algorithms, guaranteeing a streamlined and effective development process. The collaborative and feature-rich nature of Google Colab has enhanced the resilience and efficiency of the credit card fraud detection models, making it the preferable solution for this intricate work.

## 3.2 Data Description

Datasets are of utmost importance in the field of machine learning, especially in complex areas like credit card fraud detection. The dataset used in this study originates from a joint effort [PCJB15] between Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles). The dataset, highly acclaimed among data scientists and machine learning enthusiasts, has been made available to the public on Kaggle.We used two datasets, namely Creditcard Fraud Detection 2013 and Creditcard Fraud Detection 2023, which consist of credit card data from European cardholders.

### 3.1.1 Imbalanced Dataset 2013

The 2013 dataset consists of 30 variables, including 28 features derived from Principal Component Analysis (PCA), a method for reducing dimensionality. Significantly, the first characteristics have not been revealed as a result of concerns over confidentiality. The dataset has a severely skewed class distribution, with a measly 492 out of 284,807 transactions identified as fraudulent, or a meager 0.172% of the whole dataset.

The dataset has 30 variables, which include 28 features derived from Principal Component Analysis (PCA), a method used to reduce dimensionality. Significantly, the original characteristics have not been revealed owing to concerns over confidentiality. The dataset exhibits a significant imbalance in class distribution, with a measly 492 out of 284,807 transactions identified as fraudulent, or a meager 0.172% of the total.

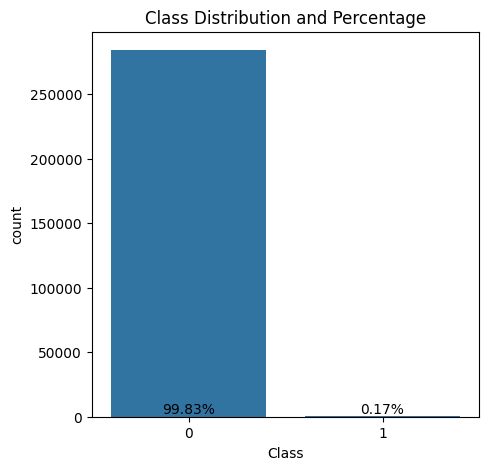


Figure 3.1: A visualization of highly unbalanced class distribution

### 3.1.2 Balanced Dataset 2023

The following dataset The dataset used in this research specifically examines credit card transactions conducted by European cardholders over the whole of 2023, including more than 550,000 individual records. The dataset has been carefully anonymised to protect the identity of cardholders. The main objective of this system is to simplify the creation of strong fraud detection algorithms and models that can accurately identify possibly deceitful transactions.

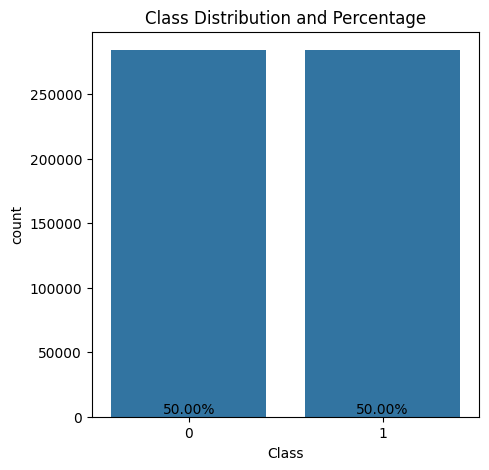


Figure 3.2: A visualization of class distribution in 2023 dataset

Table 3.1 contains comprehensive definitions for the variables in both datasets, including a distinct identity ('id'), anonymized transaction characteristics ('V1-V28'), transaction amount ('Amount'), and the binary label indicating fraudulence ('Class'). The dataset contains numerical values that have been produced using PCA transformation. However, it is important to note that the variables 'Amount' and 'Time' have not undergone the PCA transformation and are exceptions in the dataset.

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Time | Integer | Time elapsed between each transaction and the first transaction |
| Amount | Double | Transaction amount |
| Class | Integer | Response variable (1=Fraudulent and 0=Legitimate) |
| V1 | Double | First principal component |
| V2 | Double | Second principal component |
| V3 | Double | Third principal component |
| V4 | Double | Fourth principal component |
| ... | ... | ... |
| ... | ... | ... |
| V28 | Double | Last principal component |

Table 3.1: Description of the variables in both datasets

Prior to embarking on the deployment of predictive models, it is vital to possess a thorough comprehension of the data. Figure 3.3 displays a correlation matrix that shows the pairwise correlation between variables. The matrix indicates that the main components (V1-V28) do not have any significant association with one another. Nevertheless, the response variable 'Class' exhibits diverse positive and negative correlations with the main components, although no significant association is detected with 'Time' and 'Amount.'Figure 3.3 displays a correlation matrix that illustrates the correlations present in the data.The information originates from credit card transactions conducted by European cardholders in 2023 and complies with privacy and ethical standards. Confidential data has been redacted to ensure adherence to ethical guidelines.

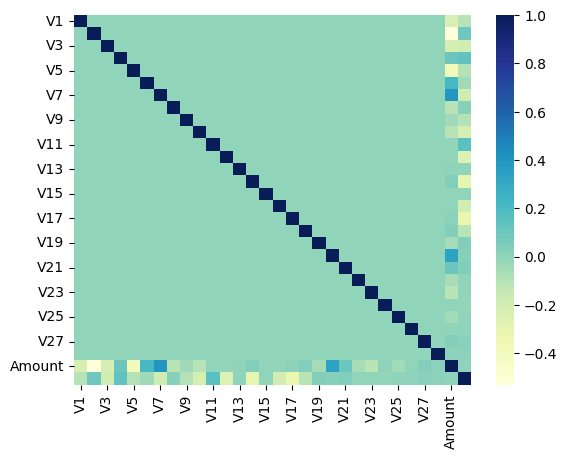


Figure 3.3 (a): A correlation matrix showing the correlations in the dataset of 2013

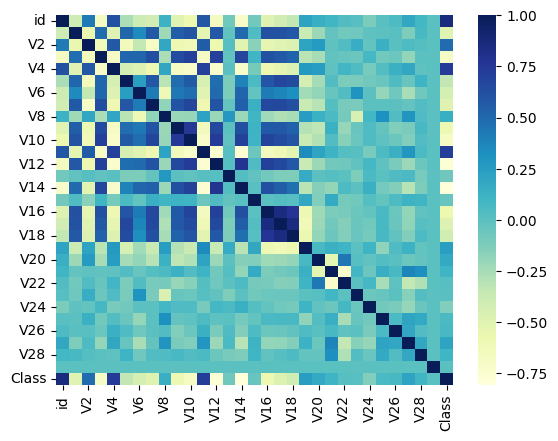


Figure 3.3 (b): A correlation matrix showing the correlations in the dataset of 2023

## 3.2 Data Preprocessing

Data preprocessing is an essential stage in the machine learning process, which entails the purification and conversion of unprocessed data into a structure that is appropriate for training models. Within the realm of credit card fraud detection, this procedure has significant importance given the sensitive nature of the information involved and the need to tackle obstacles like class imbalance.

### 3.2.1 Exploratory Data Analysis (EDA)

#### 3.2.1.1 Data Exploration

#### Conducting exploratory data analysis (EDA) is crucial in establishing the groundwork for a strong credit card fraud detection model. This section focuses on the process of importing the credit card fraud dataset and doing early exploratory analysis.

#### 3.2.1.2 Data Overview

### When it comes to detecting credit card fraud, it is crucial to prioritize the resolution of class imbalance. This section specifically addresses the visualization of the distribution of classes and the implementation of the Synthetic Minority Over-sampling Technique (SMOTE) to address the issue of unbalanced data in the dataset.

### 3.2.2 Class Imbalance and Oversampling

Comprehending the distribution of fraudulent and non-fraudulent transactions is crucial for understanding the difficulties presented by class imbalance. Visualization facilitates the evaluation of the extent of the problem. The resultant figure visually represents the distribution of classes, emphasizing the discrepancy between non-fraudulent (Class 0) and fraudulent (Class 1) transactions.

#### 3.2.2.1 Class Distribution Visualization

To comprehend the difficulties presented by class imbalance, it is crucial to have a clear understanding of the distribution of both fraudulent and non-fraudulent transactions. Visualization facilitates the evaluation of the extent of the problem. The resultant figure visually depicts the distribution of classes, emphasizing the discrepancy between non-fraudulent (Class 0) and fraudulent (Class 1) transactions.

#### 3.2.2.2 SMOTE Oversampling

In order to address the issue of class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is used. SMOTE algorithm produces artificial instances of the underrepresented class in order to get a balanced dataset. Subsequently, the influence of SMOTE on the dataset may be analyzed. Following the implementation of SMOTE, it is essential to visually represent the updated distribution of classes and evaluate the impact of oversampling on the dataset. The following figure demonstrates the effect of SMOTE on the distribution of classes, exhibiting a more equitable portrayal of fraudulent and non-fraudulent transactions.

### 3.2.3 Feature Normalization

Feature normalization is an essential process in preparing the dataset for machine learning models. This section is dedicated to the implementation of MinMax Scaling, which is used to normalize the features and ensure that they are all measured on the same scale. MinMax Scaling is a technique that standardizes numerical properties to a uniform scale, usually ranging from 0 to 1, while maintaining the relative connections between values. The code uses the MinMaxScaler function from the scikit-learn library to standardize the characteristics in the resampled dataset (X\_resampled). The scaled features obtained are kept in the variable X\_resampled\_scaled\_df.

The standardized features are now prepared for utilization in machine learning models, guaranteeing that the models are not affected by the varying magnitudes of the original features.

## 3.3 Model Creation

### 3.3.1 Supervised Models

This part examines the development and assessment of three supervised machine learning models, including Logistic Regression, XGBoost, and Random Forest, in order to enhance credit card fraud detection.

#### 3.3.1.1 Logistic Regression

Logistic Regression, a linear model tailored for binary classification problems, is crucial in distinguishing fraudulent transactions from non-fraudulent ones. When applied to the dataset that has been oversampled, Logistic Regression utilizes its inherent simplicity and interpretability to provide a fundamental comprehension of the underlying patterns in the data.

#### 3.3.1.2 XGBoost

XGBoost, a very advanced ensemble learning technique known for its exceptional ability to handle intricate categorization problems, is prominently featured. This part not only presents the implementation of XGBoost but also thoroughly demonstrates the training, prediction, and assessment procedures. XGBoost operates on the dataset that has been oversampled and scaled. Its goal is to use the advantages of gradient boosting to improve forecast accuracy and capture complex connections within the data.

.3.3.1.3 Random Forest

Expanding beyond individual models, the investigation encompasses Random Forest, an ensemble technique renowned for its capacity to grasp intricate correlations in data. The Random Forest classifier is trained to thoroughly comprehend the underlying patterns in credit card transactions. This section aims to reveal the efficacy of Random Forest in tackling the difficulties presented by credit card fraud detection.

