Credit Card Fraud Detection Design and Innovation

Date	**31-10-2023**
Team ID	**3890**
Project Name	**Credit Card Fraud Detection**

Table of Contents

1. Introduction

The project aims to develop a machine learning-based system that analyses transaction data in real-time, effectively detecting credit card fraud while minimizing false positives. This solution will help financial institutions protect against fraudulent transactions, reducing financial losses and ensuring customer trust.

2. Problem Statement

Develop an innovative and highly accurate credit card fraud detection system that addresses the evolving nature of fraud in the digital age. The primary objective is to minimize financial losses for both cardholders and financial institutions while maintaining a seamless user experience.

3. Literature Survey

Detection of Credit Card Detection using Machine Learning:

The results of a study that compares the performance of different machine learning algorithm and neural networks in detecting credit card fraud. The study evaluates the accuracy of various techniques such as KNN, Naïve Bayes, Logistic Regression. The study also finds that MLP performs better than CFLANN in terms of accuracy and time required to converge to a solution.

Review of Machine Learning Approach:

The paper provides a comparative analysis of various techniques such as Logistic Regression, K-Nearest Neighbour, Naïve Bayes, Decision Trees, and Neural Network Algorithms. The study aims to identify the most effective technique for detecting fraudulent transactions and to help financial organizations protect their customers from credit card theft.

A Novel Approach for Detecting Credit Card Fraud Detection using Machine Learning:

Decision tree and random forest algorithms are machine learning techniques used in detecting credit card fraud. The decision tree algorithm is a predictive modelling approach that groups uncommon events in a business from an accredited customer. It enforces the consideration of all the probable outcomes of a decision and creates a comprehensive analysis of the consequences. On the other hand, the random forest algorithm is an improved version of the decision tree algorithm that uses a combination of decision trees to give better results. Every single decision tree draft for the diverse condition and works on arbitrary data sets and on the decision trees. Every tree gives the possibility of the scam business and non-scam as well.

Unsupervised Machine Learning Techniques for Behavorial-Based Credit Card Users Segmentation:

This process includes data pre-processing, modelling, and generating user profiles as outputs. The implementation is illustrated through a concrete case using credit card usage data from the Commercial International Bank of Egypt. The PDF file also recommends updating the segmentation model on a quarterly basis to capture new behaviours from customers and suggests further investigations using monthly, bi-annual, and annual data to understand the similarities and dissimilarities of card customers' behaviours across Africa.

4. Design Thinking Process:

Data Collection:

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable, and it takes value 1 in case of fraud and 0 otherwise. **Dataset Link:** https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

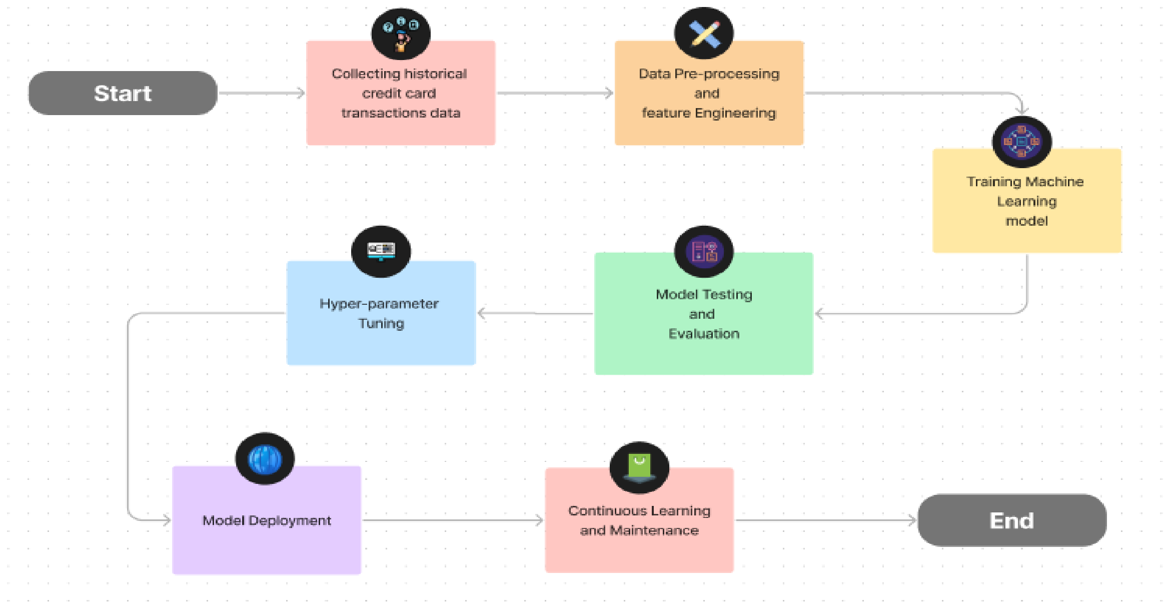
Model Selection and Training:

Innovation: Ensemble Methods and Classification Methods.

The most powerful model for classification is Logistic Regression and Decision Tree to classify the results, improve the model by changing the various feature selection. Developed the model using ensemble techniques includes Random Forest Classifier and Gradient Boosting Variants etc.

Continuous Learning:

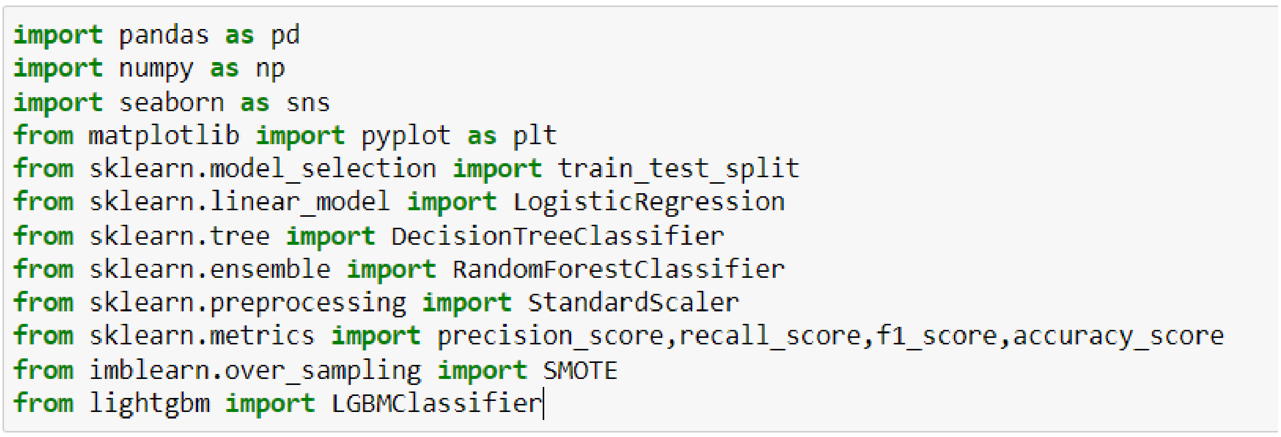
After deploying the model into a production environment, continuous monitoring is essential. It involves real-time tracking of model performance, setting up alerts to detect potential issues or drift in data patterns and regular updates.


