Credit card fraud detection using machine learning

Project Objective:

The goal of this project is to create learning models that effectively identify credit card fraud based on the available data and then compare the performance of the models. In general, we go through the following stages:

- Data analysis
- Data pre-processing
- Training, testing, and evaluating the models

Data Analysis:

The project dataset, the "Credit-card Fraud Detection dataset" on Kaggle, encompasses credit card transactions conducted by European cardholders over two days in September 2013. This selected dataset has been utilized in various research endeavors and comprises 284,807 transaction records. Upon examination of the dataset, it became evident that a very small portion of transactions (0.172%) corresponds to 492 fraudulent and genuinely forged transactions. This imbalance in the dataset implies a highly skewed distribution.

Parameter name	Total #
Total number of transactions	284,807
Total number of columns	31
Total number of features	28
Total number of label(s)	1
Total number of normal transactions	284,315
Total number of fraudulent transactions	492
% of fraudulent transactions	00.1727%
% of normal transactions	99.8273%

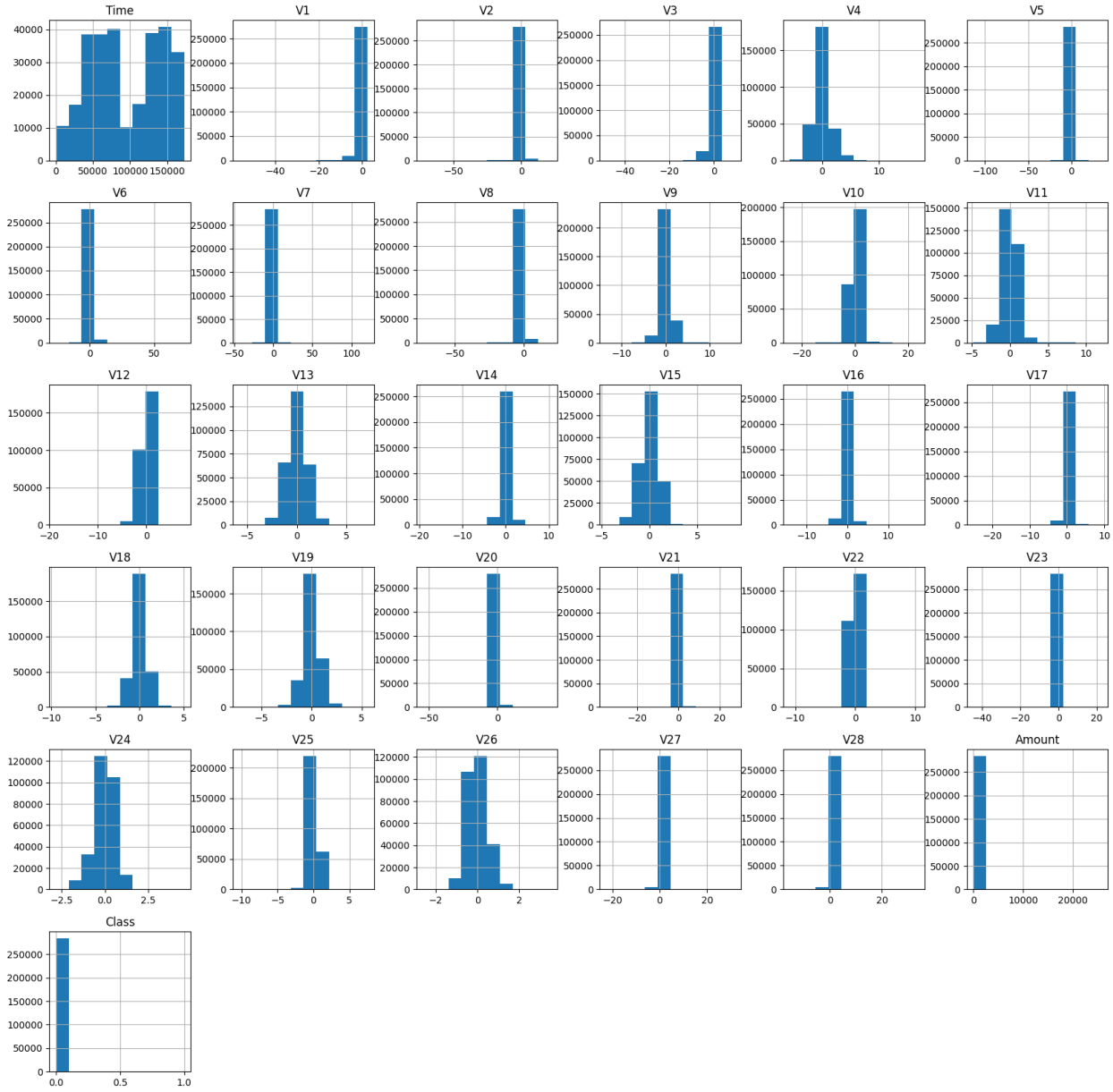
```
The dataset contains 284807 rows and 31 columns.

Normal transactions count:  284315
Fraudulent transactions count:  492
```

