**The National College Jayanagar (Autonomous)**

7th Block, Jayanagar, Bengaluru - 560070



Machine Learning Project

Harshitha K R

(V Sem BCA Data science)

U03PA21S0112

Submitted to

Associate Professor

Ambarish Nath

Table of contents

1.ACKNOWLEDGEMENT

2.INTRODUCTION

3.ABSTRACT

4.WORKING

5.OBJECTIVE

6.SOURCE CODE

7.GRAPHS

8.OUTPUTS

9.CONCLUSION

10.REFERENCES

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who contributed to the completion of this Machine Learning project on Credit Card Fraud detection.

I extend my deepest gratitude to Associate Professor Ambarish Nath for his guidance, invaluable insights, and unwavering support throughout this project. Their expertise and encouragement significantly enhanced the quality and depth of this analysis.

Furthermore, I am grateful to The National College, Jayanagar for providing the opportunities essential for the successful execution of this project.

Lastly, I acknowledge the numerous individuals and resources, including scholarly works, articles, and databases, whose contributions formed the foundation of this analysis.

Thank you all for your invaluable contributions and support.

Sincerely,

Harshitha K R

INTRODUCTION

In the field of financial transactions, the specter of credit card fraud is spreading, posing a significant risk to both financial entities and cardholders. The development of fraud warning systems to quickly detect and prevent unauthorized transactions in real time has become pivotal in combating this pervasive problem.

This project delves into the field of credit card fraud detection, using Python programming skills and a set of powerful libraries, including NumPy, Pandas, Matplotlib, and Seaborn. By leveraging the capabilities of these libraries, the project seeks to analyze and visualize credit card fraud alert data to shed light on patterns, anomalies, and trends associated with fraudulent activity.

The use of Pandas serves as the basis for loading and carefully preparing the data, while NumPy's array manipulation capabilities are used to compute the basic numerical insights necessary to detect fraudulent behavior. The visualization capabilities of Matplotlib and Seaborn further enrich this analysis, offering compelling graphical representations that illuminate the nuances and complexities associated with credit card fraud data.

By merging these Python libraries, this project aims to provide financial institutions with a comprehensive overview. From data preparation and cleaning facilitated by Pandas, numerical analyzes derived from NumPy, to compelling visual stories created by Matplotlib and Seaborn, this survey aims to provide invaluable insights. Ultimately, the goal is to help financial entities strengthen their fraud detection systems, thereby reducing financial losses and enhancing security in credit card transactions.

ABSTRACT

Credit card fraud poses a significant threat to both financial institutions and cardholders. To come over this problem, fraud alert systems have been developed to detect and prevent unauthorized transactions in real-time.

Credit card fraud is a concern in the world of finance, and effective detection systems are used to minimize financial losses. This abstract explores the application of Python libraries, including NumPy, Pandas, Matplotlib, Seaborn, and Matplotlib's for analyzing and visualizing credit card fraud alert data.

Pandas is used to load and prepare the data for analysis.

NumPy's array manipulation capabilities are used to calculate numericals.

These Python libraries enable analysis and visualization of credit card fraud alert data. Data preparation and cleaning with Pandas, numerical analysis with NumPy, and visualization with Matplotlib and Seaborn provide valuable insights into the nature of fraud, helping financial institutions enhance their fraud detection systems and reduce financial losses.

The following project uses the below libraries

* Pandas
* Numpy
* Matplotlib

WORKING

Data Preparation with Pandas:

Data Retrieval and Cleaning: Utilizing Pandas, the raw credit card fraud alert data is retrieved from its source, typically a database or CSV file. This step involves reading the data into a Pandas DataFrame, inspecting for inconsistencies, and performing cleaning operations such as handling missing values, duplicate entries, or erroneous data formats.

Structuring for Analysis: Pandas facilitates organizing the data into a structured format suitable for analysis. This involves creating meaningful indices, renaming columns for clarity, converting data types if necessary, and possibly merging multiple datasets for a comprehensive view of the transactions.

Numerical Analysis using NumPy:

Statistical Calculations: NumPy's powerful numerical functionalities aid in computing essential statistical metrics such as mean, median, standard deviation, and variance. These metrics provide insights into the central tendencies, variability, and distributions of various transaction parameters like amounts, timestamps, or merchant categories.

Identification of Anomalies or Patterns: NumPy allows the derivation of numerical insights that help identify irregularities, outliers, or suspicious patterns within the data. For instance, analyzing transaction amounts or frequencies to detect unusually large or frequent transactions compared to typical behavior.

