Are you in Cyber Security Department ?
This Project is for you.
Follow me

→

Prediction of this Project

Not Fraud Transaction **Fraud Transaction**

Approaching of this Project

↓

↓

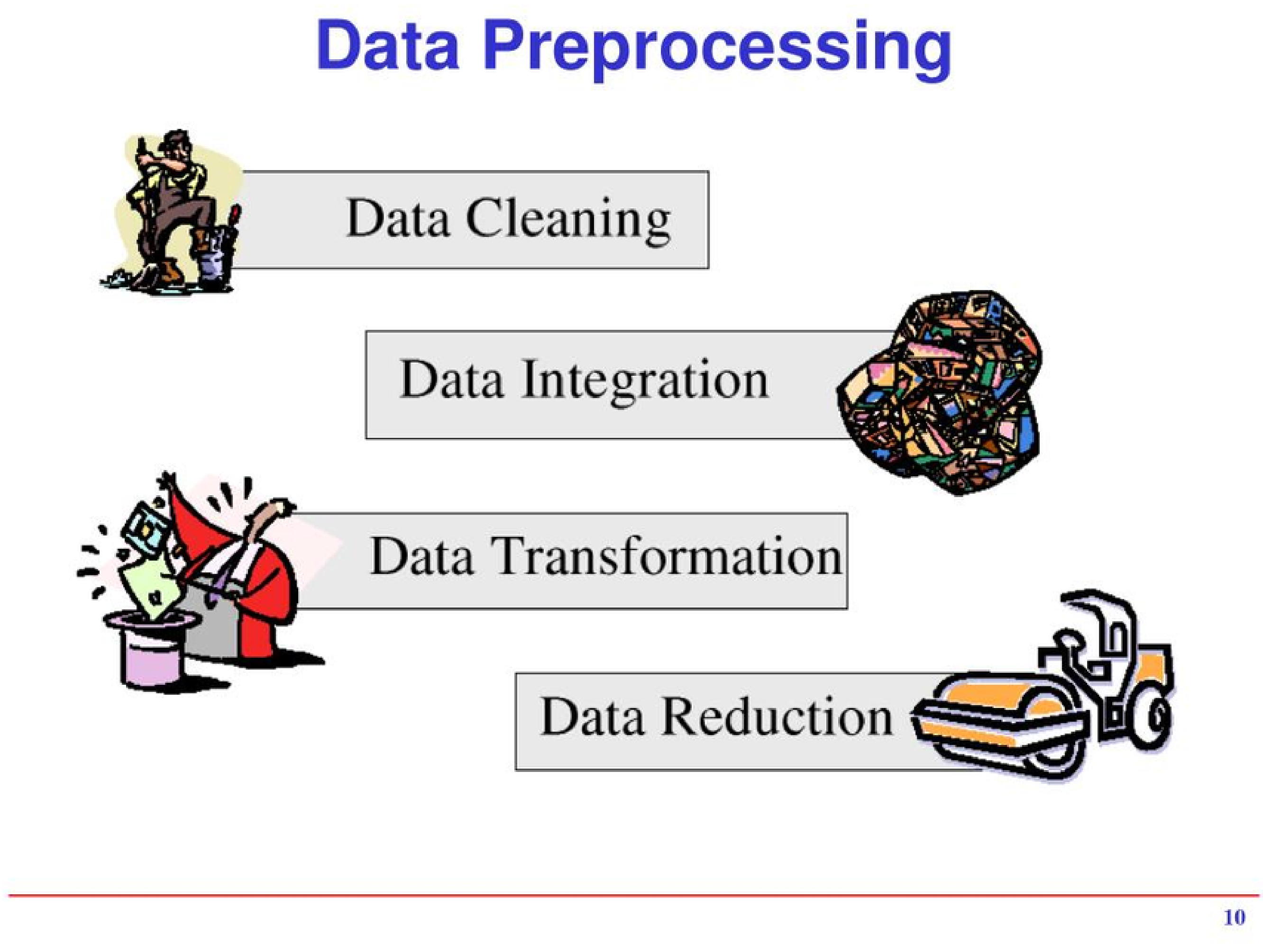
- The output is clear and Classification type(Yes or No).
- So its comes under Machine Learning -> Supervised learning -> Classification.