Start
Collecting historical credit card transactions data
Data Pre-processing and feature Engineering
Training Machine Learning model
Model Testing and Evaluation
Hyper-parameter Tuning
Model Deployment
Continuous Learning and Maintenance
End

5.Phases of Development:

5.1 Phase 1: Importing Dependencies

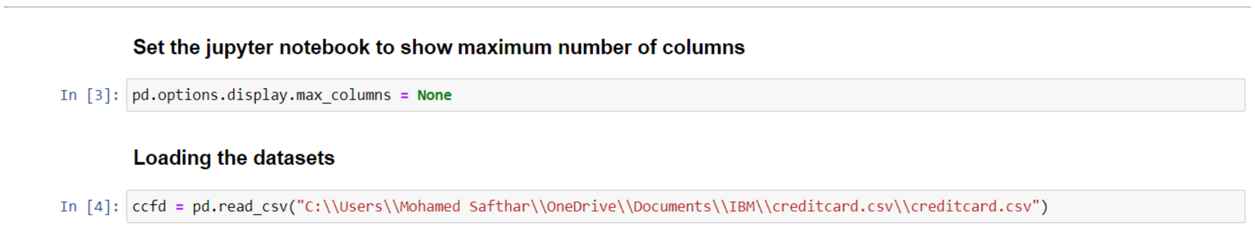
This phase involves importing necessary python libraries and modules. These libraries are required for data processing, visualization and various machine learning tasks.



```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score
from imblearn.over_sampling import SMOTE
from lightgbm import LGBMClassifier
```

5.2 Phase 2: Reading and Viewing Data

In this phase, we are importing the whole dataset into the Jupyter Notebook.



Set the jupyter notebook to show maximum number of columns

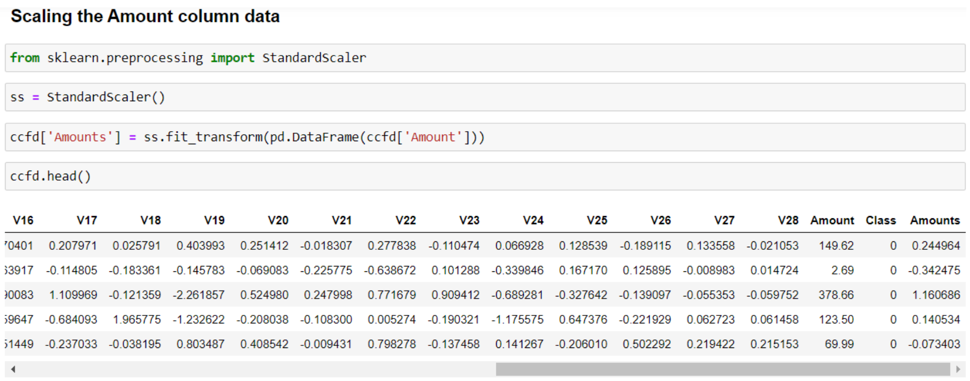
```
In [3]: pd.options.display.max_columns = None
```

Loading the datasets

```
In [4]: ccfd = pd.read_csv("C:\\Users\\Mohamed Safthar\\OneDrive\\Documents\\IBM\\creditcard.csv\\creditcard.csv")
```

5.3 Phase 3: Data Preprocessing and Exploration

Here data preprocessing and exploration occur, including column selection, column transformation, creating a new feature called Amounts. Since, in given dataset there is no outliers etc.



Scaling the Amount column data

```
from sklearn.preprocessing import StandardScaler
```

```
ss = StandardScaler()
```

```
ccfd['Amounts'] = ss.fit_transform(pd.DataFrame(ccfd['Amount']))
```

```
ccfd.head()
```

V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class	Amounts
0401	0.207971	0.025791	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0	0.244964
3917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0	-0.342475
0083	1.109969	-0.121359	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0	1.160686
9647	-0.684093	1.965775	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0	0.140534
1449	-0.237033	-0.038195	0.803487	0.408542	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0	-0.073403