Additionally, it only includes numerical input variables, the result of a PCA transformation. The features V1, V2, V3, ..., V28 are the principal components obtained through PCA, and the only features not altered by PCA are "Time" and "Amount." The "Time" feature represents the seconds elapsed between each transaction and the first transaction in the dataset, while the "Amount" feature is the total amount

for each transaction. Furthermore, the "Class" feature indicates the transaction type, with "0" referring to a normal transaction and "1" indicating a fraudulent transaction.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In the figure below, the histogram distribution for each feature in the credit card fraud detection dataset is illustrated.



Data Pre-processing:

The purpose of data pre-processing typically includes reducing data volume, discovering relationships among data, normalizing data, addressing errors and missing data, and extracting the best features and characteristics of the data.

Upon examining the dataset, as there is no missing or null data, and all columns contain only numerical values, the following pre-processing steps are necessary:

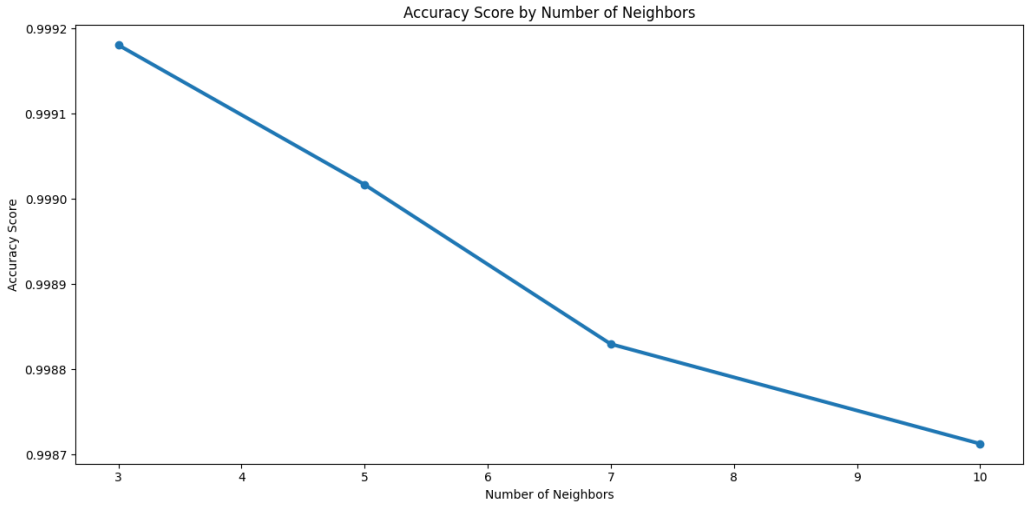
- **Data Normalization:** Normalization is performed to transform data into a consistent and comparable format. In this project, the **StandardScaler** function from the scikit-learn library is used, which operates based on z-scaling. This operation is applied only to the columns that contain the values of dataset features.

- **Checking Data Balance:** As mentioned, there is an imbalance in the used dataset between the number of fraudulent and normal transactions, with 492 instances of fraudulent transactions. Therefore, it is necessary to transform the dataset into a balanced one. In this project, the **RandomOverSampler** class is employed. In oversampling, an attempt is made to generate more samples from the minority class to equalize the class ratios.