Phases of this Project

- Data Preprocessing
- Balance the Dataset
- Feature Selection
- Exploratory Data Analysis
- Model Creation and Evaluation
- Model Deployment

Let's move to Coding Part

1.Data Preprocessing



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier

#Read the Dataset

# Loading the data
df = pd.read_csv('ccfd.csv')
df
```

```
        User  Card  Year  Month  Day   Amount            UseChip  \
0          1     1  2005      9    6   $16.68  Swipe Transaction
1          1     1  2005      9    9  $224.70 Online Transaction
2          1     1  2005      9    9  $145.61 Online Transaction
3          1     1  2005      9    9  $229.21  Swipe Transaction
4          1     1  2005      9    9   $11.00  Swipe Transaction
...      ...   ...   ...    ...  ...      ...                ...
691915  1999     4  2019      4   13   $52.42   Chip Transaction
691916  1999     4  2019      4   15    $7.57   Chip Transaction
691917  1999     4  2019      4   15    $7.27   Chip Transaction
691918  1999     4  2019      4   17    $5.39   Chip Transaction
691919  1999     4  2019      4   24   $15.59   Chip Transaction

                          MerchantName   MerchantCity MerchantCountry
\
0                   Ross Package Store        Berkley             USA

1            Digital Delivery Company 7      San Jose             NaN

2              Travel Booking Company 4      San Jose             NaN

3                  Car Rental Company 4    Belleville             USA

4                  Supermarket Chain 3     Southfield             USA

...                                ...            ...             ...

691915             Wright Beauticians  East Elmhurst             USA

691916            Bookstore Company 1        Elmhurst             USA

691917            Supermarket Chain 1        Elmhurst             USA

691918                 Acme Souvenirs  East Elmhurst             USA

691919  Fox East Elmhurst Car Cleaners  East Elmhurst             USA


            Zip   MCC IssFraud?
0       48072.0  5921        No
```

```
1            NaN  4899        No
2            NaN  4722        No
3        48111.0  3405        No
4        48075.0  5411        No
...          ...   ...       ...
691915  11370.0  7230        No
691916  11373.0  5942        No
691917  11373.0  5411        No
691918  11370.0  5947        No
691919  11370.0  7542        No

[691920 rows x 13 columns]
```

```
#User- column for user id details
#Card-Column for card number
#Year- column for year of transaction
#Month-column for month of transaction
#Day-column for Day of transaction
#Amount-for how much amount transacted
#Use Chip-for transaction is based on online or swipe transaction
#Merchant name- Name of the merchant in the transaction
#Merchant city-Merchant city name in the transaction
#Merchant state-Merchant state name in the transaction
#Zip-Postal code of the merchant area
#MCC-It is a four number pin given by bank for each card
df.columns
```

```
Index(['User', 'Card', 'Year', 'Month', 'Day', 'Amount', 'UseChip',
       'MerchantName', 'MerchantCity', 'MerchantCountry', 'Zip',
'MCC',
       'IssFraud?'],
      dtype='object')
```

```
df.isnull().sum()
```

```
User                   0
Card                   0
Year                   0
Month                  0
Day                    0
Amount                 0
UseChip                0
MerchantName           0
MerchantCity           0
MerchantCountry    73784
Zip                77856
MCC                    0
IssFraud?              0
dtype: int64
```

```
df.head()
```

```
   User  Card  Year  Month  Day   Amount              UseChip  \
0     1     1  2005      9    6   $16.68   Swipe Transaction
1     1     1  2005      9    9  $224.70  Online Transaction
2     1     1  2005      9    9  $145.61  Online Transaction
3     1     1  2005      9    9  $229.21   Swipe Transaction
4     1     1  2005      9    9   $11.00   Swipe Transaction

                 MerchantName MerchantCity MerchantCountry      Zip
MCC  \
0          Ross Package Store      Berkley             USA  48072.0
5921
1  Digital Delivery Company 7    San Jose              NaN      NaN
4899
2    Travel Booking Company 4    San Jose              NaN      NaN
4722
3        Car Rental Company 4   Belleville             USA  48111.0
3405
4          Supermarket Chain 3   Southfield             USA  48075.0
5411

