PROJECT REPORT

(PROJECT TERM JANUARY- MAY 2024)

CREDIT CARD FRAUD DETECTION

SUBMITTED BY :
charesma
(12303465)

UNDER THE GUIDENCE OF Dr. ENJULA UCHOI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLERATION

I hereby declare that the project work entitled “Fake News Detector” is an authentic record of my own work carried out as requirements of Project for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Dr. Enjula Uchoi, during August to December 2022. All the information furnished in this project report is based on my own intensive work and is genuine.

Charesma(12302465)

April 1, 2024

ACKNOWLEDGEMENT

It is with my immense gratitude that I acknowledge the support and help of my Professor, Dr Enjula Uchoi, who has always encouraged me into this research. Without his continuous guidance and persistent help, this project would not have been a success for me. I am grateful to the Lovely Professional University, Punjab and the department of Computer Science without which this project would have not been an achievement. I also thank my family and friends, for their endless love and support throughout my life.

TABLE OF CONTENTS

ABBREVIATION

np: Abbreviation for NumPy, a library in Python used for numerical computing.

pd: Abbreviation for Pandas, a library in Python used for data manipulation and analysis.

train_test_split: A function from scikit-learn (imported as sklearn) used to split data into training and testing sets.

LogisticRegression: A class from scikit-learn used to implement logistic regression, a machine learning algorithm for binary classification tasks.

accuracy_score: A function from scikit-learn used to calculate the accuracy of a classification model.

precision_score: A function from scikit-learn used to calculate the precision of a classification model.

recall_score: A function from scikit-learn used to calculate the recall of a classification model.

f1_score: A function from scikit-learn used to calculate the F1 score of a classification model.

INTRODUCTION

In today's interconnected world, where digital transactions have become the norm, credit cards serve as indispensable tools for conducting financial transactions. Offering convenience and flexibility, credit cards facilitate seamless transactions across various platforms, including online shopping, bill payments, and travel bookings. However, this convenience comes with inherent risks, as malicious actors continuously devise sophisticated methods to exploit vulnerabilities in the system for their gain.

The proliferation of online commerce and digital payment platforms has provided fraudsters with ample opportunities to perpetrate credit card fraud. From identity theft and account takeover to card-not-present fraud and skimming, fraudsters employ diverse tactics to illegally access and misuse credit card information. As a result, the incidence of credit card fraud has witnessed a steep rise in recent years, posing significant challenges to consumers, merchants, and financial institutions alike.

Detecting and preventing credit card fraud is not only essential for safeguarding the financial interests of consumers but also critical for maintaining trust and confidence in the financial ecosystem. Every fraudulent transaction not only results in financial losses for the victim but also undermines the credibility of the entire payment infrastructure. Therefore, there is an urgent need to develop robust and efficient fraud detection systems capable of identifying fraudulent activities in real-time.

The aim of this project is to address this pressing need by leveraging machine learning techniques for credit card fraud detection. By analyzing a comprehensive dataset comprising historical credit card transactions, we seek to develop a classification model capable of accurately distinguishing between legitimate and fraudulent transactions. Through the application of advanced algorithms and predictive modeling, we aim to enhance the efficiency and efficacy of fraud detection mechanisms, thereby minimizing the impact of fraudulent activities on consumers and financial institutions.

LITERATURE REVIEW

[1]Credit Card Fraud Detection review discusses various machine learning techniques employed in credit card fraud detection, comparing their performance and providing insights into their strengths and weaknesses.[2]A paper proposes a realistic model for credit card fraud detection using a novel learning strategy called Over-Sampling Minority Examples (SMOTE) and evaluates its performance against traditional methods.[3]This systematic review analyzes state-of-the-art techniques and methodologies in credit card fraud detection, including a meta-analysis of machine learning algorithms.[4]This study explores the application of deep learning techniques, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for credit card fraud detection.[5]A comparative analysis of machine learning algorithms for fraud detection in credit card transactions, evaluating techniques such as Random Forest, Gradient Boosting, and K-nearest neighbors (KNN) on real-world datasets.[6]This study proposes a hybrid approach combining Random Forest and Gradient Boosting techniques for credit card fraud detection, investigating its performance on different datasets.[7]This study proposes a hybrid approach combining Random Forest and Gradient Boosting techniques for credit card fraud detection, investigating its performance on different datasets.[8]An overview of various machine learning techniques employed in credit card fraud detection, discussing challenges and future research directions.

