Credit Card Fraud Detection

Team members:

Abhijith SS

Adarsh LS

Anandhu S

Ashkar Muhammad AM

Akhil SS

Phase 2 Submission Document

Project: Credit Card Fraud Detection

Introduction:

* **Protecting Consumers: Prevents unauthorized transactions andidentity theft.**
* **Protecting Businesses: Safeguards against revenue loss and reputational damage.**
* **Reducing Financial Losses: Minimizes financial impact onndividuals and organizations.**
* **Maintaining Trust in the Financial System: Ensures confidencein payment networks.**
* **Future Developments :**we need credit card fraud detection techniques**to protect the cardholders from false activity.** India is on its way to becoming a developed country. To achieve this, the Government of India (GoI) has launched several initiatives and one of these is Digital India Campaign.

Content for Project Phase 2 :

For analyzing data, we need some libraries. In this section, we are importing all the required libraries like pandas, NumPy, matplotlib, plotly, seaborn, and word cloud that are required for data analysis. Check the below code to import all the required libraries.

Data Source

A good data source for credit card fraud detection should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

# Data Collection and Preprocessing:

* Data collection : With Credit Card Fraud Detection, this project demonstrates the modelling of a data collection using **machine learning**. Modeling prior credit card transactions with data from those that turned out to be fraudulent is part of the Credit Card Fraud Detection Problem. The model is then used to determine whether or not a new transaction is fraudulent.

# Data preprocessing: Analyzing the effect of data preprocessing techniques using machine learning algorithms on the diagnosis of fraud detection

# Exploratory Data Analysis ( EDA ):

# This case study is focused to give you an idea of applying Exploratory Data Analysis (EDA) in a real business scenario. In this case study, apart from applying the various Exploratory Data Analysis (EDA) techniques, you will also develop a basic understanding of risk analytics and understand how data can be utilized in order to minimise the risk of losing money while lending to customers.

# Feature Engineering:

**Feature engineering** is a crucial step in credit card fraud detection. It involves selecting and transforming the most relevant features from the dataset to improve the performance of machine learning models. [In credit card fraud detection, feature engineering can help identify patterns and anomalies in transaction data that are indicative of fraudulent activity](https://medium.com/dataman-in-ai/how-to-create-good-features-in-fraud-detection-de6562f249ef).

Advanced Regression Techniques:

# There are several machine learning techniques that can be used for credit card fraud detection. One such technique is **logistic regression**. [In a study, researchers investigated the use of logistic regression to detect fraudulent credit card transactions in an imbalanced dataset where only a small fraction of transactions are fraudulent](https://ieeexplore.ieee.org/document/10112302/).Another technique is **genetic algorithm (GA)** based feature selection. A recent paper proposed a machine learning based credit card fraud detection engine using the GA algorithm for feature selection. [After the optimized features are chosen, the proposed detection engine uses the following ML classifiers: Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), Artificial Neural Network (ANN), and Naive Bayes (NB) 2](https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00573-8). [The paper also demonstrated that their proposed approach outperforms existing systems](https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00573-8).

# Model Interpretability:

Credit card fraud is a growing problem in the financial industry, with the potential to cause significant

financial losses to both customers and financial institutions. As a result, there has been a significant

amount of research in recent years on developing effective fraud detection systems. These systems rely on

a combination of statistical techniques, machine learning algorithms, and deep learning models to identify

fraudulent transactions.One of the most commonly used approaches for credit card fraud detection is

rule-based systems. These systems use predefined rules to identify transactions that are deemed suspicious.

However, rule-based systems have limitations, as they are only as good as the rules that have been

predefined, and they may not be able to detect new types of fraud. To overcome these limitations, machine

learning algorithms and statistical techniques have been applied to credit card fraud detection. These

techniques are based on analysing transaction-related data, such as the transaction amount, location, and

time, as well as other relevant factors, such as the customer’s transaction history and account details. In

recent years, deep learning models, such as convolutional neural networks (CNNs) and recurrent neural

networks (RNNs), have also been applied to credit card fraud detection. These models have shown

promising results in identifying fraudulent transactions by learning patterns in the data and improving the

accuracy of fraud detection. Overall, credit card fraud detection is a critical area of research in the financial

industry, with significant potential for improving fraud detection rates and reducing financial losse

Credit card fraud is a growing problem in the financial industry, with the potential to cause significant

financial losses to both customers and financial institutions. As a result, there has been a significant

amount of research in recent years on developing effective fraud detection systems. These systems rely on

a combination of statistical techniques, machine learning algorithms, and deep learning models to identify

fraudulent transactions.One of the most commonly used approaches for credit card fraud detection is

rule-based systems. These systems use predefined rules to identify transactions that are deemed suspicious.