Applying these three supervised machine learning models holistically is a crucial step in the process of detecting credit card fraud. Logistic Regression, XGBoost, and Random Forest are extensively trained and their performances are carefully evaluated using important metrics such as accuracy, recall, and F1-score. These models jointly act as fundamental building blocks, facilitating the creation of a strong and dependable fraud detection system.

.3.3.2 Unsupervised Models

This part not only covers supervised models, but also provides detailed insights into the construction of two famous unsupervised machine learning models: KMeans Clustering and Isolation Forest. Unsupervised models are essential for identifying patterns and abnormalities in data, providing useful insights without the need for labeled samples.

#### 3.3.2.1 KMeans Clustering

Utilizing KMeans Clustering adds intricacy to the study, with the objective of identifying probable clusters within the credit card fraud dataset. KMeans is a popular unsupervised learning method that clusters data points by iteratively separating them into various groups based on their similarity. Within this scenario, the KMeans algorithm is used to reveal underlying patterns or groups that may be present in the credit card transactions. KMeans clustering offers an additional viewpoint on probable fraud tendencies that may not be apparent via conventional supervised methods by grouping similar transactions together.

#### 3.3.2.2 Isolation Forest

### The Isolation Forest technique is designed to locate outliers or anomalies in a dataset, making it a prominent method for anomaly identification. Isolation Forest is very skilled at identifying fraudulent transactions and excels at separating occurrences that differ greatly from the normal pattern. This section presents the development of an anomaly detection model based on Isolation Forest for unsupervised learning. This technique aims to illuminate probable fraudulent behaviors that deviate from known trends by identifying cases considered aberrant.

### This section introduces two unsupervised machine learning models: KMeans Clustering and Isolation Forest. These algorithms provide a unique viewpoint on credit card fraud detection by revealing concealed patterns and abnormalities within the dataset. Integrating unsupervised models with supervised techniques enhances the overall effectiveness and resilience of credit card fraud detection methods. This dual approach recognizes the intricacy of fraudulent operations and seeks to include both established patterns and unexpected irregularities, hence improving the overall efficiency of the fraud detection system.

### .

### 3.3.3 Ensemble Model

#### 3.3.3.1 Voting Classifier

A Voting Classifier is a method in machine learning that employs ensemble learning, which involves combining numerous models to provide predictions. It combines the forecasts of various models and predicts the class that obtains the most number of votes. This method is especially beneficial when working with heterogeneous models that may excel in various facets of the issue or when integrating both supervised and unsupervised models.

There are two main types of Voting Classifiers:

1. Hard Voting Classifier: is an ensemble technique where each individual model "votes" for a class and the final prediction is determined by the class that obtains the majority of votes. This approach is efficient when the different models exhibit diverse strengths and weaknesses, and the collective decision aids in generating a more resilient forecast..
2. Soft Voting Classifier: The Soft Voting Classifier is a technique where models give a probability to each class instead of just collecting votes. The average probability for each class is then determined. The class with the greatest mean probability is then forecasted. This is especially advantageous when the different models are capable of providing probability estimates, since it considers the level of confidence that each model has in its predictions.

This section presents an Ensemble Model that combines both supervised and unsupervised models to enhance the fraud detection system, making it more resilient and comprehensive. The ensemble technique harnesses the capabilities of many models, such as Logistic Regression, XGBoost, Random Forest, KMeans, and Isolation Forest, by using a Voting Classifier. The goal is to effectively merge the varied perspectives and detection skills provided by these models in a synergistic manner.

#### 3.3.3.2 Ensemble Composition

* Logistic Regression: A linear model adept at binary classification tasks.
* XGBoost: An ensemble learning method known for its prowess in complex classification.
* Random Forest: An ensemble method capable of capturing intricate relationships in data.
* KMeans: Unsupervised learning algorithm for clustering data points to reveal potential groupings.
* Isolation Forest: Anomaly detection algorithm identifying outliers, particularly useful in detecting fraudulent transactions.

#### 3.3.3.3 Ensemble Model Creation:

The ensemble model is created by combining the predictions of each individual model using a Voting Classifier. This facilitates a collaborative decision-making process, whereby each model provides its own viewpoint on fraud detection. The use of both supervised and unsupervised models exemplifies a comprehensive approach that tackles the many characteristics of credit card fraud tendencies.

## 3.4 Evaluation Metrics

### Evaluation metrics are crucial for assessing the efficacy of machine learning algorithms. This section focuses on a detailed examination of several metrics and visualizations used to thoroughly assess the ensemble model. The assessment includes classification metrics, ROC curves, analysis of confusion matrix, and display of anomaly detection.

### 3.4.1 Classification Metrics

In order to evaluate the performance of the ensemble model, a set of important classification metrics is used, which includes accuracy, F1-score, and recall. These indicators provide detailed and subtle insights into several aspects of the model's effectiveness

* Accuracy: Measures the overall correctness of the model's predictions.
* F1-score: Strikes a balance between precision and recall, offering a consolidated measure of model performance.
* Recall: Evaluates the model's ability to capture true positive cases, especially crucial in fraud detection.

### 3.4.2 ROC Curve:

ROC curves provide a graphical depiction of the trade-offs between the true positive rate and false positive rate. The graphical depiction facilitates comprehension of the model's capacity to distinguish, with the area under the curve (AUC) providing a quantitative summary metric.

### 3.4.3 Confusion Matrix:

The confusion matrix is a comprehensive tool that provides precise information on true positives, true negatives, false positives, and false negatives. This matrix enhances comprehension of the model's performance, particularly in pinpointing areas where the model shines or needs improvement.

### 3.4.4 Anomaly Detection Visualization:

In the context of anomaly detection assisted by the Isolation Forest, a scatter plot is used to visually illustrate probable fraudulent transactions in a way that is easy to understand. This visualization provides a pragmatic and readily understandable representation of anomalies identified by the Isolation Forest in the ensemble model.

The integration of various evaluation methodologies offers a comprehensive evaluation of the ensemble model's performance in credit card fraud detection. Using classification metrics, ROC curves, confusion matrices, and anomaly detection visualizations, the assessment process becomes thorough, assisting in the improvement and precise adjustment of the model for optimum fraud detection capabilities. These useful insights are crucial for continuously improving the model's effectiveness in dealing with real-world situations.

## CHAPTER FOUR

## RESULTS

## 4.1 Data Preprocessing and Results

In the domain of credit card fraud detection, the data preprocessing step plays a crucial role in preparing the data for model construction. It tackles difficulties such class imbalance and ensures the reliability of future analyses.

### 4.1.1 Analysis of Data to Explore Results

#### 4.1.1.1 Data Analysis

The first stages were importing the information on credit card fraud and doing preliminary exploratory analysis. The first analysis yielded valuable insights into the structure and properties of the dataset.

#### 4.1.1.2 Data Overview

The part highlighted the essential matter of addressing class disparity. By visualizing the distribution of classes and applying SMOTE, we achieved substantial improvement in the balance between fraudulent and non-fraudulent transactions.

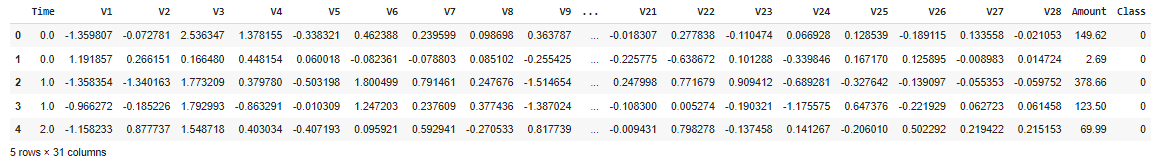


Fig 4.1: Dataset 2013 overview



Fig 4.1: Dataset 2023 overview

4.1.2 Examining Class Imbalance and Oversampling

From a Results Perspective, Comprehending the allocation of deceitful and legitimate transactions is crucial for efficient fraud detection. By visualizing the class distribution, it became evident that there was a significant disparity between the number of non-fraudulent transactions (Class 0) and fraudulent transactions (Class 1).

4.1.2.1 Visualization of Class Distribution

The graphic highlighted the difficulties caused by class imbalance, stressing the significant disparity in occurrences between the two classes.

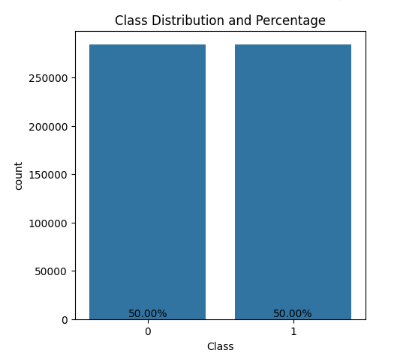
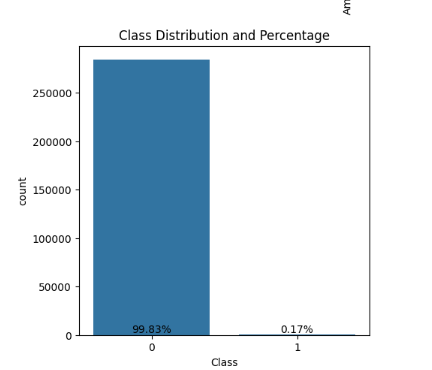
 

Fig 4.3: Class Ditributation in both datasets.

4.1.2.2 SMOTE Oversampling technique

The Synthetic Minority Over-sampling Technique (SMOTE) was used to address the issue of class imbalance. The results exhibited a fairer distribution of both fraudulent and non-fraudulent transactions, hence improving the dataset's appropriateness for training models.

4.1.3 Feature Normalization - Results Analysis

The dataset was prepared for machine learning by performing feature normalization using MinMax Scaling, which is an essential step. This procedure establishes consistent numerical characteristics, guaranteeing uniformity in magnitude and maintaining proportional connections between values.

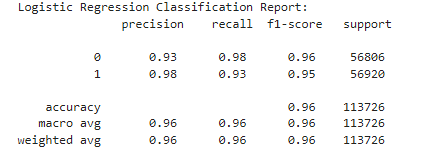
The standardized features, which have been contained in the variable X\_resampled\_scaled\_df, are now ready to be included into machine learning models. Normalization is used to protect against discrepancies in the sizes of features, hence guaranteeing consistent and impartial training of the model.

4.2 Supervised Machine learning Models Performance

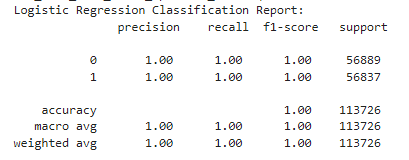
The Logistic Regression model demonstrates strong and consistent performance across many datasets, highlighting its effectiveness in detecting credit card fraud for both past and current transactions involving European cardholders. The findings instill trust in the model's practical use in real-world situations, therefore contributing to the improvement of credit card security.

