Online Payments Frauds Classification

1. Introduction:

In today's world, the online shopping system has gained immense popularity, with people preferring the convenience of purchasing goods from the comfort of their homes rather than venturing out for essential items. Particularly heightened during the COVID-19 pandemic, the trend of online transactions has evolved into a widespread phenomenon for people from all walks of life. While this online payment system has undoubtedly simplified many aspects of daily life, the surge in fraud cases has become increasingly prevalent in recent times. Globally, the challenge of detecting and preventing online payment fraud is on the rise, impacting individuals, businesses, and financial institutions. Therefore, it is imperative to establish a reliable and accurate method for identifying and thwarting fraudulent activities. In our paper, our primary objective was to develop a supervised learning model capable of categorizing a substantial volume of data and effectively detecting fraud in online transactions. The motivation driving this project was to construct a model with a high level of accuracy in detecting fraud in online payments. This endeavor is driven by the dual purpose of assisting people and recognizing the essential need for such a tool in contemporary times.

2. Motivation:

The digital era has brought with it an unprecedented ease of conducting financial transactions. With just a few clicks or taps on a device, money can change hands across the globe. However, this convenience also comes with significant risks, particularly the risk of online payment fraud. Fraudulent transactions not only result in financial losses for individuals and businesses but also undermine the trust in digital payment systems that are essential for modern commerce. The motivation for this report is to contribute to

the development of more secure online payment environments, where transactions are not only quick and easy but also reliable and safe from fraudulent activities. By enhancing fraud detection capabilities, we can support the continued growth and sustainability of the global digital economy.

3. Dataset Description:

Source: https://www.kaggle.com/

Link:https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

Reference: *Online Payments Fraud Detection Dataset*. (2022, April 17). Kaggle. https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset

In the dataset, each entry represents a unit of time, with one step equivalent to one hour. The 'type' field denotes the type of online transaction conducted, while the 'amount' field specifies the monetary value of the transaction. The 'nameOrig' field identifies the customer initiating the transaction, and 'oldbalanceOrg' indicates their account balance before the transaction. Subsequently, 'newbalanceOrig' reflects the customer's balance after the transaction.

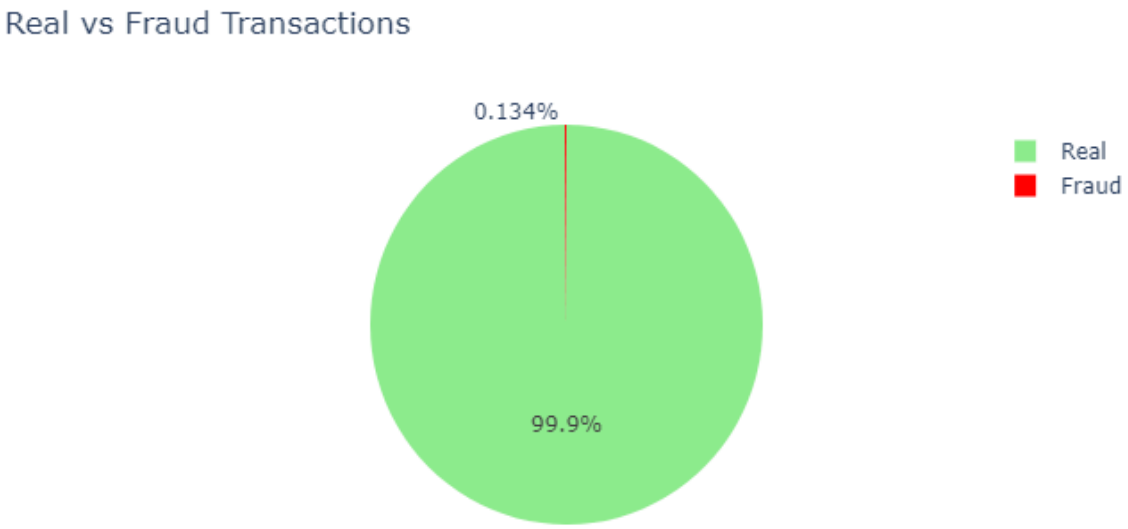
On the recipient side, 'nameDest' designates the entity receiving the transaction. The 'oldbalanceDest' field outlines the initial balance of the recipient before the transaction, and 'newbalanceDest' captures the recipient's balance after the transaction has been completed. The 'isFraud' field serves as an indicator, marking whether a given transaction is fraudulent or not. Together, these fields provide comprehensive information about the various aspects of online transactions within the dataset.