However, rule-based systems have limitations, as they are only as good as the rules that have been

predefined, and they may not be able to detect new types of fraud. To overcome these limitations, machine

learning algorithms and statistical techniques have been applied to credit card fraud detection. These

techniques are based on analysing transaction-related data, such as the transaction amount, location, and

time, as well as other relevant factors, such as the customer’s transaction history and account details. In

recent years, deep learning models, such as convolutional neural networks (CNNs) and recurrent neural

networks (RNNs), have also been applied to credit card fraud detection. These models have shown

promising results in identifying fraudulent transactions by learning patterns in the data and improving the

accuracy of fraud detection. Overall, credit card fraud detection is a critical area of research in the financial

industry, with significant potential for improving fraud detection rates and reducing financial losse

Credit Card Fraud can be defined as a case where a person uses someone else’s credit card for personal reasons while the owner and the card-issuing authorities are unaware of the fact that the card is being used. Due to the rise and acceleration of E-Commerce, there has been a tremendous use of credit cards for online shopping which led to High amount of frauds related to credit cards. In the era of digitalization, the need to identify credit card frauds is necessary. Fraud detection involves monitoring and analyzing the behaviour of various users to estimate detect or avoid undesirable behaviour. To identify credit card fraud detection effectively, we need to understand the various technologies, algorithms and types involved in detecting credit card frauds. The algorithm can differentiate transactions which are fraudulent or not. Find fraud, they need to passed dataset and knowledge of the fraudulent transaction. They analyze the dataset and classify all transactions. Fraud detection involves monitoring the activities of populations of users to estimate, perceive or avoid objectionable behaviour, which consist of fraud, intrusion, and defaulting. Machine learning algorithms are employed to analyses all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent. The investigators provide feedback to the automated system which is used to train and update the algorithm to eventually improve the fraud-detection performance over time.

# Deployment and Prediction:

Deploy the chosen regression model to credit card fraud detection

Develop a user-friendly interface for users to input property features.

Program:

### Credit Card Fraud Detection

$ pip install sklearn==0.24.2 imbalanced-learn numpy pandas matplotlib seaborn

Let's import the necessary libraries:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib import gridspec

Now we read the data and try to understand each feature's meaning. The Python module [pandas](https://pandas.pydata.org/) provide us with the functions to read data. In the next step, we will read the data from our directory where the data is saved, and then we look at the first and last five rows of the data using head(), and tail() methods:

dataset = pd.read\_csv("creditcard.csv")

dataset.head().append(dataset.tail())

╔════════╤══════════╤════════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤════════╤═══════╗

║ │ Time │ V1 │ V2 │ V3 │ V4 │ V5 │ V6 │ V7 │ V8 │ V9 │ ... │ V21 │ V22 │ V23 │ V24 │ V25 │ V26 │ V27 │ V28 │ Amount │ Class ║

╠════════╪══════════╪════════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪════════╪═══════╣

