```
 # Importing the Dependencies

 from IPython import get_ipython
 from IPython.display import display

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset to pandas dataframe
credit_card_data = pd.read_csv('creditcard.csv')

# Read the first 5 rows of the datasets
credit_card_data.head()
```

{"type":"dataframe","variable_name":"credit_card_data"}

```
# Display the last 5 rows
credit_card_data.tail()
```

{"type":"dataframe"}

```
# Get information from dataset
credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
```

```
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

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

```
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
```

```
# Check the distribution of legit transaction and fraudent
transactions
credit_card_data['Class'].value_counts()
```

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

```
# Separating data for analysis
legit = credit_card_data[credit_card_data == 0]
fraud = credit_card_data[credit_card_data == 1]
```

```
# Print the shape of legit and fraud
print(legit.shape)
print(fraud.shape)
```

```
(284807, 31)
(284807, 31)
```

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

```
count    1825.0
mean        0.0
std         0.0
min         0.0
25%         0.0
50%         0.0
75%         0.0
max         0.0
Name: Amount, dtype: float64
```

```
# Get statistical measure for the data
fraud.Amount.describe()
```

```
count    13688.0
mean         1.0
std          0.0
min          1.0
25%          1.0
50%          1.0
75%          1.0
max          1.0
Name: Amount, dtype: float64
```

```
# Compare the values for both of these transactions
credit_card_data.groupby('Class').mean()
```

{"type":"dataframe"}

```
# Dealing with unbalanced data in the dataset
legit_sample = legit.sample(n=492)

# Concatenating two dataframes
fraud_sample = fraud.sample(n=492)
new_dataset = pd.concat([legit_sample,], axis = 0)

# Check the first 5 rows of the new dataframe
new_dataset.head()
```

```
{"type":"dataframe","variable_name":"new_dataset"}
```

```
# Display the last five rows of the new dataset
new_dataset.tail()
```

```
{"type":"dataframe"}
```

```
# Check the value count for the new dataset
new_dataset['Class'].value_counts()
```

```
Class
0.0    489
Name: count, dtype: int64
```

```
fraud_sample = fraud.sample(n=492)
new_dataset = pd.concat([legit_sample, fraud_sample], axis = 0)
new_dataset.groupby('Class').max()
```

```
{"type":"dataframe"}
```

```
# Splitting data into features and targets
X = new_dataset.drop(columns = 'Class', axis = 0)
Y = new_dataset['Class']

print(X)
```

```
        Time  V1  V2  V3  V4  V5  V6  V7  V8  ...  V21  V22  V23  V24
V25  V26  V27  V28  Amount
10895    NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
39821    NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
11708    NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
250355   NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
107287   NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
...      ...  ..  ..  ..  ..  ..  ..  ..  ..  ...  ...  ...  ...  ...
...  ...  ...  ...     ...
868      NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
```

```
249887   NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
27387    NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
172322   NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN
87045    NaN NaN NaN NaN NaN NaN NaN NaN NaN  ...  NaN  NaN  NaN  NaN
NaN  NaN  NaN  NaN     NaN

[984 rows x 30 columns]
```

```
print(Y)
```

```
10895     0.0
39821     0.0
11708     0.0
250355    0.0
107287    0.0
          ...
868       NaN
249887    NaN
27387     NaN
172322    NaN
87045     NaN
Name: Class, Length: 984, dtype: float64
```

```
# Handling Nan values in the 'class' column
new_dataset.dropna(subset=['Class'],inplace=True)

# Split sata into training and testing data
X = new_dataset.drop(columns= 'Class', axis = 1)
Y = new_dataset['Class']
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size
=0.2, stratify = Y, random_state = 2)

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

```
(492, 30) (393, 30) (99, 30)
```

```
# Model training
model = LogisticRegression()

# Handling missing values
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy = 'mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/impute/_base.py:635:
UserWarning: Skipping features without any observed values: ['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']. At least one non-missing value is needed
for imputation with strategy='mean'.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/impute/_base.py:635:
UserWarning: Skipping features without any observed values: ['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']. At least one non-missing value is needed
for imputation with strategy='mean'.
  warnings.warn(
```

```
# Trainng the Logistic Regression model with training data
model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
# Model Evaluation
model_score = model.score(X_train, Y_train)
X_train_prediction = model.predict(X_train)
from sklearn.metrics import accuracy_score
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print(f"Accuracy on training datasset: {training_data_accuracy}")
```

```
Accuracy on training datasset: 0.9949109414758269
```

```
# Accuracy_score for testind data
model_score = model.score(X_test, Y_test)
X_test_prediction = model.predict(X_test)
from sklearn.metrics import accuracy_score
testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print(f"Accuracy on testing dataset: {testing_data_accuracy}")
```

```
Accuracy on testing dataset: 0.98989898989899
```