FRAUD DETECTION SYSTEM USING LINEAR REGRESSION

By: *DHANUSH TADISETTI*

Gmail: dhanushtadisetti@gmail.com

To Project Mentor: *N. MUVENDIRAN SIR*.

INTRODUCTION:

Credit card fraud detection is the process of identifying unauthorized or fraudulent transactions made using a credit card. The goal is to differentiate between legitimate transactions and suspicious activities to protect both the cardholder and the financial institution. Fraud detection systems typically use advanced analytics, machine learning algorithms, and real-time monitoring to identify anomalies or unusual spending patterns that may indicate fraud.

PROJECT OVERVIEW & DATA SET EXPLANATION:

This project focuses on detecting fraudulent credit card transactions through data preprocessing, comprehensive analysis, visualization, and machine learning models. By exploring and modelling transaction data, the project aims to identify patterns and insights to improve fraud detection accuracy.

DATA SET:

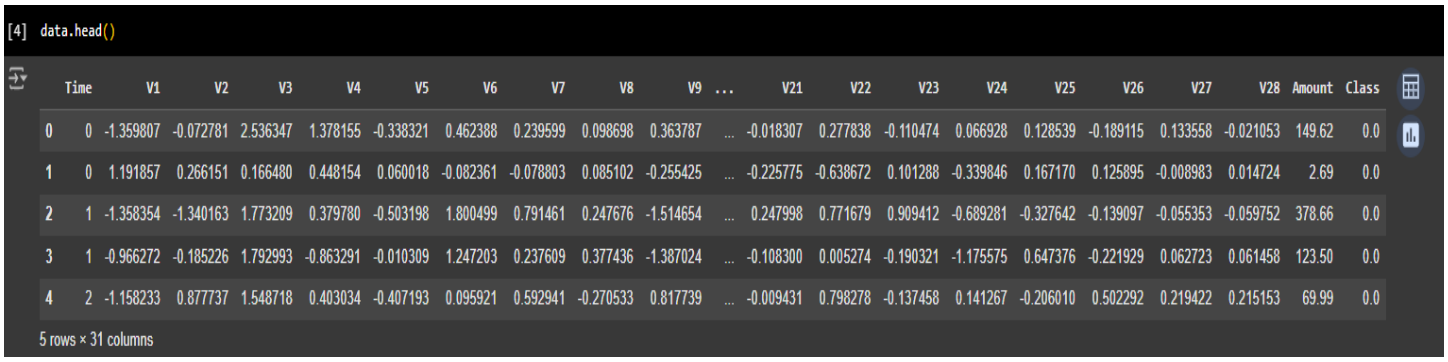
The dataset contains transactions made by credit cards in September 2013 by European cardholders. This data set presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

-The target variable Class indicates transaction type (0: Legitimate, 1: Fraudulent).

1. *DATA EXPLORATION:*

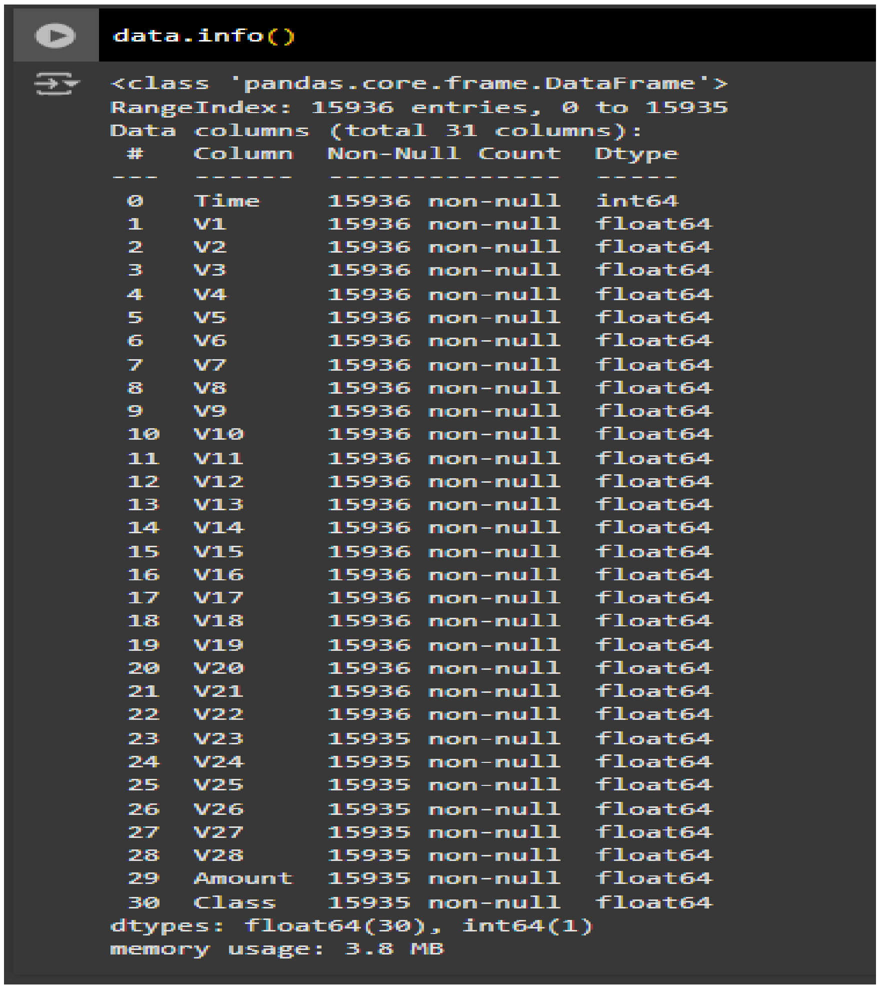
data=pd.read_csv('/content/creditcard.csv'): Reads a CSV file into a Pandas DataFrame.

data.head(): Displays the first 5 rows of the DataFrame.



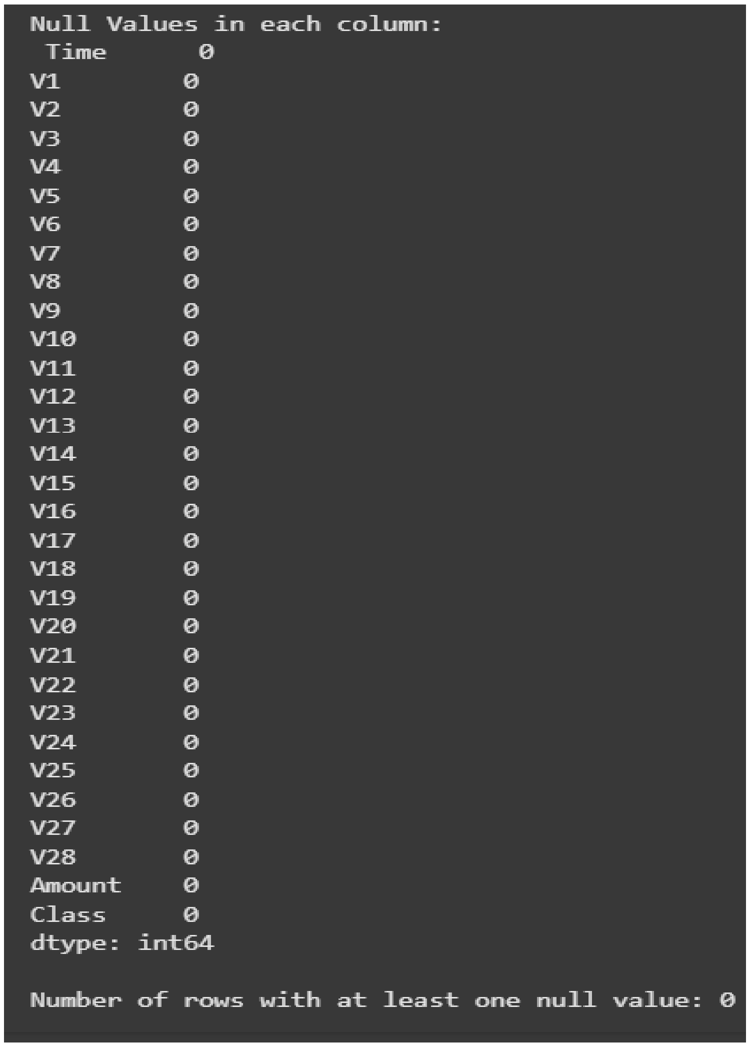
```
[4] data.head()

   Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22       V23       V24       V25       V26       V27       V28  Amount  Class
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  149.62    0.0
1     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    2.69    0.0
2     1 -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  378.66    0.0
3     1 -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  123.50    0.0
4     2 -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   69.99    0.0

5 rows × 31 columns
```

data.info(): Provides information about the dataset, such as column names, non-null counts, and data types.