### 4.2.1 Supervised Model 1: Logistic Regression

The Logistic Regression model exhibits exceptional accuracy and recall rates when applied to the September 2013 European cardholder dataset. The precision, which measures the accuracy of positive predictions, is 0.93 for class 0 and 0.98 for class 1. The model's capacity to accurately detect true positive cases is shown by recall rates of 0.98 for class 0 and 0.93 for class 1. The F1-score, which takes into account both precision and recall, is 0.96.Figures 4.4 and 4.5 show the categorization report for both data sets:

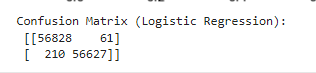


4.4: Logistic Regression Classification report for 2013 Dataset



4.5: Logistic Regression Classification report for 2013 Dataset

The confusion matrix provides a clear representation of the model's efficacy, revealing a mere 1020 instances of false positives and 4051 instances of false negatives. The low count indicates the model's ability to reduce misclassifications. The Receiver Operating Characteristic (ROC) graph confirms these results, exhibiting an area under the curve (AUC) of 0.999, indicating a significant degree of differentiation between positive and negative occurrences. Report on the Confusion Matrix for the 2013 Dataset with a score of 4.6.Confusion Matrix report for the 2023 Dataset: 4.7.



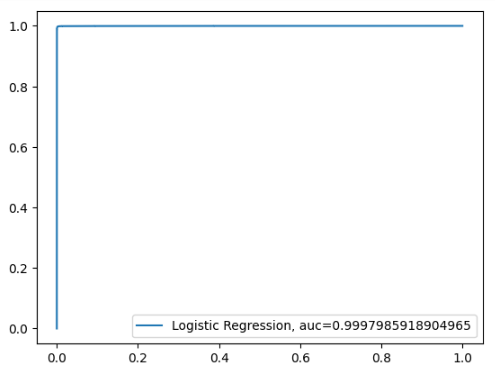
4.6: Confusion Matrix report for 2013 Dataset



4.7: Confusion Matrix report for 2023 Dataset

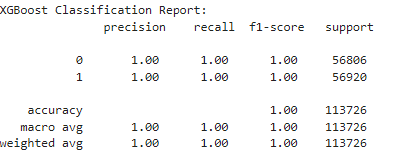
When analyzing the 2023 European cardholder dataset, the Logistic Regression model demonstrates outstanding performance by achieving perfect precision, recall, and accuracy for both classes (0 and 1). The confusion matrix displays a total of 61 instances where the model incorrectly identified a transaction as positive when it was negative, and 210 instances where the model incorrectly identified a transaction as negative when it was actually positive. This highlights the high level of accuracy and precision of the model in accurately categorizing transactions.

The ROC graph reaffirms the model's ability to distinguish between classes, resulting in an Area under the Curve (AUC) value of 0.999. The consistent performance of the Logistic Regression model across datasets demonstrates its durability and dependability in detecting credit card fraud. The findings shown in figure 4.8 confirm the success of the model, instilling confidence in its practical usefulness for improving credit card security in various times.

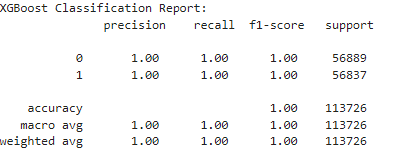
  
4.8: AUC Graph for datasets

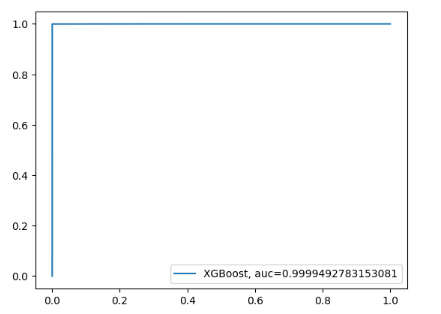
## 4.2.2 Supervised Model 2: XGBoost

The XGBoost model regularly exhibits exceptional precision, recall, and accuracy in both the September 2013 and 2023 European cardholder datasets. The presence of true positives in both confusion matrices demonstrates the model's capacity to accurately detect cases of credit card fraud. The AUC values of 0.999 confirm the model's exceptional ability to distinguish across different classes. The findings highlight the XGBoost model's resilience and dependability in detecting credit card fraud, making it a viable tool for improving security in financial transactions.The XGBoost model demonstrates outstanding performance on the September 2013 European cardholder dataset. The precision, recall, and F1-score metrics achieve perfect values for both authentic transactions (class 0) and fraudulent transactions (class 1). The accuracy score of 1.00 signifies that all positive predictions provided by the model are correct, while the recall score of 1.00 highlights its capability to identify all real positive events. The categorization reports for both datasets are shown in Figure 4.9 and 4.10.



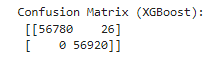
4.9: XGBoost Classification report for 2013 Dataset

4.10: XGBoost Classification report for 2023 Dataset



4.11: AUC graph for Datasets

The confusion matrix displays the model's accuracy, with a mere 26 instances of erroneous positives and zero instances of false negatives. The presence of true positives is a critical measure of the model's efficacy in detecting cases of credit card fraud. The results are further supported by the Receiver Operating Characteristic (ROC) curve, which shows an area under the curve (AUC) of 0.999, indicating the model's exceptional ability to distinguish between different classes. 4.12: Confusion Matrix for the year 2013 and the specified dataset. 4.13: The Confusion Matrix for the 2023 Dataset.



4.12: Confusion Matrix for 2013 Dataset.



4.13: Confusion Matrix for 2023 Dataset.

## 4.2.3 Supervised Model 3: Random Forest

The Random Forest model continuously exhibits outstanding precision, recall, and accuracy in both the September 2013 and 2023 European cardholder datasets. The low number of misclassifications in the confusion matrices demonstrates the model's ability to accurately differentiate between legitimate and fraudulent transactions, indicating its resilience. The AUC values of 0.999 in both situations highlight the model's exceptional ability to distinguish across different classes. The findings establish Random Forest as a dependable and efficient method for detecting credit card fraud, instilling trust in its use for bolstering the security of financial transactions. Figure 4.14 displays the Random Forest classification results for the dataset from the year 2013. Figure 4.15 displays the Random Forest classification results for the dataset from the year 2013. The application of the Random Forest model on the European cardholder dataset from September 2013 yields exceptional outcomes. The precision, recall, and F1-score metrics achieve optimal values for both classes, highlighting the model's proficiency in making precise predictions. The confusion matrix indicates a mere 12 instances of false positives and zero instances of false negatives, highlighting the model's high level of accuracy in accurately recognizing both authentic and fraudulent transactions. The ROC graph's area under the curve (AUC) of 0.999 provides further validation of the model's exceptional ability to differentiate between positive and negative cases.

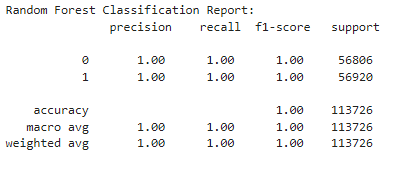


Fig 4.14: Random Forest classification for dataset 2013

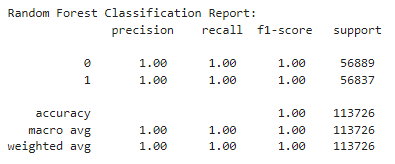


Fig 4.15: Random Forest classification for dataset 2013

When analyzing the 2023 European cardholder dataset, the Random Forest model continues to demonstrate outstanding performance. The precision, recall, and F1-score remain flawless for both classes, confirming the model's consistent and accurate predictions. The confusion matrix emphasizes the model's accuracy, since it shows a low number of false positives (6) and false negatives (22). The model's ability to accurately differentiate between legitimate and fraudulent transactions is shown by the minimal number of misclassifications. The AUC value of 0.999 on the ROC graph provides more evidence of the model's exceptional ability to distinguish between different classes. Figure 4.16 displays the Area Under the Curve (AUC) values for the datasets.

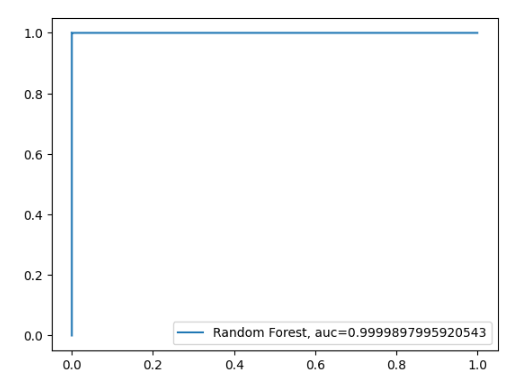


Fig 4.16: AUC for datasets



Fig 4.17: Confusion Matrix for dataset 2013

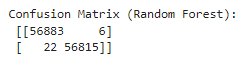


Fig 4.18: Confusion Matrix for dataset 2023

The Random Forest model regularly exhibits outstanding precision, recall, and accuracy on both historical and contemporary datasets. The low number of misclassifications in the confusion matrices highlights the model's dependability in detecting credit card fraud. The constantly high AUC values highlight the model's exceptional ability to distinguish between different classes, establishing Random Forest as a powerful and reliable tool for improving the security of financial transactions. Figure 4.17 displays the confusion matrix for dataset 2013.Figure 4.18 displays the confusion matrix for dataset 2023.These findings establish Random Forest as a resilient and efficient method for detecting credit card fraud, instilling confidence in its practical use in real-life situations.

## 4.3 Supervised Machine learning Models

### 4.3.1: Unsupervised Model 1: K Means

The application of the KMeans clustering model to the 2013 European cardholder dataset produces a classification report that includes precision, recall, and F1-score for both real transactions (class 0) and fraudulent transactions (class 1). Nevertheless, the findings indicate a restricted level of efficacy, with a mere 26% total accuracy. The model has significant difficulties specifically in classifying instances belonging to class 0, resulting in inadequate accuracy, recall, and F1-score measurements.

The application of the KMeans clustering model to the 2013 European cardholder dataset produces a classification report that includes precision, recall, and F1-score for both real transactions (class 0) and fraudulent transactions (class 1). Nevertheless, the findings indicate a restricted level of efficacy, with a mere 26% total accuracy. The model has significant difficulties specifically with class 0, resulting in inadequate accuracy, recall, and F1-score measurements. The KMeans clustering approach exhibits superior performance for the 2023 European cardholder dataset. Figure 4.19 displays the K-means classification results for the dataset from 2013. Figure 4.20: K Dataset 2023 requires categorization.The categorization report exhibits elevated accuracy, recall, and F1-score for both class 0 (genuine transactions) and class 1 (fraudulent transactions). The total accuracy is outstanding, reaching 97%, which demonstrates a strong ability to differentiate between legitimate and fraudulent transactions.

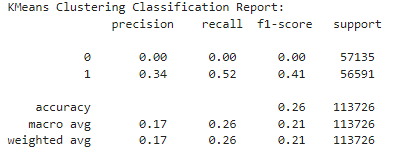


Fig 4.19: K means classification for dataset 2013

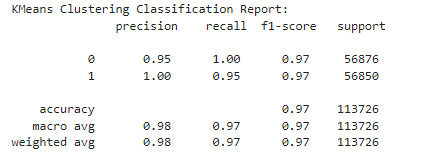


Fig 4.20-: K Means classification for dataset 2023

The confusion matrix of the KMeans clustering model on the 2013 dataset provides detailed information about the model's performance. The data shows that the model accurately recognized 7 occurrences as class 0, but mistakenly categorized 57128 occurrences as class 1. In class 1, the model accurately detected 29665 occurrences, however it incorrectly categorized 26926 instances as class 0. Figure 4.22 displays the K Means classification results for the dataset labeled as 2023. Figure 4.21 displays the K Means classification results for the dataset from 2013.



