PROJECT TITLE: Credit Card Fraud Detection

Name : Parthasarathy S **ID :** SISPLINT:121020724

Abstract

Credit cards issued by financial institutions such as banks are globally popular due to their convenience and support for cashless transactions, boosting consumer spending and financial flexibility. However, their widespread use has led to increased fraud, such as identity theft, phishing, and card skimming, affecting both online and offline transactions. These fraudulent activities pose challenges for cardholders and financial institutions, causing significant financial losses. Despite using data mining techniques to detect fraud, maintaining accurate detection remains a critical issue across sectors like government, business, and banking. In this project we detect credit card fraudulent transactions activities by performing some of the Machine learning algorithms such as Logistic Regression, Random Forest classifiers, Decision Tree classifiers, k-Nearest Neighbors and Support Vector Machine using the data visualization, data prediction, accuracy score, precision score, F1 recall score some and data mining techniques from the field of Data science.

I. Introduction

Credit card fraud has emerged as a significant and growing concern in contemporary society, particularly with the widespread adoption of internet-based services. Fraudulent activities have seen a sharp rise across various sectors, including government institutions, corporate enterprises, financial industries, and numerous other organizations. As the dependence on the internet has increased, so too has the occurrence of credit card fraud, posing significant threats to both consumers and businesses. While online transactions are commonly associated with fraud, it is important to note that fraudulent activities have also proliferated in offline environments, indicating the pervasive nature of the problem. Despite the efforts made to curb credit card fraud using data mining and machine learning techniques, the accuracy of these methods in detecting fraudulent transactions remains suboptimal. Fraud detection systems typically rely on historical data patterns to flag suspicious behavior, but this approach often fails to adapt quickly enough to evolving fraud techniques. Fraudsters continually find new methods to bypass detection systems, thus rendering many traditional data mining approaches insufficient. As a result, identifying fraudulent transactions with precision continues to be a challenge for financial institutions and businesses alike.

The primary challenge in credit card fraud detection lies in the dynamic nature of fraudulent transactions. Fraudsters frequently alter their strategies, making it difficult for static models to effectively predict or identify these activities. Moreover, the volume of legitimate transactions far outweighs fraudulent ones, resulting in highly imbalanced datasets. This imbalance complicates the detection process, as traditional models may become biased toward the majority class (i.e., legitimate transactions), leading to high rates of false negatives, where fraudulent transactions go undetected. Therefore, innovative and efficient algorithms must be developed to enhance fraud detection systems and minimize financial losses caused by fraudulent activities.

One promising solution to mitigate credit card fraud is the implementation of more sophisticated machine learning algorithms, specifically designed to handle the complexity and dynamism of fraud patterns. Advanced methods such as ensemble learning, deep learning, and hybrid models have shown potential in detecting fraud more accurately by considering multiple layers of analysis. These algorithms not only learn from past fraudulent behaviors but also adapt to new strategies employed by fraudsters. Furthermore, the use of oversampling techniques like Synthetic Minority Over-sampling Technique (SMOTE) helps to address the issue of data imbalance, allowing models to better detect fraudulent transactions without being biased toward legitimate ones. Cross-validation and hyperparameter tuning further enhance model performance by ensuring that the algorithm generalizes well across various datasets, rather than overfitting to a specific dataset. The detection of credit card fraud, however, does not solely rely on technological advances. Collaborative efforts between financial institutions, regulatory bodies, and consumers are crucial for combating fraud on a broader scale. Financial institutions must continually invest in cutting-edge fraud detection systems and encourage consumer awareness regarding safe credit card usage. Governments and regulatory bodies must establish and enforce stringent

anti-fraud regulations while providing the necessary support to institutions in adopting advanced detection methods.

II. Methodology

The primary objective of this study is to classify transactions into fraudulent and non-fraudulent categories using machine learning algorithms. These algorithms are evaluated and compared to determine which is more effective at detecting credit card fraud. The process flow for addressing the credit card fraud detection problem involves several key steps: data splitting, model training, model deployment, and evaluation based on performance metrics.

The architecture for the credit card fraud detection system encompasses multiple stages, beginning with the collection of the dataset and progressing through model development and result analysis. For this study, the dataset named “credit_card_transactions” has been collected from Kaggle. This dataset contains list of all transactions carried in US over 2019 and 2020. The first step involves pre-processing the dataset to ensure it is clean and suitable for analysis. After pre-processing, the dataset is divided into two subsets: training data and testing data. The training data is then used to build models using Logistic Regression, Decision Tree classifier, Random Forest classifier and K-NN algorithms.

Once the models are developed, their performance is evaluated using key metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into the effectiveness of each algorithm in detecting fraudulent transactions. The detailed comparison among four models is conducted to identify which algorithm more accurately detects fraud. By evaluating these performance measures, the study aims to determine the most reliable algorithm for credit card fraud detection, thereby contributing to more robust fraud prevention strategies in the financial industry. The machine learning algorithms we used in this project are Logistic Regression, k-Nearest Neighbors, Decision tree classifier and Random Forest classifier