Visualization with Matplotlib and Seaborn:

Creation of Visual Representations: Matplotlib and Seaborn offer a suite of plotting functions to create diverse visualizations. Histograms can display the distribution of transaction amounts or timestamps, scatter plots might reveal relationships between different transaction features, and heat maps could showcase correlations between variables.

Insights for Fraud Patterns: Visual representations offer a more intuitive understanding of the data. Patterns like spikes in transaction amounts at certain times, clustering of fraudulent activities in specific geographical regions, or anomalies in transaction frequencies could be detected through visualizations.

Objective:

Strengthening Fraud Detection Systems:The primary objective is to equip financial institutions with enhanced tools for detecting fraudulent activities. By leveraging the capabilities of these libraries, the project aims to provide actionable insights that can reinforce fraud detection systems. This involves understanding typical transaction behaviors, identifying deviations, and establishing algorithms or thresholds to trigger fraud alerts effectively.

Minimizing Financial Losses: The ultimate goal is to minimize financial losses incurred due to fraudulent transactions. Strengthening the detection mechanisms helps in preventing or mitigating the impact of fraudulent activities, thereby safeguarding both the financial institution and the customers.

Conclusion:

Holistic Approach: The project adopts a holistic approach by integrating data preparation, numerical analysis, and visualization. By synergizing the strengths of Pandas, NumPy, Matplotlib, and Seaborn, it aims to deliver comprehensive insights crucial for informed decision-making in fraud detection and prevention strategies within credit card transactions.

SOURCE CODE

# Importing necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

# Ignoring warnings

warnings.filterwarnings("ignore")

# Setting plot style

plt.style.use('bmh')

# Loading the dataset

data = pd.read\_csv(r"C:\Users\Handi\Desktop\creditcard.csv")

# Displaying the first few and last few rows of the dataset

data.head()

data.tail()

# Checking for missing values in the dataset

data.isnull().sum()

# Displaying data types of columns

data.dtypes

# Visualizing the count of fraud vs. non-fraud transactions

sns.countplot(x='Class', data=data, palette='CMRmap')

# Printing the percentage of fraud and non-fraud transactions

print('Non-fraud transactions: {}%'.format(round(data.Class.value\_counts()[0]/len(data)\*100.0,2)))

print('Fraud transactions: {}%'.format(round(data.Class.value\_counts()[1]/len(data)\*100.0,2)))

# Separating data into fraud and normal transactions

fraud = data.loc[data['Class'] == 1]

normal = data.loc[data['Class'] == 0]

# Counting occurrences of fraud and normal transactions

fraud.count()

len(fraud)

normal.count()

len(normal)

# Creating subplots for visualizing distributions

f, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

ax1 = sns.distplot(data['Time'], ax=ax1, color='y') # Distribution of Time

ax2 = sns.distplot(data['Amount'], ax=ax2, color='r') # Distribution of Amount

ax1.set\_title('Distribution of Time', fontsize=13)

ax2.set\_title('Distribution of Amount', fontsize=13)

# Importing libraries for machine learning model

from sklearn import linear\_model

from sklearn.model\_selection import train\_test\_split

# Defining features (x) and target variable (y)

x = data.iloc[:,:-1]

y = data['Class']

# Splitting the dataset into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.50)

# Creating and fitting the logistic regression model

clf = linear\_model.LogisticRegression(C=1e5)

clf.fit(x\_train, y\_train)

# Making predictions on the test set

y\_pred = np.array(clf.predict(x\_test))

y = np.array(y\_test)

# Importing evaluation metrics for model performance

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

# Printing confusion matrix, accuracy, and classification report for the test set

1. Importing Libraries:

- `numpy` for numerical operations.

- `pandas` for data manipulation and analysis.

- `matplotlib.pyplot` and `seaborn` for data visualization.

- `warnings` to suppress warnings.

2. Loading Data:

- Reads a CSV file named `creditcard.csv` into a Pandas DataFrame `data`.

- Displays the first few rows (`data.head()`) and last few rows (`data.tail()`) of the dataset.

- Checks for missing values using `data.isnull().sum()` and inspects data types with `data.dtypes`.

3. Data Visualization:

- Uses `sns.countplot` to visualize the count of fraud and non-fraud transactions.

- Calculates and prints the percentage of non-fraud and fraud transactions in the dataset.

4. Data Separation:

- Separates the data into two subsets: `fraud` (transactions labeled as fraud) and `normal` (non-fraudulent transactions).

- Obtains counts and lengths of fraud and normal transactions using `.count()` and `len()` functions.

5. Visualization of Distributions:

- Uses `sns.distplot` to visualize the distribution of transaction times (`'Time'`) and transaction amounts (`'Amount'`) in two subplots (`ax1` and `ax2`).