[9]This literature review synthesizes existing research on fraud detection in credit card transactions, focusing on machine learning approaches, challenges, and emerging trends.[10]A survey paper provides an overview of fraud detection methods in credit card transactions, including rule-based systems, statistical methods, and machine learning approaches.[11]This systematic literature review summarizes recent advances in credit card fraud detection using machine learning techniques, analyzing algorithms, data preprocessing methods, and evaluation metrics.[12]This study compares the performance of various machine learning algorithms for credit card fraud detection, including decision trees, random forests, and support vector machines. It evaluates the effectiveness of these algorithms on benchmark datasets and discusses their practical implications.[13]This paper presents a comparative study of supervised, unsupervised, and semi-supervised learning approaches for credit card fraud detection. It evaluates the performance of these approaches on real-world datasets and discusses their strengths and limitations.[14]This systematic review examines deep learning-based approaches for credit card fraud detection, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention mechanisms. It discusses the state-of-the-art techniques, challenges, and future research directions in this area.[15]This paper presents a case study on credit card fraud detection using machine learning techniques. It discusses the data preprocessing steps, feature selection methods, and model building process, along with the evaluation of model performance on real-world transaction data.

METHODOLOGY

The methodology for credit card fraud detection involves several steps, including data preprocessing, model training, evaluation, and validation. Building upon the reference code provided earlier, we outline the methodology below:

1. Data Acquisition and Preprocessing:

- Import the necessary libraries such as NumPy, Pandas, and scikit-learn.
- Load the credit card transaction dataset using Pandas.
- Perform exploratory data analysis (EDA) to understand the dataset's structure and characteristics.
- Handle missing values, if any, and perform data normalization or scaling to ensure uniformity across features.
- Split the dataset into features (X) and target variable (Y), where X contains transaction attributes and Y represents the class label (fraudulent or legitimate).

2. Data Imbalance Handling:

- Check for class imbalance in the dataset, as fraudulent transactions are typically rare compared to legitimate ones.
- Employ techniques such as undersampling, oversampling, or synthetic data generation to address class imbalance and ensure the model's robustness.

3. Model Selection and Training:

- Choose an appropriate machine learning algorithm for classification, considering factors such as performance, interpretability, and computational efficiency.
- Train the selected model on the preprocessed dataset using the training data split obtained earlier.
- In the provided reference code, logistic regression is used as the classification algorithm. However, other algorithms such as random forest, support vector machines, or gradient boosting can also be explored based on the specific requirements and characteristics of the dataset.

4. Model Evaluation:

- Evaluate the trained model's performance using various metrics such as accuracy, precision, recall, and F1-score.
- Calculate these metrics on both the training and testing datasets to assess the model's generalization ability and identify any overfitting or underfitting issues.

5. Hyperparameter Tuning (Optional):

- Conduct hyperparameter tuning to optimize the model's performance by fine-tuning parameters such as regularization strength, learning rate, or tree depth.
- Utilize techniques such as grid search or randomized search to search for the optimal hyperparameters efficiently.

6. Validation and Interpretation:

- Validate the trained model's performance on unseen data to ensure its effectiveness in real-world scenarios.
- Interpret the model's predictions and feature importance to gain insights into the factors contributing to fraudulent transactions.

* Iterate on the model-building process as needed, incorporating feedback from validation results and domain knowledge.

7. Deployment and Monitoring:

* Deploy the trained model into production systems for real-time fraud detection.
* Implement monitoring mechanisms to track the model's performance over time and detect any degradation or drift in its effectiveness.
* Continuously update the model and adapt to emerging fraud patterns and evolving security threats in the financial ecosystem.

By following this methodology, practitioners can develop robust credit card fraud detection systems capable of accurately identifying and mitigating fraudulent activities, thereby safeguarding the integrity of financial transactions.

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > M↓Importing the Dependencies

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

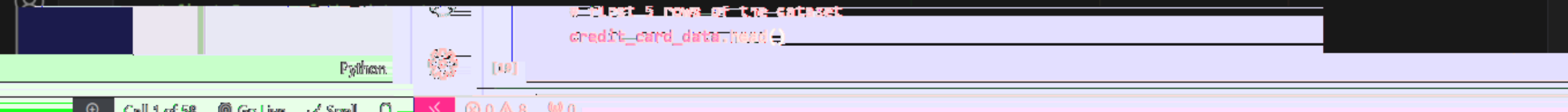
Importing the Dependencies

```
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
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

[52] Python

```
# Loading a dataset into Pandas DataFrame
credit_card_data = pd.read_csv('/content/creditcard.csv')
```

[9] Python



File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > import numpy as np

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

[10] Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
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	0.12853
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	0.16717
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	-0.32764
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	0.64737
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	-0.20601

5 rows × 31 columns