║ 0 │ 0.0 │ -1.359807 │ -0.072781 │ 2.536347 │ 1.378155 │ -0.338321 │ 0.462388 │ 0.239599 │ 0.098698 │ 0.363787 │ ... │ -0.018307 │ 0.277838 │ -0.110474 │ 0.066928 │ 0.128539 │ -0.189115 │ 0.133558 │ -0.021053 │ 149.62 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 1 │ 0.0 │ 1.191857 │ 0.266151 │ 0.166480 │ 0.448154 │ 0.060018 │ -0.082361 │ -0.078803 │ 0.085102 │ -0.255425 │ ... │ -0.225775 │ -0.638672 │ 0.101288 │ -0.339846 │ 0.167170 │ 0.125895 │ -0.008983 │ 0.014724 │ 2.69 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 2 │ 1.0 │ -1.358354 │ -1.340163 │ 1.773209 │ 0.379780 │ -0.503198 │ 1.800499 │ 0.791461 │ 0.247676 │ -1.514654 │ ... │ 0.247998 │ 0.771679 │ 0.909412 │ -0.689281 │ -0.327642 │ -0.139097 │ -0.055353 │ -0.059752 │ 378.66 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 3 │ 1.0 │ -0.966272 │ -0.185226 │ 1.792993 │ -0.863291 │ -0.010309 │ 1.247203 │ 0.237609 │ 0.377436 │ -1.387024 │ ... │ -0.108300 │ 0.005274 │ -0.190321 │ -1.175575 │ 0.647376 │ -0.221929 │ 0.062723 │ 0.061458 │ 123.50 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 4 │ 2.0 │ -1.158233 │ 0.877737 │ 1.548718 │ 0.403034 │ -0.407193 │ 0.095921 │ 0.592941 │ -0.270533 │ 0.817739 │ ... │ -0.009431 │ 0.798278 │ -0.137458 │ 0.141267 │ -0.206010 │ 0.502292 │ 0.219422 │ 0.215153 │ 69.99 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 284802 │ 172786.0 │ -11.881118 │ 10.071785 │ -9.834783 │ -2.066656 │ -5.364473 │ -2.606837 │ -4.918215 │ 7.305334 │ 1.914428 │ ... │ 0.213454 │ 0.111864 │ 1.014480 │ -0.509348 │ 1.436807 │ 0.250034 │ 0.943651 │ 0.823731 │ 0.77 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 284803 │ 172787.0 │ -0.732789 │ -0.055080 │ 2.035030 │ -0.738589 │ 0.868229 │ 1.058415 │ 0.024330 │ 0.294869 │ 0.584800 │ ... │ 0.214205 │ 0.924384 │ 0.012463 │ -1.016226 │ -0.606624 │ -0.395255 │ 0.068472 │ -0.053527 │ 24.79 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 284804 │ 172788.0 │ 1.919565 │ -0.301254 │ -3.249640 │ -0.557828 │ 2.630515 │ 3.031260 │ -0.296827 │ 0.708417 │ 0.432454 │ ... │ 0.232045 │ 0.578229 │ -0.037501 │ 0.640134 │ 0.265745 │ -0.087371 │ 0.004455 │ -0.026561 │ 67.88 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 284805 │ 172788.0 │ -0.240440 │ 0.530483 │ 0.702510 │ 0.689799 │ -0.377961 │ 0.623708 │ -0.686180 │ 0.679145 │ 0.392087 │ ... │ 0.265245 │ 0.800049 │ -0.163298 │ 0.123205 │ -0.569159 │ 0.546668 │ 0.108821 │ 0.104533 │ 10.00 │ 0 ║

╟────────┼──────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼────────┼───────╢

║ 284806 │ 172792.0 │ -0.533413 │ -0.189733 │ 0.703337 │ -0.506271 │ -0.012546 │ -0.649617 │ 1.577006 │ -0.414650 │ 0.486180 │ ... │ 0.261057 │ 0.643078 │ 0.376777 │ 0.008797 │ -0.473649 │ -0.818267 │ -0.002415 │ 0.013649 │ 217.00 │ 0 ║

╚════════╧══════════╧════════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧════════╧═══════╝

The Time is measured in seconds since the first transaction in the data collection. As a result, we may infer that this dataset contains all transactions recorded during two days. The features were prepared using PCA, so the physical interpretation of individual features does not make sense. 'Time' and 'Amount' are the only features that are not transformed to PCA. 'Class' is the response variable, and it has a value of 1 if there is fraud and 0 otherwise.

## Data Exploration and Visualization

Now we try to find out the relative proportion of valid and fraudulent credit card transactions:

print("Fraudulent Cases: " + str(len(dataset[dataset["Class"] == 1])))

print("Valid Transactions: " + str(len(dataset[dataset["Class"] == 0])))

print("Proportion of Fraudulent Cases: " + str(len(dataset[dataset["Class"] == 1])/ dataset.shape[0]))

data\_p = dataset.copy()

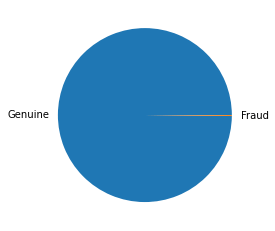
data\_p[" "] = np.where(data\_p["Class"] == 1 , "Fraud", "Genuine")

data\_p[" "].value\_counts().plot(kind="pie")

Fraudulent Cases: 492

Valid Transactions: 284315

Proportion of Fraudulent Cases: 0.001727485630620034



There is an imbalance in the data, with only 0.17% of the total cases being fraudulent.