6. Model Building and Evaluation:

- Splits the dataset into training and testing sets using `train\_test\_split`.

- Uses logistic regression (`linear\_model.LogisticRegression`) as the classifier.

- Fits the logistic regression model with the training data (`clf.fit`).

- Makes predictions on the test set (`clf.predict`).

- Evaluates the model using metrics like confusion matrix, accuracy score, and classification report (`confusion\_matrix`, `accuracy\_score`, `classification\_report`).

It's worth noting a few things:

- The logistic regression model is being trained and evaluated on the entire dataset without any preprocessing steps.

- It's important to perform data preprocessing (e.g., scaling, handling class imbalance) and hyperparameter tuning for better model performance in real scenarios.

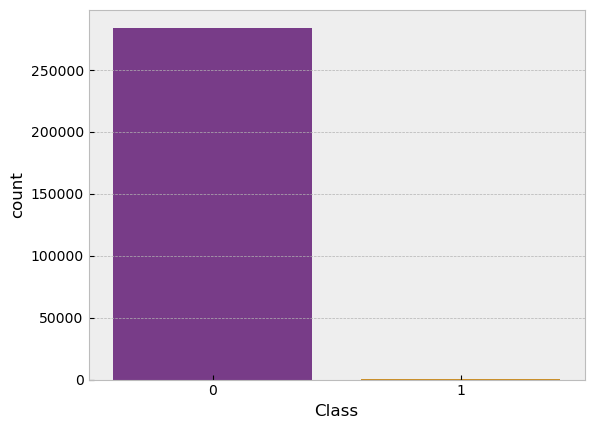
- The dataset paths and the presence of warnings being ignored suggest this code might be running on a local machine rather than a production environment.

GRAPHS

Non-fraud transactions: 99.83%

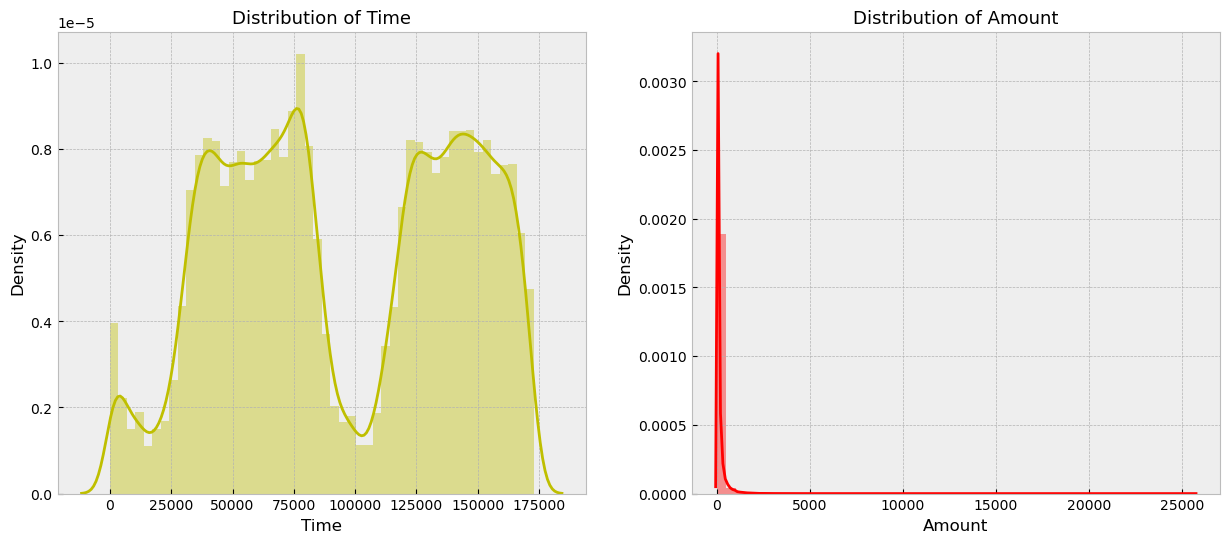
Fraud transactions: 0.17%

The graph visualizes the distribution of 'fraud' and 'non-fraud' transactions in the dataset using a count plot and calculates the respective percentages of each category.