Dropping the duplicate records

```
ccfd.duplicated().any()
```

```
True
```

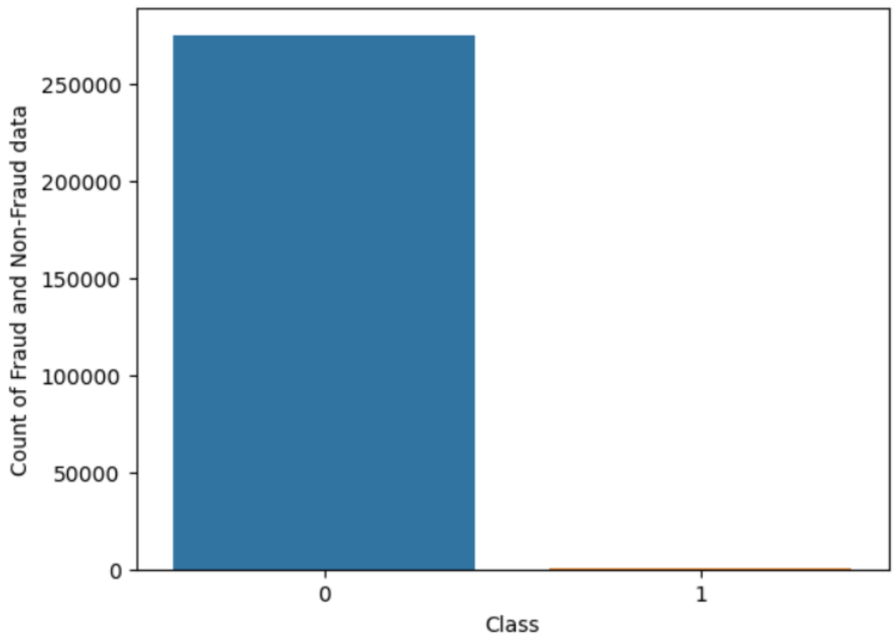
```
ccfd.drop_duplicates(inplace=True)
```

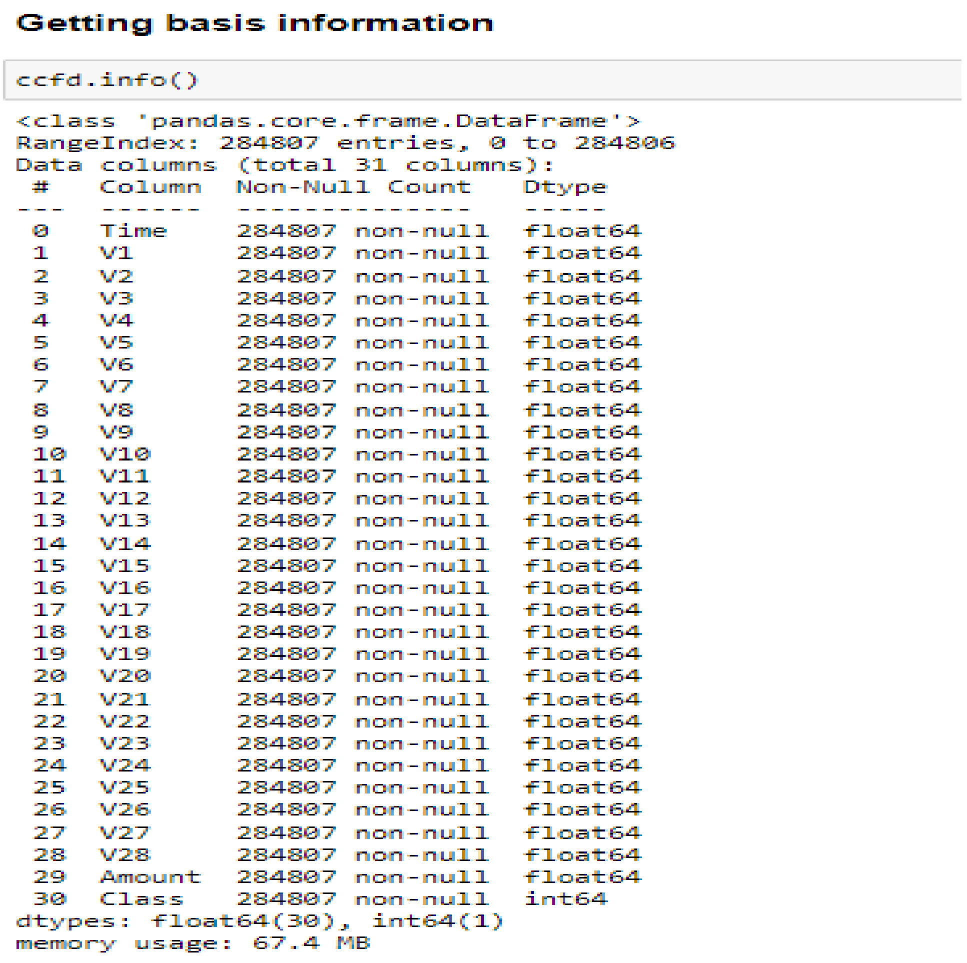
```
ccfd.shape
```

```
(275663, 30)
```

```
284807 - 275663
```

```
9144
```





