**CREDIT CARD FRAUD DETECTION**

**PHASE 5 PROJECT SUBMISSION**

**PROJECT TITLE:** Credit Card Fraud Detection

**PHASE 5:** Project Documentation & Submission

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

**INTRODUCTION:**

* Credit card fraud detection is a crucial and constantly evolving field within the financial industry that aims to identify and prevent unauthorized or fraudulent use of credit cards. With the widespread adoption of credit and debit cards for transactions, both online and offline, the need for robust fraud detection systems has become paramount.
* The primary objective of credit card fraud detection is to protect consumers and financial institutions from the financial losses and security risks associated with fraudulent transactions. It involves the use of sophisticated technologies, algorithms, and machine learning models to analyze transaction data in real-time, seeking out patterns and anomalies that may indicate potential fraud.
* Credit card fraud can take various forms, including unauthorized transactions, identity theft, card-not-present (CNP) fraud in e-commerce, and card-present fraud at physical point-of-sale locations. Detecting such fraud requires a multifaceted approach that encompasses a range of strategies, from rule-based systems to advanced AI and machine learning techniques.
* To achieve effective credit card fraud detection, financial institutions and businesses must continuously adapt and improve their strategies, staying one step ahead of increasingly sophisticated fraudsters. This often involves monitoring transaction history, analyzing customer behavior, implementing real-time fraud scoring, and collaborating with industry

peers to share threat intelligence.

* In this context, this introduction sets the stage for a deeper exploration of the various aspects of credit card fraud detection, from the technologies and methodologies employed to the challenges faced in this ongoing battle against fraud. By combining human expertise and advanced technology, the financial industry strives to maintain the trust and security of credit card transactions in an increasingly digital and interconnected world.



**OBJECTIVES:**

The primary objectives of credit card fraud detection are to safeguard the financial industry and its consumers by identifying and preventing fraudulent activities. These objectives can be summarized as follows:

* **Fraud Prevention:**The foremost goal of credit card fraud detection is to prevent unauthorized or fraudulent transactions from occurring in the first place. This helps protect consumers from financial losses and ensures the integrity of the payment system.
* **Minimize Financial Losses**: Detecting and preventing fraud helps financial institutions minimize financial losses resulting from unauthorized transactions. By identifying and blocking fraudulent activities early, they can limit the impact on both cardholders and themselves.
* **Enhance Security:** The objective is to enhance the overall security of credit card transactions. This includes protecting sensitive customer data and ensuring that legitimate cardholders can use their cards with confidence.
* **Maintain Customer Trust:** Maintaining the trust of cardholders is crucial for financial institutions. Effective fraud detection and prevention measures reassure customers that their financial assets are secure, which can lead to stronger customer relationships and brand loyalty.
* **Compliance with Regulations:** Credit card fraud detection also helps financial institutions comply with regulatory requirements and industry standards related to data security and fraud prevention. Failing to meet these standards can result in legal and financial consequences.
* **Adapt to Evolving Threats:** As fraudsters continuously develop new tactics, the objective is to adapt and stay ahead of these threats. Credit card fraud detection systems need to be agile and capable of identifying emerging fraud patterns.
* **Optimize Operations:** Efficient fraud detection systems help financial institutions optimize their operations. By reducing false positives (legitimate transactions mistakenly flagged as fraud), they can save time and resources.
* **Reduce Operational Costs:** While investing in fraud detection technology and personnel, the long-term objective is to reduce operational costs related to fraud management. This includes minimizing chargeback fees, investigations, and customer support costs associated with fraud cases.
* **Improve Customer Experience:** Effective fraud detection should not hinder legitimate transactions or inconvenience customers. The objective is to strike a balance between security and a seamless user experience.
* **Collaboration and Information Sharing:**Collaboration with other financial institutions and sharing information about fraud trends and threats is a key objective. By working together, the industry can better combat fraud.
* **Continual Improvement:** The final objective is to continually improve fraud detection systems and strategies. This involves learning from past incidents, refining algorithms, and staying current with the latest technologies and best practices in the field.
* Overall, credit card fraud detection aims to create a secure and trustworthy environment for financial transactions, protect the interests of both consumers and financial institutions, and adapt to the ever-evolving landscape of fraud threats.

**DESIGN:**

To develop a machine learning model we need to install some packages and they are:

* Pandas
* Seaborn
* Matplotlib
* RobustScaler from sklearn.preprocessing
* train\_test\_split from sklearn.model\_selection
* classification\_report,confusion\_matrix and accuracy\_matrix from sklearn.metrics

Dataset is taken from [**https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/**](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/)

Since the data in the dataset is the raw data, it needs to undergo the following stages:

* **Data Collection:** Collect historical credit card transaction data from various sources, including legitimate and fraudulent transactions.
* **Data Preprocessing:**
* Clean and preprocess the data to handle missing values,outliers, and inconsistencies.
* Normalize or scale features as needed.
* Create relevant features, such as transaction frequency, geographic patterns, and behavioral features.
* **Feature Selection:**We need to choose relevant features from the dataset, such as transaction amount, location, time, and more.
* **Model Selection:**
* We need to choose appropriate machine learning algorithms for fraud detection, considering both supervised and unsupervised techniques
* Consider using a combination of models to enhance accuracy e.g., ensemble methods.
* **Model Training:**Train models using historical data, and periodically retrain them to adapt to evolving fraud patterns.
* **Model Evaluation:** We need to continuously monitor the performance of the system using metrics like accuracy, precision, recall, F1-score, and AUC.
* **Data Privacy And Security:**
* We need to ensure that sensitive data, both at rest and in transit, is encrypted.
* Need to implement strict access controls to limit who can view and manipulate the data.
* Need to ensure compliance with relevant data privacy regulations (e.g., GDPR, HIPAA) and industry standards.
* **Scalability:**
* We will ensure that the system can handle a large volume of transactions and scale horizontally as needed.
* Also consider cloud-based solutions for scalability and reliability.

**LIBRARIES USED:**

1.Pandas

2.NumPy

3.Scikit-learn

4.TensorFlow or Pytorch

5.XGBoost or LightGBM

6.Matplotlib and Seaborn

7.Scipy

8.Statsmodels

9.keras

10.Folium

**1.Pandas:**

Pandas is a widely-used library for data manipulation and analysis. It is helpful for data preprocessing and feature engineering.

**2.NumPy:**

NumPy is used for numerical operations, making it useful for mathematical computations and working with arrays.

**3.Scikit-Learn:**

Scikit-Learn is a powerful machine learning library that provides a wide range of tools and algorithms for classification, regression, clustering, and model evaluation.

**4.TensorFlow or Pytorch:**

These deep learning frameworks are essential for implementing deep neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for fraud detection.

**5.XGBoost or LightGBM:**

These gradient boosting libraries are often used for ensemble learning and improving model accuracy.