Flowchart Diagram of Credit Card Fraud Detection using Machine Learning

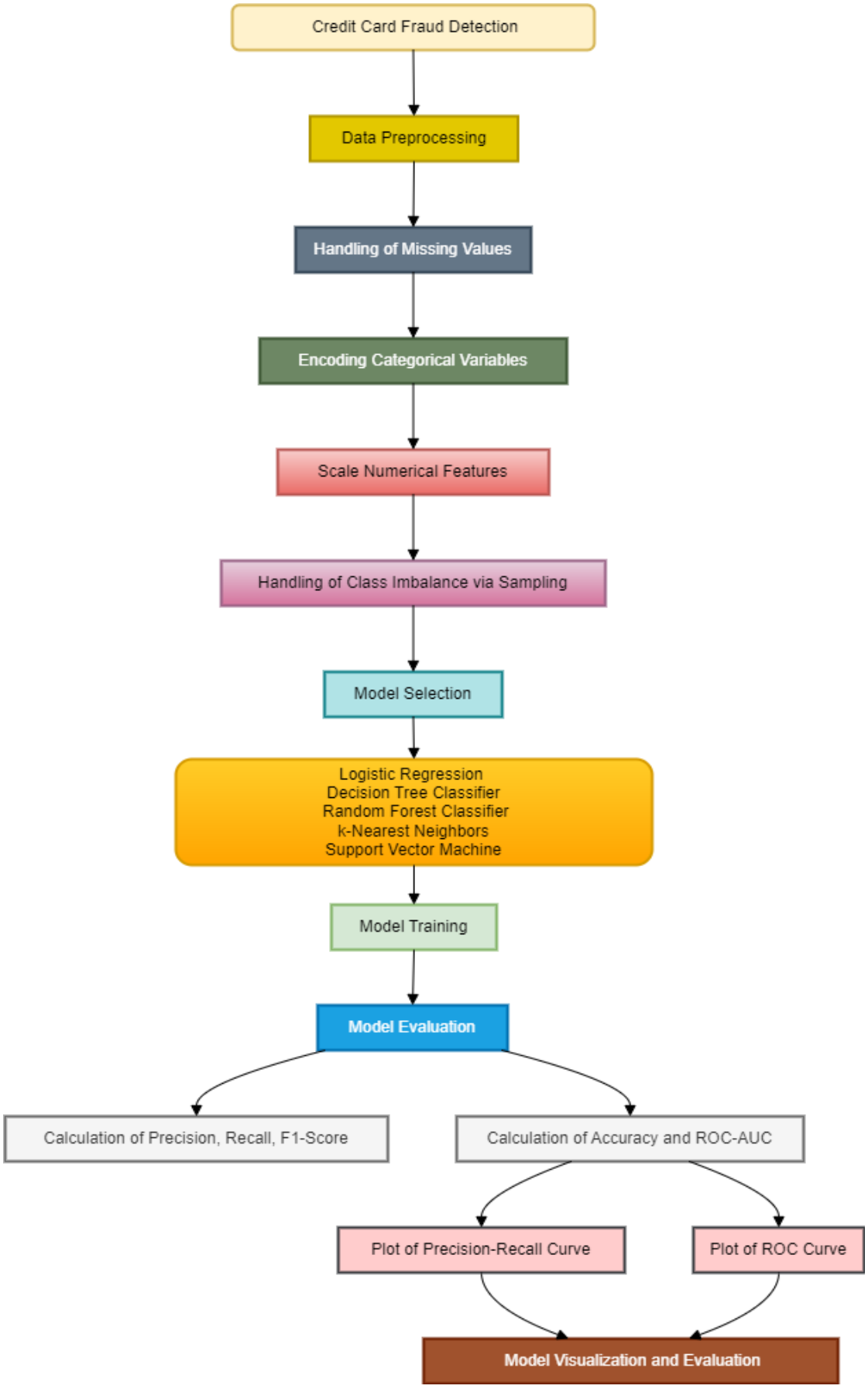


Fig 2.1 Methodologies of Credit Card Fraud Detection

1) Logistic Regression

Logistic Regression is a statistical model used for binary classification tasks. In this model of credit card fraud detection, the goal is to classify transactions as either fraudulent or non-fraudulent (i.e., binary value set as 0 for non-fraud, 1 for fraud).

Some of the features that made for our convenience are:

1. Logistic Function (Also known as Sigmoid Function): Maps any real-valued number into the range of 0 and 1. The probability output helps in the classification.

2. Decision Threshold: Default is usually 0.5, meaning if the probability of fraud is greater than 0.5, the transaction is classified as fraudulent. However the Decision Threshold parameter can be adjusted based on circumstances and necessities in finding more efficient results. So changing the Decision Threshold might affect the precision and recall score value of the dataframe.

Steps implemented in analysis and implementation of Data:

1. Data Preprocessing

Preprocessing is critical for improving the model's accuracy and dealing with data quality issues. Here's what we've done:

- Handling Missing Values:- Values that has vacant were handled by filling in missing values in the `merch_zipcode` column in our dataset using the mode (most frequent value).
- Dropping Irrelevant Columns:- Personal information like names, addresses, and transaction numbers contained in our dataset that were irrelevant to the classification task, so they were dropped to focus on the important variables.
- Encoding Categorical Variables:- Columns like `merchant`, `category`, `gender`, and `job` contain non-numerical data, presented in the dataset which must be converted to numerical values. This was done using 'Label Encoding', where each category is assigned a unique integer value.
- Feature Scaling:- Since the logistic regression model is sensitive to the scale of features, numerical features such as `amt`, `lat`, and `long` presented in our dataset were scaled using 'StandardScaler' to bring them to a similar range (mean=0, standard deviation=1). This helps improve the performance of the model by ensuring no one feature dominates because of its magnitude.
- Handling Class Imbalance:- Fraudulent transactions usually represent a small portion of the total transactions (imbalance). This was addressed using Upsampling and SMOTE (Synthetic Minority Over-Sampling Technique) where the minority class (fraudulent transactions) was oversampled to match the number of non-fraudulent transactions. This prevents the model from being biased towards the majority class.

2. Modeling

- Training the Logistic Regression Model: After preprocessing, the Logistic Regression model was trained on the cleaned and scaled dataset using cross-validation.
- Cross-Validation: This method helps to ensure that the model generalizes well to unseen data by splitting the dataset into multiple parts and training the model on different combinations of train-test splits.
- Prediction: Once the model was trained, it was used to predict whether a transaction was fraudulent or non-fraudulent.

3. Model Evaluation

- Accuracy: The proportion of correct predictions made by the model.
- Precision: The ratio of correctly predicted fraudulent transactions to all transactions predicted as fraud. This metric is important in fraud detection to reduce false positives.
- Recall: The proportion of actual fraudulent transactions that were correctly identified by the model.
- F1-Score: The harmonic mean of precision and recall, which balances both metrics.
- ROC-AUC Score: A performance metric that plots the true positive rate against the false positive rate. The AUC (Area Under the Curve) score shows how well the model distinguishes between fraud and non-fraud. Higher values are better.

2) k-Nearest Neighbors

The k-Nearest Neighbors (k-NN) algorithm is a simple, instance-based learning method used for classification tasks. It works by finding the closest points (neighbors) to a given data point in the feature space and assigning the majority class label among these neighbors. k-NN is a non-parametric algorithm, meaning it makes no assumptions about the data's underlying distribution, making it a versatile option for tasks like fraud detection.