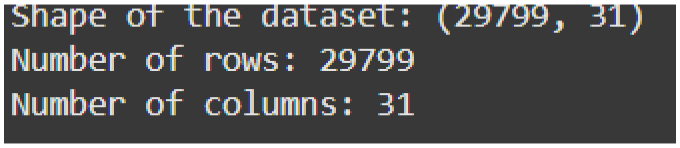
```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15936 entries, 0 to 15935
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    15936 non-null  int64  
 1   V1      15936 non-null  float64
 2   V2      15936 non-null  float64
 3   V3      15936 non-null  float64
 4   V4      15936 non-null  float64
 5   V5      15936 non-null  float64
 6   V6      15936 non-null  float64
 7   V7      15936 non-null  float64
 8   V8      15936 non-null  float64
 9   V9      15936 non-null  float64
 10  V10     15936 non-null  float64
 11  V11     15936 non-null  float64
 12  V12     15936 non-null  float64
 13  V13     15936 non-null  float64
 14  V14     15936 non-null  float64
 15  V15     15936 non-null  float64
 16  V16     15936 non-null  float64
 17  V17     15936 non-null  float64
 18  V18     15936 non-null  float64
 19  V19     15936 non-null  float64
 20  V20     15936 non-null  float64
 21  V21     15936 non-null  float64
 22  V22     15936 non-null  float64
 23  V23     15935 non-null  float64
 24  V24     15935 non-null  float64
 25  V25     15935 non-null  float64
 26  V26     15935 non-null  float64
 27  V27     15935 non-null  float64
 28  V28     15935 non-null  float64
 29  Amount  15935 non-null  float64
 30  Class   15935 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 3.8 MB
```

data.isnull().sum() : Checks the Null-Values from the data set.



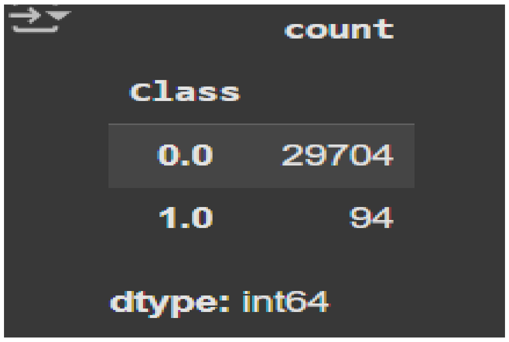
```
Null Values in each column:
 Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64

Number of rows with at least one null value: 0
```

Shape & No.of rows and columns:



```
Shape of the dataset: (29799, 31)
Number of rows: 29799
Number of columns: 31
```

Marking of Fraud(1) and Legitimate(0) classes:

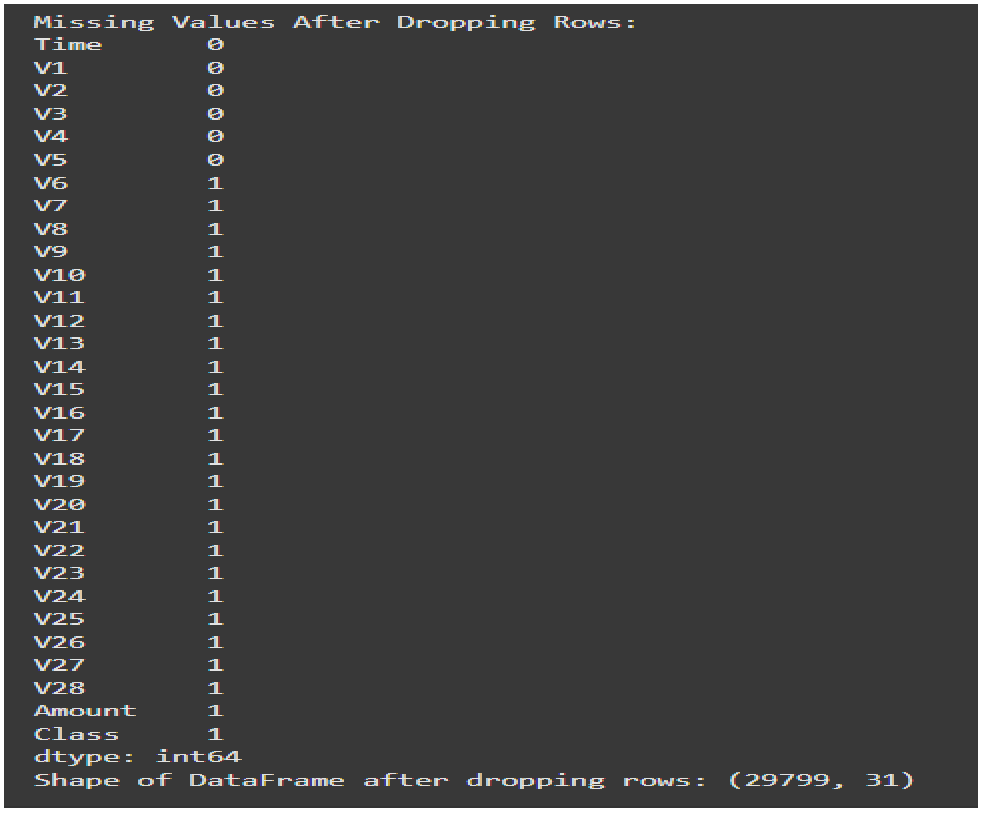


```
        count
Class
  0.0   29704
  1.0      94

dtype: int64
```

Dropping Null Rows:

df = data.dropna()

print(data.isnull().sum())

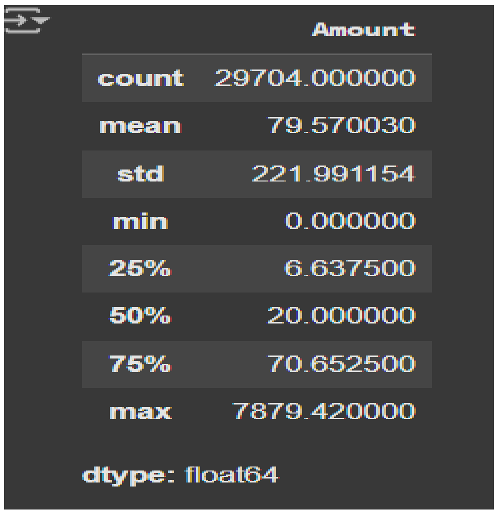


```
Missing Values After Dropping Rows:
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        1
V7        1
V8        1
V9        1
V10       1
V11       1
V12       1
V13       1
V14       1
V15       1
V16       1
V17       1
V18       1
V19       1
V20       1
V21       1
V22       1
V23       1
V24       1
V25       1
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
Shape of DataFrame after dropping rows: (29799, 31)
```

data.describe():

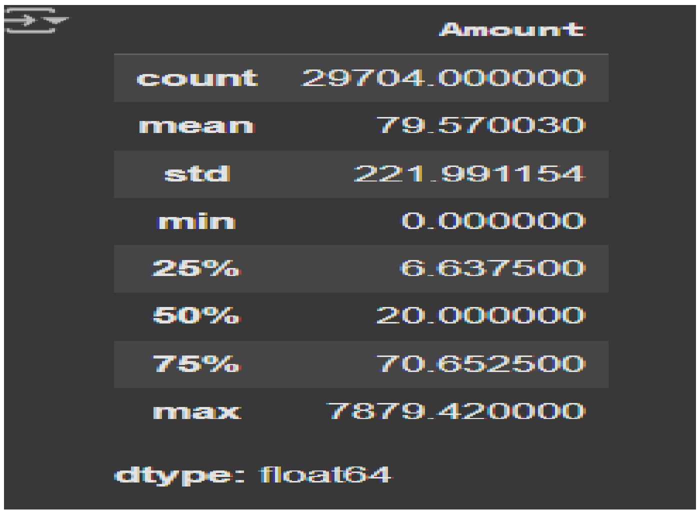
Displays a statistical summary of numerical columns, including mean, median, min, max etc.