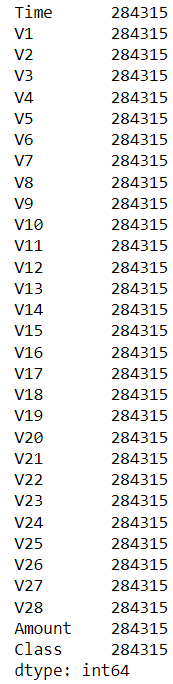
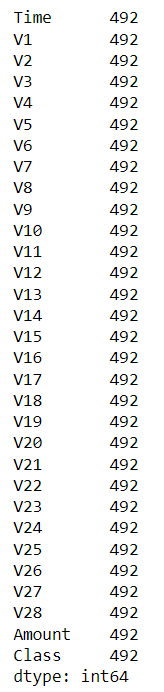
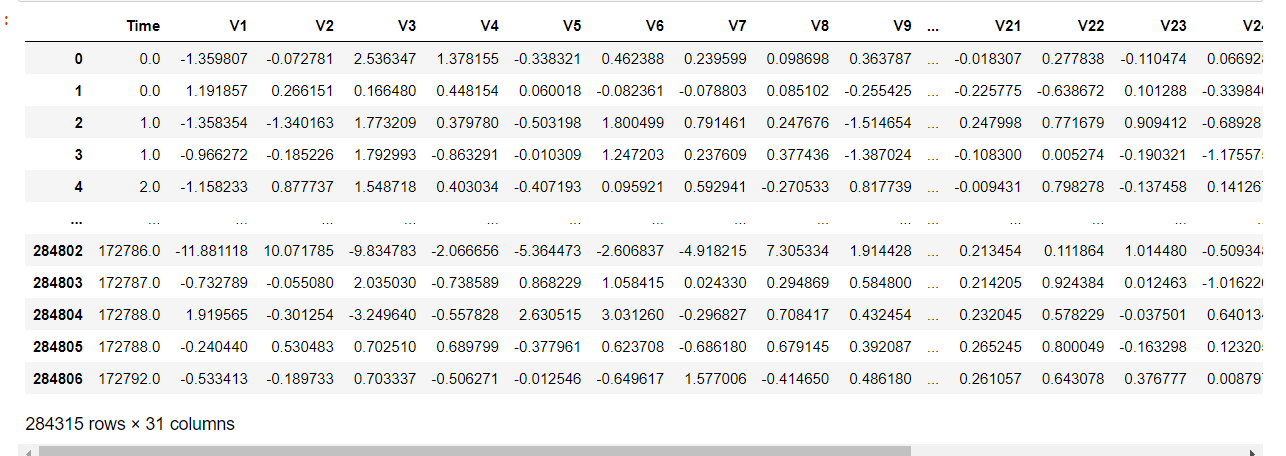
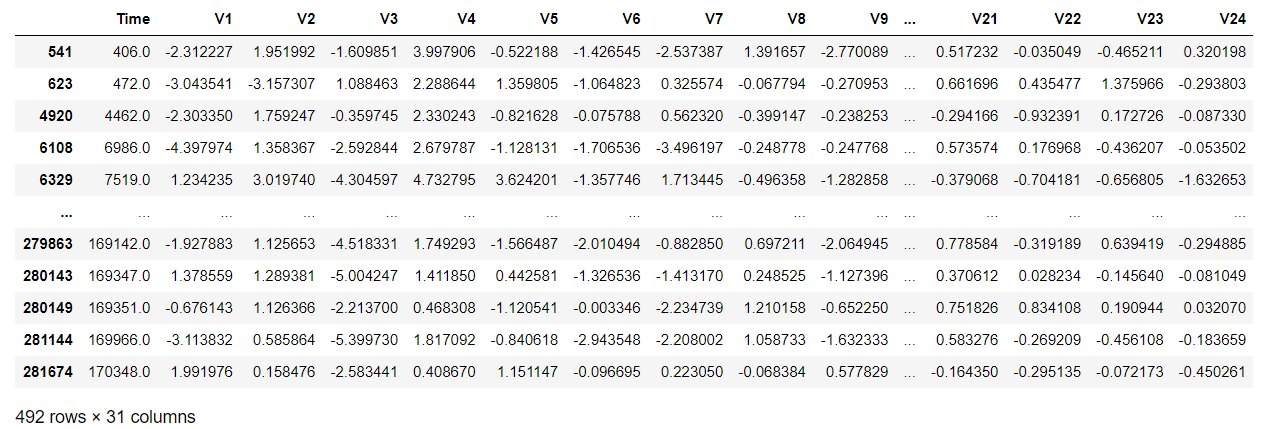
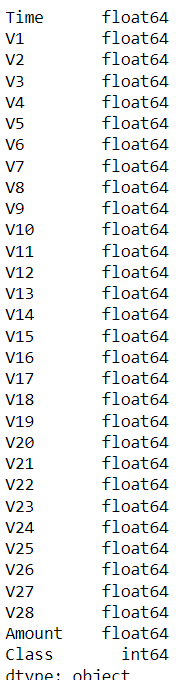
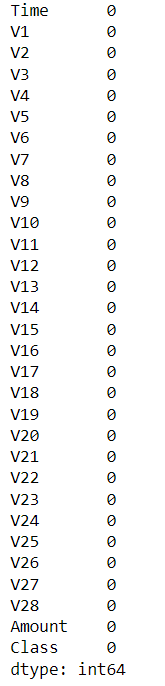
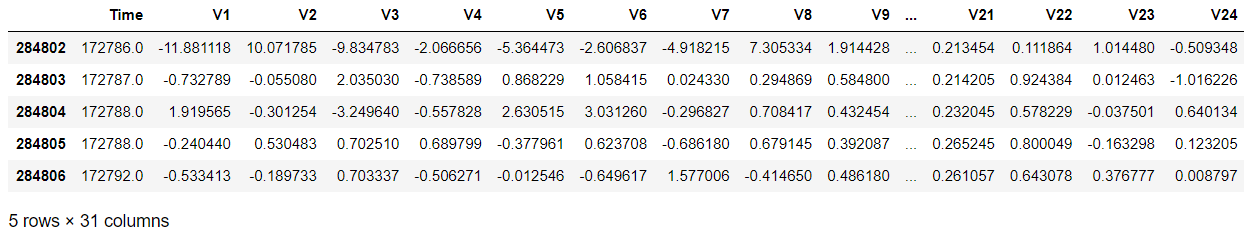
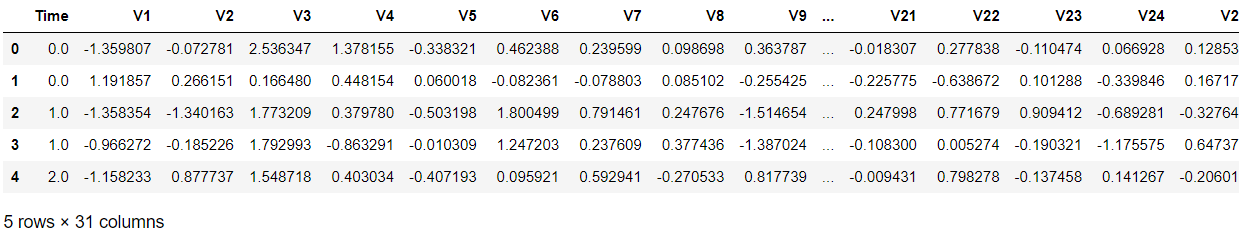