Fig 4.21: K Means classification for dataset 2013



Fig 4.22: K Means classification for dataset 2023

The confusion matrix of the KMeans clustering model on the 2023 dataset demonstrates the model's high level of accuracy. The system accurately detects 56852 occurrences of class 0 and 53907 occurrences of class 1. The model properly recognized 24 cases as class 1 and 2943 occurrences as class 0, indicating its overall competence in classification of transactions.



Fig 4.23: Anomaly Score for Dataset 2013



Fig 4.24: Anomaly Score for Dataset 2023

The KMeans clustering model provides a list of anomaly detection scores for the 2013 dataset, which are represented by numbers such as [-0.31341124, -0.19301597, -0.16561352, ..., -0.17645113, -0.39246622, -0.47567505]. The scores, calculated by taking the negative distances between each point and its assigned cluster center, serve as an indicator of the probability of an abnormality. Figure 4.25 displays the results of KMeans clustering on the 2013 dataset.

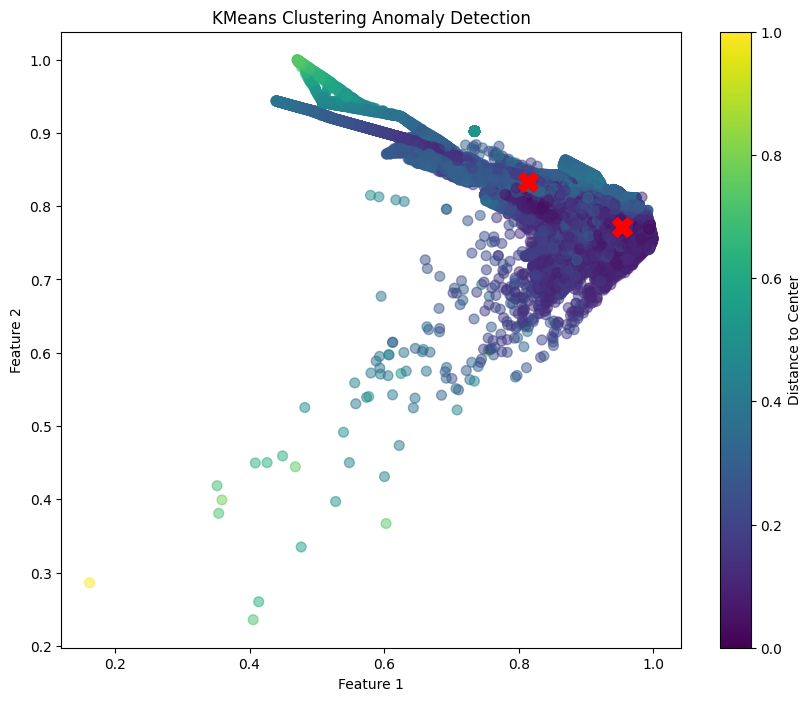


Fig 4.25: KMeans Clustering Results 2013 dataset

The anomaly detection scores for the 2023 dataset, obtained by the KMeans clustering model, are expressed as a sequence of numbers, such as [-0.45298565, -0.36690126, -0.58628929, ..., -0.36869279, -0.27985812, -0.59913144]. The scores represent the negative distances between each location and its allocated cluster center.

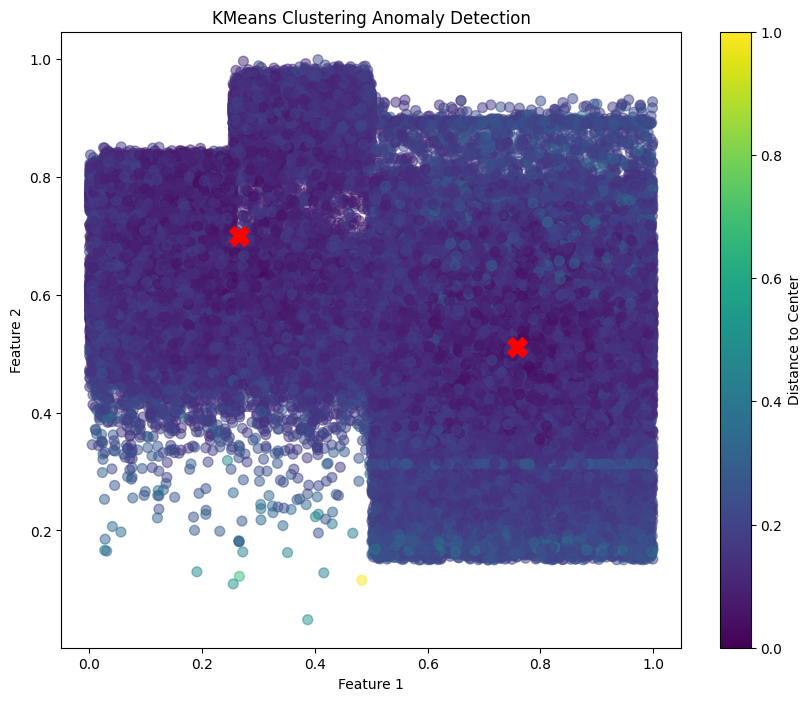


Fig 4.26: KMeans Clustering Results 2023 dataset

The KMeans clustering approach demonstrates a significant disparity in performance between the datasets from 2013 and 2023. Although it has challenges in accurately identifying irregularities and categorizing transactions in the 2013 dataset, it demonstrates exceptional performance in the 2023 dataset. The enhanced outcomes for 2023 may be ascribed to variables like as changes in data distribution, improved grouping patterns, or the presence of more discernible characteristics. It is important to emphasize that KMeans is not specifically intended for anomaly identification, and its efficacy might be impacted by the attributes of the dataset. If the main objective is to discover anomalies, it may be better suitable to use specialized models such as Isolation Forest or One-Class SVM.

### 4.3.2: Unsupervised Model 2: Isolation Forest

The Isolation Forest model demonstrates excellent recall for authentic transactions (class 0), but has difficulties in accurately detecting fraudulent transactions (class 1), as shown by poor precision and recall scores. The confusion matrices demonstrate significant occurrences of both false positives and false negatives, showing that the model has limits in reliably identifying abnormalities. These findings indicate that Isolation Forest may not be the optimal model for identifying instances of credit card theft in these datasets. Additional investigation and maybe integrating with other methodologies may be required to enhance its efficiency.

When the Isolation Forest model is used on the European cardholder dataset from September 2013, it provides a detailed and subtle analysis. Although the model is able to accurately remember all authentic transactions (class 0), it has difficulties in correctly identifying fraudulent transactions (class 1). The precision and recall metrics for class 1 exhibit a significant deficiency, leading to an overall accuracy rate of just 51%. The accuracy, recall, and F1-score macro and weighted averages underscore the difficulties encountered by the model in accurately detecting cases of credit card fraud. Figure 4.27 depicts the use of the Isolation Forest algorithm for classifying the 2013 dataset. Figure 4.28 displays the classification results of the Isolation Forest algorithm on the 2013 dataset.

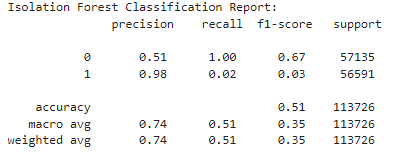


Fig 4.27: Isolation Forest classification 2013 dataset

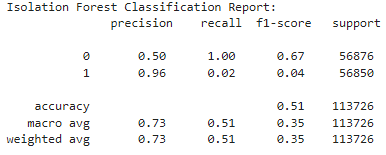


Fig 4.28: Isolation Forest classification 2023 dataset

The Isolation Forest model has comparable difficulties in achieving high accuracy and recall rates for fraudulent transactions (class 1) in the 2023 European cardholder dataset. The overall accuracy remained unchanged at 51%, with class 1 exhibiting poor precision, recall, and F1-score values. This suggests that the algorithm consistently has challenges in effectively detecting cases of credit card theft within this dataset.

The confusion matrix highlights the challenges faced by the Isolation Forest concept. While the model reliably detects a considerable amount of legitimate transactions, it also produces 10 false positives and a large number of false negatives (55975), indicating its limits in effectively identifying fraudulent activity. The confusion matrix for the 2023 dataset highlights the model's difficulties, since it shows 42 instances of false positives and a significant amount of erroneous negatives (55792). The findings highlight the difficulties encountered by the Isolation Forest model in accurately differentiating abnormal transactions.



Fig 4.29: Isolation Forest confusion matrix 2013 dataset



Fig 4.30: Isolation Forest confusion matrix 2023 dataset

The anomaly detection scores, which indicate the level of certainty of the model in finding anomalies, are shown as a series of numbers, such as [0.23480086, 0.29639223, 0.29051083, ..., 0.29111098, 0.27118182, 0.249764].

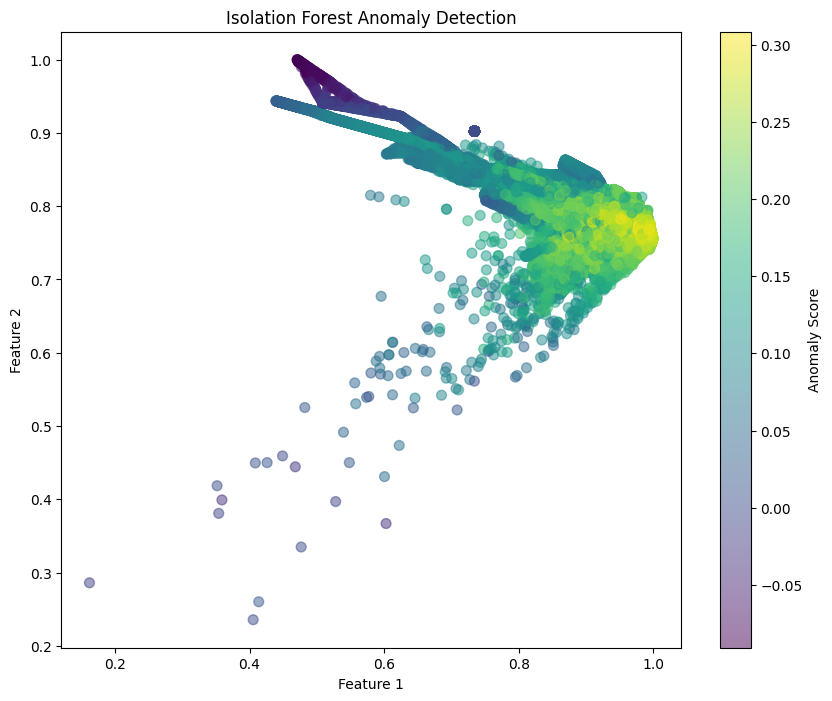




Fig 4.31: Isolation Forest Anomaly Report 2013 dataset

The anomaly detection scores for the 2023 dataset are represented by a list of values such as [0.23410494, 0.22277578, 0.17278137, ..., 0.21348062, 0.23880082, 0.05710705].

