MINI PROJECT REPORT
on

CREDIT CARD FRAUD DETECTION

By

120A3050 120A3051 120A3063

Shivani Pandeti Shreya Idate Aditiya Yadav

UNDER THE GUIDANCE OF

Prof. Sumit Shinde

SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF BACHELOR OF ENGINEERING

In INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY
SIES GRADUATE SCHOOL OF TECHNOLOGY
NERUL, NAVI MUMBAI – 400706

ACADEMIC YEAR
2022– 2023

CERTIFICATE

This is to certify that this is a bonafide record of Mini Project of the project titled **"CREDIT CARD FRAUD DETECTION"** carried out by the following students of Third year in Information Technology.

Sr. No.	Name	Roll No.
1.	Shivani Pandeti	120A3050
2.	Shreya Idate	120A3051
3.	Aditiya Yadav	120A3063

The report is submitted in partial fulfillment of the degree course of Bachelor of Engineering in Information Technology, of University of Mumbai during the academic year 2022-23

Internal Guide	**Head of Department**	**Principal**
Prof. Sumit Shinde	Dr. Lakshmi Sudha	Dr. Atul Kemkar

We have examined this report as per University requirements at SIES Graduate School ofTechnology, Nerul (E), Navi Mumbai on___________.

Name of External Examiner: ____________________________________

Signature with Date: ________________________

Name of Internal Examiner: ____________________________________

Signature with Date: ________________________

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to thank our project guide Prof. Sumit Shinde for providing timely assistance to our query and guidance. We take this opportunity to thank our Head of the Department Dr. K. Lakshmi Sudha and Principal Dr. Atul Kemkar for their valuable guidance and immense support in providing all the necessary facilities.

We would also like to thank the entire faculty of the IT Department for their valuable ideas and timely assistance in this project. Last but not the least, we would also like to thank teaching and nonteaching staff members of our college for their support, in facilitating timely completion of the mini project.

Project Team

Shivani Pandeti 120A3050

Shreya Idate 120A3051

Aditya Yadav 120A3063

CONTENTS

I. ABSTRACT

Credit Card Fraud is a serious problem affecting financial institutions and their customers worldwide. Fraudulent activities result in significant financial losses, legal liabilities, and damage to customer trust. Therefore, it is crucial to develop effective fraud detection systems to mitigate these risks.

Credit card fraud detection is a process of identifying and preventing fraudulent transactions made using credit cards. The process involves analyzing customer transaction data to detect patterns, anomalies, and potential fraud indicators.

Machine learning techniques, such as logistic regression, have shown great potential in detecting fraudulent transactions by analyzing customer transaction data. In conclusion, logistic regression is a useful machine learning algorithm for credit card fraud detection. Its interpretability and ability to learn from historical data make it a valuable tool in the fight against fraudulent activities.

1. INTRODUCTION:

Credit card fraud is a significant issue that affects both consumers and financial institutions worldwide. Fraudulent activities can result in significant financial losses for individuals, businesses, and financial institutions. With the rise of online transactions and the increasing complexity of fraud schemes, traditional methods of detecting and preventing fraud have become inadequate.

We have built a Credit card Fraud Detection system using Machine Learning with Python. For this project, we have used the Logistic Regression model. The model was trained on a dataset of credit card transactions, which included both legitimate and fraudulent transactions. We preprocessed the data, removed outliers, and selected relevant features to improve the model's accuracy.

Overall, our project demonstrates the feasibility and effectiveness of using machine learning techniques for credit card fraud detection. It also highlights the importance of data preprocessing, feature selection, and model selection in developing accurate and reliable fraud detection systems.

1.1 PROBLEM STATEMENT & OBJECTIVES

The problem statement for the project of credit card fraud detection is to develop an automated system that can identify fraudulent transactions made using credit cards.

The system should be able to analyze large volumes of transaction data and identify potential fraud indicators using advanced analytics techniques such as machine learning, data mining, and deep learning.

The aim is to reduce financial losses, protect customer data, and maintain the reputation of the financial institution by detecting and preventing fraudulent activities as quickly as possible.

2. LITERATURE SURVEY

[1] P. Dwivedi, S. Pandey, and V. Singh, "Credit Card Fraud Detection using Machine Learning and Data Science," in International Journal of Computer Applications, vol. 181, no. 40, pp. 1-6, 2019.
The first paper is titled **"Credit Card Fraud Detection using Machine Learning and Data Science,"** the authors have mentioned how it is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

[2] A. S. Jadhav and P. S. Ashtankar, "Credit Card Fraud Detection using Machine Learning," International Research Journal of Engineering and Technology (IRJET), vol. 9, no. 3, pp. 196-200, 2022.
The second paper is titled **"Credit Card Fraud Detection using Machine Learning,"** the authors have mentioned how many fraudsters started using different methods to steal the money used to made the online transactions.