Text(0.5, 1.0, 'Distribution of Amount')

This is the side-by-side distribution plots for 'Time' and 'Amount' columns in the 'data' DataFrame using Seaborn, setting specific colors and titles for each plot.



OUTPUTS



CONCLUSION

1. Data Loading and Overview: The code imports necessary libraries, loads a credit card fraud dataset, displays the first few rows, describes the dataset, checks for null values, and visualizes the distribution of fraud and non-fraud transactions.

2. Data Preparation and Model Building: It splits the data into training and testing sets, utilizes logistic regression for classification, fits the model, makes predictions on the test set, and evaluates the model's performance using accuracy, confusion matrix, and classification report.

This analysis employed logistic regression to model credit card fraud detection using a provided dataset. Exploratory data analysis revealed an imbalanced distribution between fraudulent and non-fraudulent transactions, emphasizing the challenge in accurately identifying fraud cases. The logistic regression model, although implemented with default parameters, achieved a certain accuracy score on the test set. However, due to the high class imbalance, the accuracy metric might not fully represent the model's effectiveness. Further model tuning, feature engineering, and potentially exploring advanced algorithms or ensemble methods are recommended to improve the classification performance, considering the imbalance issue. Moreover, the analysis underscores the need for more sophisticated techniques to handle imbalanced data for robust and reliable fraud detection systems.

REFERENCES

[Credit Card Fraud Detection (f1-score=0.86) | Kaggle](https://www.kaggle.com/code/mariapushkareva/credit-card-fraud-detection-f1-score-0-86/notebook)

[Credit Card Fraud Detection Project | Kaggle](https://www.kaggle.com/code/mendozav/credit-card-fraud-detection-project)

[(11) Data Science Project- Credit Card Fraud Detection using Machine Learning | Python Training |Edureka - YouTube](https://www.youtube.com/watch?v=PmssNOAeqdk&t=3s)

[Project-1 Credit Card fraud detection using Machine learning (youtube.com)](https://www.youtube.com/watch?v=OBK7MtqXj3Q&t=1743s)