**6.Matplotlib and Seaborn:**

These libraries are used for data visualization, helping to visualize patterns, anomalies, and model performance.

**7.Scipy:**

Scipy provides a variety of scientific and statistical functions that can be useful in fraud detection, especially for advanced statistical analysis.

**8.Statsmodels:**

Statsmodels is useful for statistical analysis and hypothesis testing, which can be relevant for fraud detection research.

**9.keras:**

If you prefer a high-level API for deep learning, Keras can be an alternative to TensorFlow or PyTorch.

**10.Folium:**

Folium is a Python library for creating interactive maps, which can be helpful when analyzing geographic patterns in fraud detection.

**INNOVATIVE TECHNOLOGIES AND MODULES:**

**1.Machine Learning Algorithms:**

* **Supervised Learning:** Use algorithms like Random Forest, Gradient Boosting, and Support Vector Machines (SVM) to classify transactions as either legitimate or fraudulent based on historical data.
* **Unsupervised Learning:** Employ techniques like clustering (e.g., k-means or DBSCAN) to identify unusual patterns or outliers in transaction data.
* **Deep Learning:** Implement neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for fraud detection. Deep learning can capture complex patterns in transaction sequences.

**2.** **Feature Engineering:**

* Create meaningful features from transaction data, such as transaction frequency, time of day, location, and merchant type. Feature engineering can help algorithms better distinguish between genuine and fraudulent transactions.

**3. Behavioral Analytics:**

* Utilize behavioral analytics to build profiles of users' normal transaction behavior. Any deviation from these profiles can raise red flags for potential fraud.

**4. Anomaly Detection:**

* Implement anomaly detection techniques like Isolation Forests or Autoencoders to identify transactions that deviate significantly from the norm.

**5. Real-time Monitoring:**

* Develop real-time fraud detection systems that can analyze and flag transactions as they occur, enabling immediate action to prevent further fraud.

**6. Machine Learning Explainability:**

* Employ explainable AI techniques to make the model's decisions interpretable, which can help investigators understand why a transaction was flagged as fraudulent.

**7. Hybrid Models and Continuous Learning:**

* Combine multiple models and techniques, such as ensemble methods, to improve the overall accuracy and robustness of fraud detection systems.
* Implement models that can adapt and learn from new data to stay up-to-date with evolving fraud patterns.

**8. User Behavior Biometrics:**

* Utilize biometric data, such as fingerprint or facial recognition, as an additional layer of authentication for online transactions.

**9. Blockchain Technology:**

* Explore blockchain-based solutions for enhancing security and traceability in financial transactions.

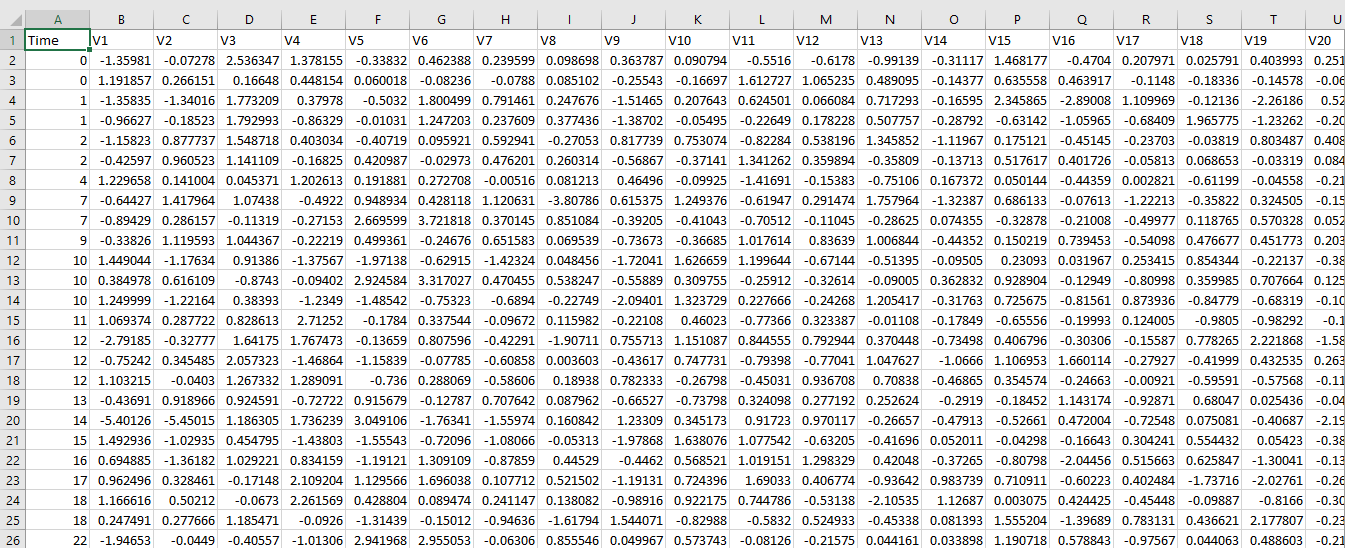
**LIBRARY INSTALLATION AND IMPORTING:**

* Ensure required libraries are installed,like

1. pip install numpy
2. pip install pandas
3. pip install scikit-learn

* Importing libraries in the program,

1. import numpy as np
2. import pandas as pd
3. from sklearn.model\_selection import train\_test\_split
4. from sklearn.ensemble import RandomForestClassifier
5. from sklearn.metrics import classification\_report, accuracy\_score

**SAMPLE DATASET:  
**

**DATA PRE-PROCESSING:**

Data preprocessing is a crucial step in credit card fraud detection to ensure that the data is clean, relevant, and suitable for modeling. The quality of your data greatly influences the performance of your fraud detection models. Here's a step-by-step guide for data preprocessing in credit card fraud detection:

**Data Collection:**

* Gather historical credit card transaction data, including features like transaction amount, timestamp, cardholder information, and transaction outcome (fraudulent or legitimate).

**Data Exploration and Visualization:**

* Analyze the dataset to understand its characteristics and identify any patterns, trends, or anomalies.
* Visualize the data using histograms, scatter plots, and other techniques to gain insights.

**Data Cleaning:**

* Handle missing values, either by removing rows with missing data or imputing missing values.
* Detect and handle outliers, which could be due to errors or fraudulent activities. Outliers can be treated using techniques like Z-score or IQR.

**Feature Selection:**

* Identify relevant features and eliminate irrelevant ones to reduce dimensionality and improve model performance.
* Consider using techniques like feature importance from tree-based models or correlation analysis.

**Data Transformation:**

* Convert categorical variables (e.g., cardholder information) into numerical representations using techniques like one-hot encoding or label encoding.
* Scale or standardize numerical features to bring them to a similar scale to improve model convergence.

**Handling Imbalanced Data:**

* Credit card fraud datasets are often highly imbalanced, with the majority of transactions being legitimate. You may need to address this imbalance using techniques like oversampling, undersampling, or synthetic data generation (e.g., SMOTE).