Now we look at the distribution of the two named features in the dataset. For Time, it is clear that there was a particular duration in the day when most of the transactions took place:

f, axes = plt.subplots(1, 2, figsize=(18,4), sharex = True)

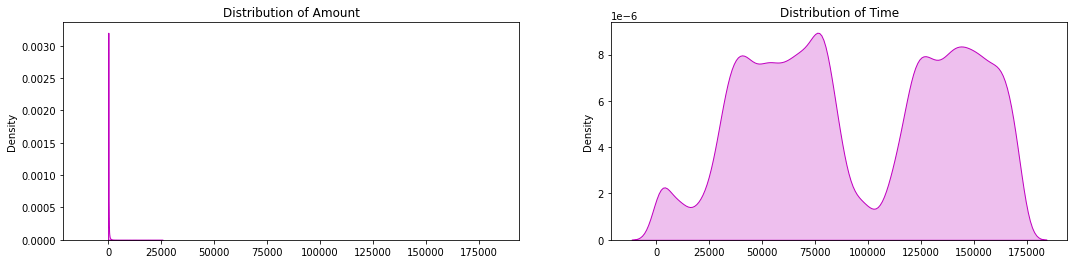
amount\_value = dataset['Amount'].values # values

time\_value = dataset['Time'].values # values

sns.distplot(amount\_value, hist=False, color="m", kde\_kws={"shade": True}, ax=axes[0]).set\_title('Distribution of Amount')

sns.distplot(time\_value, hist=False, color="m", kde\_kws={"shade": True}, ax=axes[1]).set\_title('Distribution of Time')

plt.show()



Let us check if there is any difference between valid transactions and fraudulent transactions:

print("Average Amount in a Fraudulent Transaction: " + str(dataset[dataset["Class"] == 1]["Amount"].mean()))

print("Average Amount in a Valid Transaction: " + str(dataset[dataset["Class"] == 0]["Amount"].mean()))

Average Amount in a Fraudulent Transaction: 122.21132113821133

Average Amount in a Valid Transaction: 88.29102242225574

As we can notice from this, the average money transaction for the fraudulent ones is more. It makes this problem crucial to deal with. Now let us try to understand the distribution of values in each feature. Let's start with the Amount:

print("Summary of the feature - Amount" + "\n-------------------------------")

print(dataset["Amount"].describe())

Summary of the feature - Amount

-------------------------------

count 284807.000000

mean 88.349619

std 250.120109

min 0.000000

25% 5.600000

50% 22.000000

75% 77.165000

max 25691.160000

Name: Amount, dtype: float64

The rest of the features don't have any physical interpretation and will be seen through histograms. Here the values are subgrouped according to class (valid or fraud):

data\_plot = dataset.copy()

amount = data\_plot['Amount']

data\_plot.drop(labels=['Amount'], axis=1, inplace = True)

data\_plot.insert(0, 'Amount', amount)

columns = data\_plot.iloc[:,0:30].columns

plt.figure(figsize=(12,30\*4))

grids = gridspec.GridSpec(30, 1)

for grid, index in enumerate(data\_plot[columns]):

ax = plt.subplot(grids[grid])

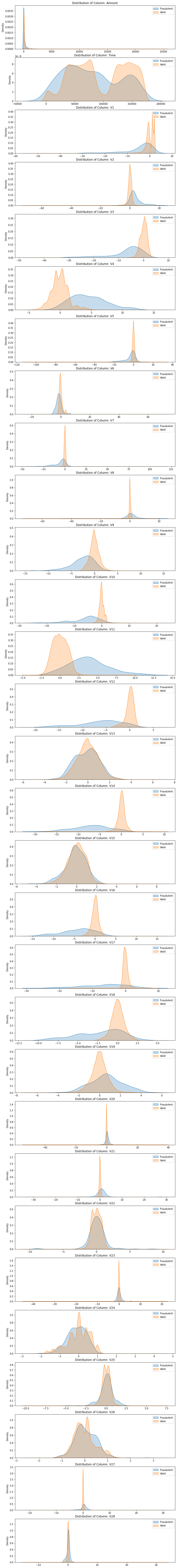
sns.distplot(data\_plot[index][data\_plot.Class == 1], hist=False, kde\_kws={"shade": True}, bins=50)

sns.distplot(data\_plot[index][data\_plot.Class == 0], hist=False, kde\_kws={"shade": True}, bins=50)

ax.set\_xlabel("")

ax.set\_title("Distribution of Column: " + str(index))

plt.show()



## Data Preparation

Since the features are created using PCA, feature selection is unnecessary as many features are tiny. Let's see if there are any missing values in the dataset:

dataset.isnull().shape[0]

print("Non-missing values: " + str(dataset.isnull().shape[0]))

print("Missing values: " + str(dataset.shape[0] - dataset.isnull().shape[0]))

Non-missing values: 284807

Missing values: 0

As there are no missing data, we turn to standardization. We standardize only Time and Amount using [RobustScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html" \o "RobustScaler" \t "_blank):

from sklearn.preprocessing import RobustScaler

scaler = RobustScaler().fit(dataset[["Time", "Amount"]])

dataset[["Time", "Amount"]] = scaler.transform(dataset[["Time", "Amount"]])

dataset.head().append(dataset.tail())