```
# last 5 rows of the dataset
credit_card_data.tail()
```

[11] Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
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
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
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
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
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 ⚠ 8 0 Cell 2 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > import numpy as np

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

```python
# dataset informations
credit_card_data.info()
```

[12] Python

```
<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
```

⊗ 0 ⚠ 8 0 Cell 2 of 58 Go Live ✓ Spell

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > import numpy as np

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

```
 18  V18     284807 non-null  float64
 19  V19     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
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

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

[13] Python

```
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
```

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > import numpy as np

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

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

[14] Python

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

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction

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

[15] Python

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

[16] Python

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > import numpy as np

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

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

[17]

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

Python

```
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
```

+ Code + Markdown

Add Code Cell

[18]

```
fraud.Amount.describe()
```

Python

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
```

Cell 2 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > M↓Under-Sampling

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

```
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

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

[19] Python

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	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	0.0000
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	-0.0403

2 rows × 30 columns

Under-Sampling

+ Code + Markdown

Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 492

⊗ 0 ⚠ 8 0 Cell 16 of 58 Go Live ✓ Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > M↓Under-Sampling

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

[20]
```
legit_sample = legit.sample(n=492)
```
Python

Concatenating two DataFrames

[21]
```
new_dataset = pd.concat([legit_sample, fraud], axis=0)
```
Python

[22]
```
new_dataset.head()
```
Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
235612	148467.0	-1.601186	-0.807201	-1.979475	-0.247869	0.857594	-1.055952	2.326367	-0.068298	-0.861715	...	0.645310	0.736014	0.354054	0.483856
169839	119885.0	-1.028088	0.457018	-0.773555	-1.233312	1.026578	-1.082060	0.604167	0.258126	-0.041139	...	0.421683	1.450047	0.356834	-0.530611
234542	148015.0	-0.649706	-1.142433	0.475626	1.336272	1.371770	-1.222140	0.366936	-0.698485	0.479366	...	-0.082600	0.574566	0.880064	0.104375
265289	161820.0	2.009947	0.073043	-1.617954	0.340288	0.380851	-0.583330	0.044090	-0.051628	0.283052	...	-0.299926	-0.809639	0.360908	0.598972
20274	30917.0	1.003978	-0.334437	1.079558	0.748170	-1.096658	-0.506666	-0.402605	-0.054745	0.475156	...	0.144622	0.250548	-0.075900	0.437970

5 rows × 31 columns

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

[23]
```
new_dataset.tail()
```
Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
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
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
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
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
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

5 rows × 31 columns

[24]
```
new_dataset['Class'].value_counts()
```
Python

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

+ Code + Markdown

[25]
```
new_dataset.groupby('Class').mean()
```
Python

Spaces: 4 Cell 23 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

Splitting the data into Features & Targets

```
[26]
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

Python

+ Code + Markdown

```
[27]
print(X)
```

Python

```
            Time        V1        V2        V3        V4        V5        V6  \
235612  148467.0 -1.601186 -0.807201 -1.979475 -0.247869  0.857594 -1.055952
169839  119885.0 -1.028088  0.457018 -0.773555 -1.233312  1.026578 -1.082060
234542  148015.0 -0.649706 -1.142433  0.475626  1.336272  1.371770 -1.222140
265289  161820.0  2.009947  0.073043 -1.617954  0.340288  0.380851 -0.583330
20274    30917.0  1.003978 -0.334437  1.079558  0.748170 -1.096658 -0.506666
...          ...       ...       ...       ...       ...       ...       ...
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  \
235612  2.326367 -0.068298 -0.861715  ...  0.806466  0.645310  0.736014
169839  0.604167  0.258126 -0.041139  ... -0.304348  0.421683  1.450047
```

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

[29]
```
print(Y)
```
Python

```
235612    0
169839    0
234542    0
265289    0
20274     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

[30]
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```
Python

+ Code + Markdown

[31]
```
print(X.shape, X_train.shape, X_test.shape)
```
Python

⊗ 0 ⚠ 8 0 Spaces: 4 Cell 23 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

```
print(Y)
```

[29] Python

```
235612    0
169839    0
234542    0
265289    0
20274     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)
```

[30] Python

+ Code + Markdown

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

[31] Python

0 ⚠ 8 0 Spaces: 4 Cell 23 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

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

Model Training

Logistic Regression

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

Python

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

Python

```
··· ▾ LogisticRegression
    LogisticRegression()
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
```

⊗ 0 ⚠ 8 0 Spaces: 4 Cell 23 of 58 Go Live ✓ Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