**Temporal Features:**

* If your data includes timestamps, consider creating temporal features, such as day of the week, time of day, or time since the last transaction, which can be useful for fraud detection.

**Feature Engineering:**

* Create new features that may help improve model performance. For example, you can calculate aggregated statistics for each cardholder or perform clustering to identify groups of similar transactions.

**Splitting the Data:**

* Split the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters, and the testing set is used to evaluate model performance.

**Model-Specific Preprocessing:**

* Some machine learning models may require specific preprocessing steps. For instance, deep learning models might benefit from sequence data representations if transaction order matters.

**Regularization and Data Leakage:**

* Be cautious about data leakage, where information from the future is used to predict the past. Ensure that your preprocessing steps do not inadvertently introduce data leakage.
* Apply regularization techniques to prevent overfitting.

**Evaluation and Monitoring:**

* Continuously monitor model performance in a production environment and retrain the model as new data becomes available. Make sure to update preprocessing steps if necessary.

**FEATURE ENGINEERING:**

Feature engineering is a critical step in preparing the data for training machine learning models. It involves creating new features or transforming existing ones to improve model performance and fraud detection accuracy.

some feature engineering techniques and ideas for credit card fraud detection:

**Transaction Frequency Features:** Transaction frequency features help identify unusual transaction patterns based on the number of transactions within specific time intervals. In this case, we calculate the number of transactions in the last hour for each cardholder.

**Code:**

>>>df['transactions\_last\_hour'] = df.groupby('cardholder')['Time'].transform(lambda x: x.diff().lt(3600).cumsum())

**Transaction Amount Features:** Transaction amount features are important to understand the spending behavior of cardholders. We compute statistics like the average, maximum, and minimum transaction amounts for each cardholder.

**Code:**

>>>df['avg\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('mean')

>>>df['max\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('max')

>>>df['min\_transaction\_amount'] = df.groupby('cardholder')['Amount'].transform('min')

**Time-Based Patterns:** Time-based features capture patterns in transaction times. By extracting information such as the hour of the day, day of the week, and time since the last transaction, we can identify suspicious temporal behavior.

**Code:**

>>>df['hour'] = pd.to\_datetime(df['Time']).dt.hour

>>>df['day\_of\_week'] = pd.to\_datetime(df['Time']).dt.dayofweek

>>>df['time\_since\_last\_transaction'] = df.groupby('cardholder')['Time'].diff()

**Cardholder Behavior:** Features related to cardholder behavior provide insights into spending habits. We calculate the average number of transactions per day and the average amount spent per transaction.

**Code:**

>>>df['avg\_transactions\_per\_day'] = df.groupby(['cardholder', ‘day\_of\_week']) ['Amount'] .transform('count')

>>>df['avg\_amount\_per\_transaction'] = df.groupby(['cardholder'])['Amount'].transform('mean')

**Merchant-Based Features**: Merchant-based features focus on the characteristics of merchants. These features include the average transaction amount at each merchant and the number of transactions in specific merchant categories.

**Code:**

>>>df['avg\_transaction\_amount\_merchant'] = df.groupby ('MerchantID') ['Amount']. transform('mean')

>>>df['num\_transactions\_merchant\_category'] = df.groupby ('MerchantCategory') ['Amount']. Transform('count')

**MODEL TRAINING:**

Model training involves selecting the right machine learning algorithms, tuning hyperparameters, and training the models to detect credit card fraud effectively.

**Data Collection:**

* Gather a dataset of historical credit card transactions. This dataset should include information about each transaction, including transaction amount, timestamp, and features that you've engineered during the feature engineering process.
* The dataset should also include labels indicating whether each transaction is legitimate (non-fraudulent) or fraudulent.

**Data Preprocessing:**

* Clean and preprocess the data, handling missing values and outliers appropriately.
* Normalize or scale numerical features to ensure they have a similar range. This step is essential for many machine learning algorithms.

**Data Splitting:**

* Divide the dataset into training, validation, and test sets. Common splits include 70-80% for training, 10-15% for validation, and the remaining 10-15% for testing. The validation set is used to fine-tune hyperparameters, while the test set is used to evaluate the model's performance.

**Choosing an Algorithm:**Selecting a suitable machine learning algorithm for credit card fraud detection. Common choices include:

* Logistic Regression
* Random Forest
* Gradient Boosting (e.g., XGBoost, LightGBM)
* Neural Networks (Deep Learning)
* Anomaly Detection algorithms (e.g., Isolation Forest, One-Class SVM)

We using Logistic Regression and Random Forest

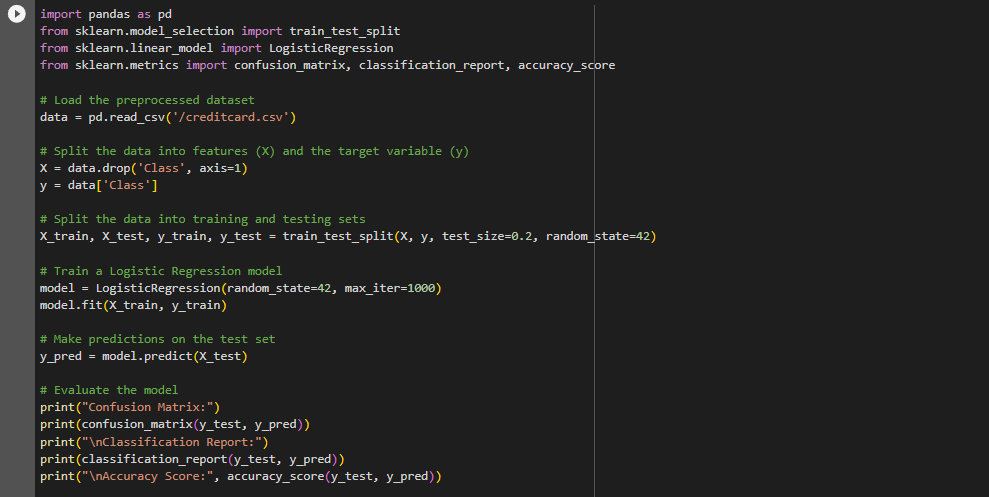
**MODEL SELECTION:**

Choose machine learning models suitable for credit card fraud detection. Common options include Logistic Regression, Random Forest, and Gradient Boosting models.