Key Components of k-NN:

- k: The number of neighbors to consider when classifying a new data point.
- Distance Metric: Typically, Euclidean distance is used to measure the similarity between data points.
- Majority Voting: The class label is assigned based on the majority class among the k-nearest neighbors.

Procedures involved in k-NN

1. Data Preprocessing

As with any machine learning algorithm, preprocessing is vital to ensure that the k-NN model performs well. For k-NN, the following preprocessing steps were crucial:

- Handling Missing Values: Missing values in columns like `merch_zipcode` were filled using the mode (most frequent value). k-NN does not handle missing data directly, so this step ensures no missing values in the dataset.
- Dropping Irrelevant Columns: Columns containing personal details like `first`, `last`, `street`, `city`, and `trans_num` were dropped, as they provide no useful information for fraud detection.
- Encoding Categorical Variables: k-NN requires all features to be numerical. Therefore, categorical variables such as `merchant`, `category`, `gender`, and `job` were converted into numerical values using Label Encoding.
- Feature Scaling: Unlike decision tree-based methods, k-NN is sensitive to feature scaling because it relies on distance measurements. Therefore, all numerical features (e.g., `amt`, `lat`, `long`) were standardized to have zero mean and unit variance using StandardScaler. This ensures that features with larger ranges do not dominate the distance calculations.
- Handling Class Imbalance: Fraudulent transactions typically make up a small portion of the dataset, leading to class imbalance. We addressed this by upsampling the minority class (fraudulent transactions) to match the majority class (non-fraudulent transactions). This ensures that the model does not become biased towards predicting non-fraudulent transactions.

2. Modeling

- Training the k-NN Classifier: After the data have been preprocessed, a k-NN Classifier was trained. The model predicts the class of a data point based on the majority class label among the k nearest neighbors. The choice of `k` is important, as a small value can lead to noisy predictions, while a large value can result in oversmoothing of the dataset.
- Distance Metric: The Euclidean distance was used as the distance metric to calculate the similarity between data points. Other distance metrics, such as Manhattan distance, can also be used, but Euclidean distance is a common choice.
- Choosing the Number of Neighbors (k): We experimented with different values of `k` using cross-validation. Typically, odd values of `k` are preferred to avoid ties when classifying a new point. The optimal value of `k` was chosen based on model performance (precision, recall, and F1-score).
- Weighting of Neighbors: In this implementation, each neighbor contributed equally to the prediction. However, we could also use distance-weighted k-NN, where closer neighbors are given more weight in the decision-making process.

3. Model Evaluation

- Accuracy: It measures the proportion of correctly classified transactions (fraudulent and non-fraudulent) out of the total number of transactions.
- Precision: It measures how many of the predicted fraudulent transactions were actually fraudulent. This is critical in fraud detection because false positives (legitimate transactions predicted as fraud) can be costly.

- Recall (Sensitivity): Measures how many actual fraudulent transactions were correctly identified. This is important because missing fraudulent transactions (false negatives) can result in undetected fraud.
- F1-Score: The harmonic mean of precision and recall, providing a balance between the two metrics.
- ROC-AUC Score: The Receiver Operating Characteristic Area Under the Curve score evaluates the model's ability to distinguish between fraudulent and non-fraudulent transactions. A higher AUC score indicates better performance.

3) Decision Tree Classifier

A Decision Tree Classifier is a supervised learning algorithm used for classification tasks. It works by recursively splitting the data into subsets based on feature values, forming a tree-like structure. At each node, the decision tree selects the feature that best separates the classes (in our case, fraudulent and non-fraudulent transactions).

Key Components involved in Decision Tree Classifier algorithm:

- Root Node: The initial feature used to split the data.
- Internal Nodes: Features that further divide the data.
- Leaf Nodes: Final classifications (fraud or non-fraud).
- Splitting Criterion: Typically, Gini impurity or information gain (entropy) is used to decide the best split at each node.

Methodology for Credit Card Fraud Detection Using Decision Tree Classifier

1. Data Preprocessing

Just like with logistic regression, data preprocessing is the crucial step for improving the performance of the Decision Tree Classifier. The necessary steps in Data preprocessing are:

- Handling Missing Values: Missing values in columns like `merch_zipcode` were handled by imputing the mode (most frequent value) of the column. This ensures that the dataset has no gaps, which could affect the splitting process of the decision tree.
- Dropping Irrelevant Columns: Personal information (like names, street addresses, and transaction numbers) were irrelevant to the classification task, so they were removed to keep the model focused on meaningful features.
- Encoding Categorical Variables: Features like `merchant`, `category`, `gender`, and `job` were categorical, and needed to be converted into numeric format using Label Encoding. Decision trees can handle categorical variables if they are numerically encoded.
- Feature Scaling: Unlike logistic regression, decision trees are not sensitive to feature scaling, so this step was skipped. Decision trees can naturally handle different ranges of data, making scaling unnecessary.
- Handling Class Imbalance: Fraudulent transactions are rare compared to non-fraudulent ones, creating class imbalance. We handled this by upsampling the

minority class (fraudulent transactions) to match the number of non-fraudulent transactions. This ensures the model doesn't over-predict non-fraudulent transactions.

2. Modeling

- Training the Decision Tree Classifier: After the preprocessing steps, a Decision Tree Classifier was trained on the balanced dataset.
- Splitting Criterion: We used Gini impurity to determine the best feature splits at each node. Gini impurity measures how often a randomly chosen element would be incorrectly classified if it was randomly assigned a class label based on the distribution of class labels in the subset.
- Tree Depth and Pruning: To avoid overfitting, the depth of the tree can be restricted using hyperparameters like `max_depth`. Additionally, pruning (removing branches that have little importance or predictive power) helps simplify the tree and prevent overfitting to the training data.

3. Model Evaluation

- Accuracy: Measures how many transactions were correctly classified as fraudulent or non-fraudulent.
- Precision: The ratio of correctly predicted fraudulent transactions to all transactions predicted as fraud. Precision is important for fraud detection to minimize false positives (i.e., legitimate transactions classified as fraud).
- Recall: Measures the proportion of actual fraudulent transactions that the model correctly identified.
- F1-Score: The harmonic mean of precision and recall, balancing the two metrics.
- ROC-AUC Score: The Receiver Operating Characteristic Area Under the Curve score measures the model's ability to distinguish between classes. A higher AUC indicates better performance in separating fraudulent from non-fraudulent transactions.