LEGITIMATE CLASSES **FRAUD CLASSES**



	Amount
count	29704.000000
mean	79.570030
std	221.991154
min	0.000000
25%	6.637500
50%	20.000000
75%	70.652500
max	7879.420000

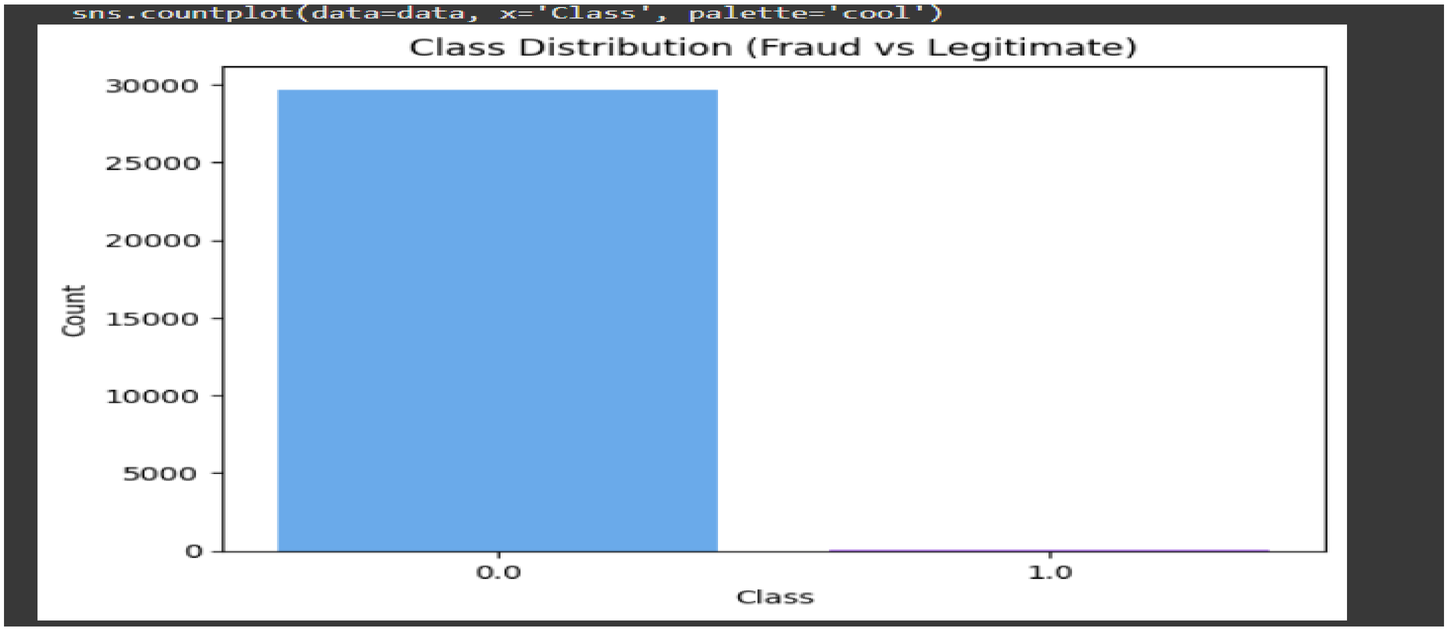
dtype: float64



	Amount
count	29704.000000
mean	79.570030
std	221.991154
min	0.000000
25%	6.637500
50%	20.000000
75%	70.652500
max	7879.420000

dtype: float64

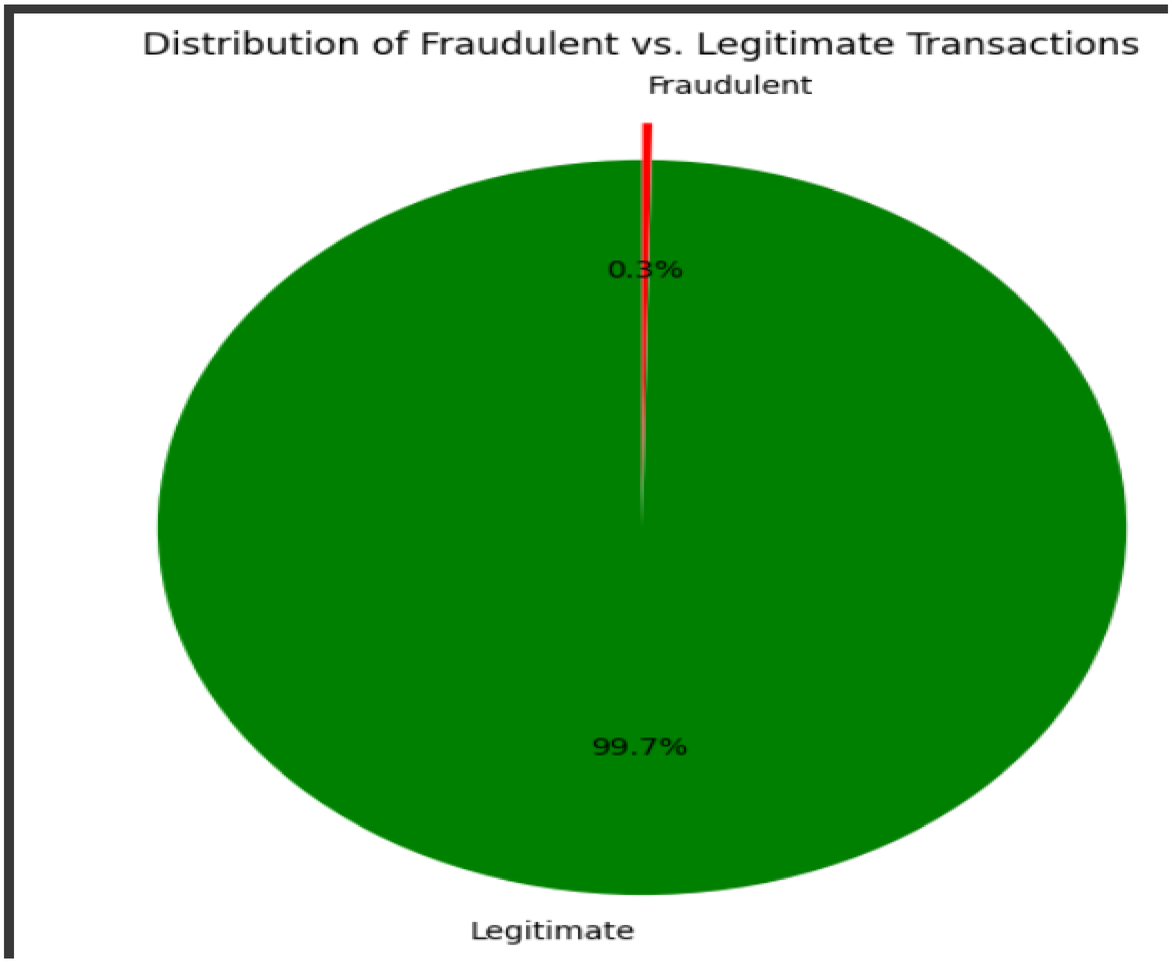
CLASS DISTRIBUTION:



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

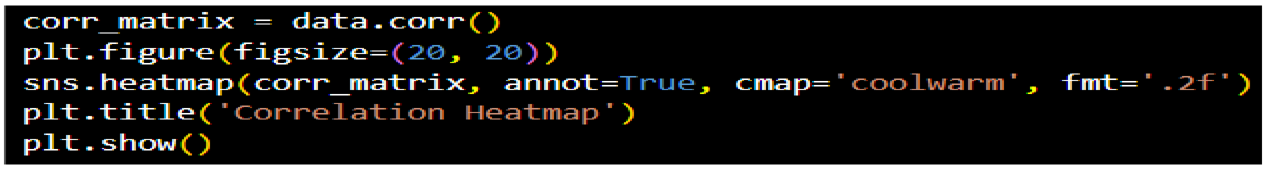
From the above Bar-Chart, We can classify Legitimate Users are nearly 29,000+(i.e upto 29,704+) and Fraud users are upto 94.