Learning Models

In this project, four different models of machine learning algorithms for classification have been utilized:

- **Random Forest Algorithm:**
  It consists of a large number of individual decision trees that act as a group and make decisions about the output based on majority votes. The results of this algorithm are usually accompanied by lower errors. The reason for the good performance of Random Forest lies in the fact that the trees protect each other against individual errors. While some trees may predict the wrong answer, others in the group will correct the final prediction. Thus, as a group of trees, they can move in the right direction.
- **Naive Bayes Algorithm:**
  A probabilistic algorithm used for classification and prediction of classes in a dataset based on Bayesian theory and the assumption of independence between variables.
- **Logistic Regression Algorithm:**
  A method for analyzing a dataset in which one or more independent variables are used to determine an output. In contrast to linear regression, which yields continuous numeric values, this method is used for assigning data to a discrete set of classes.
- **k-Nearest Neighbors Algorithm (k-NN):**
  Predicts by identifying similar data points through calculating the distance between the target and its nearest neighbors. Essentially, it considers the labels of the k-nearest neighbors to predict the label of a new data point.
  In this project, to find the optimal value for k, the model was trained based on various values of k=3, 5, 7, and 10. Ultimately, k=3 was utilized for classification, as it demonstrated higher accuracy.



Model Evaluation

To evaluate these models, 70% of the dataset was used for training, and 30% for testing and experimentation. For assessing the performance of the four classifiers, the values of True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) are initially calculated. Using these values, the confusion matrix, ROC curve, and the following metrics are examined.

$$Precision = \frac{TP}{TP + FP} \quad Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

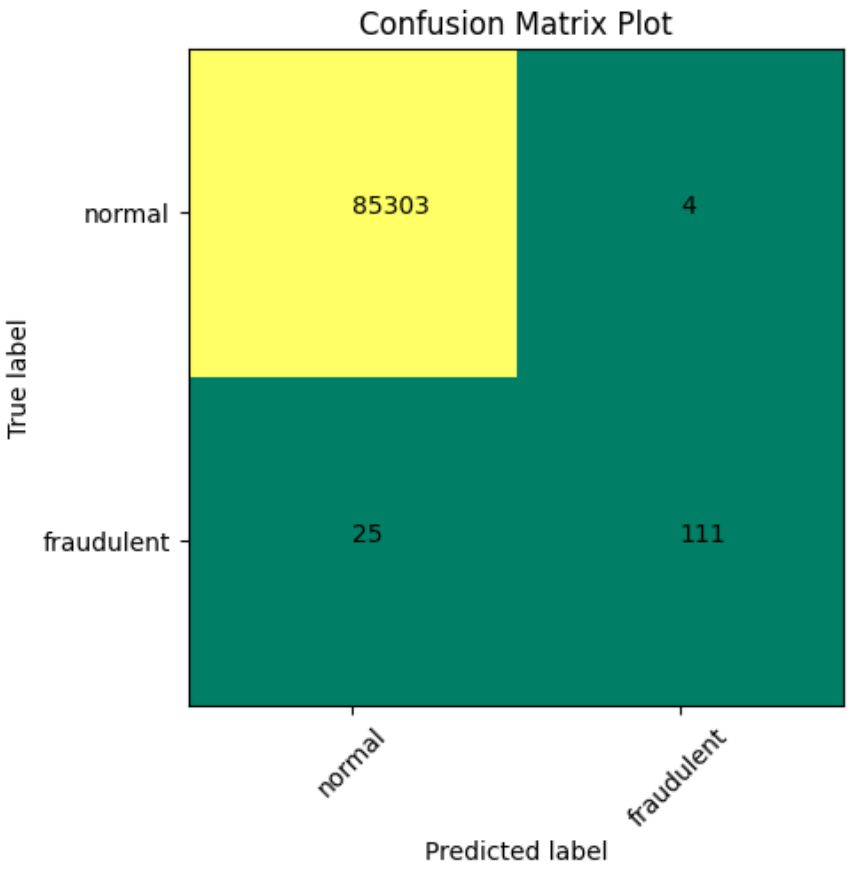
$$F1 = \frac{2 * Recall * Precision}{Recall * Precision} \quad Recall = \frac{TP}{TP + FN}$$