  IssFraud?
0        No
1        No
2        No
3        No
4        No
```

```
df['IssFraud?'].value_counts()
```

```
IssFraud?
No     691048
Yes       872
Name: count, dtype: int64
```

```
independent=df[['User', 'Card', 'Year', 'Month', 'Day','UseChip',
       'MerchantName', 'MerchantCity', 'MerchantCountry', 'Zip',
'MCC']]

dependent=df[['IssFraud?']]
print(independent)
```

```
        User  Card  Year  Month  Day             UseChip  \
0          1     1  2005      9    6   Swipe Transaction
1          1     1  2005      9    9  Online Transaction
2          1     1  2005      9    9  Online Transaction
3          1     1  2005      9    9   Swipe Transaction
4          1     1  2005      9    9   Swipe Transaction
...      ...   ...   ...    ...  ...                 ...
691915  1999     4  2019      4   13    Chip Transaction
```

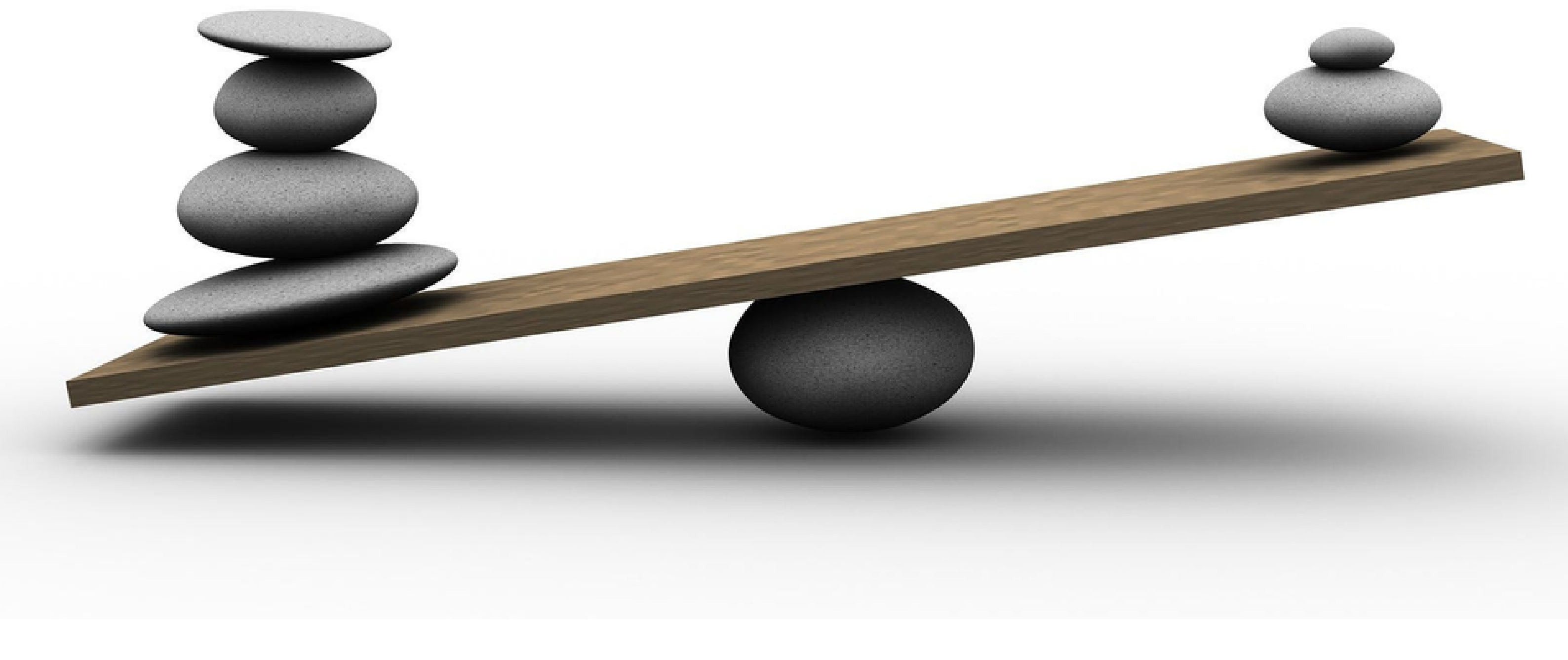
```
691916  1999     4  2019      4   15    Chip Transaction
691917  1999     4  2019      4   15    Chip Transaction
691918  1999     4  2019      4   17    Chip Transaction
691919  1999     4  2019      4   24    Chip Transaction

                          MerchantName   MerchantCity MerchantCountry
\
0                   Ross Package Store        Berkley             USA

1           Digital Delivery Company 7      San Jose              NaN

2             Travel Booking Company 4      San Jose              NaN

3                Car Rental Company 4     Belleville              USA

4                 Supermarket Chain 3     Southfield              USA

...                                ...            ...             ...