PIE CHART DISTRIBUTION VISUALSATION:

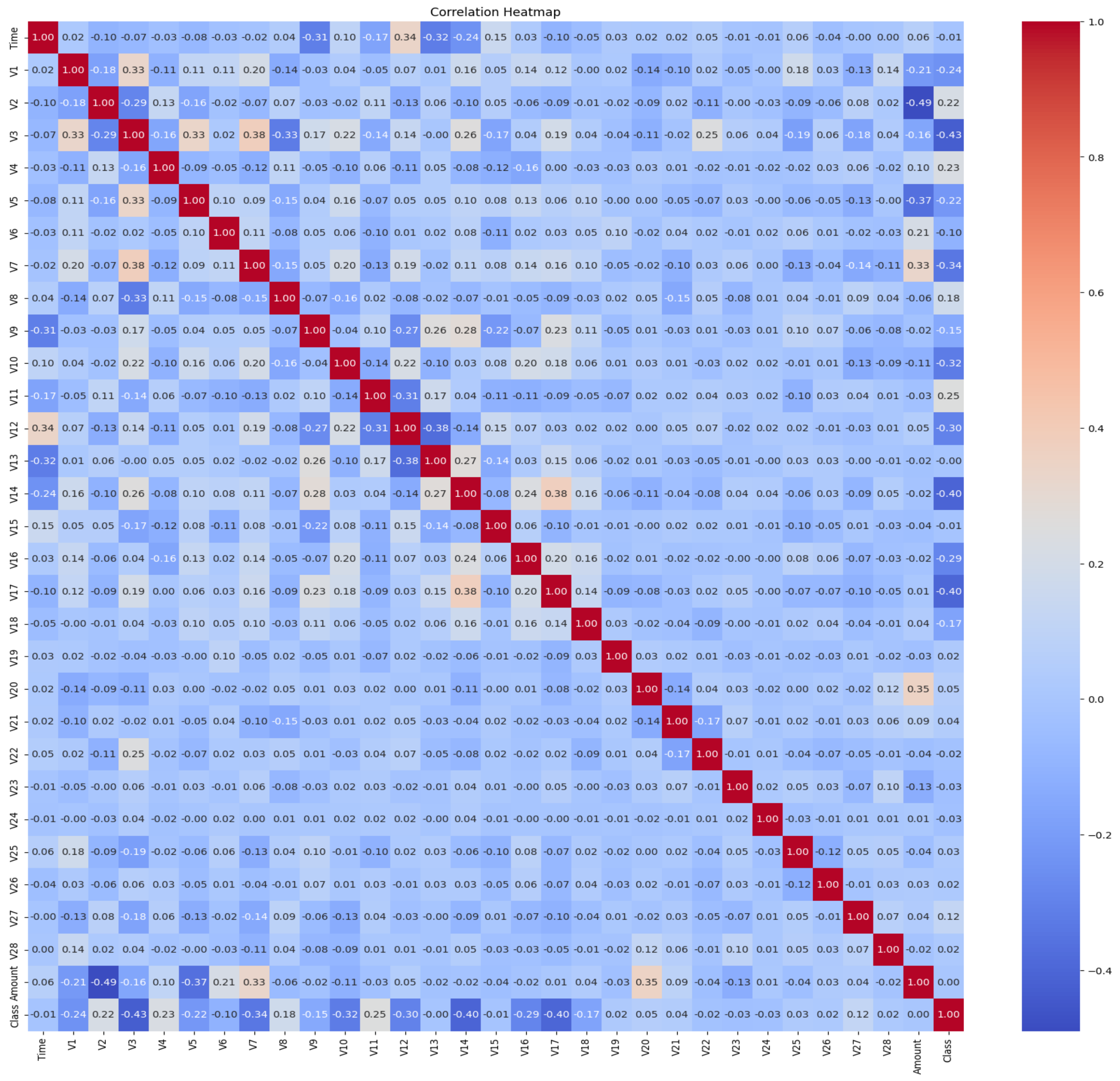


CO-RELATION HEATMAP:

This code calculates the correlations between numerical features in a dataset, visualizes those correlations using a heatmap with annotations and a color scheme, and then displays the resulting plot. This type of visualization is very useful for quickly identifying strong relationships between different features in a dataset.