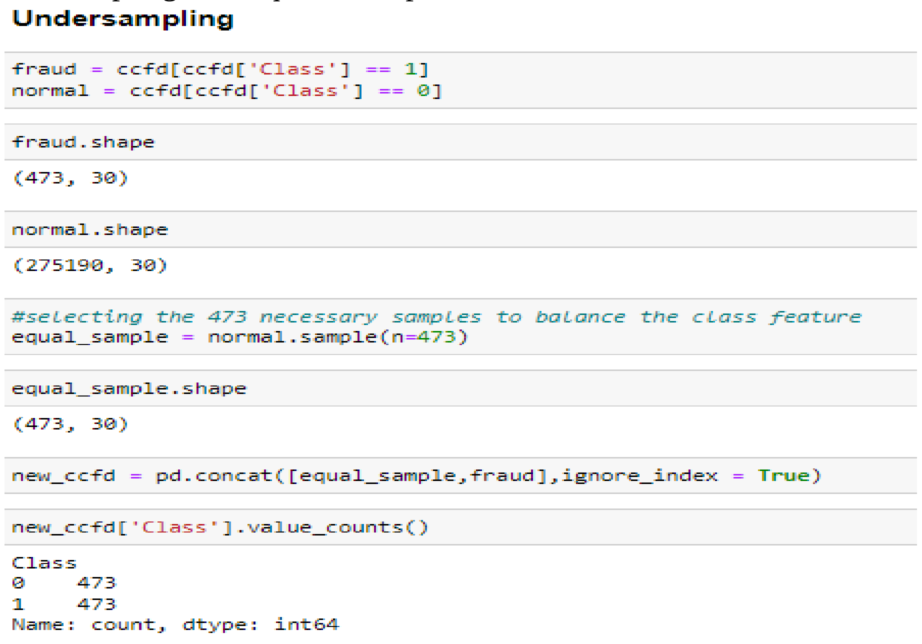
Getting basis information

```
ccfd.info()
```

```
<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
```

5.4 Phase 4: Model Training

This phase includes splitting the dataset into training and testing sets and preparing the features and target values for machine learning models. Apply under sampling and oversampling technique in this phase.



Undersampling

```
fraud = ccfd[ccfd['Class'] == 1]
normal = ccfd[ccfd['Class'] == 0]
```

```
fraud.shape
```

```
(473, 30)
```

```
normal.shape
```

```
(275190, 30)
```

```
#selecting the 473 necessary samples to balance the class feature
equal_sample = normal.sample(n=473)
```

```
equal_sample.shape
```

```
(473, 30)
```

```
new_ccfd = pd.concat([equal_sample,fraud],ignore_index = True)
```

```
new_ccfd['Class'].value_counts()
```

```
Class
0    473
1    473
Name: count, dtype: int64
```

Oversampling Technique:

```
from imblearn.over_sampling import SMOTE
```

```
x2 = ccfd.drop('Class',axis=1)
```

```
x2.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V1
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.46817
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.63555
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.34586
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.63141
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.17512