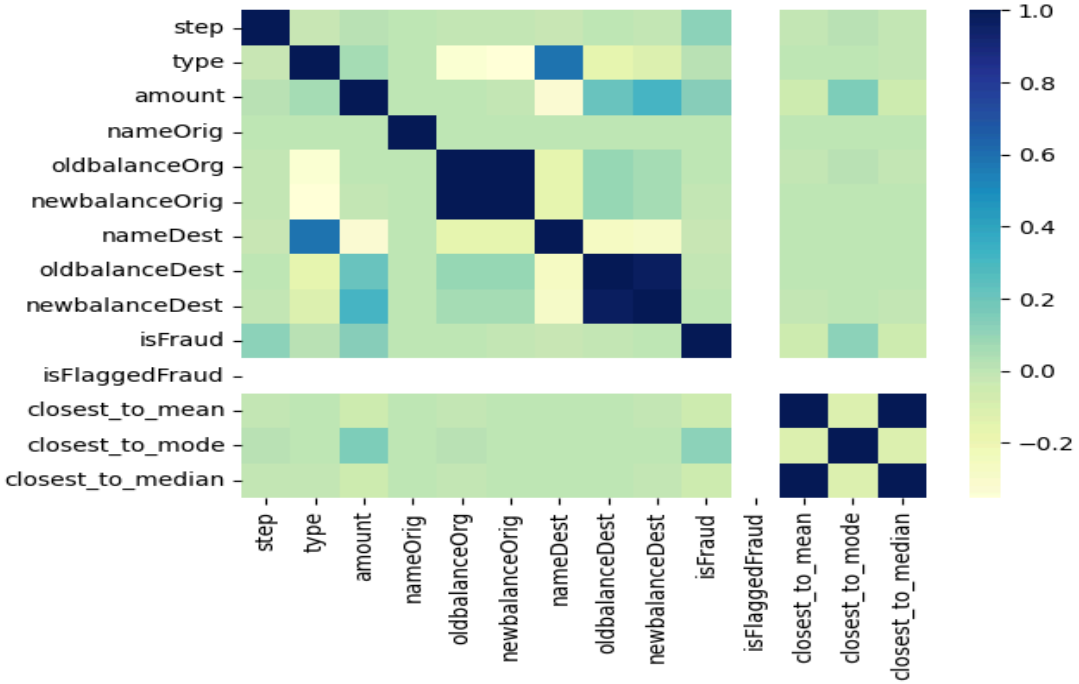
Our dataset has 699869 unique values which is the datapoint of our dataset and it is divided into 11 columns so the overall feature is 11. Our model worked on classification problems as it was detecting whether the transaction was fraud or not, it was addressed as a classification problem due to its inherent characteristics. The primary goal is to categorize transactions into either fraudulent or non-fraudulent classes, aligning well with a binary classification framework. Utilizing supervised learning on labeled datasets, classification algorithms discern patterns, allowing them to generalize and predict new, unseen transactions. Imbalances in class distribution, where fraudulent transactions are a minority, are effectively handled by classification algorithms. Specific evaluation metrics, such as accuracy, precision, and recall, are tailored to assessing the model's performance in identifying and differentiating between fraudulent and non-fraudulent transactions. We have both quantitative and categorical values in our dataset, the step, amount, oldbalance, newbalance, oldbalance dest, newbalance dest, isfraud, isflaggedfraud are all quantitative while the categorical values can be seen in nameorig, type, namedest. From our heatmap we can see that we have strong correlation between our features. The figure is given here.

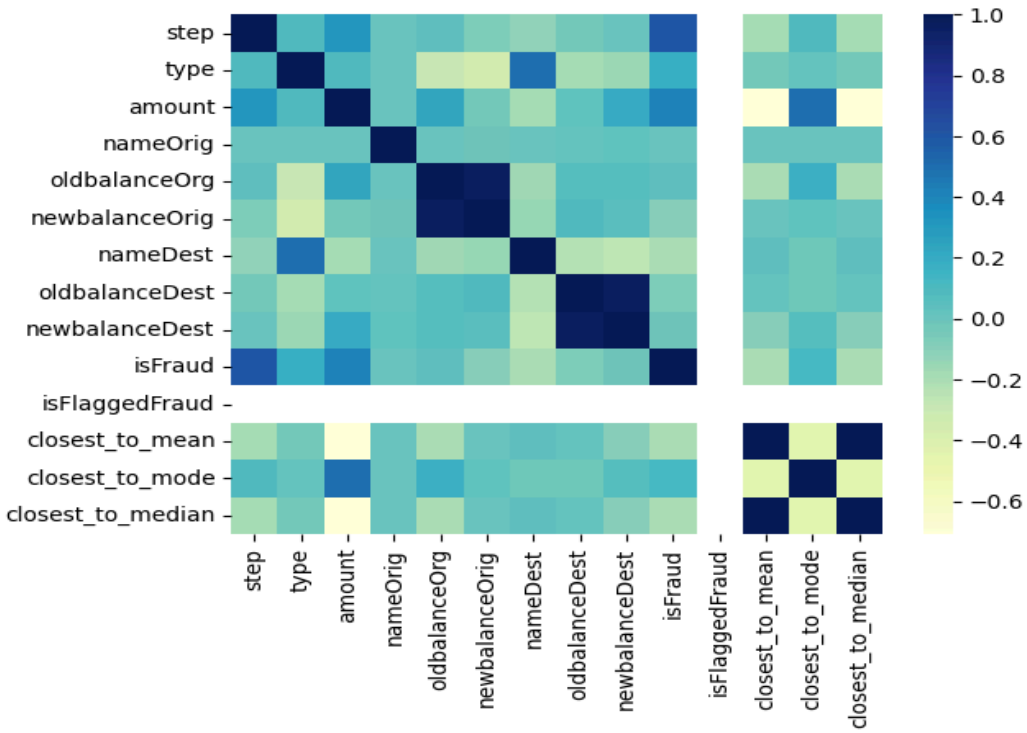
Imbalanced dataset:

In our dataset among 699869 data, 698930 data is unique and 939 dataset falls under fraudulent, which gives us an accuracy of 99.9 % and a fraud percentage of 0.134 %. To address the issue of imbalance in the dataset, we employed oversampling techniques along with the stratify method. The integration of oversampling with the stratified method is a widely adopted strategy for addressing imbalanced datasets, especially in classification scenarios like fraud detection. This approach enhances the robustness of the training process, ensuring a more balanced representation of both classes. Consequently,

it improves the model's capability to accurately identify and classify instances belonging to the minority class.







4. Data Preprocessing:

```
step                 0
type                 0
amount            1123
nameOrig             0
oldbalanceOrg        0
newbalanceOrig       0
nameDest             0
oldbalanceDest       0
newbalanceDest       0
isFraud              0
isFlaggedFraud     657
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699869 entries, 0 to 699868
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   step            699869 non-null  float64
 1   type            699869 non-null  object
 2   amount          698746 non-null  float64
 3   nameOrig        699869 non-null  object
 4   oldbalanceOrg   699869 non-null  float64
 5   newbalanceOrig  699869 non-null  float64
 6   nameDest        699869 non-null  object
 7   oldbalanceDest  699869 non-null  float64
 8   newbalanceDest  699869 non-null  float64
 9   isFraud         699869 non-null  float64
 10  isFlaggedFraud  699212 non-null  float64
dtypes: float64(8), object(3)
memory usage: 58.7+ MB
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1.0	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0.0	0.0
1	1.0	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0.0	0.0
2	1.0	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1.0	0.0
3	1.0	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1.0	0.0
4	1.0	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0.0	0.0

Before Pre-processing

Problem 1: There are 1123 null values in 1 ‘amount’ column and ‘657 null values in the ‘isFlaggedFraud’ column.

Solution: . For the ‘amount’ column, handling missing values, SimpleImputer is created with a strategy of 'median', meaning that missing values will be filled with the median of the column. The imputer is then fitted to the amount column, and the transformation (imputation) is applied to the same column. The value that has been imputed here is the median of the amount column. This is because the strategy specified is 'median'. This choice is typically made because the median is considered a more robust measure against outliers than the mean. Also median will also give more precise value for handing the null values.

For the second column, as ‘isFlaggedFraud’ column is obviously a redundant column for our training so we will drop the Column As

Problem 2: There are some columns that have categorical values like ‘type’, ‘nameOrig’, ‘nameDest’. Categorical values need to be converted into a numerical format to be used effectively in machine learning algorithms.

Solution: ‘LabelEncoder’ was imported to solve this problem. It will help to preprocess categorical columns in a Pandas data frame using Label Encoding. This will select columns with data type 'object' from the DataFrame 'dataset' and store their column names in the variable 'object_cols'. In the context of Pandas, 'object' usually refers to columns containing strings or a mix of data types. Then later they were converted to numeric values in the process. As these columns have good correlation with the target variable so we will need to use them for working with machine learning algorithms that require numerical input.

```
step                0
type                0
amount              0
nameOrig            0
oldbalanceOrg       0
newbalanceOrig      0
nameDest            0
oldbalanceDest      0
newbalanceDest      0
isFraud             0
isFlaggedFraud      0
closest_to_mean     0
closest_to_mode     0
closest_to_median   0
dtype: int64
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1.0	3	9839.64	83456	170136.0	160296.36	185644	0.0	0.0	0.0
1	1.0	3	1864.28	240750	21249.0	19384.72	193453	0.0	0.0	0.0
2	1.0	4	181.00	110023	181.0	0.00	51187	0.0	0.0	1.0
3	1.0	1	181.00	641382	181.0	0.00	45579	21182.0	0.0	1.0
4	1.0	3	11668.14	378491	41554.0	29885.86	94618	0.0	0.0	0.0

After Pre-processing

5. Feature scaling:

```
[ ]  1 scaler= StandardScaler()
     2 scaler.fit(X_trainN)
     3 print(scaler.mean_)
     4 X_trainN = scaler.transform(X_trainN)
     5 X_testN = scaler.transform(X_testN)
```

Here we used the fitted scaler to transform (standardize) both the training and testing sets. The transformation is done using the mean and standard deviation calculated from the training set. This ensures that both sets are scaled consistently.