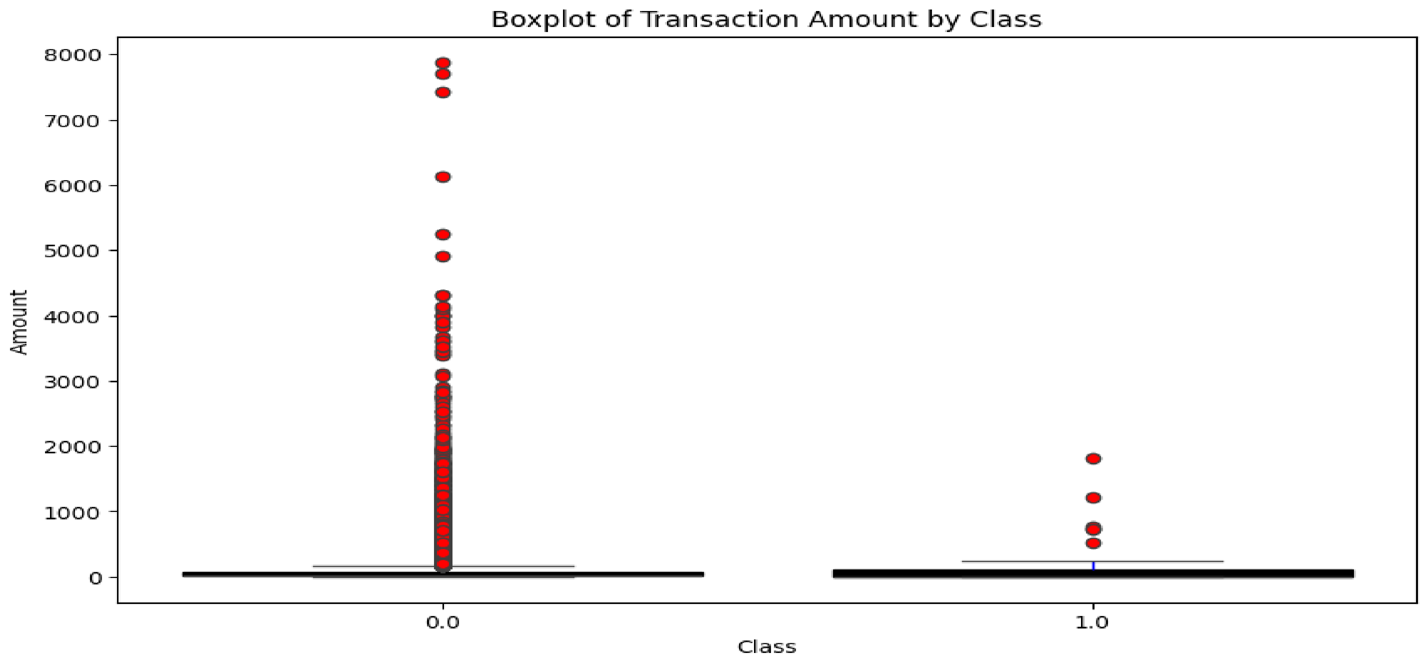
```
corr_matrix = data.corr()
plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
Time	1.00	0.02	-0.10	-0.07	-0.03	-0.08	-0.03	-0.02	0.04	-0.31	0.10	-0.17	0.34	-0.32	-0.24	0.15	0.03	-0.10	-0.05	0.03	0.02	0.02	0.05	-0.01	-0.01	0.06	-0.04	-0.00	0.00	0.06	-0.01
V1	0.02	1.00	-0.18	0.33	-0.11	0.11	0.11	0.20	-0.14	-0.03	0.04	-0.05	0.07	0.01	0.16	0.05	0.14	0.12	-0.00	0.02	-0.14	-0.10	0.02	-0.05	-0.00	0.18	0.03	-0.13	0.14	-0.21	-0.24
V2	-0.10	-0.18	1.00	-0.29	0.13	-0.16	-0.02	-0.07	0.07	-0.03	-0.02	0.11	-0.13	0.06	-0.10	0.05	-0.06	-0.09	-0.01	-0.02	-0.09	0.02	-0.11	-0.00	-0.03	-0.09	-0.06	0.08	0.02	-0.49	0.22
V3	-0.07	0.33	-0.29	1.00	-0.16	0.33	0.02	0.38	-0.33	0.17	0.22	-0.14	0.14	-0.00	0.26	-0.17	0.04	0.19	0.04	-0.04	-0.11	-0.02	0.25	0.06	0.04	-0.19	0.06	-0.18	0.04	-0.16	-0.43
V4	-0.03	-0.11	0.13	-0.16	1.00	-0.09	-0.05	-0.12	0.11	-0.05	-0.10	0.06	-0.11	0.05	-0.08	-0.12	-0.16	0.00	-0.03	-0.03	0.03	0.01	-0.02	-0.01	-0.02	-0.02	0.03	0.06	-0.02	0.10	0.23
V5	-0.08	0.11	-0.16	0.33	-0.09	1.00	0.10	0.09	-0.15	0.04	0.16	-0.07	0.05	0.05	0.10	0.08	0.13	0.06	0.10	-0.00	0.00	-0.05	-0.07	0.03	-0.00	-0.06	-0.05	-0.13	-0.00	-0.37	-0.22
V6	-0.03	0.11	-0.02	0.02	-0.05	0.10	1.00	0.11	-0.08	0.05	0.06	-0.10	0.01	0.02	0.08	-0.11	0.02	0.03	0.05	0.10	-0.02	0.04	0.02	-0.01	0.02	0.06	0.01	-0.02	-0.03	0.21	-0.10
V7	-0.02	0.20	-0.07	0.38	-0.12	0.09	0.11	1.00	-0.15	0.05	0.20	-0.13	0.19	-0.02	0.11	0.08	0.14	0.16	0.10	-0.05	-0.02	-0.10	0.03	0.06	0.00	-0.13	-0.04	-0.14	-0.11	0.33	-0.34
V8	0.04	-0.14	0.07	-0.33	0.11	-0.15	-0.08	-0.15	1.00	-0.07	-0.16	0.02	-0.08	-0.02	-0.07	-0.01	-0.05	-0.09	-0.03	0.02	0.05	-0.15	0.05	-0.08	0.01	0.04	-0.01	0.09	0.04	-0.06	0.18
V9	-0.31	-0.03	-0.03	0.17	-0.05	0.04	0.05	0.05	-0.07	1.00	-0.04	0.10	-0.27	0.26	0.28	-0.22	-0.07	0.23	0.11	-0.05	0.01	-0.03	0.01	-0.03	0.01	0.10	0.07	-0.06	-0.08	-0.02	-0.15
V10	0.10	0.04	-0.02	0.22	-0.10	0.16	0.06	0.20	-0.16	-0.04	1.00	-0.14	0.22	-0.10	0.03	0.08	0.20	0.18	0.06	0.01	0.03	0.01	-0.03	0.02	0.02	-0.01	0.01	-0.13	-0.09	-0.11	-0.32
V11	-0.17	-0.05	0.11	-0.14	0.06	-0.07	-0.10	-0.13	0.02	0.10	-0.14	1.00	-0.31	0.17	0.04	-0.11	-0.11	-0.09	-0.05	-0.07	0.02	0.02	0.04	0.03	0.02	-0.10	0.03	0.04	0.01	-0.03	0.25
V12	0.34	0.07	-0.13	0.14	-0.11	0.05	0.01	0.19	-0.08	-0.27	0.22	-0.31	1.00	-0.38	-0.14	0.15	0.07	0.03	0.02	0.02	0.00	0.05	0.07	-0.02	0.02	0.02	-0.01	-0.03	0.01	0.05	-0.30
V13	-0.32	0.01	0.06	-0.00	0.05	0.05	0.02	-0.02	-0.02	0.26	-0.10	0.17	-0.38	1.00	0.27	-0.14	0.03	0.15	0.06	-0.02	0.01	-0.03	-0.05	-0.01	-0.00	0.03	0.03	-0.00	-0.01	-0.02	-0.00
V14	-0.24	0.16	-0.10	0.26	-0.08	0.10	0.08	0.11	-0.07	0.28	0.03	0.04	-0.14	0.27	1.00	-0.08	0.24	0.38	0.16	-0.06	-0.11	-0.04	-0.08	0.04	0.04	-0.06	0.03	-0.09	0.05	-0.02	-0.40
V15	0.15	0.05	0.05	-0.17	-0.12	0.08	-0.11	0.08	-0.01	-0.22	0.08	-0.11	0.15	-0.14	-0.08	1.00	0.06	-0.10	-0.01	-0.01	-0.00	0.02	0.02	0.01	-0.01	-0.10	-0.05	0.01	-0.03	-0.04	-0.01
V16	0.03	0.14	-0.06	0.04	-0.16	0.13	0.02	0.14	-0.05	-0.07	0.20	-0.11	0.07	0.03	0.24	0.06	1.00	0.20	0.16	-0.02	0.01	-0.02	-0.02	-0.00	-0.00	0.08	0.06	-0.07	-0.03	-0.02	-0.29
V17	-0.10	0.12	-0.09	0.19	0.00	0.06	0.03	0.16	-0.09	0.23	0.18	-0.09	0.03	0.15	0.38	-0.10	0.20	1.00	0.14	-0.09	-0.08	-0.03	0.02	0.05	-0.00	-0.07	-0.07	-0.10	-0.05	0.01	-0.40
V18	-0.05	-0.00	-0.01	0.04	-0.03	0.10	0.05	0.10	-0.03	0.11	0.06	-0.05	0.02	0.06	0.16	-0.01	0.16	0.14	1.00	0.03	-0.02	-0.04	-0.09	-0.00	-0.01	0.02	0.04	-0.04	-0.01	0.04	-0.17
V19	0.03	0.02	-0.02	-0.04	-0.03	-0.00	0.10	-0.05	0.02	-0.05	0.01	-0.07	0.02	-0.02	-0.06	-0.01	-0.02	-0.09	0.03	1.00	0.03	0.02	0.01	-0.03	-0.01	-0.02	-0.03	0.01	-0.02	-0.03	0.02
V20	0.02	-0.14	-0.09	-0.11	0.03	0.00	-0.02	-0.02	0.05	0.01	0.03	0.02	0.00	0.01	-0.11	-0.00	0.01	-0.08	-0.02	0.03	1.00	-0.14	0.04	0.03	-0.02	0.00	0.02	-0.02	0.12	0.35	0.05
V21	0.02	-0.10	0.02	-0.02	0.01	-0.05	0.04	-0.10	-0.15	-0.03	0.01	0.02	0.05	-0.03	-0.04	0.02	-0.02	-0.03	-0.04	0.02	-0.14	1.00	-0.17	0.07	-0.01	0.02	-0.01	0.03	0.06	0.09	0.04
V22	0.05	0.02	-0.11	0.25	-0.02	-0.07	0.02	0.03	0.05	0.01	-0.03	0.04	0.07	-0.05	-0.08	0.02	-0.02	0.02	-0.09	0.01	0.04	-0.17	1.00	-0.01	0.01	-0.04	-0.07	-0.05	-0.01	-0.04	-0.02
V23	-0.01	-0.05	-0.00	0.06	-0.01	0.03	-0.01	0.06	-0.08	-0.03	0.02	0.03	-0.02	-0.01	0.04	0.01	-0.00	0.05	-0.00	-0.03	0.03	0.07	-0.01	1.00	0.02	0.05	0.03	-0.07	0.10	-0.13	-0.03
V24	-0.01	-0.00	-0.03	0.04	-0.02	-0.00	0.02	0.00	0.01	0.01	0.02	0.02	0.02	-0.00	0.04	-0.01	-0.00	-0.00	-0.01	-0.01	-0.02	-0.01	0.01	0.02	1.00	-0.03	-0.01	0.01	0.01	0.01	-0.03
V25	0.06	0.18	-0.09	-0.19	-0.02	-0.06	0.06	-0.13	0.04	0.10	-0.01	-0.10	0.02	0.03	-0.06	-0.10	0.08	-0.07	0.02	-0.02	0.00	0.02	-0.04	0.05	-0.03	1.00	-0.12	0.05	0.05	-0.04	0.03
V26	-0.04	0.03	-0.06	0.06	0.03	-0.05	0.01	-0.04	-0.01	0.07	0.01	0.03	-0.01	0.03	0.03	-0.05	0.06	-0.07	0.04	-0.03	0.02	-0.01	-0.07	0.03	-0.01	-0.12	1.00	-0.01	0.03	0.03	0.02
V27	-0.00	-0.13	0.08	-0.18	0.06	-0.13	-0.02	-0.14	0.09	-0.06	-0.13	0.04	-0.03	-0.00	-0.09	0.01	-0.07	-0.10	-0.04	0.01	-0.02	0.03	-0.05	-0.07	0.01	0.05	-0.01	1.00	0.07	0.04	0.12
V28	0.00	0.14	0.02	0.04	-0.02	-0.00	-0.03	-0.11	0.04	-0.08	-0.09	0.01	0.01	-0.01	0.05	-0.03	-0.03	-0.05	-0.01	-0.02	0.12	0.06	-0.01	0.10	0.01	0.05	0.03	0.07	1.00	-0.02	0.02
Amount	0.06	-0.21	-0.49	-0.16	0.10	-0.37	0.21	0.33	-0.06	-0.02	-0.11	-0.03	0.05	-0.02	-0.02	-0.04	-0.02	0.01	0.04	-0.03	0.35	0.09	-0.04	-0.13	0.01	-0.04	0.03	0.04	-0.02	1.00	0.00
Class	-0.01	-0.24	0.22	-0.43	0.23	-0.22	-0.10	-0.34	0.18	-0.15	-0.32	0.25	-0.30	-0.00	-0.40	-0.01	-0.29	-0.40	-0.17	0.02	0.05	0.04	-0.02	-0.03	-0.03	0.03	0.02	0.12	0.02	0.00	1.00

BOX-PLOT:

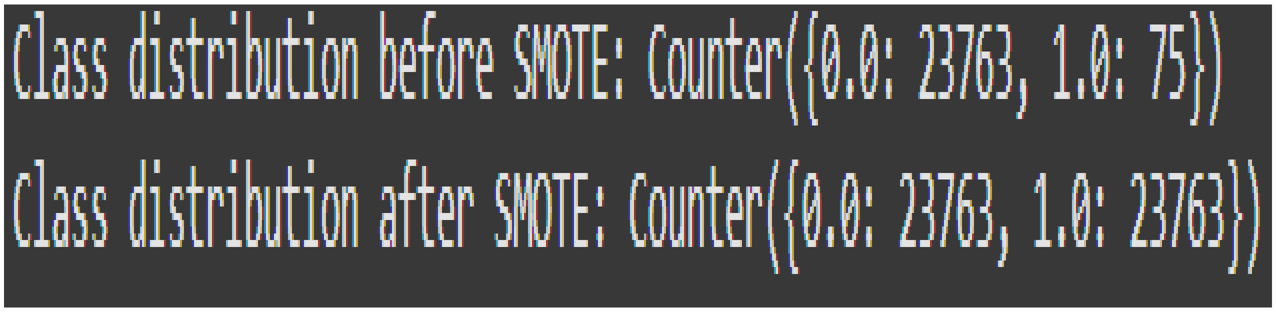
This code snippet generates a box plot that visually compares the distribution of transaction amounts ('Amount') for different transaction classes ('Class', likely fraudulent or legitimate). It uses color and style customizations to enhance the plot's readability.



SMOTE:

- In this code, SMOTE is used to address the issue of **class imbalance** in the credit card fraud dataset. Class imbalance occurs when one class (in this case, legitimate transactions) has significantly more instances than the other class (fraudulent transactions). This can lead to machine learning models being biased towards the majority class and performing poorly on the minority class, which is often the class of interest.
- SMOTE is applied to the training data (X_train, y_train) to **oversample** the minority class (fraudulent transactions). This means that it creates synthetic samples of the minority class to balance the class distribution, making the dataset more suitable for training a machine learning model.
- SMOTE works by generating synthetic samples of the minority class based on the existing minority class samples. Here's a simplified explanation of the process:

1.**Identify the k-nearest neighbors 2. Generate synthetic samples 3. Repeat**

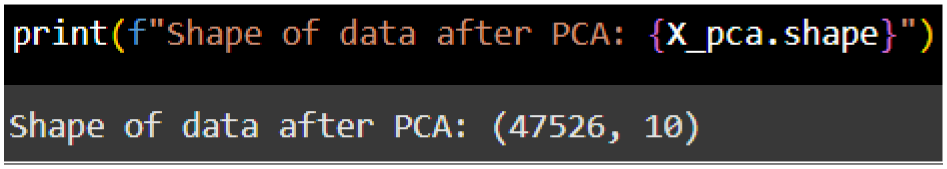