```
y2 = ccfd.Class
```

```
y2
```

```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 275663, dtype: int64
```

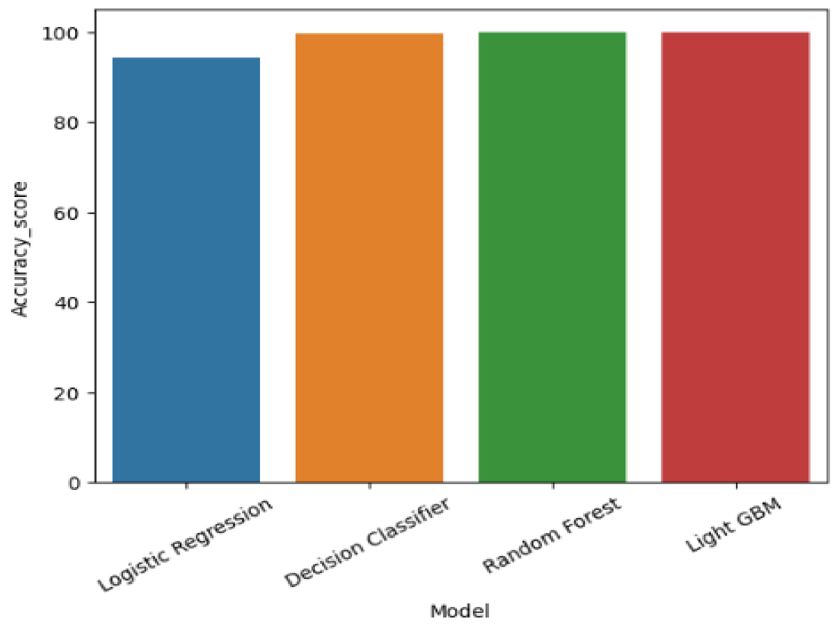
```
X_res,y_res = SMOTE().fit_resample(x2,y2)
```

```
y_res.value_counts()
```

```
Class
0    275190
1    275190
Name: count, dtype: int64
```

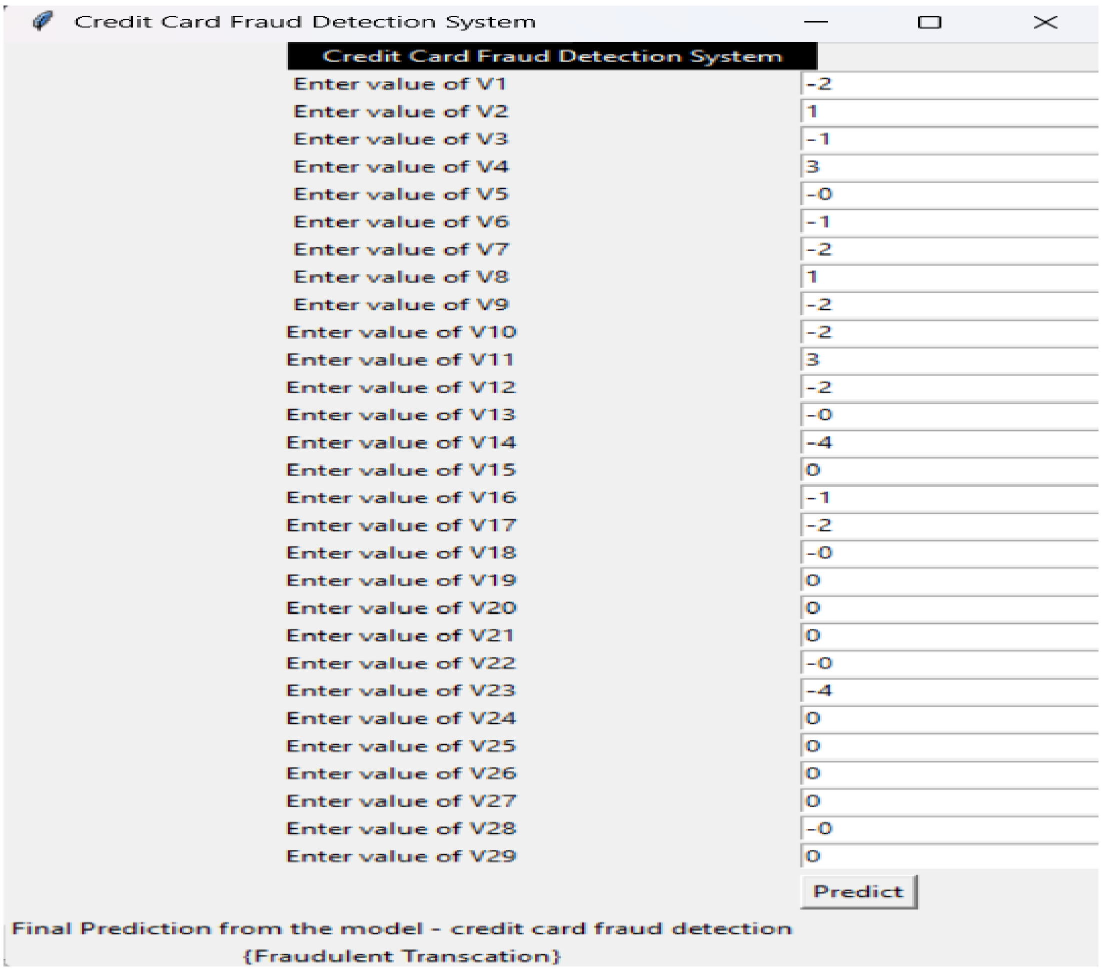
	Model	Accuracy_score
0	Logistic Regression	94.438388
1	Decision Classifier	99.824667
2	Random Forest	99.991824
3	Light GBM	99.906428