Implementation Results

The results of each of the four techniques employed are presented below:

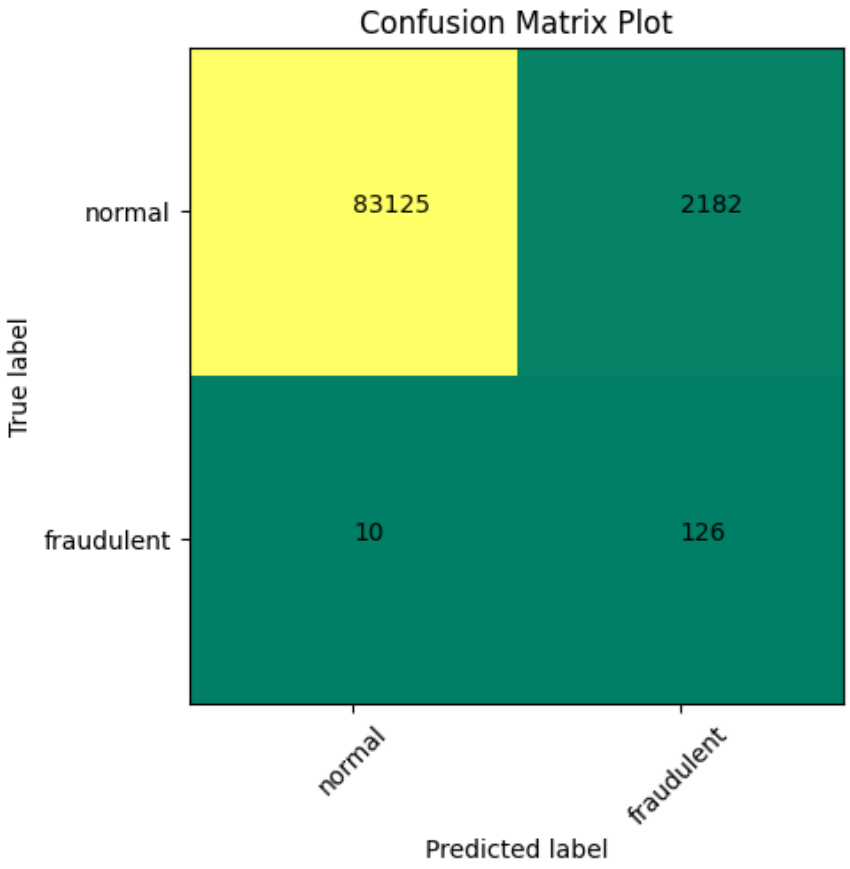
- **Results of Random Forest Algorithm Evaluation:**
  Using this model, we achieved an accuracy close to 100%. The evaluation results based on other metrics are also outlined below.

```
=== RandomForest Classifier ===
Model Accuracy:  100.0%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85307
           1       0.97      0.82      0.88       136

    accuracy                           1.00     85443
   macro avg       0.98      0.91      0.94     85443
weighted avg       1.00      1.00      1.00     85443
```