Standardizing features is essential when using certain machine learning algorithms, such as those based on distance measures or gradient descent, to ensure that all features

contribute equally to the model and prevent one feature from dominating due to a difference in scale.

6. Dataset Splitting:

Dataset splitting is a critical step in machine learning, involving the division of data into training, validation, and test sets. The training set is used for model learning, the validation set for hyperparameter tuning and performance assessment, and the test set for final model evaluation. Stratified splitting maintains class distribution, crucial for classification tasks, while randomization ensures representative subsets. Libraries like Scikit-Learn provide tools for easy implementation. We are utilizing the "train_test_split" function to partition the dataset into training and testing sets. The goal is to facilitate effective model training, optimization, and unbiased evaluation for robust machine-learning models. The X variable encompasses all features in the dataset except for the target variable (isFlaggedFraud). Therefore, we will exclude this category 'isFlaggedFraud'. The train-test-split function is employed to randomly divide the dataset into training and testing sets, with the test size parameter indicating a 70:30 ratio (30% for testing and 70% for training). Stratify is set to y, ensuring a proportional representation of fraudulent transactions in both the training and testing sets, aligning with the original dataset's distribution. The use of a constant random state argument (2) ensures consistent splits for reproducible and reliable results with each code execution. Here, Stratify is employed in dataset splitting to address class imbalance, especially in classification tasks. In cases where certain classes are underrepresented, such as fraudulent transactions in fraud detection, stratified splitting ensures that the proportion of each class remains consistent in both the training and testing sets. This prevents potential bias in model performance caused by an uneven distribution of classes,

fostering a more accurate reflection of real-world conditions during both training and evaluation.

```
[ ]  train_df2 = newDataset.drop(['isFlaggedFraud','oldbalanceOrg','oldbalanceDest','nameOrig'], axis=1)
```

```
[ ]  nX = train_df2.drop("isFraud",axis=1)
     ny = train_df2["isFraud"]
     X_trainN, X_testN, y_trainN, y_testN = train_test_split(nX, ny, test_size=0.3,stratify=ny,random_state=2)
```

7. Model Training & testing:

KNN:

K-Nearest Neighbors, or KNN for short, is like finding the most common vote among a group of neighbors. You choose a number (let's call it K) to decide how many nearby points you want to consider. Then, you figure out which K points in your data are closest to the new point you're looking at. You look at what most of these K points are classified as, and that's what you guess the new point should be classified as too. It's like taking a poll of the closest K points.

For this dataset, we use KNN by teaching it with cleaned-up data. This includes dividing it up to test the KNN, and then making guesses on new data. After we've taught the model, we test it with some of the data we used for teaching it to see how well it does. We look at the guesses the model makes and check them against the real answers.

KNN	Training Accuracy	Testing Accuracy	Precision	Recall
	0.98	0.97	0.93	0.84

After Training and testing with the model

Logistic Regression:

Logistic regression is a way to predict yes-or-no outcomes, like if a transaction is legit or not, based on given data points. In our case, it helps us figure out if transactions are fraudulent. We gather and prep our data, pinpointing which factors to consider. Then, the model learns from this data by adjusting its focus on these factors to best match what we've seen happen. Once it's up to speed, we can feed it new information, and it'll give us the odds of a transaction being fraudulent. We use logistic regression because it's good for this kind of yes-or-no question, crunching numbers to give us a probability between 0 (no) and 1 (yes).

Logistic Regression	**Training Accuracy**	**Testing Accuracy**	**Precision**	**Recall**
	0.95	0.95	0.93	0.61

After Training and testing with the model

Naive Bayes:

Naive Bayes is chosen for this dataset to decide if a transaction is a fraud or not. It's a simple yet effective algorithm based on the idea that each thing you're looking at contributes independently to the outcome. This "naive" assumption isn't always true, but the algorithm often works well, particularly with specific types of data like categories or text. We're using the Gaussian version because it treats data features as if they're normally distributed (think of the classic bell curve), and it predicts by figuring out the average and spread of each feature for each category. Then, it uses these calculations to guess the category of new transactions, picking the category that seems most likely.