4) Random Forest Classifier

The Random Forest Classifier is a powerful ensemble learning algorithm that operates by constructing multiple decision trees and aggregating their predictions to produce more accurate and robust results. It is well-suited for binary classification tasks, such as detecting fraudulent transactions in a credit card fraud detection dataset.

Key Components:

- Ensemble Method: Random Forest builds multiple decision trees (an ensemble of decision trees) and averages their outputs to improve performance and reduce overfitting.
- Bootstrap Aggregation (Bagging): Each tree is trained on a random sample (with replacement) of the data, making the model less sensitive to noise and variations in the dataset.
- Random Feature Selection: For each split in the decision tree, Random Forest only considers a random subset of features, which increases diversity among trees and reduces the likelihood of overfitting.

Steps involved in Random Forest Classifier

1. Data Preprocessing

Preprocessing the data is essential for ensuring that the Random Forest Classifier can be trained effectively. The following steps were carried out:

- Handling Missing Values: Missing values in the dataset, specifically in columns like `merch_zipcode`, were filled using the mode (most frequent value). This ensures that no data is missing during training, as the algorithm cannot handle missing values.
- Dropping Irrelevant Columns: Certain columns, such as personal details (`first`, `last`, `street`, `city`, and `trans_num`), were irrelevant to the prediction task and were dropped. This reduces noise and focuses the model on meaningful features.
- Encoding Categorical Variables: Random Forest requires all input features to be numerical, so categorical variables like `merchant`, `category`, `gender`, and `job` were converted into numerical values using Label Encoding.
- Feature Scaling: Random Forest does not require feature scaling because it is based on decision trees, which are insensitive to feature magnitude. Therefore, no scaling of numerical features (e.g., `amt`, `lat`, `long`) was necessary.
- Handling Class Imbalance: In the dataset, fraudulent transactions represent only a small portion of the total transactions (imbalance problem). To ensure the model focuses on fraud detection, upsampling of the minority class (fraudulent transactions) was performed, increasing the number of fraudulent cases in the training set to match the majority class (non-fraudulent transactions).

2. Modeling

- Training the Random Forest Classifier: After preprocessing, a Random Forest Classifier was trained on the balanced dataset. It constructed several decision trees, each trained on different random subsets of the data (bagging) and features.
- Number of Trees: The number of decision trees (`n_estimators`) in the forest was set to 100 by default, but this can be adjusted for better performance.
- Splitting Criterion: Like decision trees, the splitting criterion for each tree was based on Gini impurity to determine the best feature splits at each node.
- Random Feature Selection: At each node, only a random subset of features was considered for splitting, which introduces more diversity among the trees and reduces overfitting.
- Model Aggregation: The final prediction of the Random Forest model is based on the majority vote across all the decision trees. This helps in producing a robust, generalized model with lower variance than individual trees.

3. Model Evaluation of the data

- Accuracy: Measures how many transactions were correctly classified as fraudulent or non-fraudulent out of the total number of transactions.
- Precision: The ratio of correctly predicted fraudulent transactions to all transactions predicted as fraud. This is particularly important in fraud detection because it measures how well the model avoids false positives.

- Recall (Sensitivity): Measures how many actual fraudulent transactions were correctly identified by the model. This metric is critical in ensuring that we capture as many fraudulent transactions as possible.
- F1-Score: The harmonic mean of precision and recall, which balances the trade-off between the two metrics.
- ROC-AUC Score: The Receiver Operating Characteristic Area Under the Curve score evaluates the model's ability to distinguish between fraud and non-fraud. A higher score indicates better model performance.

This entire methodology ensures that the model is well-prepared, robust, and balanced for detecting credit card fraud, with an emphasis on reducing false positives while capturing as many true fraudulent transactions as possible.

III. Results and discussion

Algorithms of Logistic Regression, k-NN, Decision Tree classifier and Random Forest classifier on supervised learning methods have been therefore used to detect the credit card fraudulent transaction and data visualization techniques also been performed. In our datset the category under the credit card transactions that were legit was 1042569 entries whose binary value is '0' whereas the fraudulent transactions denoted as binary value '1' has 6006 entries. i.e. In our Data, the number of legitimate transactions is 1042569 whereas those in fraudulent transactions are 6006. The precision score, accuracy score, recall score, classification report, confusion matrix and data plot of each algorithm have been discussed below.

i. Logistic Regression

Precision-Recall Curve - This graph shows the trade-off between precision and recall. It's especially useful when dealing with imbalanced datasets like fraudulent detection.