```
Class distribution before SMOTE: Counter({0.0: 23763, 1.0: 75})
Class distribution after SMOTE: Counter({0.0: 23763, 1.0: 23763})
```

PCA ANALYSIS:

PCA is used to reduce the dimensionality of the training data (X_train_smote) from its original number of features to just 10 principal components, stored in X_pca. This can be useful for things like:

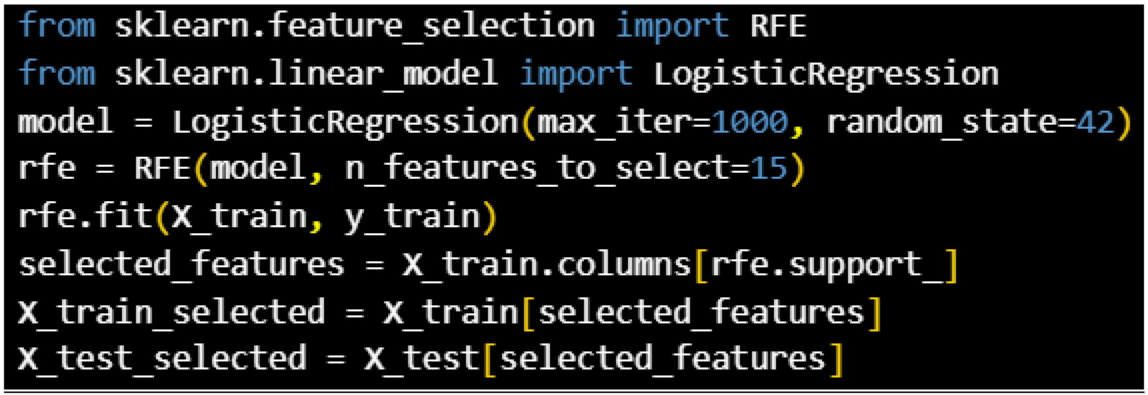
- Speeding up machine learning algorithms: Training algorithms on lower-dimensional data is often faster.
- Visualizing data: It's easier to visualize data in 2 or 3 dimensions (which can be achieved by setting n_components to 2 or 3).
- Removing noise: PCA can help to remove irrelevant features and noise from the data.