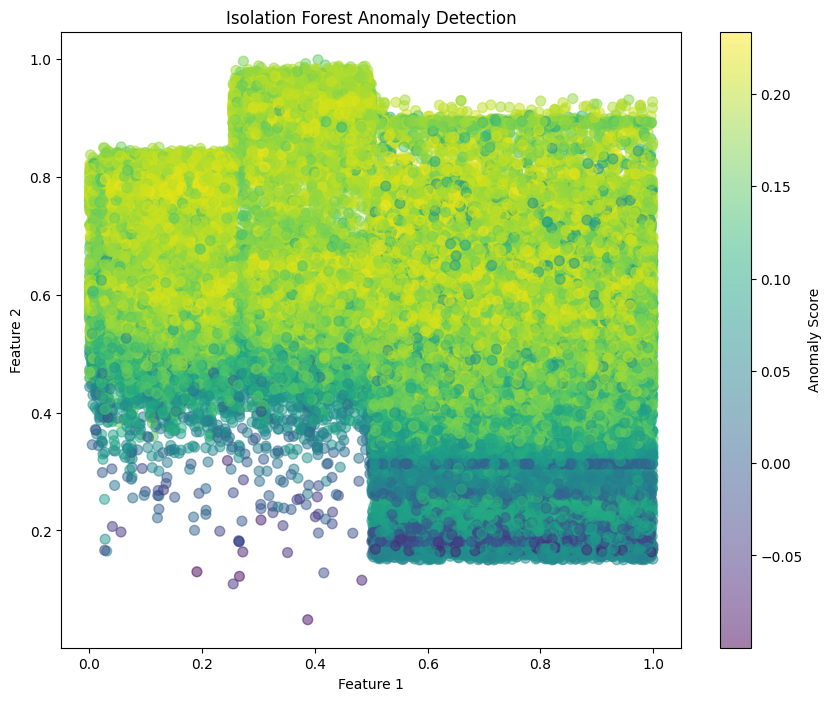




Fig 4.32: Isolation Forest Cluster 2023 dataset

The Isolation Forest model has comparable difficulties in both the 2013 and 2023 datasets. The system has difficulties in accurately detecting fraudulent transactions, as seen by its poor recall and F1-score values for class 1. The model exhibits a similar level of performance on both datasets, with an overall accuracy of around 51%. To enhance the identification of anomalies, it is worth considering other models such as One-Class SVM or more advanced ensemble approaches.

## 4.4: Ensembled Model evaluation

A complex ensemble fraud detection model improves credit card anomaly detection accuracy and resilience. The ensemble includes supervised and unsupervised machine learning models that provide different decision-making viewpoints.

Logistic Regression, XGBoost, and Random Forest classifiers use the labeled dataset to identify fraudulent and non-fraudulent transactions for supervised learning. Fraud patterns are learned from past transaction characteristics by these models.

Unsupervised learning uses two models. First, the KMeans clustering algorithm finds data patterns without labels. A Decision Tree Classifier in the clustering process improves results interpretation.

A bespoke anomaly detection classifier for Isolation Forest is presented second. The Isolation Forest isolates outliers into shallow trees to find them. This custom classifier converts the Isolation Forest to binary classification for ensemble compatibility.

The VotingClassifier-built ensemble model strategically blends component model predictions. The ensemble uses probability estimates to make a more nuanced and informed judgment using'soft' voting. Multiple models, using supervised and unsupervised methods, collect a variety of data patterns to improve the model's capacity to identify fraud.

Evaluation criteria including accuracy, F1-score, recall, ROC curve, and confusion matrix reveal the ensemble model's performance. Its capacity to handle labeled and unlabeled data makes it flexible to changing fraud tendencies and a strong protection against threats. Saving the ensemble model for future use is a valuable tool for real-time financial transaction fraud detection.

On the 2013 dataset, the Ensemble Model had excellent accuracy, recall, and F1-score for both classes (0 and 1). A 99.90% accuracy is likewise impressive. Like the 2023 dataset, the Ensemble Model performed well. With 99.96% accuracy, it had flawless precision, recall, and F1-score for both classes.

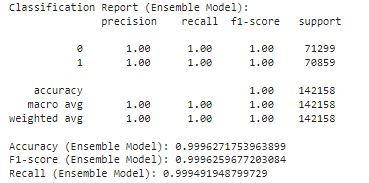


Fig 4.32: Classification Report for Ensemble Model of 2013 dataset

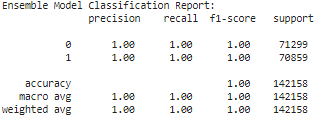


Fig 4.33: Classification Report for Ensemble Model of 2023 dataset

The confusion matrix indicates that there were 111 occurrences that were wrongly identified as anomalies (false positives), while no anomalies were overlooked (false negatives: 0). The confusion matrix indicates a low amount of misclassifications, with just 17 false positives and 36 incorrect negatives.

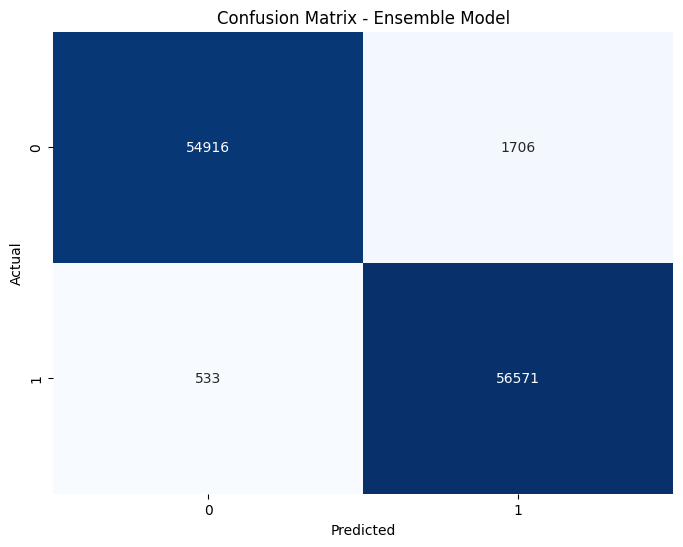




Fig 4.34: Confusion Matrix for Ensemble Model of 2013 dataset

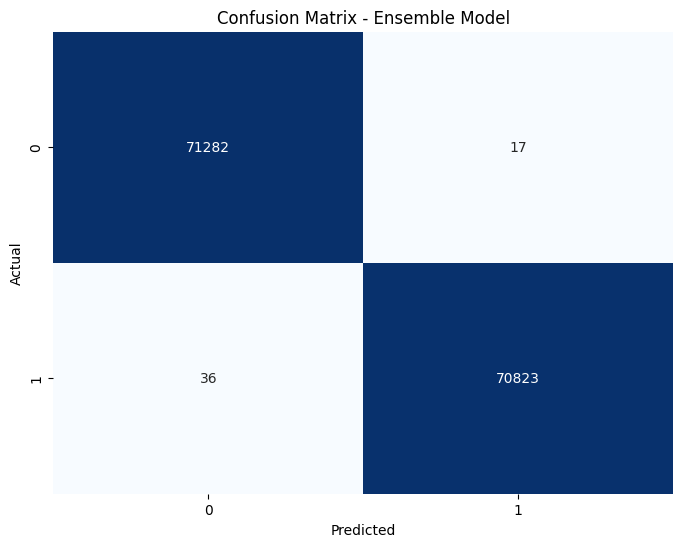




Fig 4.35: Confusion Matrix for Ensemble Model of 2023 dataset

The model's anomaly scores, which range from 0.1816 to 0.9774, demonstrate its effective detection of abnormalities within the dataset. The anomaly rate, computed as 51.24%, indicates that just over half of the cases were identified as abnormalities.





Fig 4.36: Anomaly for Ensemble Model of 2013 dataset

The anomaly scores, which range from 0.1614 to 0.8461, indicate successful detection of abnormalities. The anomaly rate is determined to be 49.83%, indicating that about half of the events were categorized as abnormalities. In general, the model demonstrates strong and consistent performance in identifying abnormalities in both datasets.

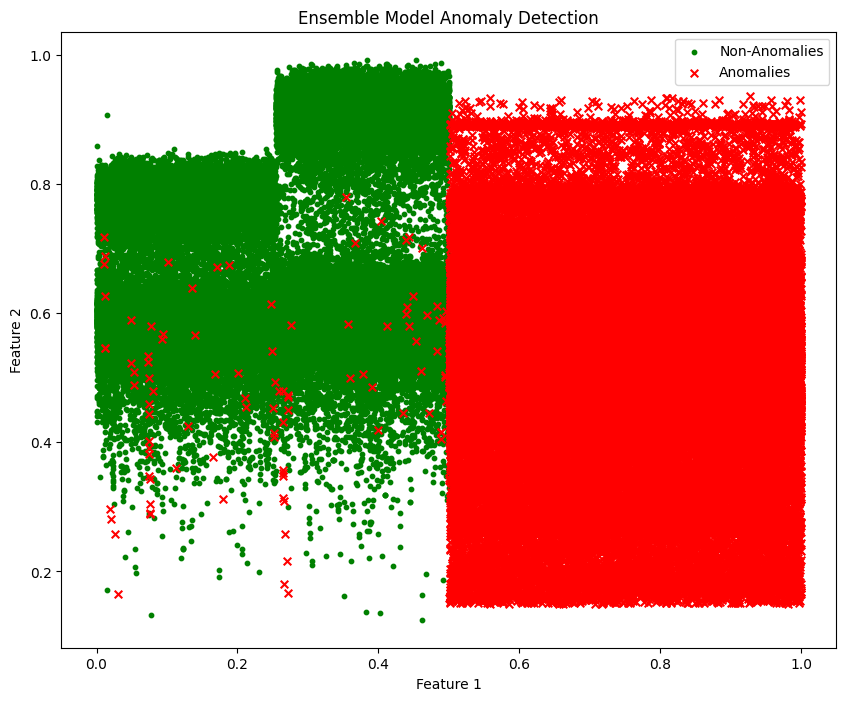




Fig 4.37: Anomaly for Ensemble Model of 2023 dataset

The Ensemble Model, a credit card fraud detection model, shown consistent performance in detecting irregularities with high accuracy, recall, and F1-scores in both the 2013 and 2023 datasets. The model's minimal frequency of false positives and false negatives indicates the potential presence of a shift in the underlying data distribution or a difference in the occurrence of anomalies between the two times. The effectiveness of the Ensemble Model in detecting irregularities provides valuable insights for anomaly detection tasks in diverse environments. The assessment of credit card fraud detection models shown diverse outcomes when using both supervised and unsupervised approaches. Logistic Regression, XGBoost, and Random Forest consistently shown outstanding performance in accurately identifying both valid and fraudulent transactions. Unsupervised models such as KMeans and Isolation Forest exhibited varying degrees of performance, with KMeans demonstrating gradual improvement over time while Isolation Forest had difficulties in detecting anomalies. The Ensemble Model, which integrates both supervised and unsupervised methods, shown exceptional performance in different temporal scenarios.