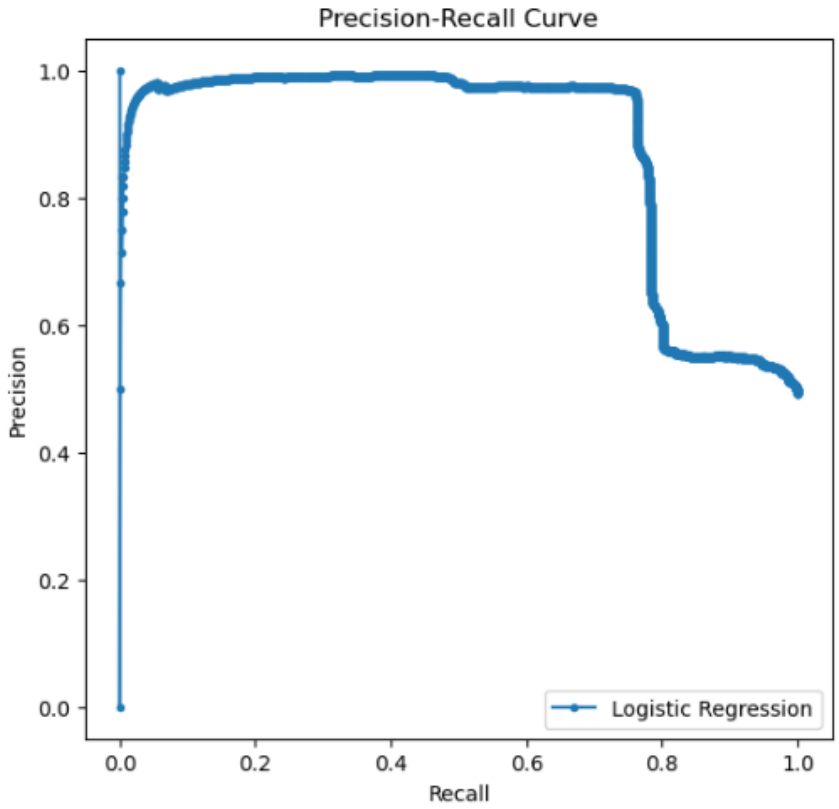


Fig 3.1.1 plot of Precision-Recall Curve

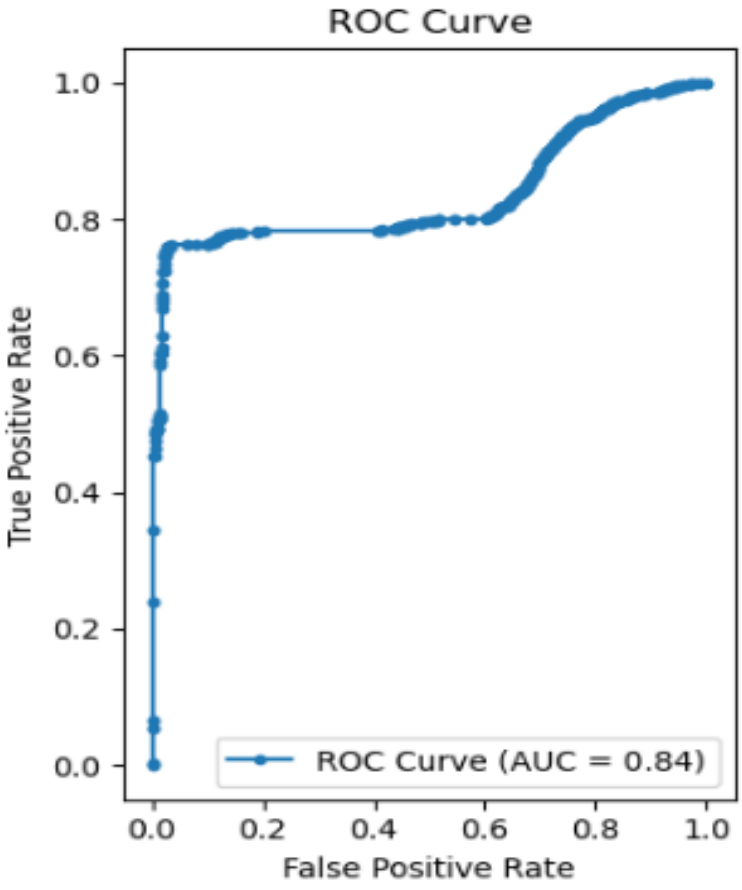


Fig 3.1.2 Plot of ROC Curve

ROC Curve - Visual representation of the trade-off between true positive and false positive rates. The closer the curve is to the top-left corner, the better the model.

Classification Report

	precision	recall	f1-score	support
0	0.81	0.96	0.87	1828
1	0.94	0.76	0.84	1776
accuracy			0.86	3604
macro avg	0.87	0.86	0.86	3604
weighted avg	0.87	0.86	0.86	3604

0.8363371577266545

According to these reports in prediction and accuracy, our model has performed well