691915             Wright Beauticians  East Elmhurst              USA

691916            Bookstore Company 1       Elmhurst              USA

691917            Supermarket Chain 1       Elmhurst              USA

691918                Acme Souvenirs  East Elmhurst              USA

691919  Fox East Elmhurst Car Cleaners  East Elmhurst              USA


            Zip   MCC
0       48072.0  5921
1           NaN  4899
2           NaN  4722
3       48111.0  3405
4       48075.0  5411
...         ...   ...
691915  11370.0  7230
691916  11373.0  5942
691917  11373.0  5411
691918  11370.0  5947
691919  11370.0  7542

[691920 rows x 11 columns]
```

```
def quanQual(df):
    quan=[]
    qual=[]
    for columnName in df.columns:
    #print(columnName)
        if(df[columnName].dtypes=='O'):
```

```
        #print("qual")
            qual.append(columnName)
        else:
        #print("quan")
             quan.append(columnName)
    return quan,qual

quan,qual=quanQual(df)

quan

['User', 'Card', 'Year', 'Month', 'Day', 'Zip', 'MCC']

qual

['Amount',
 'UseChip',
 'MerchantName',
 'MerchantCity',
 'MerchantCountry',
 'IssFraud?']

import numpy as np
from sklearn.impute import SimpleImputer
imp=SimpleImputer(missing_values=np.nan,strategy="mean",copy=True)
imp.fit(df[quan])
datan=imp.transform(df[quan])

datan

array([[1.00000000e+00, 1.00000000e+00, 2.00500000e+03, ...,
        6.00000000e+00, 4.80720000e+04, 5.92100000e+03],
       [1.00000000e+00, 1.00000000e+00, 2.00500000e+03, ...,
        9.00000000e+00, 5.16946769e+04, 4.89900000e+03],
       [1.00000000e+00, 1.00000000e+00, 2.00500000e+03, ...,
        9.00000000e+00, 5.16946769e+04, 4.72200000e+03],
       ...,
       [1.99900000e+03, 4.00000000e+00, 2.01900000e+03, ...,
        1.50000000e+01, 1.13730000e+04, 5.41100000e+03],
       [1.99900000e+03, 4.00000000e+00, 2.01900000e+03, ...,
        1.70000000e+01, 1.13700000e+04, 5.94700000e+03],
       [1.99900000e+03, 4.00000000e+00, 2.01900000e+03, ...,
        2.40000000e+01, 1.13700000e+04, 7.54200000e+03]])

datan=pd.DataFrame(datan,columns=quan)

import numpy as np
from sklearn.impute import SimpleImputer
imp=SimpleImputer(missing_values=np.nan,strategy="most_frequent")
imp.fit(df[qual])
datal=imp.transform(df[qual])
```

```
datal

array([['$16.68', 'Swipe Transaction', 'Ross Package Store',
'Berkley',
        'USA', 'No'],
       ['$224.70', 'Online Transaction', 'Digital Delivery Company 7',
        'San Jose ', 'USA', 'No'],
       ['$145.61', 'Online Transaction', 'Travel Booking Company 4',
        'San Jose ', 'USA', 'No'],
       ...,
       ['$7.27', 'Chip Transaction', 'Supermarket Chain 1',
'Elmhurst',
        'USA', 'No'],
       ['$5.39', 'Chip Transaction', 'Acme Souvenirs', 'East
Elmhurst',
        'USA', 'No'],
       ['$15.59', 'Chip Transaction', 'Fox East Elmhurst Car
Cleaners',
        'East Elmhurst', 'USA', 'No']], dtype=object)

datal=pd.DataFrame(datal,columns=qual)

df=pd.concat([datan,datal],axis=1)

csv=df.to_csv("Preprocessed_credit_card_detection.csv",index=False)