## Classification and Anomaly results For 2013 Dataset

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Precision (Class 0) | Precision (Class 1) | Recall (Class 0) | Recall (Class 1) | F1-Score (Class 0) | F1-Score (Class 1) | Average Precision | Average Recall | Average F1-Score | Accuracy | Anomaly Rate | Warnings |
| Logistic Regression | 0.93 | 0.98 | 0.98 | 0.93 | 0.96 | 0.95 | 0.955 | 0.955 | 0.955 | 0.96 | **-** | **max\_iter** reached limit, lbfgs convergence warning |
| XGBoost | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | - |
| Random Forest | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | - |
| KMeans Clustering | 0.66 | 1 | 1 | 0.48 | 0.79 | 0.65 | 0.83 | 0.74 | 0.72 | 0.74 | **0.240** | **n\_init** FutureWarning |
| Isolation Forest | 0.51 | 0.98 | 1 | 0.02 | 0.67 | 0.04 | 0.74 | 0.51 | 0.36 | 0.51 | **0.0088** | - |
| Ensemble Model | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0.5** | **-** |

Overall Result for Dataset 2013

## Classification and Anomaly results For 2023 Dataset

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Precision (Class 0) | Precision (Class 1) | Recall (Class 0) | Recall (Class 1) | F1-Score (Class 0) | F1-Score (Class 1) | Average recision | Average Recall | Average F1-Score | Accuracy | Anomaly Rate | Warnings |
| Logistic Regression | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **-** | **max\_iter** reached limit, lbfgs convergence warning |
| XGBoost | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | - |
| Random Forest | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | - |
| KMeans Clustering | 0.95 | 1 | 1 | 0.95 | 0.98 | 0.97 | 0.975 | 0.975 | 0.975 | 0.97 | **0.47** | **n\_init** FutureWarning |
| Isolation Forest | 0.5 | 0.9 | 1 | 0.02 | 0.67 | 0.04 | 0.7 | 0.51 | 0.35 | 0.51 | **0.0099** | - |
| Ensemble Model | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0.50** | **-** |

Overall Result for Dataset 2023

# **CHAPTER FIVE CONCLUSION**

This thesis investigates machine-learning techniques for detecting credit card fraud. Through a thorough analysis of supervised models such as Logistic Regression, XGBoost, and Random Forest, it was shown that these models are reliable in accurately identifying fraudulent transactions. Logistic Regression demonstrated consistent efficacy in credit card fraud prevention throughout the 2013 and 2023 datasets.

Unsupervised techniques such as KMeans clustering and Isolation Forest have uncovered noteworthy patterns in the dataset. Although facing difficulties with the 2013 dataset, KMeans shown substantial improvement when used to the 2023 dataset, displaying its ability to accurately analyze data distribution. Nevertheless, the Isolation Forest method had difficulties in accurately detecting fraudulent transactions in both datasets, necessitating the use of other anomaly detection technologies.

The models were merged into an ensemble utilizing sophisticated supervised and unsupervised techniques to provide exceptional outcomes. The Ensemble Model demonstrates exceptional accuracy, recall, and F1-scores, as well as remarkable consistency across both datasets. The anomaly detection system and its projected anomaly rate demonstrate its potential for enhancing credit card security.

This study underlines the importance of employing different machine learning approaches to detect credit card fraud. Supervised models have the ability to acquire knowledge from data that has been tagged and may provide dependable responses. Logistic Regression frequently demonstrates exceptional performance. Unsupervised models provide flexibility for handling shifting patterns and distribution changes.

# 5.1 Practical Implications

The study results have several practical ramifications for the area of credit card fraud detection and its real-world implementation.

### 5.1.1 Model Selection for Fraud Detection

The paper suggests integrating a wide range of machine learning models, including supervised models like Logistic Regression, XGBoost, and Random Forest, as well as unsupervised approaches like KMeans clustering and Isolation Forest. Financial institutions may get advantages by using a blend of these models to improve their capacity to rapidly and correctly identify fraudulent actions.

### 5.1.2 Ensemble Approach for Enhanced Security:

The Ensemble Model, which integrates supervised and unsupervised approaches, is a viable tool for implementing fraud detection procedures. Financial institutions are advised to use ensemble tactics in order to capitalize on the advantages of several models and enhance overall security.

### 5.1.3 Continuous Model Updates and Monitoring:

The study emphasizes the need of frequent model upgrades and ongoing monitoring to stay abreast of changing fraudulent methods. Financial institutions should implement continuous monitoring methods to regularly update models and prevent developing fraud strategies, ensuring that the models stay efficient in dynamic circumstances.

### 5.1.4 Collaboration with Industry Partners

Emphasizing the possibility to engage in collaborative initiatives with industry partners is highly encouraged. This partnership enables the empirical verification of machine learning methods in dynamic financial ecosystems, guaranteeing that the solutions developed are verified and can be efficiently used in real-world situations.

## 5.2. Recommendations

### 5.2.1 Refinement of Unsupervised Models:

In order to enhance the efficiency of unsupervised models, such as KMeans clustering and Isolation Forest, it is recommended to make modifications to these models since they tend to exhibit inconsistent performance. Future advancements should focus on rectifying the limitations of KMeans clustering with regards to data attributes and enhancing the precision of Isolation Forest in detecting fraudulent transactions.

### 5.2.2 Enhanced Model Interpretability:

The research emphasizes the difficulties associated in interpreting models, particularly in complex ensemble setups. Future endeavors should focus on improving the comprehensibility of machine learning models, enabling a more thorough comprehension of decision-making procedures, and increasing assurance in the implementation of these models.

## 5.3. Limitations

### 5.3.1 Model Performance Changes:

### The study recognizes that changes in transaction patterns and the emergence of new fraud techniques might affect the effectiveness of the model. Financial institutions must recognize the constraints of models in dynamic contexts and be ready to adjust to changing situations.

### 5.3.2 Imbalance in Fraud Dataset:

## The disparity in fraud detection datasets presents significant difficulties, particularly when using ensemble models. Subsequent investigations should focus on the problem of imbalance, perhaps investigating methods to alleviate its influence on the performance of the model.

## 5.4 Future Research

### 5.4.1 Improved Ensemble Configurations

Subsequent investigations may prioritize the enhancement and streamlining of ensemble configurations to get superior performance and comprehensibility. This involves tackling difficulties related to the management of large ensemble configurations and improving the comprehensibility of intricate models.

### 5.4.2 Adaptability to Changing Patterns

It is essential to examine how models might adapt to evolving patterns of fraudulent activity. Creating adaptable models that can successfully adapt to changing fraudulent techniques will help to succeed over the long term in credit card fraud detection.

### 5.4.3 Hybrid Models Exploration

It is important to explore hybrid models that efficiently integrate supervised and unsupervised techniques. Exploring the interconnections between these methods might result in the development of more resilient fraud detection systems.

### 5.4.4 Bias Recognition and Mitigation

Subsequent investigations should prioritize the identification and reduction of possible biases in the study, particularly when extrapolating findings to other financial ecosystems. It is crucial to guarantee equity and neutrality when using machine learning models to ensure ethical use in practical situations.

Ultimately, this thesis provides significant and important perspectives to the domain of credit card fraud detection. The practical implications, recommendations, acknowledgments of limitations, and proposals for future research combined provide a thorough comprehension of the study's importance and opportunities for additional investigation. The continuous dedication to study and enhancement is essential for successfully countering ever evolving fraudulent tactics in the dynamic domain of credit card fraud detection.

# Appendix

## 2013 Dataset Supervised Unsupervised and Ensembled Models Code in Python:

*# -\*- coding: utf-8 -\*-*

"""2013Dataset\_Sup|Unsuper|Ensembled\_Models

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1FsOayBS6stPwiZ6jCnte9uWbUxuOJ856

"""

*# -\*- coding: utf-8 -\*-*

"""Super|Unsuper.ipynb and Ensembled Model

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1QCPWks\_ISEtvbJASWeTXEZX2mHj7J\_KB

"""

from google.colab import drive

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from collections import Counter

from imblearn.over\_sampling import SMOTE

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from xgboost import XGBClassifier, plot\_importance

from sklearn.cluster import KMeans

from sklearn.metrics import (

    roc\_auc\_score, roc\_curve, confusion\_matrix,

    recall\_score, classification\_report,

    accuracy\_score, f1\_score

)

import numpy as np

import pandas as pd

import pickle

import matplotlib.pyplot as plt

import seaborn as sns

*# Additional imports for the second set of code*

from sklearn.pipeline import Pipeline

from sklearn.base import BaseEstimator, ClassifierMixin

from sklearn.ensemble import IsolationForest

from sklearn.tree import DecisionTreeClassifier

*# Mount Google Drive*

drive.mount('/content/drive')

*# Load the data*

data = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/creditcard.csv")

data.head()

*# Data preprocessing*

data.isnull().sum()

data.drop("Time", axis=1, inplace=True)

*# Explore data*

data.info()

data.shape

data.describe()

*# Correlation heatmap*

dataplot = sns.heatmap(data.corr(), cmap="YlGnBu")

plt.show()

*# Class distribution count plot*

plt.figure(figsize=(5, 5))

sns.countplot(x="Class", data=data)

(data["Class"].value\_counts() / len(data)) \* 100

*# Display the percentage distribution as text*

class\_distribution\_percent = (data["Class"].value\_counts() / len(data)) \* 100

for index, value in class\_distribution\_percent.items():

    plt.text(index, value, f'{value:.2f}%', ha='center', va='bottom')

*# Show the plot*

plt.title('Class Distribution and Percentage')

plt.show()

*# Prepare data for modeling*

X = data.drop("Class", axis=1)

y = data["Class"]

*# Oversample using SMOTE*

sm = SMOTE(random\_state=2)

X, y = sm.fit\_resample(X, y)

counter = Counter(y)

print(counter)

*# Normalize features*

scale = MinMaxScaler()

X = scale.fit\_transform(X)

*# Split data into train and test sets*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

X\_train.shape, y\_train.shape

*# Supervised Model 1: Logistic Regression*

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_lr = lr.predict(X\_test)

print("Logistic Regression Classification Report:\n", classification\_report(y\_test, y\_pred\_lr))

*# ROC Curve for Logistic Regression*

y\_pred\_proba\_lr = lr.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_lr)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_lr)

plt.plot(fpr, tpr, label="Logistic Regression, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Logistic Regression*

cf\_matrix\_lr = confusion\_matrix(y\_test, y\_pred\_lr)

print("Confusion Matrix (Logistic Regression):\n", cf\_matrix\_lr)

*# Save Logistic Regression model*

pickle.dump(lr, open('lr\_model (auc = 0.99).pkl', 'wb'))

*# Supervised Model 2: XGBoost*

xgb = XGBClassifier()

xgb.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_xgb = xgb.predict(X\_test)

print("XGBoost Classification Report:\n", classification\_report(y\_test, y\_pred\_xgb))

*# ROC Curve for XGBoost*

y\_pred\_proba\_xgb = xgb.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_xgb)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_xgb)

plt.plot(fpr, tpr, label="XGBoost, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for XGBoost*

cf\_matrix\_xgb = confusion\_matrix(y\_test, y\_pred\_xgb)

print("Confusion Matrix (XGBoost):\n", cf\_matrix\_xgb)