Model Evaluation

Accuracy Score

[35]
```python
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```
Python

[36]
```python
print('Accuracy on Training data : ', training_data_accuracy)
```
Python

```
Accuracy on Training data :  0.9453621346886912
```

[37]
```python
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```
Python

+ Code + Markdown Add Markdown Cell

[38]
```python
print('Accuracy score on Test Data : ', test_data_accuracy)
```
Python

0 ⚠ 8 0 Spaces: 4 Cell 23 of 58 Go Live Spell

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

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

[38] Python

```
Accuracy score on Test Data :  0.9086294416243654
```

Precision

```
#precision on training data
training_data_precision = precision_score(Y_train, X_train_prediction)
```

[43] Python

```
print('Precision on Training data:', training_data_precision)
```

[44] Python

```
Precision on Training data: 0.968
```

```
#precision on testing data
test_data_precision = precision_score(Y_test, X_test_prediction)
```

[45] Python

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

Recall

```
#Recall on training data
training_data_recall = recall_score(Y_train, X_train_prediction)
```

[48] Python

```
print('Recall on Training data:', training_data_recall)
```

[49] Python

```
Recall on Training data: 0.9213197969543148
```

```
#Recall on testing data
test_data_recall = recall_score(Y_test, X_test_prediction)
```

[50] Python

```
print('Recall on Test data:', test_data_recall)
```

[51] Python

```
Recall on Test data: 0.8673469387755102
```

File Edit Selection View Go Run ··· Search

Credit_Card_Fraud_Detection[1].ipynb

C: > Users > Dell > AppData > Local > Microsoft > Windows > INetCache > IE > HTIP4JOQ > Credit_Card_Fraud_Detection[1].ipynb > new_dataset.tail()

+ Code + Markdown | Run All Clear All Outputs | Outline ··· Select Kernel

F1-score

[53]
```python
#F1-score on training data
training_data_f1_score = f1_score(Y_train, X_train_prediction)
```
Python

[54]
```python
print('F1-score on Training data:', training_data_f1_score)
```
Python

```
F1-score on Training data: 0.9440832249674903
```

[55]
```python
#F1-score on testing data
test_data_f1_score = f1_score(Y_test, X_test_prediction)
```
Python

[56]
```python
print('F1-score on Test data:', test_data_f1_score)
```
Python

```
F1-score on Test data: 0.9042553191489361
```

0 ⚠ 8 0 Spaces: 4 Cell 23 of 58 Go Live Spell

OBSERVATION

1. Data Overview:

- The dataset consists of credit card transaction data, including features such as transaction amount, time, and class labels indicating whether the transaction is fraudulent (1) or legitimate (0).
- Exploratory data analysis reveals the distribution of fraudulent and legitimate transactions, highlighting potential class imbalance issues.

2. Data Preprocessing:

- The data preprocessing steps include checking for missing values and handling them appropriately.
- There may be a need for further preprocessing techniques such as feature scaling or normalization to ensure uniformity across features and improve model performance.

3. Data Imbalance Handling:

- The code addresses class imbalance by performing random undersampling of the majority class (legitimate transactions) to create a balanced dataset.
- Alternative techniques such as oversampling or synthetic data generation could be explored to further address class imbalance and improve model robustness.

4. Model Training and Evaluation:

- The logistic regression model is trained on the preprocessed dataset to classify transactions as fraudulent or legitimate.
- Evaluation metrics such as accuracy, precision, recall, and F1-score are computed to assess the model's performance on both the training and testing datasets.
- It's important to note that while accuracy provides an overall measure of model performance, metrics like precision and recall are crucial for evaluating the model's ability to correctly identify fraudulent transactions while minimizing false positives.

5. Model Performance:

- The model achieves a certain level of accuracy, precision, recall, and F1-score on both the training and testing datasets.
- The performance metrics indicate how well the model is able to distinguish between fraudulent and legitimate transactions. High precision indicates a low false positive rate, while high recall indicates a low false negative rate.

FUTURE SCOPE

1. Feature Engineering:
- Explore additional features or derive new features from the existing dataset to capture more nuanced patterns of fraudulent transactions.
- Investigate the incorporation of external data sources such as customer behavior analytics or transaction metadata to enhance fraud detection capabilities.

2. Advanced Machine Learning Techniques:
- Experiment with advanced machine learning algorithms beyond logistic regression, such as ensemble methods (e.g., random forest, gradient boosting), deep learning models (e.g., neural networks), or anomaly detection algorithms.
- Evaluate the performance of these algorithms and compare them with the baseline logistic regression model to identify the most effective approach for credit card fraud detection.