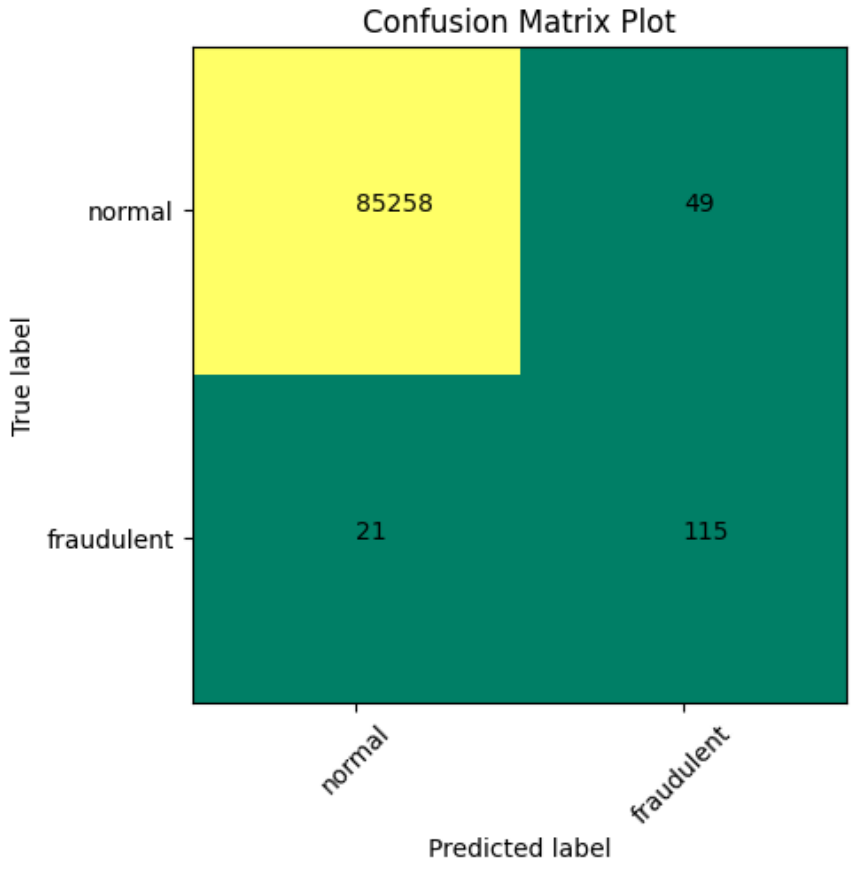
- **Results of Logistic Regression Algorithm Evaluation:**
  Using this model, we achieved an accuracy close to 97.4%. The evaluation results based on other metrics are also outlined below.

```
=== Logistic Regression ===
Model Accuracy:  97.39999999999999%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99     85307
           1       0.05      0.93      0.10       136

    accuracy                           0.97     85443
   macro avg       0.53      0.95      0.55     85443
weighted avg       1.00      0.97      0.99     85443
```



- **Results of k-Nearest Neighbors Algorithm Evaluation:**
  Using this model, we achieved an accuracy close to 99.9%. The evaluation results based on other metrics are also outlined below.

```
=== K-Neighbors Classifier ===
Model Accuracy:  99.9%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85307
           1       0.70      0.85      0.77       136

    accuracy                           1.00     85443
   macro avg       0.85      0.92      0.88     85443
weighted avg       1.00      1.00      1.00     85443
```