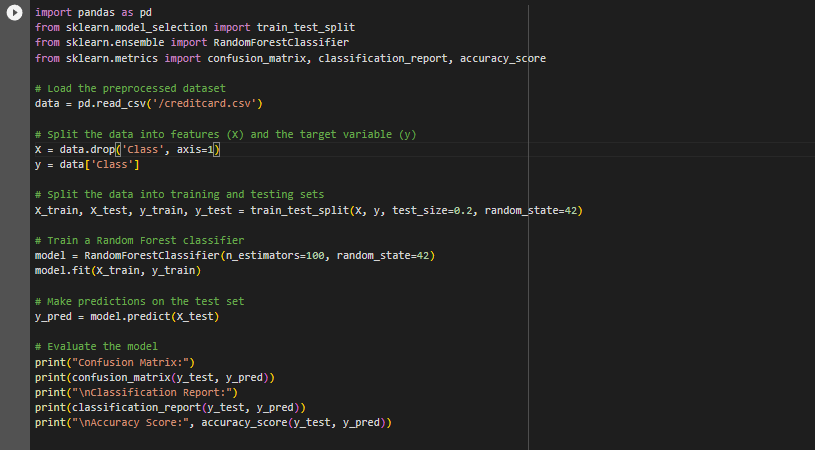
**Hyperparameter Tuning:** Tune the hyperparameters of selected models to optimize their performance. Techniques like Grid Search or Random Search can be used.

**Model Training:** Train the chosen algorithm on the training dataset. The model learns to distinguish between legitimate and fraudulent transactions based on the labeled data.

**The below example using Logistic regression:**

****

**The below example using Random Forest:**

****

**EVALUATION:**

Evaluating the model's performance is crucial to assess its effectiveness in detecting credit card fraud.

**Accuracy:**

* Accuracy measures the proportion of correctly classified transactions. However, in the presence of imbalanced data, accuracy can be misleading because it can be high even when the model fails to detect most of the fraud cases.

**Precision (Positive Predictive Value**):

* Precision measures the fraction of true positive predictions (correctly identified fraud cases) out of all positive predictions (total cases predicted as fraud). A higher precision indicates that the model is better at not misclassifying legitimate transactions as fraud.
* **Precision = True Positives / (True Positives + False Positives)**

**Recall (Sensitivity, True Positive Rate):**

* Recall measures the proportion of actual fraud cases that were correctly identified by the model. A higher recall means that the model is better at detecting most of the fraud cases.
* Recall = True Positives / (True Positives + False Negatives)

**F1-Score:**

* The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, taking both false positives and false negatives into account. It is especially useful when there is an imbalance between the classes.
* F1-Score = 2 \* (Precision \* Recall) / (Precision + Recall)

**Evaluation Metrics:**

* Define the evaluation metrics to assess the performance of the models. Common metrics include precision, recall, F1-score, and the area under the Receiver Operating Characteristic (ROC-AUC) curve.

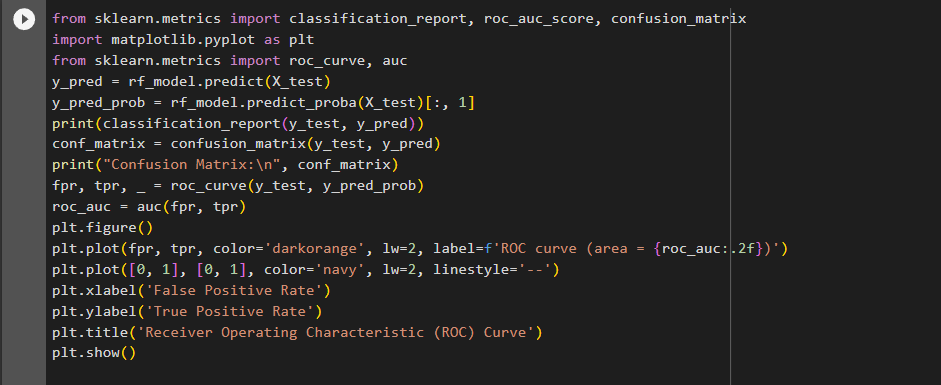
**Model Performance:**

* Present the results of model evaluation on the test data, including relevant metrics and visualizations like confusion matrices and ROC curves.

**Confusion Matrix:**

* Examining the confusion matrix, which includes True Positives, False Positives, True Negatives, and False Negatives, provides a more detailed view of model performance.

**Code:**



**Coding:**

*#importing the dependencies*

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**from** sklearn.preprocessing **import** StandardScaler, RobustScaler

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV

**from** imblearn.over\_sampling **import** SMOTE, RandomOverSampler

*#from sklearn.metrics import classification\_report, average\_precision\_score, plot\_precision\_recall\_curve, plot\_roc\_curve*

**from** sklearn.feature\_selection **import** mutual\_info\_classif

*#Importing models*

**from** sklearn.linear\_model **import** LogisticRegression, SGDClassifier

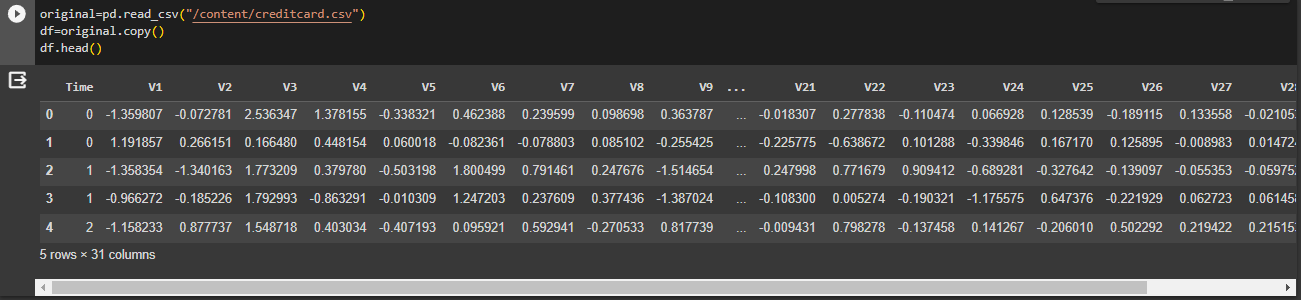
**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.tree **import** DecisionTreeClassifier

original**=**pd**.**read\_csv("creditcard.csv")

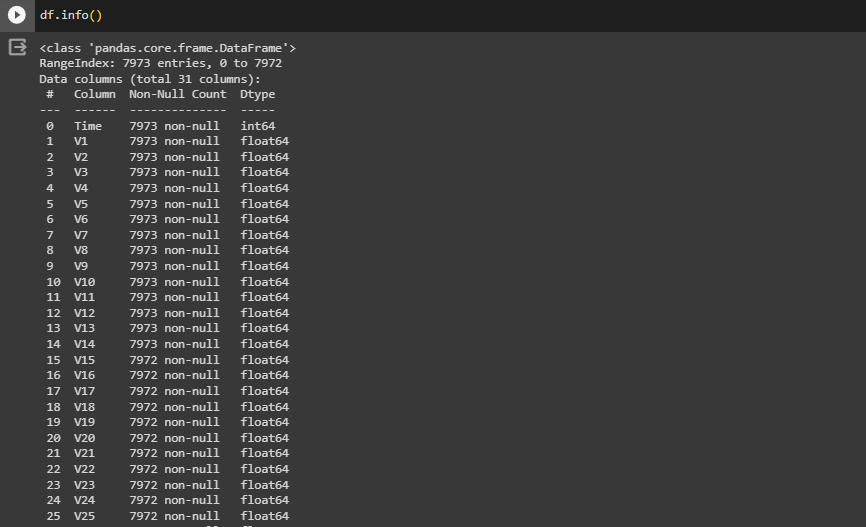
df**=**original**.**copy()

df**.**head()