Naive Bayes	Training Accuracy	Testing Accuracy	Precision	Recall
	0.93	0.93	0.89	0.46

After Training and testing with the model

7. Model Selection/ Comparison Analysis:

We selected three models for our analysis: K-Nearest Neighbors (KNN), Logistic Regression, and Gaussian Naive Bayes (GNB), these gave us an approximate accuracy of 97%, 95%, and 93% respectively for KNN, Logistic Regression, and Gnb. The training accuracy of KNN is 0.98, and the testing accuracy stands at 0.97. Additionally, the precision is recorded at 0.93, while the recall is measured at 0.84. The logistic regression model exhibits an accuracy of 0.952 on the training data and 0.95 on the test data, the precision is calculated at 0.92, while the recall is observed to be 0.61. The precision for the Gaussian Naive Bayes (Gnb) model is 0.890, and the recall is 0.464. The training accuracy is measured at 0.93, and the testing accuracy aligns closely at 0.93 as well. We have presented comprehensive bar charts that compare and illustrate the confusion matrices for all our selected models below. Our confusion matrix categorizes predictions into true positives, true negatives, false positives, and false negatives, providing a detailed breakdown of the model's accuracy and errors. This matrix is crucial for evaluating metrics like precision, recall, and overall model effectiveness.

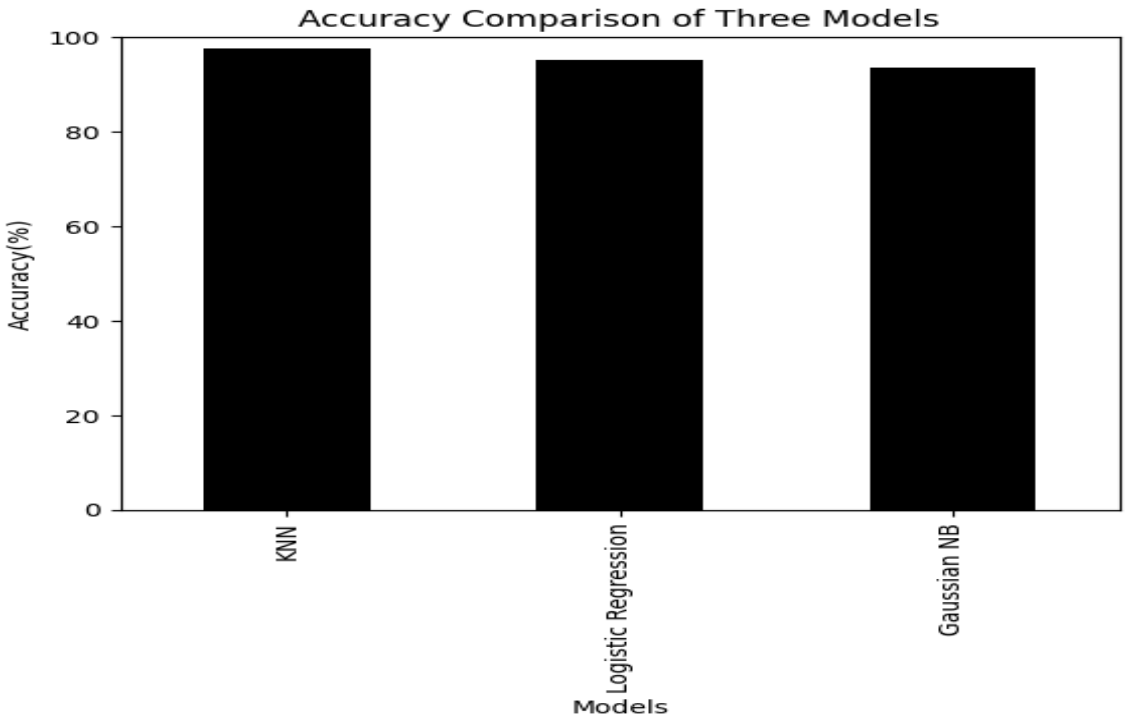


Fig: Accuracy comparison of three models

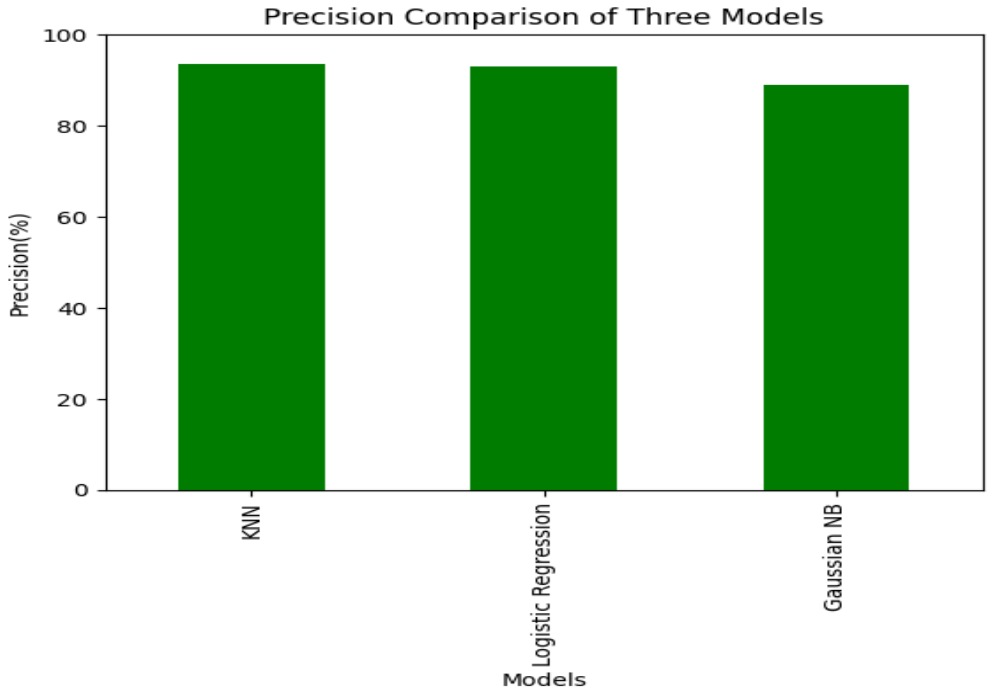
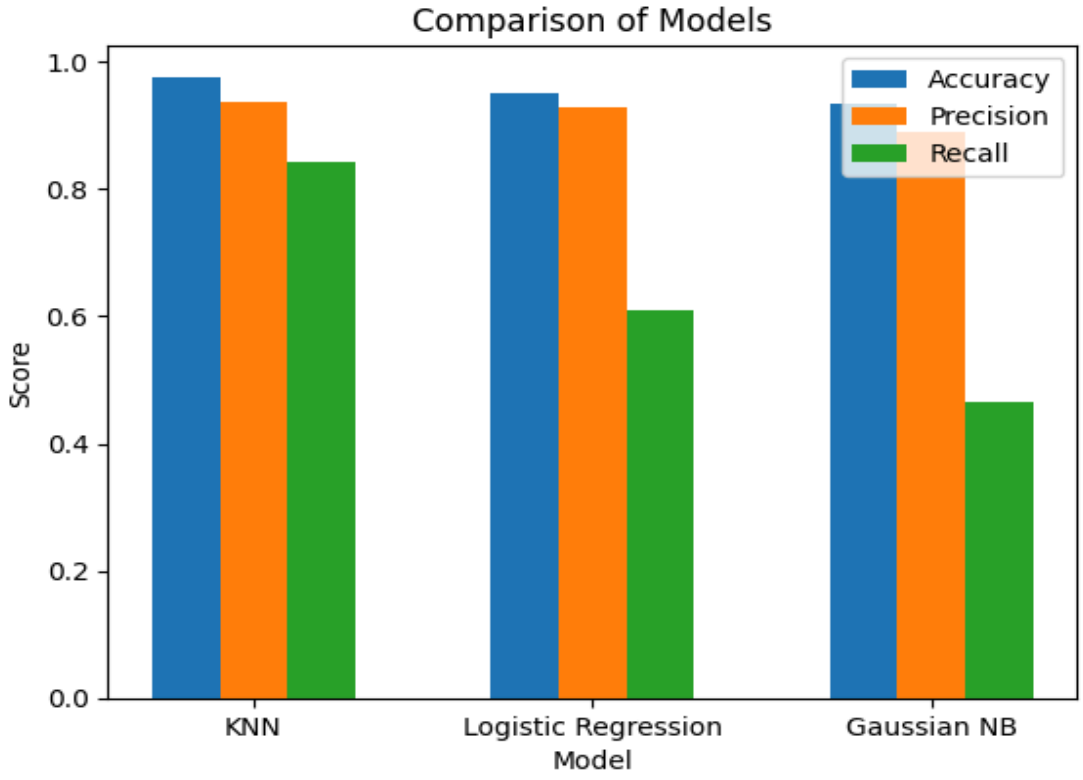