ii. Decision Tree Classifier

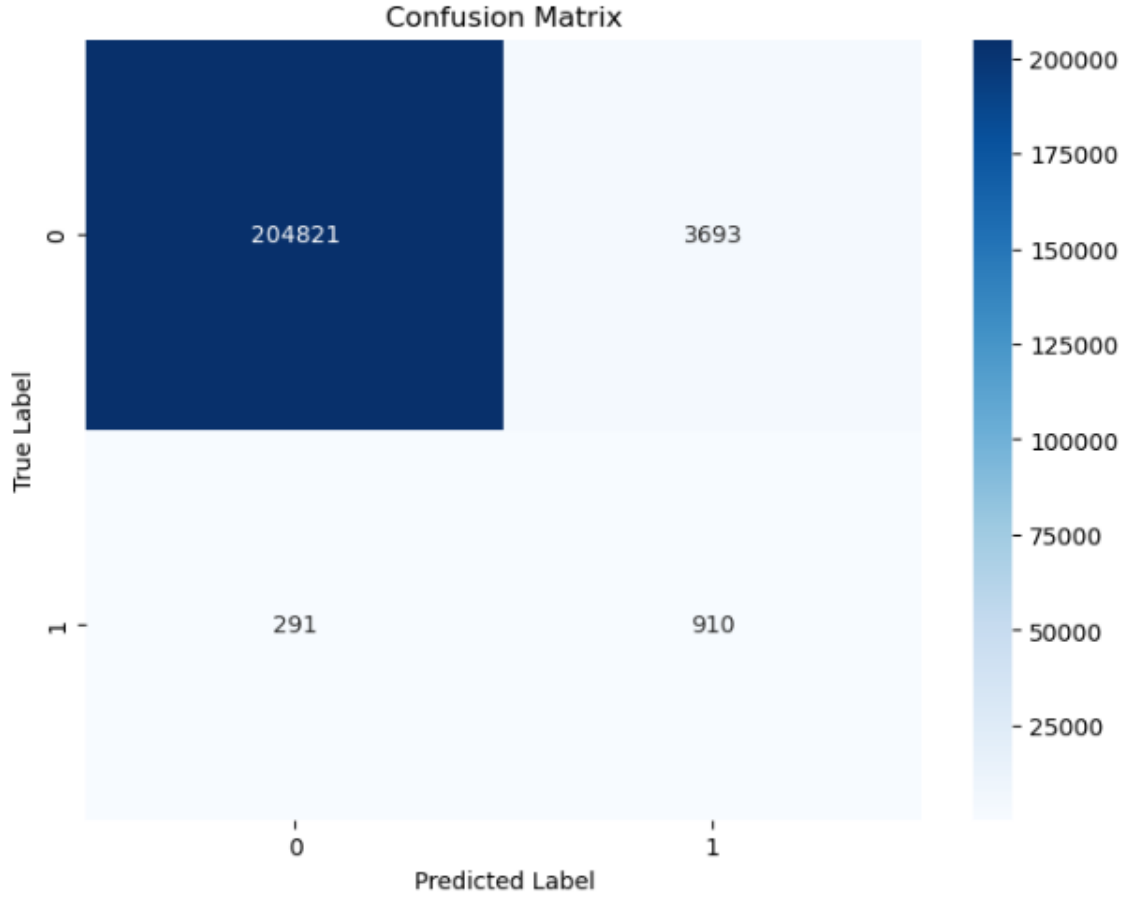


Fig 3.2.1 Plot of Confusion Matrix

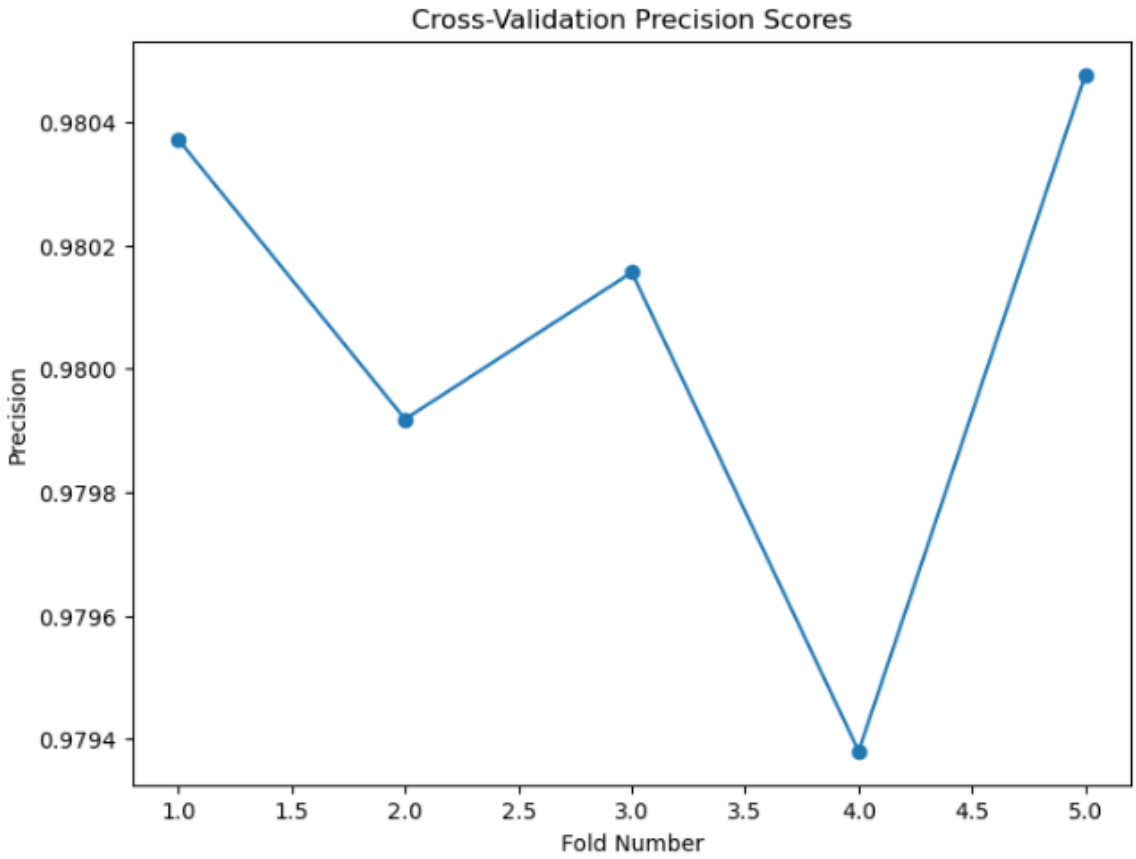


Fig 3.2.2 Plot of Cross-Validation Precision Scores

ROC and Precision-Recall Curves: Like in logistic regression, we can also visualize the ROC curve and precision-recall curve to assess the model's performance.

Classification Report and ROC-AUC Score

Precision: 0.19769715402998045
Recall: 0.7577019150707743
F1 Score: 0.31357684355616816
Cross-validated Precision: 0.980060346579395
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	208514
1	0.20	0.76	0.31	1201
accuracy			0.98	209715
macro avg	0.60	0.87	0.65	209715
weighted avg	0.99	0.98	0.99	209715

Confusion Matrix:
[[204821 3693]
[291 910]]