As we saw previously, the Amount column has outliers, that's why we chose RobustScaler() as it's robust to outliers. Output:

╔════════╤═══════════╤════════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════╗

║ │ Time │ V1 │ V2 │ V3 │ V4 │ V5 │ V6 │ V7 │ V8 │ V9 │ ... │ V21 │ V22 │ V23 │ V24 │ V25 │ V26 │ V27 │ V28 │ Amount │ Class ║

╠════════╪═══════════╪════════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════╣

║ 0 │ -0.994983 │ -1.359807 │ -0.072781 │ 2.536347 │ 1.378155 │ -0.338321 │ 0.462388 │ 0.239599 │ 0.098698 │ 0.363787 │ ... │ -0.018307 │ 0.277838 │ -0.110474 │ 0.066928 │ 0.128539 │ -0.189115 │ 0.133558 │ -0.021053 │ 1.783274 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 1 │ -0.994983 │ 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 │ -0.269825 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 2 │ -0.994972 │ -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 │ 4.983721 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 3 │ -0.994972 │ -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 │ 1.418291 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 4 │ -0.994960 │ -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 │ 0.670579 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 284802 │ 1.034951 │ -11.881118 │ 10.071785 │ -9.834783 │ -2.066656 │ -5.364473 │ -2.606837 │ -4.918215 │ 7.305334 │ 1.914428 │ ... │ 0.213454 │ 0.111864 │ 1.014480 │ -0.509348 │ 1.436807 │ 0.250034 │ 0.943651 │ 0.823731 │ -0.296653 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 284803 │ 1.034963 │ -0.732789 │ -0.055080 │ 2.035030 │ -0.738589 │ 0.868229 │ 1.058415 │ 0.024330 │ 0.294869 │ 0.584800 │ ... │ 0.214205 │ 0.924384 │ 0.012463 │ -1.016226 │ -0.606624 │ -0.395255 │ 0.068472 │ -0.053527 │ 0.038986 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 284804 │ 1.034975 │ 1.919565 │ -0.301254 │ -3.249640 │ -0.557828 │ 2.630515 │ 3.031260 │ -0.296827 │ 0.708417 │ 0.432454 │ ... │ 0.232045 │ 0.578229 │ -0.037501 │ 0.640134 │ 0.265745 │ -0.087371 │ 0.004455 │ -0.026561 │ 0.641096 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 284805 │ 1.034975 │ -0.240440 │ 0.530483 │ 0.702510 │ 0.689799 │ -0.377961 │ 0.623708 │ -0.686180 │ 0.679145 │ 0.392087 │ ... │ 0.265245 │ 0.800049 │ -0.163298 │ 0.123205 │ -0.569159 │ 0.546668 │ 0.108821 │ 0.104533 │ -0.167680 │ 0 ║

╟────────┼───────────┼────────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼─────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───────╢

║ 284806 │ 1.035022 │ -0.533413 │ -0.189733 │ 0.703337 │ -0.506271 │ -0.012546 │ -0.649617 │ 1.577006 │ -0.414650 │ 0.486180 │ ... │ 0.261057 │ 0.643078 │ 0.376777 │ 0.008797 │ -0.473649 │ -0.818267 │ -0.002415 │ 0.013649 │ 2.724796 │ 0 ║

╚════════╧═══════════╧════════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════╝

10 rows × 31 columns

Next, let's divide the data into features and targets. We also make the train-test split of the data:

y = dataset["Class"] # target

X = dataset.iloc[:,0:30]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size = 0.2, random\_state = 42)

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

Output:

((227845, 30), (56962, 30), (227845,), (56962,))

Let's import all the necessary libraries for the tutorial:

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import GridSearchCV, cross\_val\_score, RandomizedSearchCV

kf = StratifiedKFold(n\_splits=5, random\_state = None, shuffle = False)

from imblearn.pipeline import make\_pipeline ## Create a Pipeline using the provided estimators .

from imblearn.under\_sampling import NearMiss ## perform Under-sampling based on NearMiss methods.

from imblearn.over\_sampling import SMOTE ## PerformOver-sampling class that uses SMOTE.

from sklearn.metrics import roc\_curve, roc\_auc\_score, accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

Conclusion and Future Work (Phase 2)

# Project Conclusion:

In the Phase 2 conclusion: Recap Key Points: Stress the importance of fraud detection form financial security.Emphasize the Ongoing Need for Vigilance: Highlight that fraud prevention is an evolving process

Future Work: we need credit card fraud detection techniques**to protect the cardholders from false activity.** India is on its way to becoming a developed country. To achieve this, the Government of India (GoI) has launched several initiatives and one of these is Digital India Campaign. .