```
print(f"Shape of data after PCA: {X_pca.shape}")
```

```
Shape of data after PCA: (47526, 10)
```

LOGISTIC REGRESSION MODEL IMPORTING FOR TRAINING AND TESTING MACHINE:

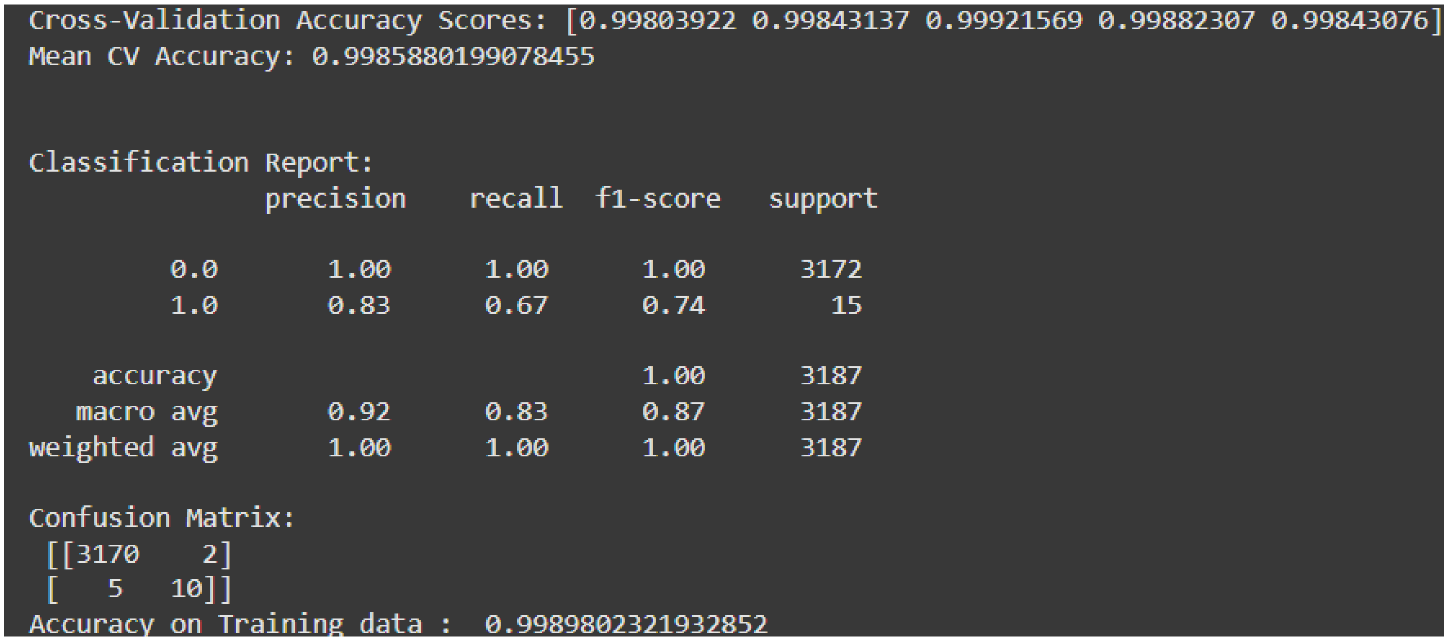


```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000, random_state=42)
rfe = RFE(model, n_features_to_select=15)
rfe.fit(X_train, y_train)
selected_features = X_train.columns[rfe.support_]
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]
```

- In summary, this code snippet uses RFE to select the 15 most important features for predicting credit card fraud based on a Logistic Regression model. The resulting datasets with selected features are then used for further model training and evaluation.
- RFE selector to the training data (X_train, y_train). RFE works by recursively removing features and building a model with the remaining features. It ranks features based on their importance to the model's performance.
- New datasets (X_train_selected, X_test_selected) containing only the selected features for both the training and testing sets. This is done to train and evaluate the model using only the most important features.

TRAINING THE MODEL AND VALIDATING TRAIN MODEL ACCURACY:

This section of the code aims to train a Logistic Regression model, thoroughly evaluate its performance using cross-validation and various evaluation metrics, and gain insights into its predictions using a classification report and confusion matrix. This approach helps ensure that the model generalizes well to new, unseen data and provides a comprehensive understanding of its strengths and weaknesses.



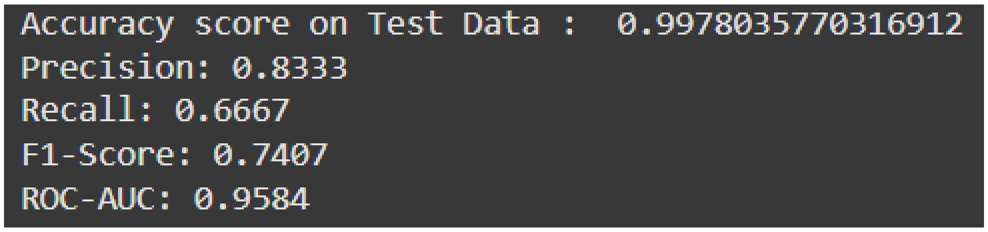
```
Cross-Validation Accuracy Scores: [0.99803922 0.99843137 0.99921569 0.99882307 0.99843076]
Mean CV Accuracy: 0.9985880199078455


Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3172
         1.0       0.83      0.67      0.74        15

    accuracy                           1.00      3187
   macro avg       0.92      0.83      0.87      3187
weighted avg       1.00      1.00      1.00      3187

Confusion Matrix:
 [[3170    2]
 [   5   10]]
Accuracy on Training data :  0.9989802321932852
```

TEST DATA ACCURACY, Precision, Recall, F1-Score, ROC-AUC VALUES.

This section of the code assesses the performance of the model on unseen test data using a range of metrics to provide a comprehensive evaluation of its effectiveness in detecting fraudulent transactions.

- This section calculates and prints several key performance metrics:
    - **Accuracy:** The overall proportion of correctly classified samples.
    - **Precision:** Out of all the samples predicted as positive (fraudulent), what proportion was actually positive? It focuses on minimizing false positives.

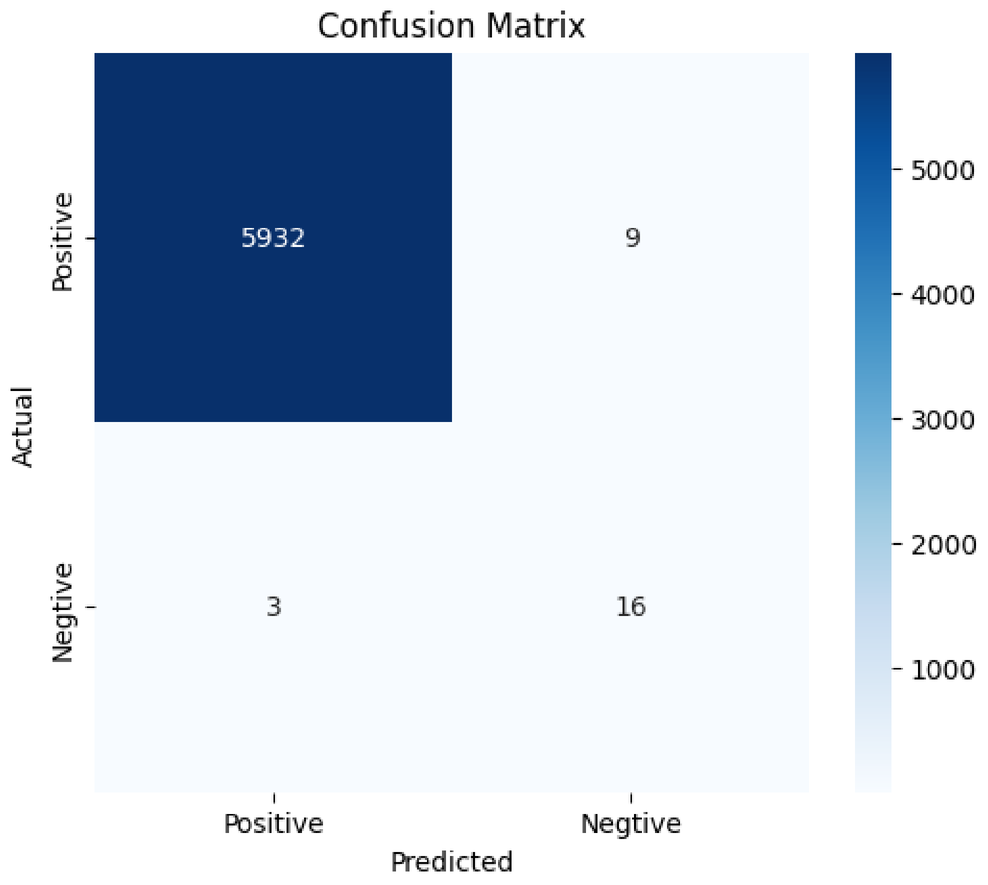
- **Recall:** Out of all the actual positive samples, what proportion did the model correctly identify? It focuses on minimizing false negatives.
- **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.
- **ROC-AUC:** Area Under the Receiver Operating Characteristic Curve, which measures the model's ability to distinguish between classes. A higher AUC indicates better performance.