The code reads a CSV file named "creditcard.csv" into a Pandas dataframe "original", creates a copy of it in another dataframe "df", and displays

df**.**info()



*#Draw distribution plot for Time and Amount*

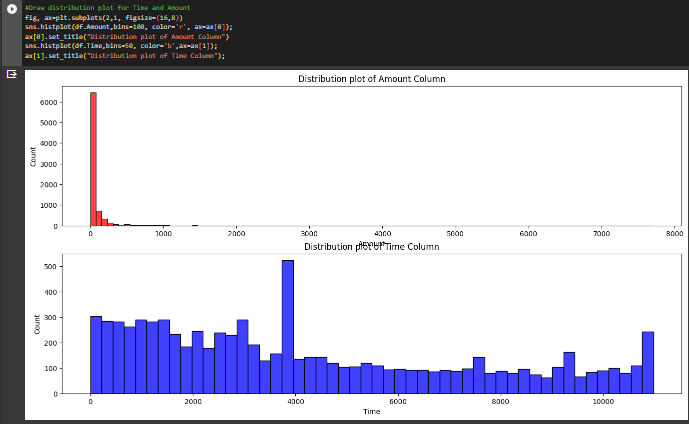
fig, ax**=**plt**.**subplots(2,1, figsize**=**(20,10))

sns**.**histplot(df**.**Amount,bins**=**500, color**=**'r', ax**=**ax[0]);

ax[0]**.**set\_title("Distribution plot of Amount Column")

sns**.**histplot(df**.**Time,bins**=**50, color**=**'b',ax**=**ax[1]);

ax[1]**.**set\_title("Distribution plot of Time Column");

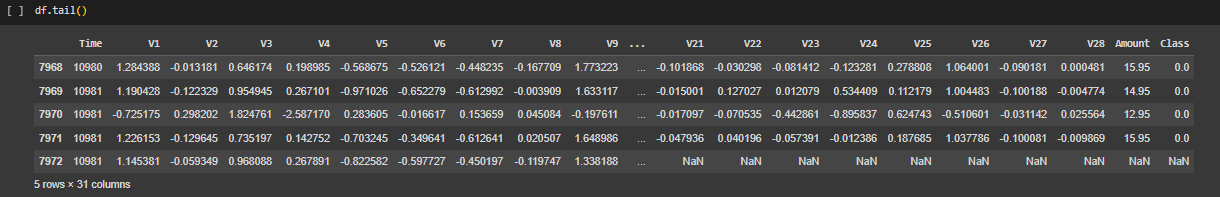
****

The code creates a 2x1 figure with subplots, plots a histogram of the 'Amount' column in red with 500 bins and a title "Distribution plot of Amount Column" on the first subplot and a histogram of the 'Time' column in blue with 50 bins and a title "Distribution plot of Time Column" on the second subplot using seaborn's "histplot" function.

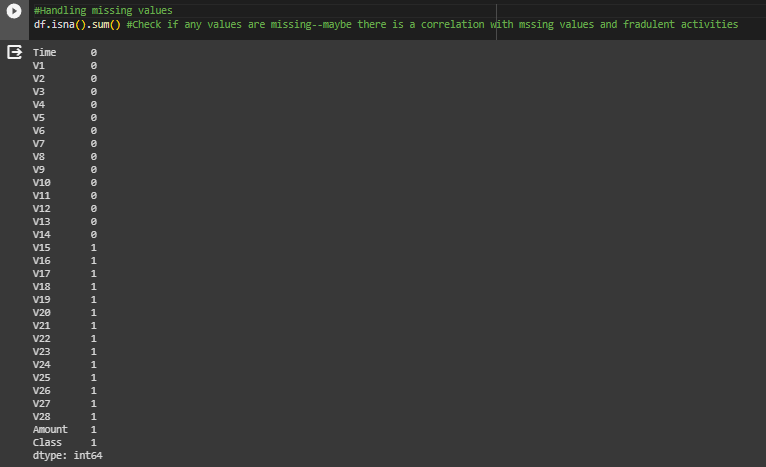
max(df**.**Time) *# this is equivalent to 2 days almost so try to convert out time into hrs(total time period is 48 hrs)*



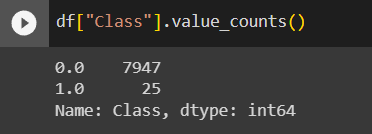
df**.**tail()



df**.**isna()**.**sum() *#Check if any values are missing--maybe there is a correlation with mssing values and fradulent activities*



df["Class"]**.**value\_counts()