csv
```

2 . Balancing the Dataset



```python
import pandas as pd

df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df
```

```
          User  Card    Year  Month   Day           Zip     MCC
Amount  \
0          1.0   1.0  2005.0    9.0   6.0  48072.000000  5921.0
$16.68
1          1.0   1.0  2005.0    9.0   9.0  51694.676895  4899.0
$224.70
2          1.0   1.0  2005.0    9.0   9.0  51694.676895  4722.0
$145.61
3          1.0   1.0  2005.0    9.0   9.0  48111.000000  3405.0
$229.21
4          1.0   1.0  2005.0    9.0   9.0  48075.000000  5411.0
$11.00
...        ...   ...     ...    ...   ...           ...     ...      .
..
691915  1999.0   4.0  2019.0    4.0  13.0  11370.000000  7230.0
$52.42
691916  1999.0   4.0  2019.0    4.0  15.0  11373.000000  5942.0
$7.57
691917  1999.0   4.0  2019.0    4.0  15.0  11373.000000  5411.0
$7.27
691918  1999.0   4.0  2019.0    4.0  17.0  11370.000000  5947.0
$5.39
691919  1999.0   4.0  2019.0    4.0  24.0  11370.000000  7542.0
$15.59

                   UseChip                   MerchantName
MerchantCity  \
0         Swipe Transaction             Ross Package Store
Berkley
1        Online Transaction      Digital Delivery Company 7      San
Jose
2        Online Transaction        Travel Booking Company 4      San
Jose
3         Swipe Transaction          Car Rental Company 4
Belleville
4         Swipe Transaction            Supermarket Chain 3
Southfield
...                     ...                             ...
...
691915     Chip Transaction             Wright Beauticians  East
Elmhurst
691916     Chip Transaction            Bookstore Company 1
Elmhurst
691917     Chip Transaction            Supermarket Chain 1
Elmhurst
```

```
691918    Chip Transaction                  Acme Souvenirs  East
Elmhurst
691919    Chip Transaction  Fox East Elmhurst Car Cleaners  East
Elmhurst

       MerchantCountry IssFraud?
0                  USA        No
1                  USA        No
2                  USA        No
3                  USA        No
4                  USA        No
...                ...       ...
691915             USA        No
691916             USA        No
691917             USA        No
691918             USA        No
691919             USA        No

[691920 rows x 13 columns]
```

```
df.columns
```

```
Index(['User', 'Card', 'Year', 'Month', 'Day', 'Zip', 'MCC', 'Amount',
       'UseChip', 'MerchantName', 'MerchantCity', 'MerchantCountry',
       'IssFraud?'],
      dtype='object')
```

```
df["IssFraud?"].value_counts()
```

```
IssFraud?
No     691048
Yes       872
Name: count, dtype: int64
```

```
df.isnull().sum()
```

```
User               0
Card               0
Year               0
Month              0
Day                0
Zip                0
MCC                0
Amount             0
UseChip            0
MerchantName       0
MerchantCity       0
MerchantCountry    0
IssFraud?          0
dtype: int64
```

```python
independent=df[['User', 'Card', 'Year', 'Month', 'Day', 'Amount',
'UseChip',
       'MerchantName', 'MerchantCity', 'MerchantCountry', 'Zip',
'MCC']]
independent
```

```
          User  Card    Year  Month   Day   Amount            UseChip
\
0          1.0   1.0  2005.0    9.0   6.0   $16.68   Swipe Transaction

1          1.0   1.0  2005.0    9.0   9.0  $224.70  Online Transaction

2          1.0   1.0  2005.0    9.0   9.0  $145.61  Online Transaction

3          1.0   1.0  2005.0    9.0   9.0  $229.21   Swipe Transaction

4          1.0   1.0  2005.0    9.0   9.0   $11.00   Swipe Transaction

...        ...   ...     ...    ...   ...      ...                 ...

691915  1999.0   4.0  2019.0    4.0  13.0   $52.42    Chip Transaction

691916  1999.0   4.0  2019.0    4.0  15.0    $7.57    Chip Transaction

691917  1999.0   4.0  2019.0    4.0  15.0    $7.27    Chip Transaction

691918  1999.0   4.0  2019.0    4.0  17.0    $5.39    Chip Transaction

691919  1999.0   4.0  2019.0    4.0  24.0   $15.59    Chip Transaction


                     MerchantName   MerchantCity MerchantCountry
\
0              Ross Package Store        Berkley             USA

1         Digital Delivery Company 7       San Jose             USA

2          Travel Booking Company 4       San Jose             USA

3             Car Rental Company 4     Belleville             USA

4              Supermarket Chain 3     Southfield             USA

...                           ...            ...             ...