Fig: Precision comparison of three models

Fig: Recall comparison of three models



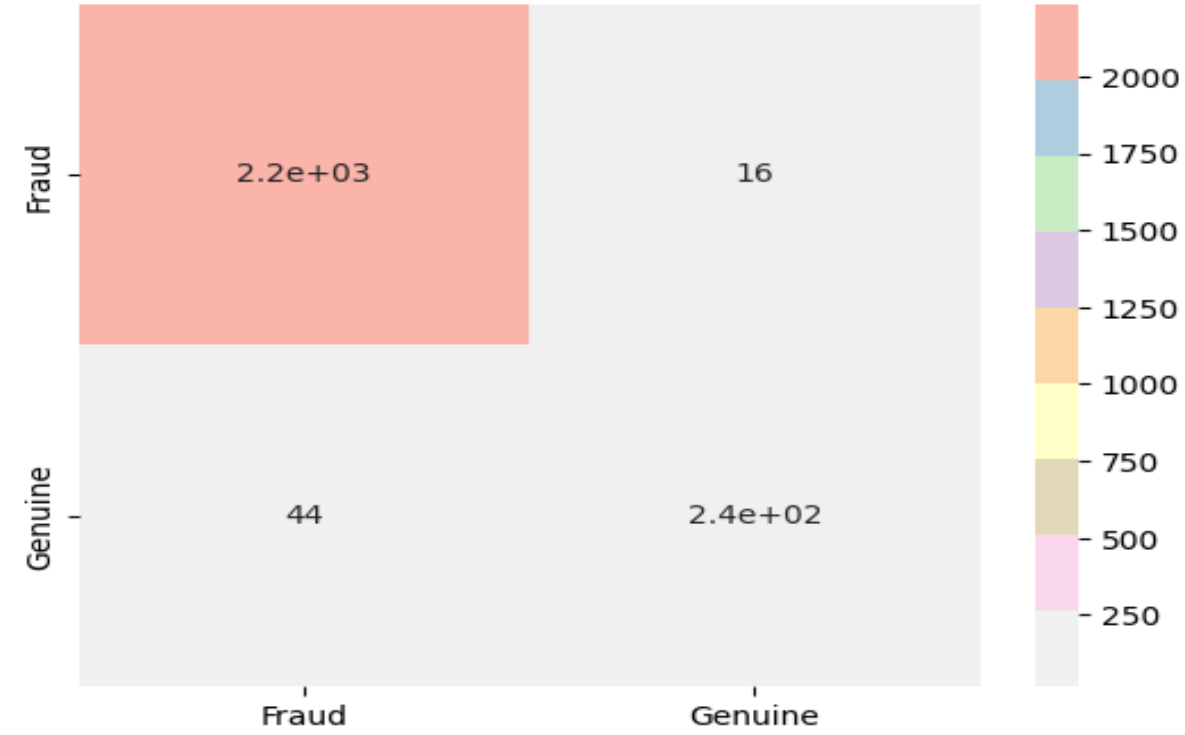


Fig: Confusion Matrix for KNN

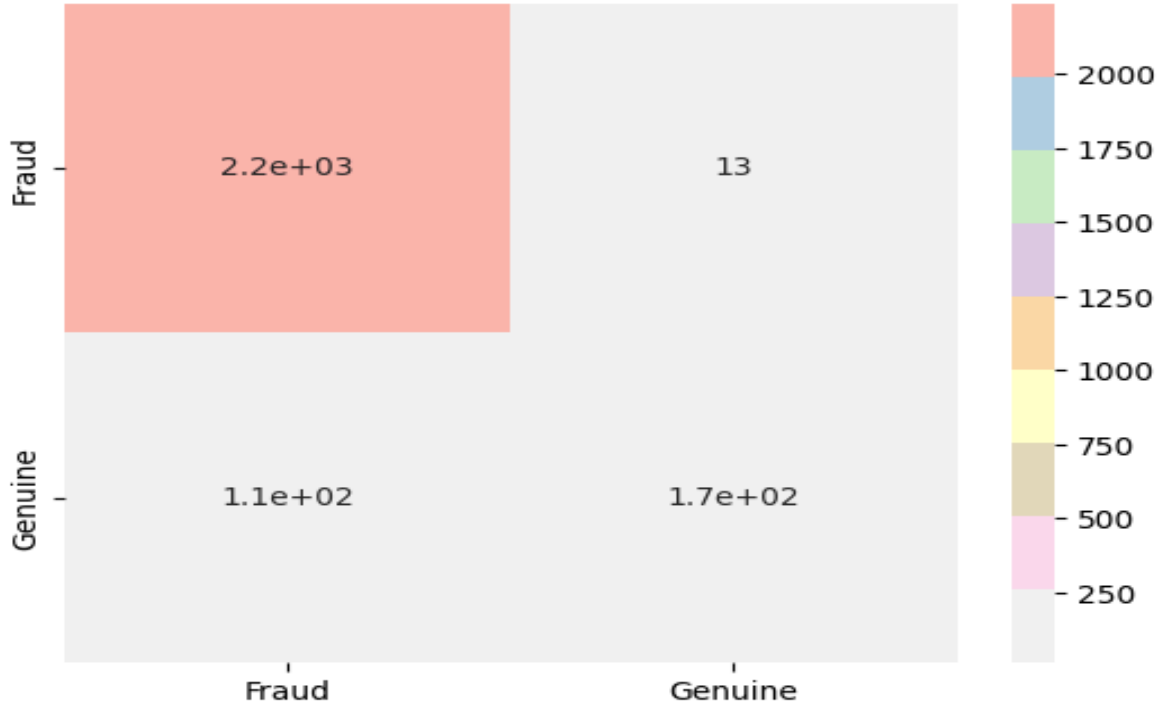


Fig: Confusion Matrix for Logistic Regression

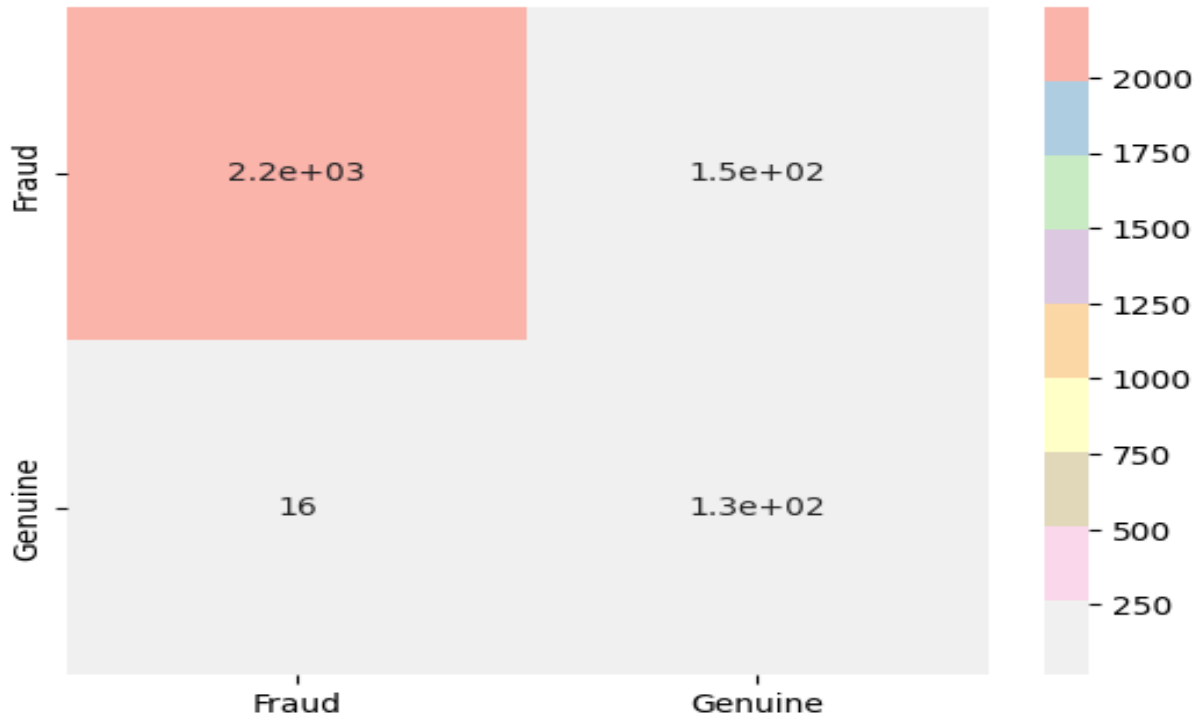


Fig: Confusion Matrix for Gnb

9. Conclusion:

Our research reveals that the K-Nearest Neighbors (KNN) algorithm achieved the highest accuracy, followed by logistic regression, while Gaussian Naive Bayes (GNB) performed less optimally. Despite GNB being the least-performing algorithm, we are satisfied with its accuracy, which reached 93%. This underscores the significance of understanding the relative strengths and weaknesses of different algorithms and emphasizes the acceptable performance achieved by GNB in the context of our study. Our research is dedicated to identifying fraudulent payments in our model, and we are obtaining satisfactory results. We believe that the approach developed in our work can be applied to other models,

providing an effective means of accurately assessing fraud percentages. Detecting fraudulent activities in online transactions poses a significant challenge, requiring the development of a supervised model capable of categorizing large volumes of data. Through our research, we aim to contribute to the creation of such supervised models using established machine-learning algorithms. We hope that this research not only advances the understanding of existing algorithms but also provides insights into their respective accuracy percentages when applied to diverse models. Ultimately, we aspire to offer valuable information and inspire others in the field to explore and implement effective solutions for combating online transaction fraud.

10. Future Work:

Our future plans involve integrating the SVM model into our detection process to streamline operations and aspire to achieve higher accuracy levels. Additionally, we aim to incorporate artificial intelligence to automate and enhance the efficiency of the entire process. These advancements are geared towards making our fraud detection system more robust and effective in ensuring the security of online transactions.