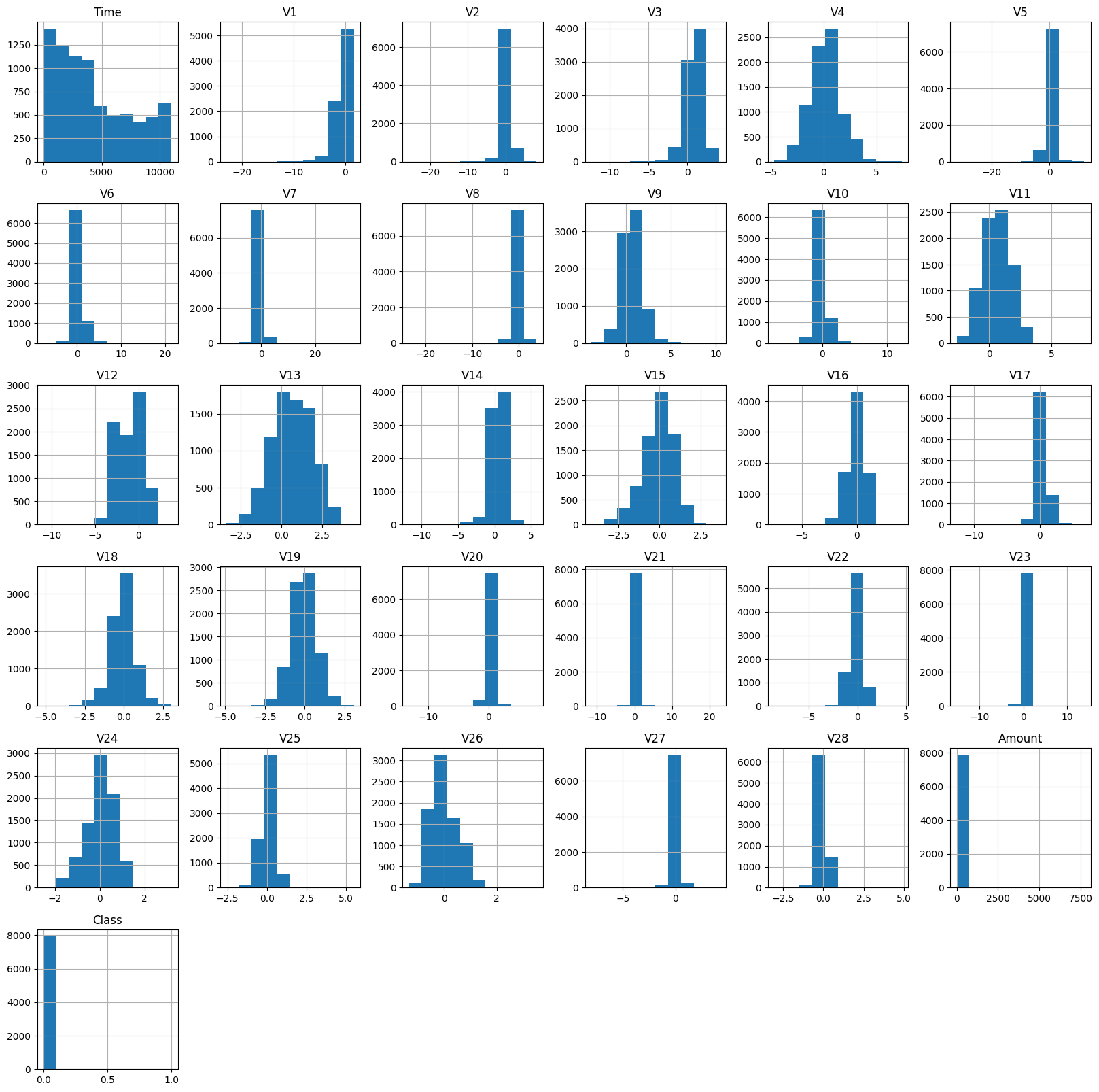


This Dataset is highly unbalanced

class 0 represents Normal Transaction class 1 represents fraudulent transaction

df**.**hist(figsize **=** (20, 20))

plt**.**show()



*#Lets visualize this*

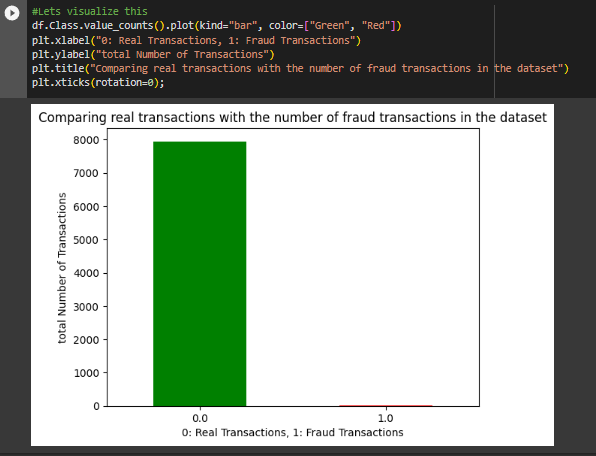
df**.**Class**.**value\_counts()**.**plot(kind**=**"bar", color**=**["Green", "Red"])

plt**.**xlabel("0: Real Transactions, 1: Fraud Transactions")

plt**.**ylabel("total Number of Transactions")

plt**.**title("Comparing real transactions with the number of fraud transactions in the dataset")

plt**.**xticks(rotation**=**0);



As you can see, the dataset is highly imbalanced which can significantly affect out ML model. A balanced dataset is best for training purposes

*#See if the transactions occur at a given time period*

plt**.**figure(figsize**=**(15,10))

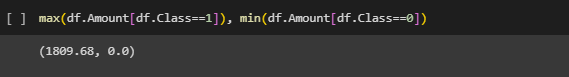
plt**.**scatter(df**.**Time[df**.**Class**==**1], df**.**Class[df**.**Class**==**1], c**=**"red")

plt**.**scatter(df**.**Time[df**.**Class**==**0], df**.**Class[df**.**Class**==**0], c**=**"green");



There is no pattern for when the fraudulent activites occur. They happen at random

max(df**.**Amount[df**.**Class**==**1]), min(df**.**Amount[df**.**Class**==**0])



plt**.**figure(figsize**=**(10,6))

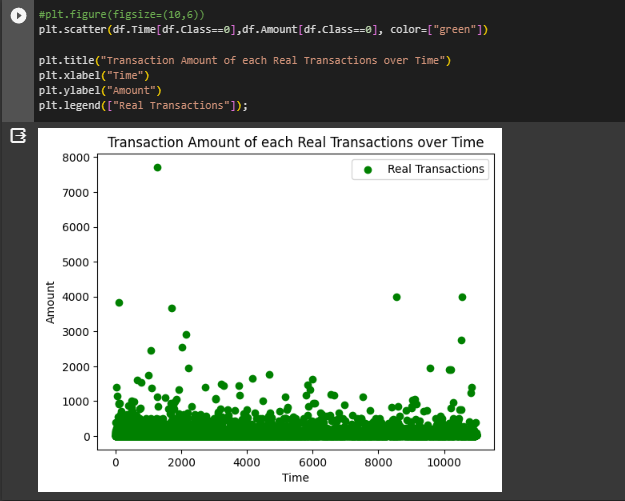
plt**.**scatter(df**.**Time[df**.**Class**==**0],df**.**Amount[df**.**Class**==**0], color**=**["green"])

plt**.**title("Transaction Amount of each Real Transactions over Time")

plt**.**xlabel("Time")

plt**.**ylabel("Amount")

plt**.**legend(["Real Transactions"]);



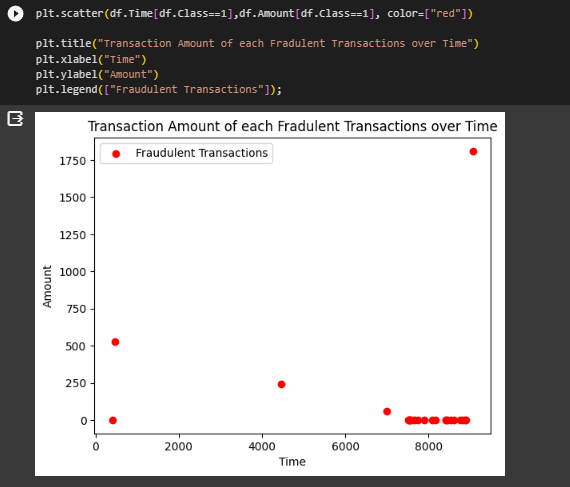
plt**.**scatter(df**.**Time[df**.**Class**==**1],df**.**Amount[df**.**Class**==**1], color**=**["red"])

plt**.**title("Transaction Amount of each Fradulent Transactions over Time")

plt**.**xlabel("Time")

plt**.**ylabel("Amount")

plt**.**legend(["Fraudulent Transactions"]);