```
Accuracy score on Test Data :  0.9978035770316912
Precision: 0.8333
Recall: 0.6667
F1-Score: 0.7407
ROC-AUC: 0.9584
```

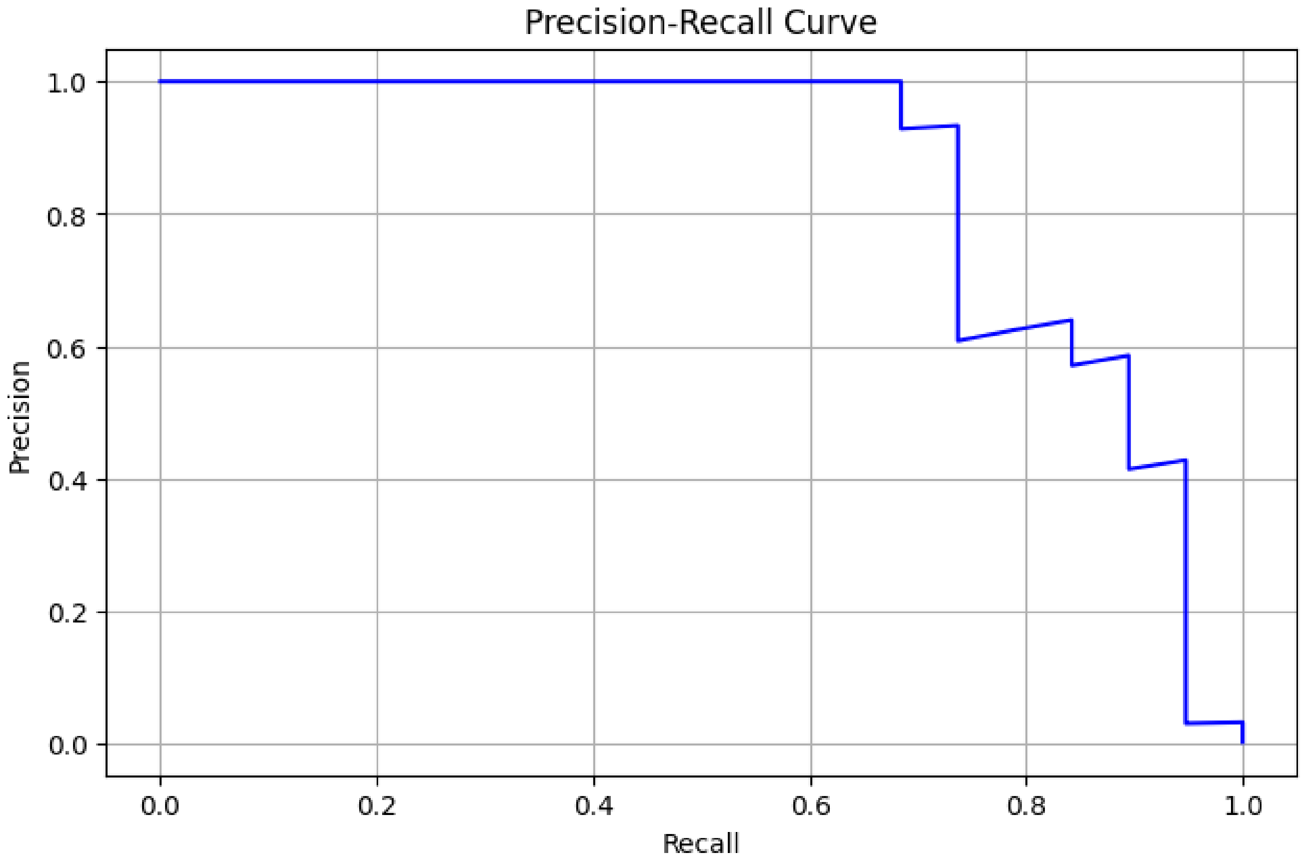
CO-RELATION MARIX VISUALISATION:

Visualizes the confusion matrix for a classification model, providing a clear picture of its performance in terms of true positives, true negatives, false positives, and false negatives.



Precision-Recall Curve:

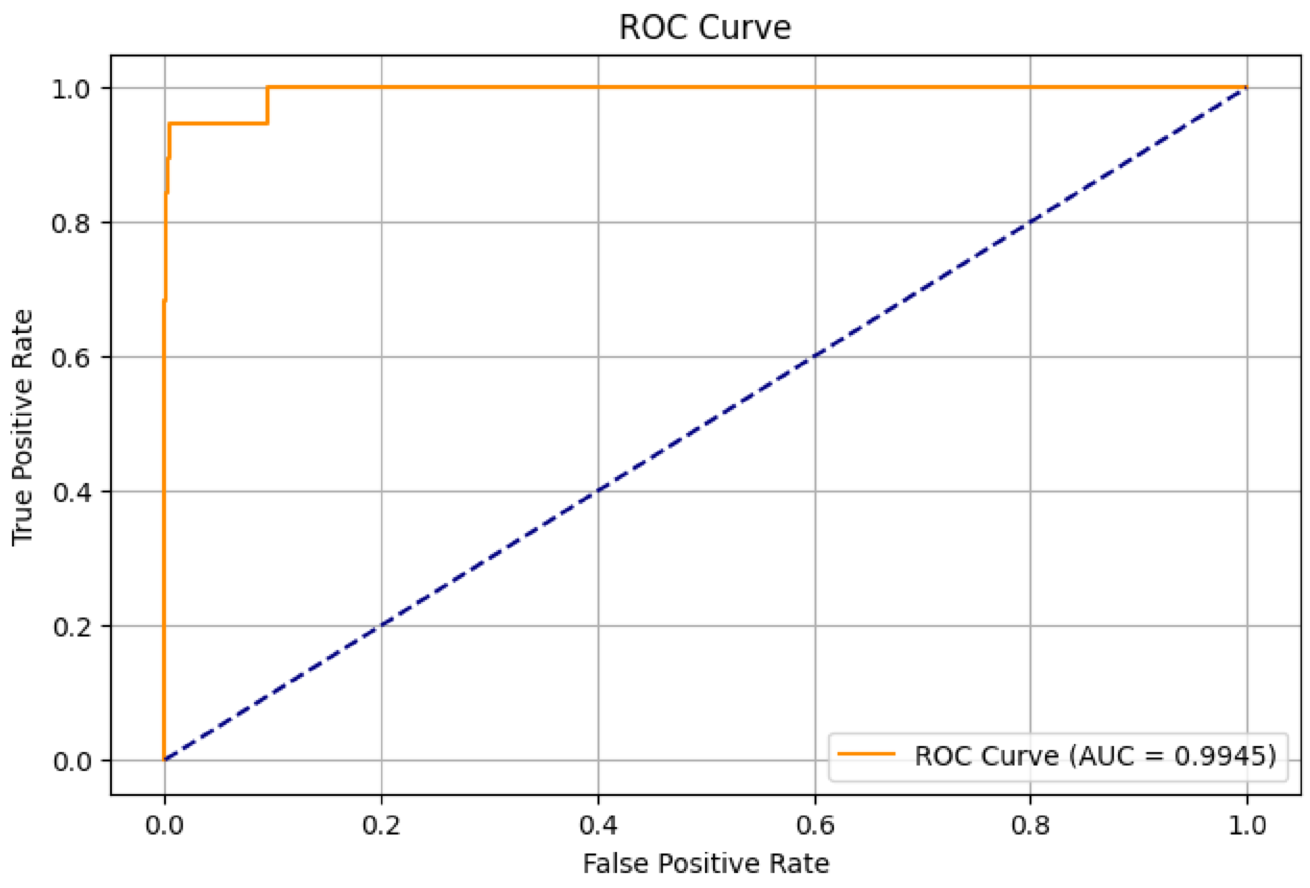
Calculates and visualizes the confusion matrix for a classification model, providing a clear picture of its performance in terms of true positives, true negatives, false positives, and false negatives.



ROC Curve:

ROC Curve (Receiver Operating Characteristic Curve) is a graphical representation of a classification model's performance at various threshold settings. It's created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at different classification thresholds.

- True Positive Rate (TPR): Also known as sensitivity or recall, it measures the proportion of actual positive cases that are correctly identified by the model.
- False Positive Rate (FPR): Also known as the fall-out, it measures the proportion of actual negative cases that are incorrectly classified as positive.
- ROC curve and its AUC provide a visual and numerical assessment of the model's ability to distinguish between the positive and negative classes. A higher AUC indicates better performance, with an AUC of 1 representing a perfect classifier and an AUC of 0.5 representing a random classifier.



CONCLUSION:

This project showcases the use of machine learning models such as Logistic Regression for fraud detection. By addressing class imbalance and leveraging advanced visualization techniques, it offers actionable insights to improve financial security. The models were evaluated using key metrics, highlighting their potential in real-world applications.

COLAB WORKSPACE LINK:

Dhanush_Milestone3.ipynb

THANK YOU

SUBMITTED BY:

DHANUSH TADISETTI

MILESTONE-3