ROC-AUC Score: 0.8699954370427583

iii. Random Forest Classifier

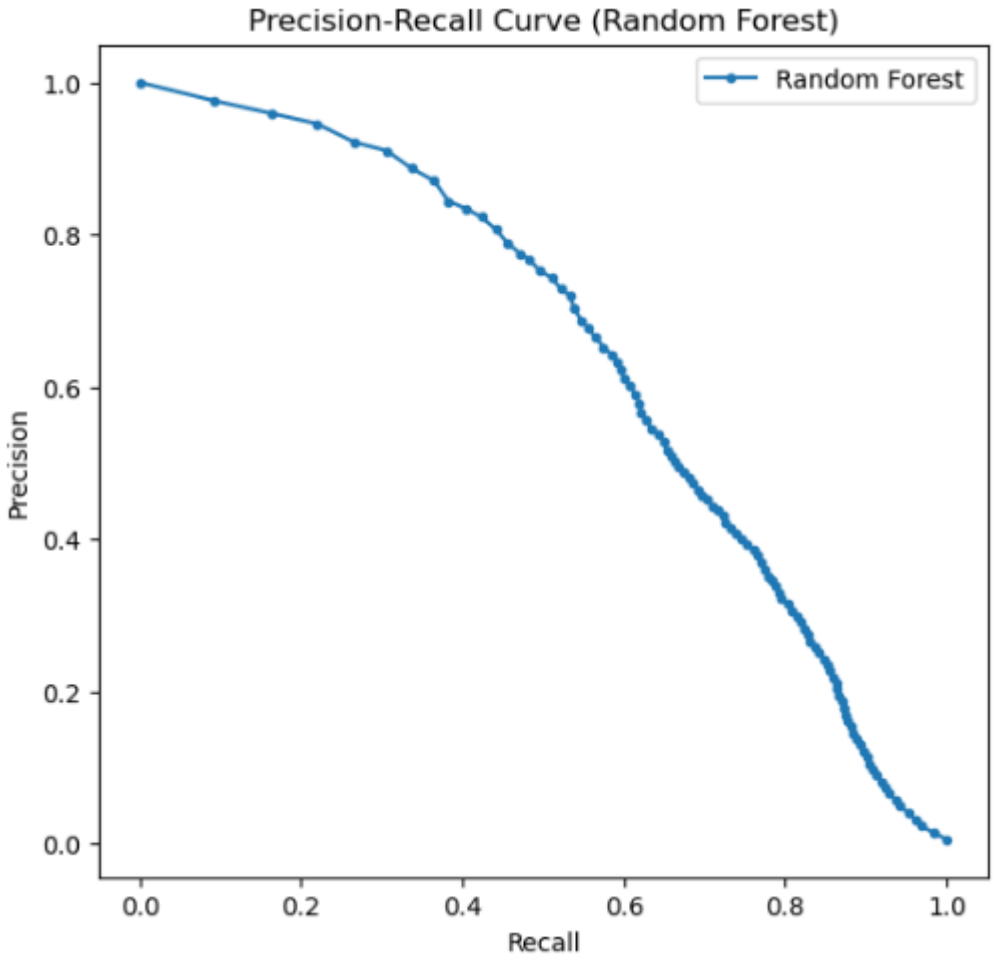


Fig 3.3.1 Plot of Precision-Recall Curve

Precision-Recall Curve: This curve is useful for understanding the trade-off between precision and recall, particularly important in imbalanced datasets like fraud detection.

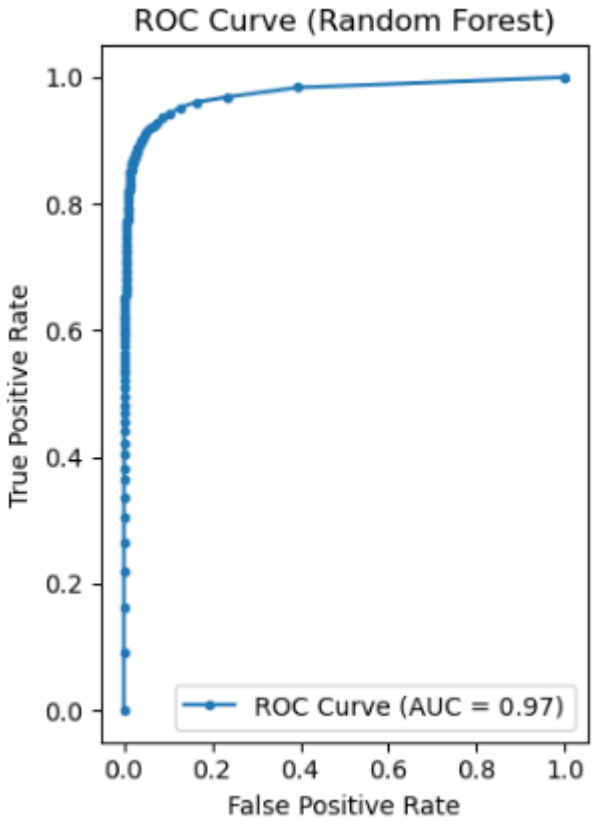


Fig 3.3.2 Plot of ROC Curve

ROC Curve - The ROC curve plots the true positive rate (recall) against the false positive rate, showing the model's ability to distinguish between the classes at different threshold levels.

Classification Report

	precision	recall	f1-score	support
0	1.00	0.99	1.00	312815
1	0.42	0.73	0.53	1758
accuracy			0.99	314573
macro avg	0.71	0.86	0.76	314573
weighted avg	1.00	0.99	0.99	314573