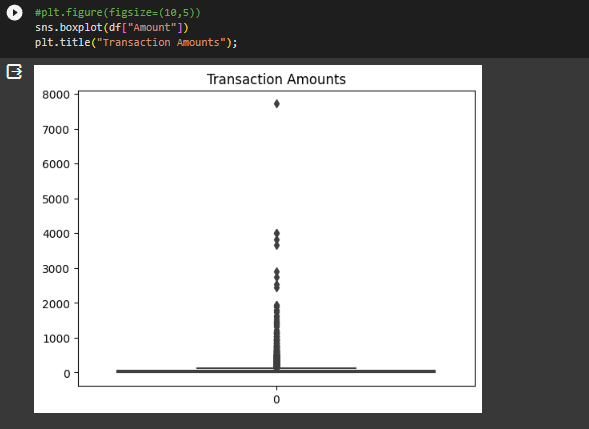


The Transaction amount for each fraudulent transaction is significantly lower compared to the Real transactions. This makes it a lot more difficult to detect fraud activities. The max amount is approxiamtely $2100.

plt**.**figure(figsize**=**(10,5))

sns**.**boxplot(df["Amount"])

plt**.**title("Transaction Amounts");



df**.**Amount**.**skew()

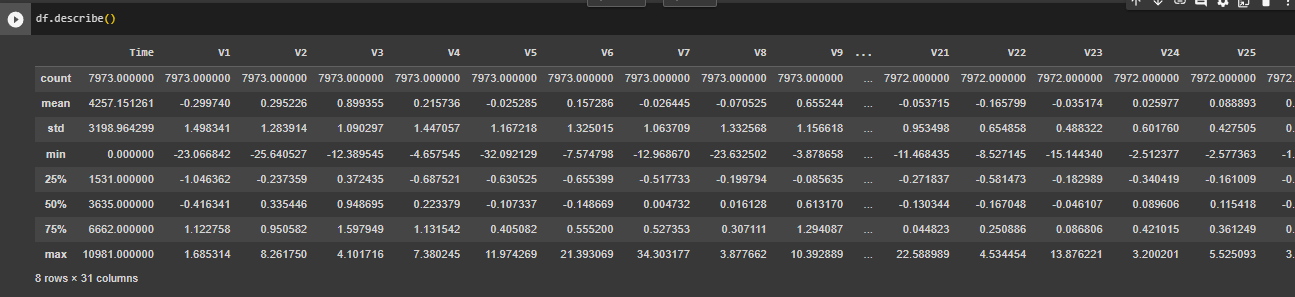
Out[18]:

16.977724453761024

The Amount is heavily right-skewed with a lot of outliers which we can fix using BoxCox transformation

**Lets look at the Mean and Standard Deviations for V1-V28 feature**

df**.**describe()



vs **=** df**.**drop(labels**=**["Time", "Amount","Class"], axis**=**1)

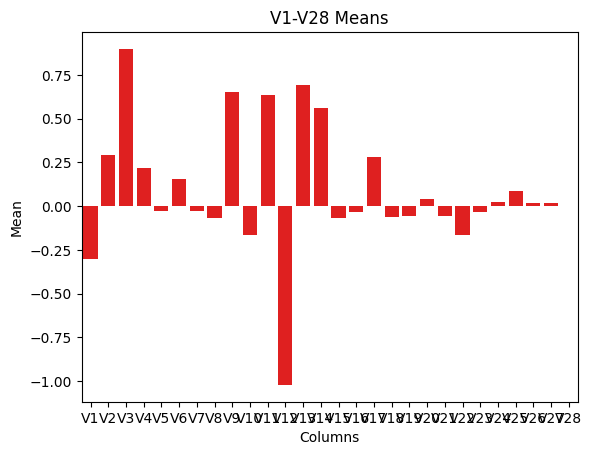
plt**.**figure(figsize**=**(10,10))

sns**.**barplot(x**=**vs**.**columns, y**=**vs**.**mean(), color**=**"red")

plt**.**xlabel("Columns")

plt**.**ylabel("Mean")

plt**.**title("V1-V28 Means");



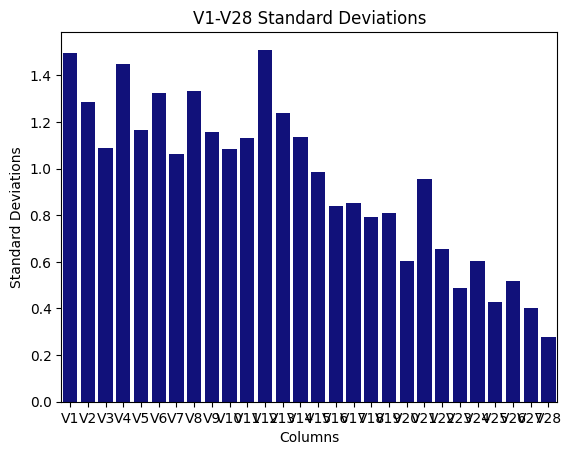
plt**.**figure(figsize**=**(10,10))

sns**.**barplot(x**=**vs**.**columns, y**=**vs**.**std(), color**=**"darkblue")

plt**.**xlabel("Columns")

plt**.**ylabel("Standard Deviations")

plt**.**title("V1-V28 Standard Deviations");



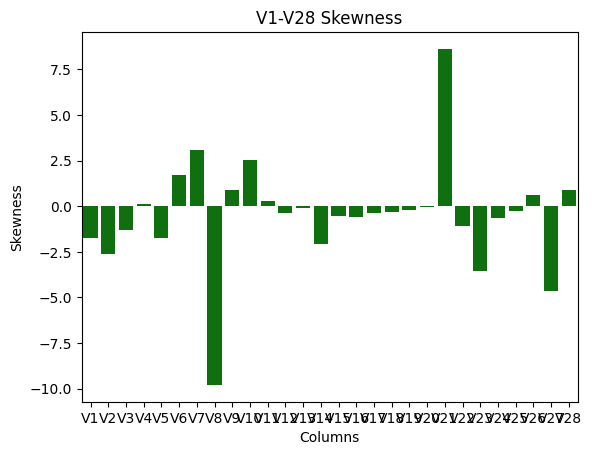
plt**.**figure(figsize**=**(10,10))

sns**.**barplot(x**=**vs**.**columns, y**=**vs**.**skew(), color**=**"green")

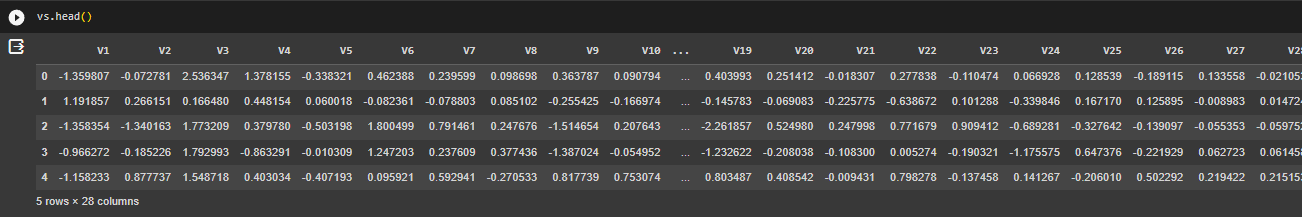
plt**.**xlabel("Columns")

plt**.**ylabel("Skewness")

plt**.**title("V1-V28 Skewness");



vs**.**head()