*# Save XGBoost model*

pickle.dump(xgb, open('xgb\_model (auc = 0.99).pkl', 'wb'))

*# Plot Feature Importances for XGBoost*

plot\_importance(xgb)

plt.show()

*# Supervised Model 3: Random Forest*

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_rf = rf.predict(X\_test)

print("Random Forest Classification Report:\n", classification\_report(y\_test, y\_pred\_rf))

*# ROC Curve for Random Forest*

y\_pred\_proba\_rf = rf.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_rf)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_rf)

plt.plot(fpr, tpr, label="Random Forest, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Random Forest*

cf\_matrix\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

print("Confusion Matrix (Random Forest):\n", cf\_matrix\_rf)

*# Save Random Forest model*

pickle.dump(rf, open('rf\_model (auc = 0.99).pkl', 'wb'))

*# Unsupervised Model 1: KMeans Clustering*

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X\_train)

*# Predictions and Evaluation*

y\_pred\_kmeans = kmeans.predict(X\_test)

print("KMeans Clustering Classification Report:\n", classification\_report(y\_test, y\_pred\_kmeans))

*# Anomaly Detection for KMeans*

distance\_to\_center\_kmeans = np.min(kmeans.transform(X\_test), axis=1)

anomaly\_rate\_kmeans = np.mean(y\_pred\_kmeans)

*# Print and visualize anomalies for KMeans*

print("KMeans Clustering Anomaly Detection Report:")

print("Distance to Center:\n", distance\_to\_center\_kmeans)

print("Anomaly Rate:", anomaly\_rate\_kmeans)

*# Visualization of KMeans Clustering Anomaly Detection*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=distance\_to\_center\_kmeans, cmap='viridis', s=50, alpha=0.5)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', s=200, marker='X')

plt.title('KMeans Clustering Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.colorbar(label='Distance to Center')

plt.show()

*# Unsupervised Model 2: Isolation Forest*

from sklearn.ensemble import IsolationForest

from sklearn.svm import OneClassSVM

isolation\_forest = IsolationForest(contamination=0.01, random\_state=42)

isolation\_forest.fit(X\_train)

*# Predictions and Evaluation*

y\_pred\_iso\_forest = isolation\_forest.predict(X\_test)

y\_pred\_iso\_forest[y\_pred\_iso\_forest == 1] = 0

y\_pred\_iso\_forest[y\_pred\_iso\_forest == -1] = 1

print("Isolation Forest Classification Report:\n", classification\_report(y\_test, y\_pred\_iso\_forest))

*# Confusion Matrix for Isolation Forest*

cf\_matrix\_iso\_forest = confusion\_matrix(y\_test, y\_pred\_iso\_forest)

print("Confusion Matrix (Isolation Forest):\n", cf\_matrix\_iso\_forest)

*# Print and visualize anomalies for Isolation Forest*

print("Isolation Forest Anomaly Detection Report:")

print("Anomaly Scores:\n", anomaly\_scores\_iso\_forest)

print("Anomaly Rate:", anomaly\_rate\_iso\_forest)

*# Visualization of Isolation Forest Anomaly Detection*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=anomaly\_scores\_iso\_forest, cmap='viridis', s=50, alpha=0.5)

plt.title('Isolation Forest Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.colorbar(label='Anomaly Score')

plt.show()

*# Custom Classifier for IsolationForest*

class IsolationForestClassifier(BaseEstimator, ClassifierMixin):

    def \_\_init\_\_(self, contamination=0.01, random\_state=None):

*self*.contamination = contamination

*self*.random\_state = random\_state

*self*.isolation\_forest = IsolationForest(contamination=*self*.contamination, random\_state=*self*.random\_state)

    def fit(self, X, y=None):

*self*.isolation\_forest.fit(X)

        return *self*

    def predict(self, X):

*# Convert IsolationForest output to binary predictions*

        return np.where(*self*.isolation\_forest.predict(X) == -1, 1, 0)

    def decision\_function(self, X):

*# Return anomaly score as a decision function*

        return -*self*.isolation\_forest.decision\_function(X)

    def predict\_proba(self, X):

*# Simulate probabilities based on anomaly scores*

        decision\_function = *self*.decision\_function(X)

        min\_decision = np.min(decision\_function)

        max\_decision = np.max(decision\_function)

        normalized\_decision = (decision\_function - min\_decision) / (max\_decision - min\_decision)

*# Probability is the complement of the anomaly score*

        probabilities = 1 - normalized\_decision

        return np.column\_stack([1 - probabilities, probabilities])

*# Supervised Model 1: Logistic Regression*

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

*# Supervised Model 2: XGBoost*

xgb = XGBClassifier()

xgb.fit(X\_train, y\_train)

*# Supervised Model 3: RandomForest*

rf = RandomForestClassifier(random\_state=42)

rf.fit(X\_train, y\_train)

*# Unsupervised Model 1: KMeans Clustering*

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans\_pipeline = Pipeline([

    ("kmeans", kmeans),

    ("classifier", DecisionTreeClassifier())  *# Using a classifier, e.g., DecisionTreeClassifier*

])

*# Unsupervised Model 2: Isolation Forest*

isolation\_forest\_classifier = IsolationForestClassifier(contamination=0.01, random\_state=42)

*# Hybrid Model: Ensemble of Logistic Regression, XGBoost, Random Forest, KMeans, Isolation Forest*

ensemble\_model = VotingClassifier(estimators=[

    ('logistic\_regression', lr),

    ('xgboost', xgb),

    ('kmeans', kmeans\_pipeline),

     ('random\_forest', rf),

    ('isolation\_forest', isolation\_forest\_classifier)

], voting='soft')

ensemble\_model.fit(X\_train, y\_train)

*# Predictions for the Ensemble Model*

y\_pred\_ensemble = ensemble\_model.predict(X\_test)

*# Classification Report for Ensemble Model*

classification\_rep = classification\_report(y\_test, y\_pred\_ensemble)

print("Classification Report (Ensemble Model):\n", classification\_rep)

*# Calculate Accuracy, F1-score, and Recall*

accuracy = accuracy\_score(y\_test, y\_pred\_ensemble)

f1 = f1\_score(y\_test, y\_pred\_ensemble)

recall = recall\_score(y\_test, y\_pred\_ensemble)

*# Print Accuracy, F1-score, and Recall*

print("Accuracy (Ensemble Model):", accuracy)

print("F1-score (Ensemble Model):", f1)

print("Recall (Ensemble Model):", recall)

*# ROC Curve for Ensemble Model*

y\_pred\_proba\_ensemble = ensemble\_model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_ensemble)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_ensemble)

plt.plot(fpr, tpr, label="Ensemble Model, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Ensemble Model*

cf\_matrix\_ensemble = confusion\_matrix(y\_test, y\_pred\_ensemble)

print("Confusion Matrix (Ensemble Model):\n", cf\_matrix\_ensemble)

*# Save Ensemble Model*

pickle.dump(ensemble\_model, open('ensemble\_model.pkl', 'wb'))

*# Confusion Matrix for Ensemble Model*

cf\_matrix\_ensemble = confusion\_matrix(y\_test, y\_pred\_ensemble)

print("Confusion Matrix (Ensemble Model):\n", cf\_matrix\_ensemble)

*# Plot Confusion Matrix*

plt.figure(figsize=(8, 6))

sns.heatmap(cf\_matrix\_ensemble, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title("Confusion Matrix - Ensemble Model")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

*# Bar Plot for Fraud and Non-Fraud Counts*

plt.figure(figsize=(8, 6))

sns.countplot(x=y\_test, hue=y\_pred\_ensemble)

plt.title("Fraud and Non-Fraud Counts - Ensemble Model")

plt.xlabel("Actual Class")

plt.ylabel("Count")

plt.legend(title="Predicted Class", loc="upper right", labels=["Non-Fraud", "Fraud"])

plt.show()

*# Assuming 'X\_test' is your test data*

*# Predictions for the Ensemble Model*

y\_pred\_ensemble = ensemble\_model.predict(X\_test)

*# Anomaly Detection Scores*

anomaly\_scores = ensemble\_model.predict\_proba(X\_test)[:, 1]

*# Anomaly Detection Rates*

anomaly\_rate = np.mean(y\_pred\_ensemble)

*# Print and visualize anomalies*

print("Ensemble Model Classification Report:\n", classification\_report(y\_test, y\_pred\_ensemble))

*# Visualize anomalies*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[y\_pred\_ensemble == 0][:, 0], X\_test[y\_pred\_ensemble == 0][:, 1], label='Non-Anomalies', c='green', s=10)

plt.scatter(X\_test[y\_pred\_ensemble == 1][:, 0], X\_test[y\_pred\_ensemble == 1][:, 1], label='Anomalies', c='red', s=30, marker='x')

plt.title('Ensemble Model Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

*# Print Anomaly Scores and Rate*

print("Anomaly Scores:\n", anomaly\_scores)

print("Anomaly Rate:", anomaly\_rate)

## 2023 Dataset Supervised Unsupervised and Ensembled Models Code in Python:

*# -\*- coding: utf-8 -\*-*

"""2023Dataset\_Sup|Unsuper|Ensembled\_Models

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1JyTUEsklAgCvcUw\_W3qFYjxoTlMvmFLO

"""

*# -\*- coding: utf-8 -\*-*

"""Super|Unsuper.ipynb and Ensembled Model

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1QCPWks\_ISEtvbJASWeTXEZX2mHj7J\_KB

"""

from google.colab import drive

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from collections import Counter

from imblearn.over\_sampling import SMOTE

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from xgboost import XGBClassifier, plot\_importance

from sklearn.cluster import KMeans

from sklearn.metrics import (

    roc\_auc\_score, roc\_curve, confusion\_matrix,

    recall\_score, classification\_report,

    accuracy\_score, f1\_score

)

import numpy as np

import pandas as pd

import pickle

import matplotlib.pyplot as plt

import seaborn as sns

*# Additional imports for the second set of code*

from sklearn.pipeline import Pipeline

from sklearn.base import BaseEstimator, ClassifierMixin

from sklearn.ensemble import IsolationForest

from sklearn.tree import DecisionTreeClassifier

*# Mount Google Drive*

drive.mount('/content/drive')

*# Load the data*

data = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/creditcard\_2023.csv")

data.head()

*# Data preprocessing*

data.isnull().sum()

*# Explore data*

data.info()

data.shape

data.describe()

*# Correlation heatmap*

dataplot = sns.heatmap(data.corr(), cmap="YlGnBu")

plt.show()

*# Class distribution count plot*

plt.figure(figsize=(5, 5))

sns.countplot(x="Class", data=data)

(data["Class"].value\_counts() / len(data)) \* 100

*# Display the percentage distribution as text*

class\_distribution\_percent = (data["Class"].value\_counts() / len(data)) \* 100

for index, value in class\_distribution\_percent.items():

    plt.text(index, value, f'{value:.2f}%', ha='center', va='bottom')

*# Show the plot*

plt.title('Class Distribution and Percentage')

plt.show()

*# Prepare data for modeling*

X = data.drop("Class", axis=1)

y = data["Class"]