0.9742541556791073

iv. k-Nearest Neighbors

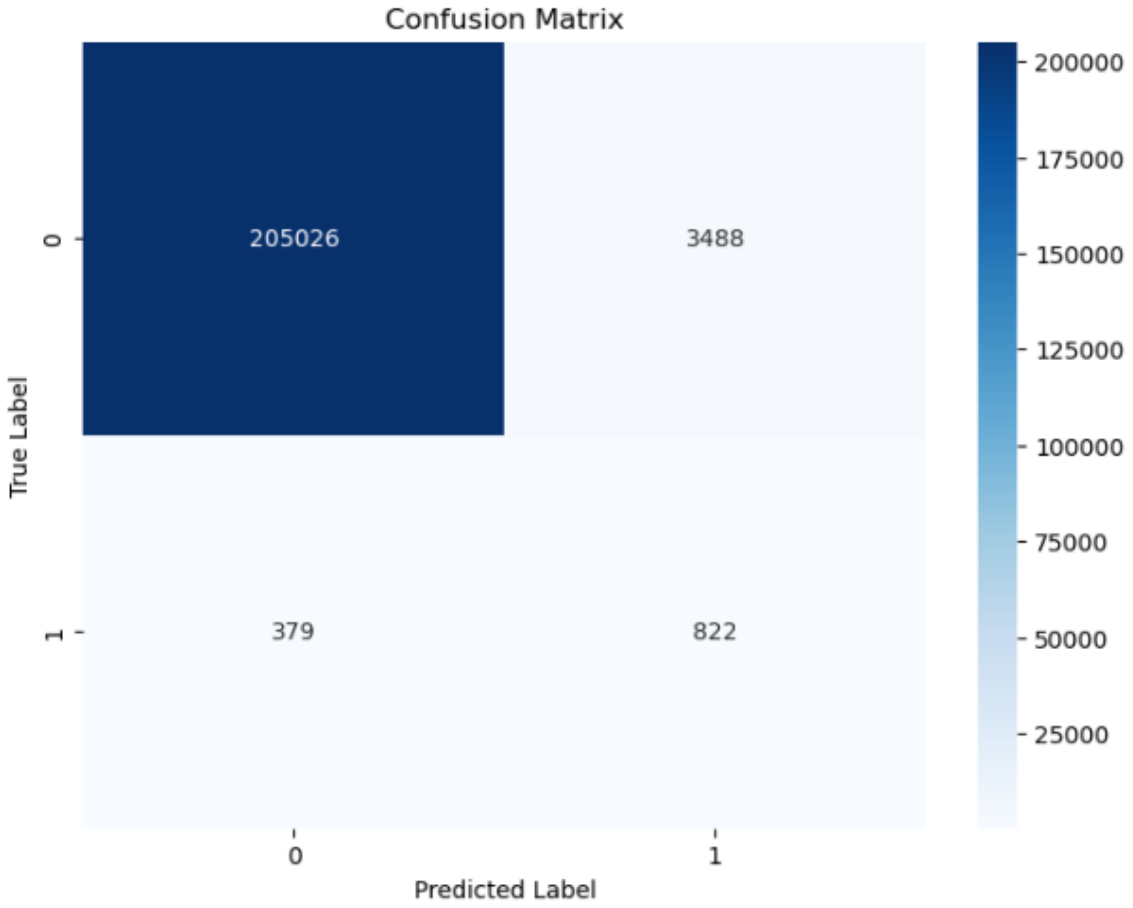
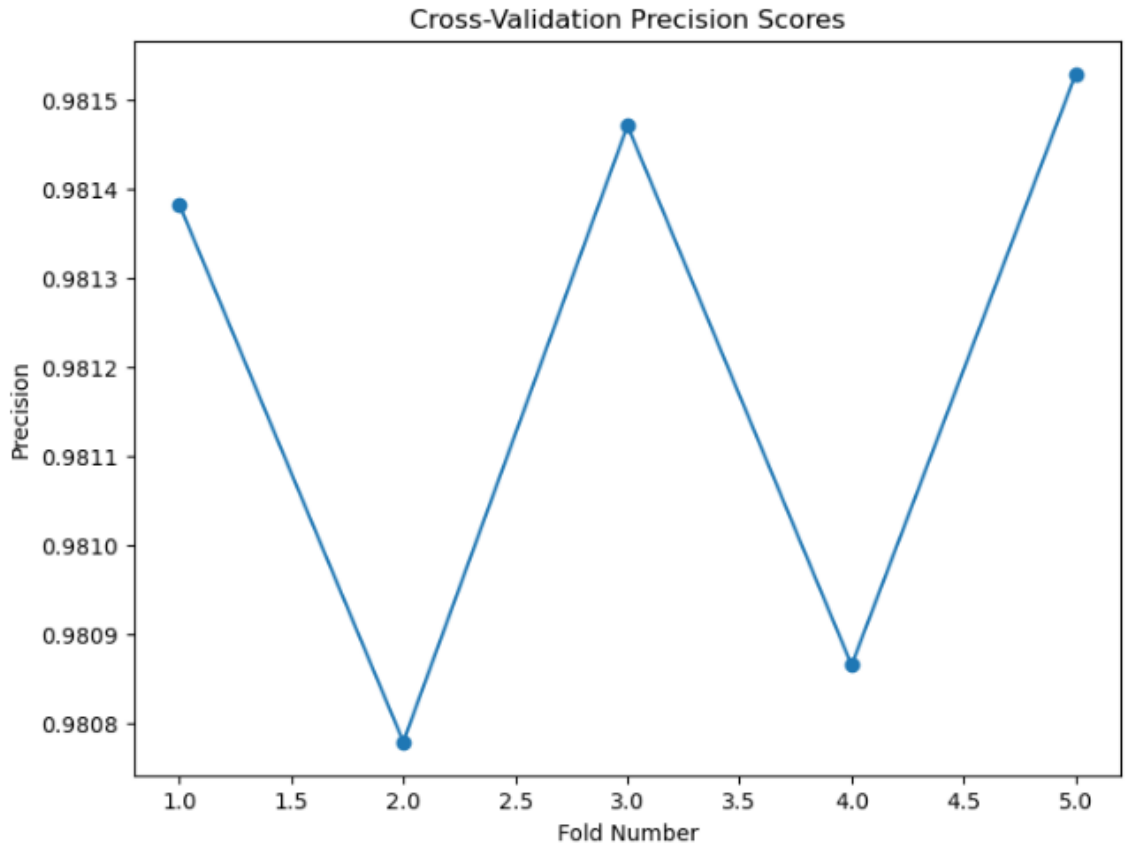


Fig 3.4.1 Plot of Confusion matrix

A confusion matrix was used to visualize the true positives, true negatives, false positives, and false negatives of the model's predictions. This helped in understanding where the model was making errors (e.g., misclassifying fraudulent transactions as legitimate).



This curve shows the trade-off between precision and recall at different threshold levels. It is especially useful in cases of imbalanced datasets like fraud detection.

ROC Curve - The ROC curve plots the true positive rate (recall) against the false positive rate, showing the model's performance across different classification thresholds.

Fig 3.4.2 Plot of cross-Validation Precision scores

Classification Report

Precision: 0.19071925754060326

Cross-validated Precision: 0.981205278374514

	precision	recall	f1-score	support
0	1.00	0.98	0.99	208514
1	0.19	0.68	0.30	1201
accuracy			0.98	209715
macro avg	0.59	0.83	0.64	209715

weighted avg 0.99 0.98 0.99 209715

Confusion Matrix:

[[205026 3488]

[379 822]]

ROC-AUC Score: 0.8338508741971669

As we discussed about the datasets that fitted on models, all the algorithms we've been used so far has predicted the credit card fraud detection well, as we seen that the performance of the prediction were precise and all of our accuracy score in each model that we've tested came with proper results and also the precision scores as well as the F1 recall scores were perfectly calculated as well. So in conclusion, all of our trained models with the datasets we've chosen was correct and precise.

IV. Conclusion

In the credit card fraud detection dataset, we implemented Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and k-Nearest Neighbors (k-NN) to predict fraudulent transactions. Logistic Regression offers simplicity but may struggle with complex patterns. Decision Trees are easy to interpret but prone to overfitting, while Random Forests provide higher accuracy and robustness but are computationally expensive. k-NN can capture non-linear relationships but is inefficient with large datasets. As we have seen, among the four algorithms the highest accuracy score we obtained is 0.9742541556791073 from the RandomForest Classifier. The Random Forest Classifier gives us the best accuracy results in the detection of fraudulent Credit Card Transactions. Future challenges in credit card fraud detection include managing the severe class imbalance and ensuring model scalability as transaction volumes increase. Models must adapt to evolving fraud tactics while maintaining privacy and security in data sharing. Combining advanced AI techniques with human expertise is crucial to improve accuracy and minimize false positives.