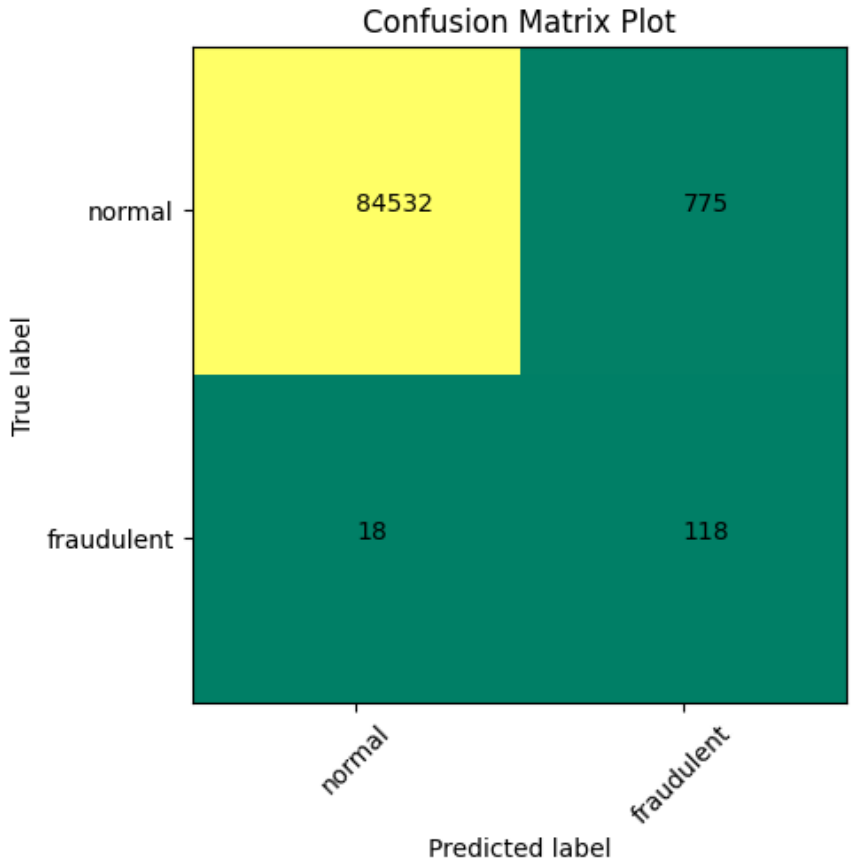


- **Results of Naive Bayes Algorithm Evaluation:**
  Using this model, we achieved an accuracy close to 99.1%. The evaluation results based on other metrics are also outlined below.

```
=== Naive Baiye Classifier ===
Model Accuracy:  99.1%
```

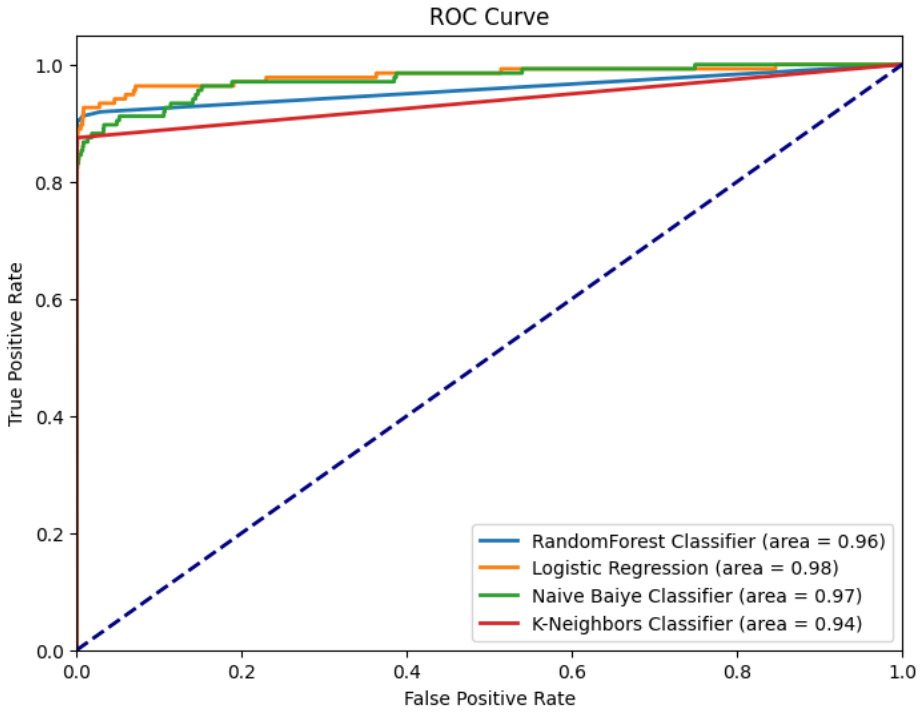
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	85307
1	0.13	0.87	0.23	136
accuracy			0.99	85443
macro avg	0.57	0.93	0.61	85443
weighted avg	1.00	0.99	0.99	85443



Results of Model Evaluations Based on ROC Curve:

According to this curve, all models have performed very well and have been able to accurately detect counterfeit credit cards. This is evident from the high True Positive rates in all four models.



Conclusion:

As anticipated, machine learning models demonstrated acceptable performance in detecting fraudulent credit card transactions. With proper preprocessing operations on the utilized dataset, as done in this project, it is possible to achieve very high accuracy in identifying fraudulent transactions. Random Forest and k-Nearest Neighbors (k-NN) models, which were examined in this project, also showed better performance with accuracy close to 100% compared to other models for this type of problem.