*# Normalize features*

scale = MinMaxScaler()

X = scale.fit\_transform(X)

*# Split data into train and test sets*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

X\_train.shape, y\_train.shape

*# Supervised Model 1: Logistic Regression*

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_lr = lr.predict(X\_test)

print("Logistic Regression Classification Report:\n", classification\_report(y\_test, y\_pred\_lr))

*# ROC Curve for Logistic Regression*

y\_pred\_proba\_lr = lr.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_lr)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_lr)

plt.plot(fpr, tpr, label="Logistic Regression, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Logistic Regression*

cf\_matrix\_lr = confusion\_matrix(y\_test, y\_pred\_lr)

print("Confusion Matrix (Logistic Regression):\n", cf\_matrix\_lr)

*# Save Logistic Regression model*

pickle.dump(lr, open('lr\_model (auc = 0.99).pkl', 'wb'))

*# Supervised Model 2: XGBoost*

xgb = XGBClassifier()

xgb.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_xgb = xgb.predict(X\_test)

print("XGBoost Classification Report:\n", classification\_report(y\_test, y\_pred\_xgb))

*# ROC Curve for XGBoost*

y\_pred\_proba\_xgb = xgb.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_xgb)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_xgb)

plt.plot(fpr, tpr, label="XGBoost, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for XGBoost*

cf\_matrix\_xgb = confusion\_matrix(y\_test, y\_pred\_xgb)

print("Confusion Matrix (XGBoost):\n", cf\_matrix\_xgb)

*# Save XGBoost model*

pickle.dump(xgb, open('xgb\_model (auc = 0.99).pkl', 'wb'))

*# Plot Feature Importances for XGBoost*

plot\_importance(xgb)

plt.show()

*# Supervised Model 3: Random Forest*

rf = RandomForestClassifier()

rf.fit(X\_train, y\_train)

*# Evaluation and visualization*

y\_pred\_rf = rf.predict(X\_test)

print("Random Forest Classification Report:\n", classification\_report(y\_test, y\_pred\_rf))

*# ROC Curve for Random Forest*

y\_pred\_proba\_rf = rf.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_rf)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_rf)

plt.plot(fpr, tpr, label="Random Forest, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Random Forest*

cf\_matrix\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

print("Confusion Matrix (Random Forest):\n", cf\_matrix\_rf)

*# Save Random Forest model*

pickle.dump(rf, open('rf\_model (auc = 0.99).pkl', 'wb'))

*# Unsupervised Model 1: KMeans Clustering*

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X\_train)

*# Predictions and Evaluation*

y\_pred\_kmeans = kmeans.predict(X\_test)

print("KMeans Clustering Classification Report:\n", classification\_report(y\_test, y\_pred\_kmeans))

*# Anomaly Detection for KMeans*

distance\_to\_center\_kmeans = np.min(kmeans.transform(X\_test), axis=1)

anomaly\_rate\_kmeans = np.mean(y\_pred\_kmeans)

*# Print and visualize anomalies for KMeans*

print("KMeans Clustering Anomaly Detection Report:")

print("Distance to Center:\n", distance\_to\_center\_kmeans)

print("Anomaly Rate:", anomaly\_rate\_kmeans)

*# Visualization of KMeans Clustering Anomaly Detection*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=distance\_to\_center\_kmeans, cmap='viridis', s=50, alpha=0.5)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', s=200, marker='X')

plt.title('KMeans Clustering Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.colorbar(label='Distance to Center')

plt.show()

*# Unsupervised Model 2: Isolation Forest*

from sklearn.ensemble import IsolationForest

from sklearn.svm import OneClassSVM

isolation\_forest = IsolationForest(contamination=0.01, random\_state=42)

isolation\_forest.fit(X\_train)

*# Predictions and Evaluation*

y\_pred\_iso\_forest = isolation\_forest.predict(X\_test)

y\_pred\_iso\_forest[y\_pred\_iso\_forest == 1] = 0

y\_pred\_iso\_forest[y\_pred\_iso\_forest == -1] = 1

print("Isolation Forest Classification Report:\n", classification\_report(y\_test, y\_pred\_iso\_forest))

*# Confusion Matrix for Isolation Forest*

cf\_matrix\_iso\_forest = confusion\_matrix(y\_test, y\_pred\_iso\_forest)

print("Confusion Matrix (Isolation Forest):\n", cf\_matrix\_iso\_forest)

*# Anomaly Detection for Isolation Forest*

anomaly\_scores\_iso\_forest = isolation\_forest.decision\_function(X\_test)

anomaly\_rate\_iso\_forest = np.mean(y\_pred\_iso\_forest)

*# Print and visualize anomalies for Isolation Forest*

print("Isolation Forest Anomaly Detection Report:")

print("Anomaly Scores:\n", anomaly\_scores\_iso\_forest)

print("Anomaly Rate:", anomaly\_rate\_iso\_forest)

*# Visualization of Isolation Forest Anomaly Detection*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=anomaly\_scores\_iso\_forest, cmap='viridis', s=50, alpha=0.5)

plt.title('Isolation Forest Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.colorbar(label='Anomaly Score')

plt.show()

*# Custom Classifier for IsolationForest*

class IsolationForestClassifier(BaseEstimator, ClassifierMixin):

    def \_\_init\_\_(self, contamination=0.01, random\_state=None):

*self*.contamination = contamination

*self*.random\_state = random\_state

*self*.isolation\_forest = IsolationForest(contamination=*self*.contamination, random\_state=*self*.random\_state)

    def fit(self, X, y=None):

*self*.isolation\_forest.fit(X)

        return *self*

    def predict(self, X):

*# Convert IsolationForest output to binary predictions*

        return np.where(*self*.isolation\_forest.predict(X) == -1, 1, 0)

    def decision\_function(self, X):

*# Return anomaly score as a decision function*

        return -*self*.isolation\_forest.decision\_function(X)

    def predict\_proba(self, X):

*# Simulate probabilities based on anomaly scores*

        decision\_function = *self*.decision\_function(X)

        min\_decision = np.min(decision\_function)

        max\_decision = np.max(decision\_function)

        normalized\_decision = (decision\_function - min\_decision) / (max\_decision - min\_decision)

*# Probability is the complement of the anomaly score*

        probabilities = 1 - normalized\_decision

        return np.column\_stack([1 - probabilities, probabilities])

*# Supervised Model 1: Logistic Regression*

lr = LogisticRegression()

lr.fit(X\_train, y\_train)

*# Supervised Model 2: XGBoost*

xgb = XGBClassifier()

xgb.fit(X\_train, y\_train)

*# Supervised Model 3: RandomForest*

rf = RandomForestClassifier(random\_state=42)

rf.fit(X\_train, y\_train)

*# Unsupervised Model 1: KMeans Clustering*

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans\_pipeline = Pipeline([

    ("kmeans", kmeans),

    ("classifier", DecisionTreeClassifier())  *# Using a classifier, e.g., DecisionTreeClassifier*

])

*# Unsupervised Model 2: Isolation Forest*

isolation\_forest\_classifier = IsolationForestClassifier(contamination=0.01, random\_state=42)

*# Hybrid Model: Ensemble of Logistic Regression, XGBoost, Random Forest, KMeans, Isolation Forest*

ensemble\_model = VotingClassifier(estimators=[

    ('logistic\_regression', lr),

    ('xgboost', xgb),

    ('kmeans', kmeans\_pipeline),

     ('random\_forest', rf),

    ('isolation\_forest', isolation\_forest\_classifier)

], voting='soft')

ensemble\_model.fit(X\_train, y\_train)

*# Predictions for the Ensemble Model*

y\_pred\_ensemble = ensemble\_model.predict(X\_test)

*# Classification Report for Ensemble Model*

classification\_rep = classification\_report(y\_test, y\_pred\_ensemble)

print("Classification Report (Ensemble Model):\n", classification\_rep)

*# Calculate Accuracy, F1-score, and Recall*

accuracy = accuracy\_score(y\_test, y\_pred\_ensemble)

f1 = f1\_score(y\_test, y\_pred\_ensemble)

recall = recall\_score(y\_test, y\_pred\_ensemble)

*# Print Accuracy, F1-score, and Recall*

print("Accuracy (Ensemble Model):", accuracy)

print("F1-score (Ensemble Model):", f1)

print("Recall (Ensemble Model):", recall)

*# ROC Curve for Ensemble Model*

y\_pred\_proba\_ensemble = ensemble\_model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_proba\_ensemble)

auc = roc\_auc\_score(y\_test, y\_pred\_proba\_ensemble)

plt.plot(fpr, tpr, label="Ensemble Model, auc="+str(auc))

plt.legend(loc=4)

plt.show()

*# Confusion Matrix for Ensemble Model*

cf\_matrix\_ensemble = confusion\_matrix(y\_test, y\_pred\_ensemble)

print("Confusion Matrix (Ensemble Model):\n", cf\_matrix\_ensemble)

*# Save Ensemble Model*

pickle.dump(ensemble\_model, open('ensemble\_model.pkl', 'wb'))

*# Confusion Matrix for Ensemble Model*

cf\_matrix\_ensemble = confusion\_matrix(y\_test, y\_pred\_ensemble)

print("Confusion Matrix (Ensemble Model):\n", cf\_matrix\_ensemble)

*# Plot Confusion Matrix*

plt.figure(figsize=(8, 6))

sns.heatmap(cf\_matrix\_ensemble, annot=True, fmt='g', cmap='Blues', cbar=False)

plt.title("Confusion Matrix - Ensemble Model")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

*# Bar Plot for Fraud and Non-Fraud Counts*

plt.figure(figsize=(8, 6))

sns.countplot(x=y\_test, hue=y\_pred\_ensemble)

plt.title("Fraud and Non-Fraud Counts - Ensemble Model")

plt.xlabel("Actual Class")

plt.ylabel("Count")

plt.legend(title="Predicted Class", loc="upper right", labels=["Non-Fraud", "Fraud"])

plt.show()

*# Assuming 'X\_test' is your test data*

*# Predictions for the Ensemble Model*

y\_pred\_ensemble = ensemble\_model.predict(X\_test)

*# Anomaly Detection Scores*

anomaly\_scores = ensemble\_model.predict\_proba(X\_test)[:, 1]

*# Anomaly Detection Rates*

anomaly\_rate = np.mean(y\_pred\_ensemble)

*# Print and visualize anomalies*

print("Ensemble Model Classification Report:\n", classification\_report(y\_test, y\_pred\_ensemble))

*# Visualize anomalies*

plt.figure(figsize=(10, 8))

plt.scatter(X\_test[y\_pred\_ensemble == 0][:, 0], X\_test[y\_pred\_ensemble == 0][:, 1], label='Non-Anomalies', c='green', s=10)

plt.scatter(X\_test[y\_pred\_ensemble == 1][:, 0], X\_test[y\_pred\_ensemble == 1][:, 1], label='Anomalies', c='red', s=30, marker='x')

plt.title('Ensemble Model Anomaly Detection')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

*# Print Anomaly Scores and Rate*

print("Anomaly Scores:\n", anomaly\_scores)

print("Anomaly Rate:", anomaly\_rate)