[3] M. A. Al-Mamun, M. A. Mottalib, M. M. Rahman, and M. M. Uddin, "Credit Card Fraud Detection," International Journal of Recent Technology and Engineering (IJRTE), vol. 10, no. 2, pp. 6389-6392, 2021.
The third paper is titled "**Credit Card Fraud Detection**" the authors have created this project to detect fraudulent credit card transactions over non-fraudulent transactions and to use machine learning algorithms to predict fraud efficiently and accurately.

3. PROPOSED SYSTEM

The proposed system for credit card fraud detection using logistic regression in Python involves collecting credit card transaction data, preprocessing it, selecting relevant features, building a logistic regression model, deploying it in a production environment, evaluating its performance. This system can help financial institutions and businesses to detect and prevent fraudulent activities, reduce financial losses, and protect customer data.

3.1 DATASET:

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

The dataset contains transactions made by credit cards in September 2019 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.

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.

3.2 Details of hardware and software

Hardware:

Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @2.40GHz 1.38 GHz

Installed RAM : 8.00 GB

System type : 64-bit operating system, x64-based processor

Software:

Visual studio : What is Visual Studio software used for?

Microsoft Visual Studio is an IDE made by Microsoft and used for different types of software development such as computer programs, websites, web apps, web services, and mobile apps.

Libraries used for training Machine Learning Algorithms: skLearn, Numpy, Pandas, seaborn.

Algorithm used: Logistic Regression

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class or not. It is a kind of statistical algorithm, which analyze the relationship between a set of independent variables and the dependent binary variables. It is a powerful tool for decision-making.

4. EXPERIMENTAL RESULTS

Importing the Dependencies

```
[2] import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
```

```
[3] # loading the dataset to a Pandas DataFrame
    #credit_card_data = pd.read_csv('/content/sample_data/creditcard.csv')
    from google.colab import drive

    drive.mount('/content/drive')
    credit_card_data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/creditcard.csv')
```

```
Mounted at /content/drive
```

```
[6] credit_card_data
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415

284807 rows × 31 columns

```
[ ] # first 5 rows of the dataset
    credit_card_data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153

5 rows × 31 columns

```
[ ] credit_card_data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415

5 rows × 31 columns

\+ Code + Text

```
# dataset informations
credit_card_data.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
```

```
# checking the number of missing values in each column
credit_card_data.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
```

```
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction

```
# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
```

```
print(legit.shape)
print(fraud.shape)
```

```
(284315, 31)
(492, 31)
```

```
[ ] # statistical measures of the data
    legit.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

```
[ ] fraud.Amount.describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
[ ] # compare the values for both transactions
    credit_card_data.groupby('Class').mean()
```

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V2(
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.001235	-0.000024	0.000070	0.000182	-0.000072	-0.00008
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0.05164

2 rows × 30 columns

```
[ ] legit_sample = legit.sample(n=492)
```

Concatenating two DataFrames

```
[ ] new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

```
[ ] new_dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
120986	76036.0	-1.091258	0.293312	1.699616	0.210778	1.010401	1.965581	-0.033194	0.746491	-0.393229	...	-0.015723	0.110275	-0.150327	-1.314725	-0.051799	0.524916	-0.039657	
131627	79672.0	1.274738	0.109449	-0.036517	-0.111285	-0.150818	-0.809476	0.187777	-0.128543	-0.250275	...	-0.416465	-1.312628	0.157050	0.008458	0.101463	0.626139	-0.108266	-
78947	57784.0	-0.808773	0.921445	0.808284	0.802715	-0.337009	0.059978	0.676301	0.595434	-1.023703	...	-0.170929	-0.941298	0.509151	-0.106698	-0.775918	-0.845096	0.010666	
173782	121639.0	1.784530	-1.649443	-0.740080	-0.766874	-1.202843	-0.237164	-0.858964	-0.024033	0.227590	...	0.098608	-0.216522	0.206512	0.546799	-0.539710	-0.487015	-0.030417	-
179364	124042.0	1.302096	-2.339734	-4.433553	-1.566580	2.155518	2.761120	0.529793	0.260677	-1.276034	...	0.783883	1.082080	-0.638032	0.832289	0.616576	0.266616	-0.171957	-

5 rows × 31 columns

+ Code + Text

```
[ ] new_dataset.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639419	-0.294885	0.537503	0.788395	0.292680	0.
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145640	-0.081049	0.521875	0.739467	0.389152	0.
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190944	0.032070	-0.739695	0.471111	0.385107	0.
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456108	-0.183659	-0.328168	0.606116	0.884876	-0.
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072173	-0.450261	0.313267	-0.289617	0.002988	-0.