*# separating data for further analysis*

valid **=** df[df**.**Class **==** 0]

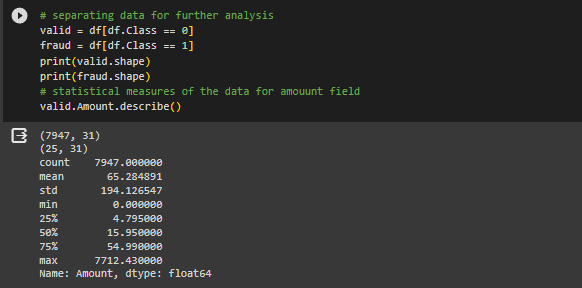
fraud **=** df[df**.**Class **==** 1]

print(valid**.**shape)

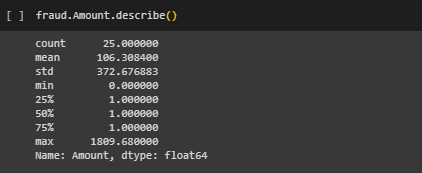
print(fraud**.**shape)

*# statistical measures of the data for amouunt field*

valid**.**Amount**.**describe()



fraud**.**Amount**.**describe()



# Correlation Matrix to see how each variable is related to each other and the target variable

**!**pip install --upgrade numpy

Requirement already satisfied: numpy in c:\users\sohel\appdata\local\programs\python\python311\lib\site-packages (1.24.1)

corr**=**df**.**corr()

mask **=** np**.**zeros\_like(corr, dtype**=**np**.**bool\_)

mask[np**.**triu\_indices\_from(mask)] **=** **True**

*# Set up the matplotlib figure*

f, ax **=** plt**.**subplots(figsize**=**(30, 30))

*# Generate a custom diverging colormap*

cmap **=** sns**.**diverging\_palette(220, 10, as\_cmap**=True**)

*# Draw the heatmap with the mask and correct aspect ratio*

sns**.**heatmap(corr, annot**=True**, mask**=**mask, cmap**=**cmap, vmax**=**.3, center**=**0,

square**=True**, linewidths**=**.5, cbar\_kws**=**{"shrink": .5});

# 

# Correlation matrix does not make sense. There is no correlation between the V# variables due to the PCA performed on the features. A correlation matrix only makes sense if the values are linear. Therefore, we are going to have to use other techniques to visualize and fit the data to the Machine Learning model

X**=**df**.**drop("Class", axis**=**1)

y**=**df**.**Class

X\_train, X\_test, y\_train, y\_test**=**train\_test\_split(X,y, test\_size**=**0.3, random\_state**=**42)

X\_train**.**shape, X\_test**.**shape, y\_train**.**shape, y\_test**.**shape

# ((199364, 30), (85443, 30), (199364,), (85443,))

y\_train**.**value\_counts(), y\_test**.**value\_counts()

**(0 199008**

**1 356**

**Name: Class, dtype: int64,**

**0 85307**

**1 136**

**Name: Class, dtype: int64)**

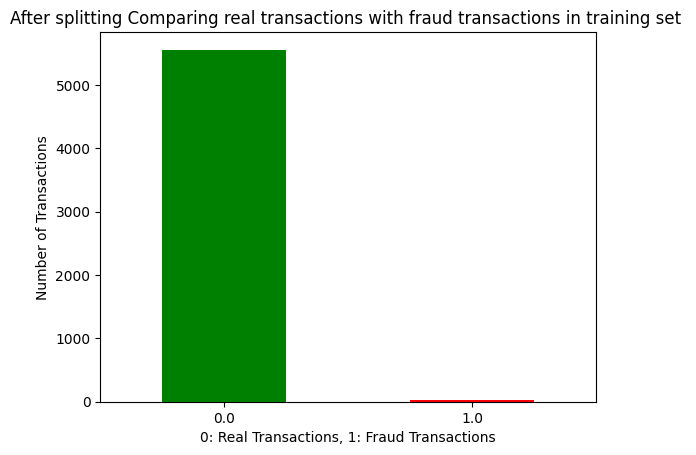
y\_train**.**value\_counts()**.**plot(kind**=**"bar", color**=**["Green", "Red"])

plt**.**xlabel("0: Real Transactions, 1: Fraud Transactions")

plt**.**ylabel("Number of Transactions")

plt**.**title("After splitting Comparing real transactions with fraud transactions in training set ")

plt**.**xticks(rotation**=**0);

****

# StandardScaler

ss**=**StandardScaler()

X\_train**=**ss**.**fit\_transform(X\_train)

X\_test**=**ss**.**transform(X\_test)

X\_train**.**shape, X\_test**.**shape

**((199364, 30), (85443, 30))**

X\_train[0]

**array([-1.95144063, -1.16681856, -0.28654908, 0.53924737, -1.20368154,**

**0.58967831, -1.23993846, 0.75411155, -0.45490037, 1.2064479 ,**

**-0.39869658, 0.32182001, 0.58170129, 0.11222719, -0.22290476,**

**1.38644463, -1.0221033 , -0.35174842, -0.7336555 , -0.49340827,**

**-1.06710984, -0.28780091, 1.26352311, 1.37067196, 0.69925952,**

**0.59643511, -1.6211327 , 0.96676 , -0.4486209 , -0.33974783])**

**RobustScaler:**

Lets compare RandomOverSampler and SMOTE to see which yields better results

# RandomOverSamples:

X\_train, y\_train

# 

# Making sure is actually correctly balancing the dataset out:

lol**=**pd**.**DataFrame(data**=**y\_train, columns**=**["Class"])

lol**.**Class**.**value\_counts()**.**plot**.**bar(color**=**["Red", "Green"]);

plt**.**xlabel("Classes")

plt**.**ylabel("Count")

plt**.**xticks(rotation**=**0)

plt**.**title("Count of \"Class\" feature in training set after ")

# 

# CONCLUSION:

# The overall conclusion for credit card fraud detection is that it is a crucial aspect of maintaining financial security for both individuals and businesses. Advanced technologies and machine learning algorithms have significantly improved the accuracy and efficiency of fraud detection systems. However, it's important to note that no system is foolproof, and continuous monitoring, periodic updates, and user awareness are essential components of a robust fraud prevention strategy. Additionally, collaboration between financial institutions, law enforcement, and cybersecurity experts is crucial in staying ahead of evolving fraud tactics.