```
sns.barplot(x = 'Model',y = 'Accuracy_score',data = stats_oversampling)
plt.xticks(rotation=30)
plt.show()
```



5.5 Phase 5: Model Deployment

For better user experience, the model has been deployed on desktop application, this app is created by tkinter module in python.

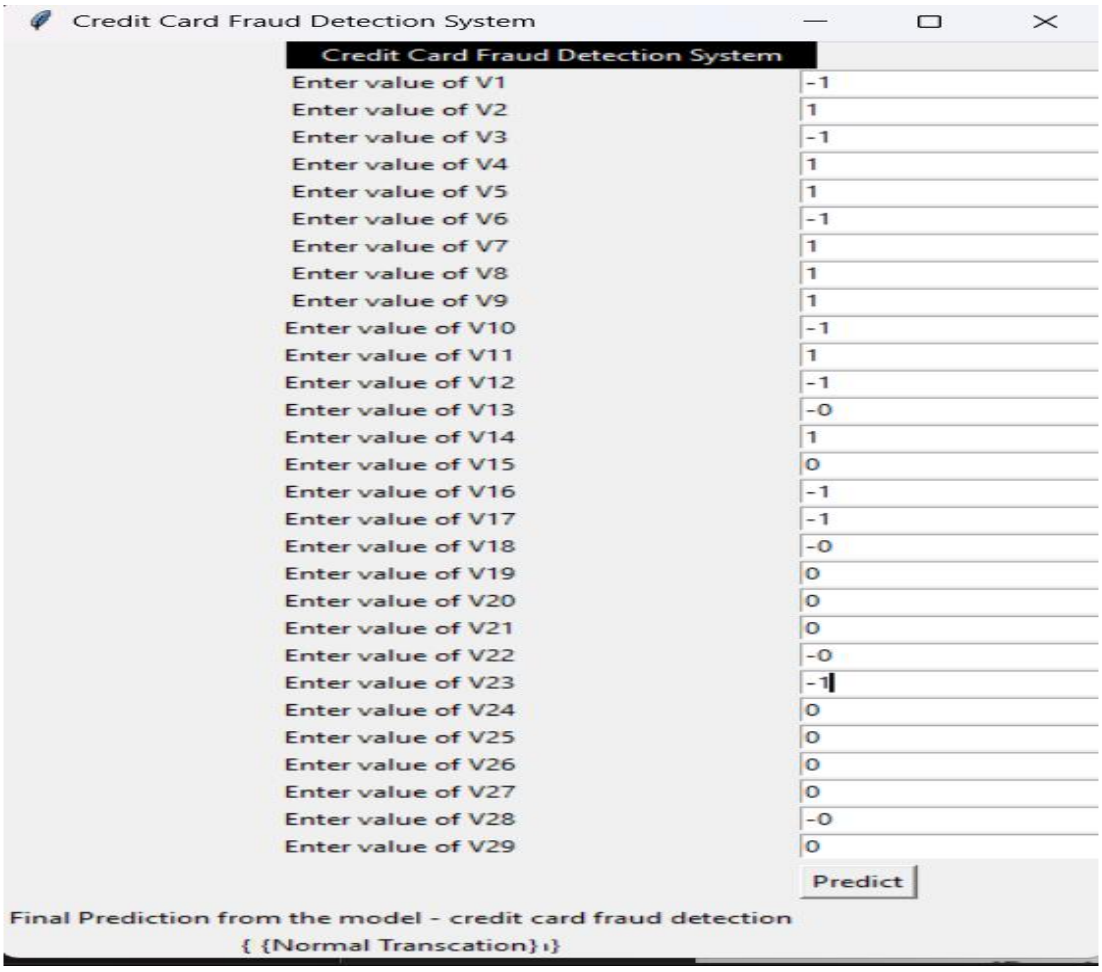


Credit Card Fraud Detection System

Credit Card Fraud Detection System

Enter value of V1	-2
Enter value of V2	1
Enter value of V3	-1
Enter value of V4	3
Enter value of V5	-0
Enter value of V6	-1
Enter value of V7	-2
Enter value of V8	1
Enter value of V9	-2
Enter value of V10	-2
Enter value of V11	3
Enter value of V12	-2
Enter value of V13	-0
Enter value of V14	-4
Enter value of V15	0
Enter value of V16	-1
Enter value of V17	-2
Enter value of V18	-0
Enter value of V19	0
Enter value of V20	0
Enter value of V21	0
Enter value of V22	-0
Enter value of V23	-4
Enter value of V24	0
Enter value of V25	0
Enter value of V26	0
Enter value of V27	0
Enter value of V28	-0
Enter value of V29	0

Final Prediction from the model - credit card fraud detection
{Fraudulent Transcation}



Credit Card Fraud Detection System

Credit Card Fraud Detection System

Enter value of V1	-1
Enter value of V2	1
Enter value of V3	-1
Enter value of V4	1
Enter value of V5	1
Enter value of V6	-1
Enter value of V7	1
Enter value of V8	1
Enter value of V9	1
Enter value of V10	-1
Enter value of V11	1
Enter value of V12	-1
Enter value of V13	-0
Enter value of V14	1
Enter value of V15	0
Enter value of V16	-1
Enter value of V17	-1
Enter value of V18	-0
Enter value of V19	0
Enter value of V20	0
Enter value of V21	0
Enter value of V22	-0
Enter value of V23	-1
Enter value of V24	0
Enter value of V25	0
Enter value of V26	0
Enter value of V27	0
Enter value of V28	-0
Enter value of V29	0

Final Prediction from the model - credit card fraud detection

{ {Normal Transcation} ,}

6. Conclusion:

In conclusion, our credit card fraud detection project demonstrates the power of machine learning in safeguarding financial transactions. By harnessing advanced data pre-processing techniques and a diverse range of classification algorithms, we've developed a robust model capable of identifying fraudulent activities. Our innovative approach, which includes ensemble methods and Light GBM, enhances the accuracy and reliability of our predictions. This project not only protects consumers and financial institutions from fraudulent activities but also sets a strong foundation for further advancements in the field of fraud detection and prevention through machine learning.