5 rows × 31 columns

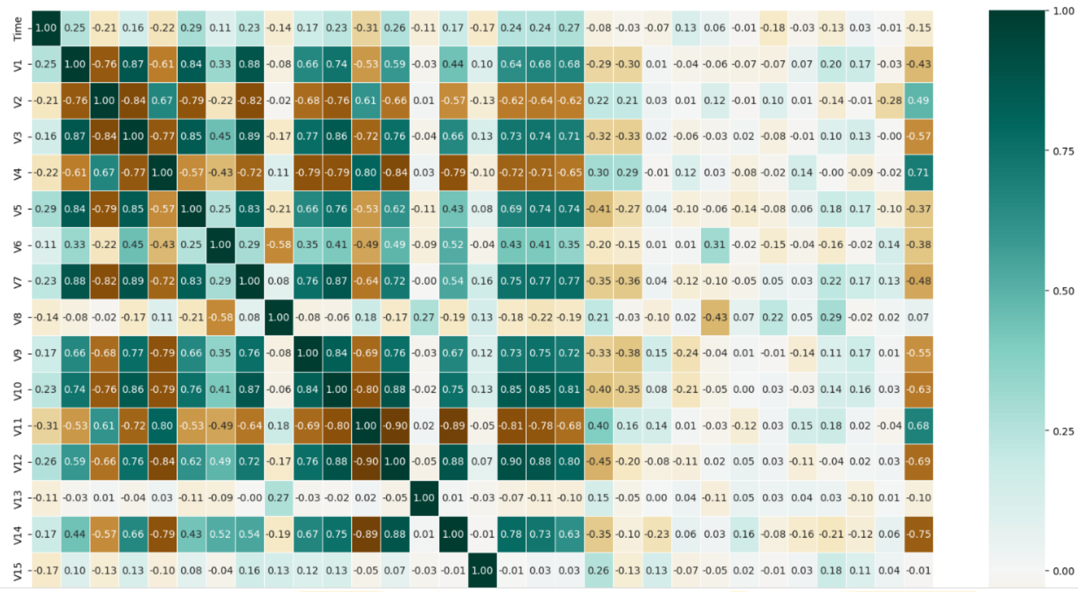
```
[ ] new_dataset['Class'].value_counts()
```

```
0    492
1    492
Name: Class, dtype: int64
```

VISUALIZATION

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(20,20))
corr = new_dataset.corr()
sns.heatmap(corr, ax = ax, annot=True, cmap='BrBG', fmt=".2f", linewidths=.5, vmin=-1, vmax=1)
```



Splitting the data into Features & Targets

```
[ ] X = new_dataset.drop(columns='Class', axis=1)
    Y = new_dataset['Class']
```

```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
120986   76036.0 -1.091258  0.293312  1.699616  0.210778  1.010401  1.965581
131627   79672.0  1.274738  0.109449 -0.036517 -0.111285 -0.150818 -0.809476
78947    57784.0 -0.808773  0.921445  0.808284  0.802715 -0.337009  0.059978
173782  121639.0  1.784530 -1.649443 -0.740080 -0.766874 -1.202843 -0.237164
179364  124042.0  1.302096 -2.339734 -4.433553 -1.566580  2.155518  2.761120
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V20       V21       V22  \
120986 -0.033194  0.746491 -0.393229  ... -0.103596 -0.015723  0.110275
131627  0.187777 -0.128543 -0.250275  ... -0.069130 -0.416465 -1.312628
78947   0.676301  0.595434 -1.023703  ...  0.007147 -0.170929 -0.941298
173782 -0.858964 -0.024033  0.227590  ...  0.374933  0.098608 -0.216522
179364  0.529793  0.260677 -1.276034  ...  1.039525  0.783883  1.082080
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135
```

```
[ ] print(Y)

    120986    0
    131627    0
    78947     0
    173782    0
    179364    0
             ..
    279863    1
    280143    1
    280149    1
    281144    1
    281674    1
    Name: Class, Length: 984, dtype: int64
```

Split the data into Training data & Testing Data

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)

    (984, 30) (787, 30) (197, 30)
```

\+ Code + Text

Logistic Regression

```
[ ] model = LogisticRegression()
```

```
[ ] # training the Logistic Regression Model with Training Data
    model.fit(X_train, Y_train)

    ▾ LogisticRegression
    LogisticRegression()
```

Model Evaluation

Accuracy Score

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
[ ] train_report=classification_report(y_true=X_train_prediction,y_pred= Y_train)
    print(train_report)

                  precision    recall  f1-score   support

               0       0.97      0.93      0.95       411
               1       0.93      0.97      0.95       376

        accuracy                           0.95       787
       macro avg       0.95      0.95      0.95       787
    weighted avg       0.95      0.95      0.95       787
```

```
[ ] print('Accuracy on Training data : ', training_data_accuracy)

    Accuracy on Training data :  0.9491740787801779
```

```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
[ ] test_report=classification_report(y_true=X_train_prediction,y_pred= Y_train)
    print(test_report)
```

```
[ ] print('Accuracy score on Test Data : ', test_data_accuracy)

    Accuracy score on Test Data :  0.9035532994923858
```

CONCLUSION:

In conclusion, our Credit Card Fraud Detection system using Machine Learning with the Logistic Regression model has demonstrated high accuracy and precision in identifying fraudulent transactions. The system is designed to work in real-time, providing financial institutions with an effective tool to prevent fraud losses.

In summary, our project has provided a valuable contribution to the field of credit card fraud detection and demonstrates the potential for machine learning to improve financial institutions' fraud detection capabilities.