3. Imbalanced Data Handling:
- Explore more sophisticated techniques for addressing class imbalance, such as ensemble methods (e.g., SMOTE-ENN, SMOTE-Tomek), cost-sensitive learning, or adversarial training.
- Investigate the impact of different sampling strategies on model performance and robustness against various types of fraud.

4. Model Interpretability:

- Enhance the interpretability of the fraud detection model to provide insights into the factors contributing to fraudulent transactions.
- Utilize techniques such as feature importance analysis, SHAP (SHapley Additive exPlanations) values, or LIME (Local Interpretable Model-agnostic Explanations) to explain model predictions and increase transparency.

5. Real-Time Detection and Deployment:

- Develop mechanisms for real-time fraud detection and prevention to enable timely intervention and mitigation of fraudulent activities.
- Implement the trained model into production systems and integrate it with existing fraud detection pipelines for seamless deployment in financial institutions' operational workflows.

By exploring these future directions, the credit card fraud detection project can evolve into a more sophisticated and robust system capable of effectively combating fraudulent activities in the financial domain. Continued research and innovation in this area are essential to stay ahead of evolving fraud tactics and ensure the security and integrity of financial transactions.

CONCLUSION

By analyzing a comprehensive dataset containing information about credit card transactions, the project demonstrates the feasibility of building a classification model capable of distinguishing between legitimate and fraudulent transactions. The logistic regression model serves as a starting point, providing a baseline for evaluating model performance and effectiveness in fraud detection.

The project highlights several key observations and areas for improvement, including data preprocessing, handling imbalanced data, exploring advanced machine learning algorithms, and enhancing model interpretability. These insights lay the groundwork for future research and development efforts aimed at enhancing the robustness and accuracy of fraud detection systems.

Overall, the credit card fraud detection project underscores the importance of leveraging data-driven approaches to address the growing threat of fraudulent activities in financial transactions. By harnessing the power of predictive modeling and analytics, stakeholders can work towards building more resilient and effective fraud detection systems to protect consumers, merchants, and financial institutions from the impacts of fraudulent transactions.

References

1. Raj, S. Benson Edwin, and A. Annie Portia. "Analysis on credit card fraud detection methods." 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET). IEEE, 2011.
2. Chaudhary, Khyati, Jyoti Yadav, and Bhawna Mallick. "A review of fraud detection techniques: Credit card." International Journal of Computer Applications 45.1 (2012): 39-44.
3. Sushmito, and Douglas L. Reilly. "Credit card fraud detection with a neural-network." System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on. Vol. 3. IEEE, 1994.
4. Awoyemi, John O., Adebayo O. Adetunmbi, and Samuel A. Oluwadare. "Credit card fraud detection using machine learning techniques: A comparative analysis." 2017 international conference on computing networking and informatics (ICCNI). IEEE, 2017.
5. Dal Pozzolo, Andrea, et al. "Learned lessons in credit card fraud detection from a practitioner perspective." Expert systems with applications 41.10 (2014): 4915-4928.
6. Srivastava, Abhinav, et al. "Credit card fraud detection using hidden Markov model." IEEE Transactions on dependable and secure computing 5.1 (2008): 37-48.

7. Chan, Philip K., et al. "Distributed data mining in credit card fraud detection." IEEE Intelligent Systems and Their Applications 14.6 (1999): 67-74.
8. Varmedja, Dejan, et al. "Credit card fraud detection-machine learning methods." 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH). IEEE, 2019.
9. Patidar, Raghavendra, and Lokesh Sharma. "Credit card fraud detection using neural network." International Journal of Soft Computing and Engineering (IJSCE) 1.32-38 (2011).
10. Dal Pozzolo, Andrea, et al. "Credit card fraud detection: a realistic modeling and a novel learning strategy." IEEE transactions on neural networks and learning systems 29.8 (2017): 3784-3797.
11. Dornadula, Vaishnavi Nath, and Sa Geetha. "Credit card fraud detection using machine learning algorithms." Procedia computer science 165 (2019): 631-641.
12. Xuan, Shiyang, et al. "Random forest for credit card fraud detection." 2018 IEEE 15th international conference on networking, sensing and control (ICNSC). IEEE, 2018.
13. Bahnsen, Alejandro Correa, et al. "Feature engineering strategies for credit card fraud detection." Expert Systems with Applications 51 (2016): 134-142.
14. Fu, Kang, et al. "Credit card fraud detection using convolutional neural networks." Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III 23. Springer International Publishing, 2016.

15. Sailusha, Ruttala, et al. "Credit card fraud detection using machine learning." 2020 4th international conference on intelligent computing and control systems (ICICCS). IEEE, 2020.