691915          Wright Beauticians  East Elmhurst             USA

691916           Bookstore Company 1       Elmhurst             USA

691917           Supermarket Chain 1       Elmhurst             USA
```

```
691918                     Acme Souvenirs  East Elmhurst             USA

691919  Fox East Elmhurst Car Cleaners  East Elmhurst             USA


                 Zip     MCC
0       48072.000000  5921.0
1       51694.676895  4899.0
2       51694.676895  4722.0
3       48111.000000  3405.0
4       48075.000000  5411.0
...              ...     ...
691915  11370.000000  7230.0
691916  11373.000000  5942.0
691917  11373.000000  5411.0
691918  11370.000000  5947.0
691919  11370.000000  7542.0

[691920 rows x 12 columns]
```

```
dependent=df[['IssFraud?']]
dependent
```

```
       IssFraud?
0             No
1             No
2             No
3             No
4             No
...          ...
691915        No
691916        No
691917        No
691918        No
691919        No

[691920 rows x 1 columns]
```

```
from imblearn.under_sampling import RandomUnderSampler
ros=RandomUnderSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(independent,dependent)
```

```
x_ros.shape
```

```
(1744, 12)
```

```
y_ros.value_counts()
```

```
IssFraud?
No             872
```

```
Yes          872
Name: count, dtype: int64

x_ros

          User  Card    Year  Month   Day   Amount              UseChip
\
609890  1750.0   0.0  2015.0    7.0  16.0   $21.42    Swipe Transaction

677647  1959.0   1.0  2016.0    5.0   5.0   $76.99     Chip Transaction

59562    182.0   2.0  2012.0   11.0  23.0    $2.19    Swipe Transaction

155077   458.0   2.0  2019.0    5.0  15.0   $45.73     Chip Transaction

674259  1949.0   0.0  2018.0    1.0   4.0    $1.25     Chip Transaction

...        ...   ...     ...    ...   ...      ...                  ...

691161  1998.0   2.0  2013.0    1.0  26.0  $193.24    Swipe Transaction

691871  1999.0   3.0  2020.0    1.0  26.0  $221.96    Swipe Transaction

691872  1999.0   3.0  2020.0    1.0  26.0   $26.69    Swipe Transaction

691873  1999.0   3.0  2020.0    1.0  26.0  $103.95     Chip Transaction

691874  1999.0   3.0  2020.0    1.0  26.0    $0.24   Online Transaction


                                  MerchantName MerchantCity
MerchantCountry  \
609890                         Lukass Theaters       Tiffin
USA
677647                        Jadens Wholesale    Lancaster
USA
59562                     Supermarket Chain 3      Houston
USA
155077                    Supermarket Chain 3        Flint
USA
674259              Convenience Store Chain 1     Brooklyn
USA
...                                        ...          ...
...
691161  Neufelder Tegucigalpa Wine and Liquor  Tegucigalpa
Honduras
691871                        Abrils Wholesale  Saint Louis
USA
691872                        Abrils Wholesale  Saint Louis
USA
691873             Cox Saint Louis Restaurant  Saint Louis
```

```
USA
691874              Digital Content Company 2    San Jose
USA

                 Zip     MCC
609890  44883.000000  7832.0
677647  93535.000000  5300.0
59562   77096.000000  5411.0
155077  48532.000000  5411.0
674259  11213.000000  5499.0
...              ...     ...
691161  51694.676895  5921.0
691871  63146.000000  5300.0
691872  63146.000000  5300.0
691873  63146.000000  5812.0
691874  51694.676895  5815.0

[1744 rows x 12 columns]

# Convert the undersampled arrays to a Pandas DataFrame
undersampled_data = pd.DataFrame(data=x_ros,
columns=independent.columns)

# Add the 'dependent' variable as a new column to the DataFrame
undersampled_data['target'] = y_ros

# Replace 'undersampled_data.csv' with the desired filename
output_file = 'undersampled_data.csv'

# Save the DataFrame to a new CSV file
undersampled_data.to_csv(output_file, index=False)

print(f"Undersampled data has been saved to {output_file}.")

Undersampled data has been saved to undersampled_data.csv.
```

3.Feature Selction

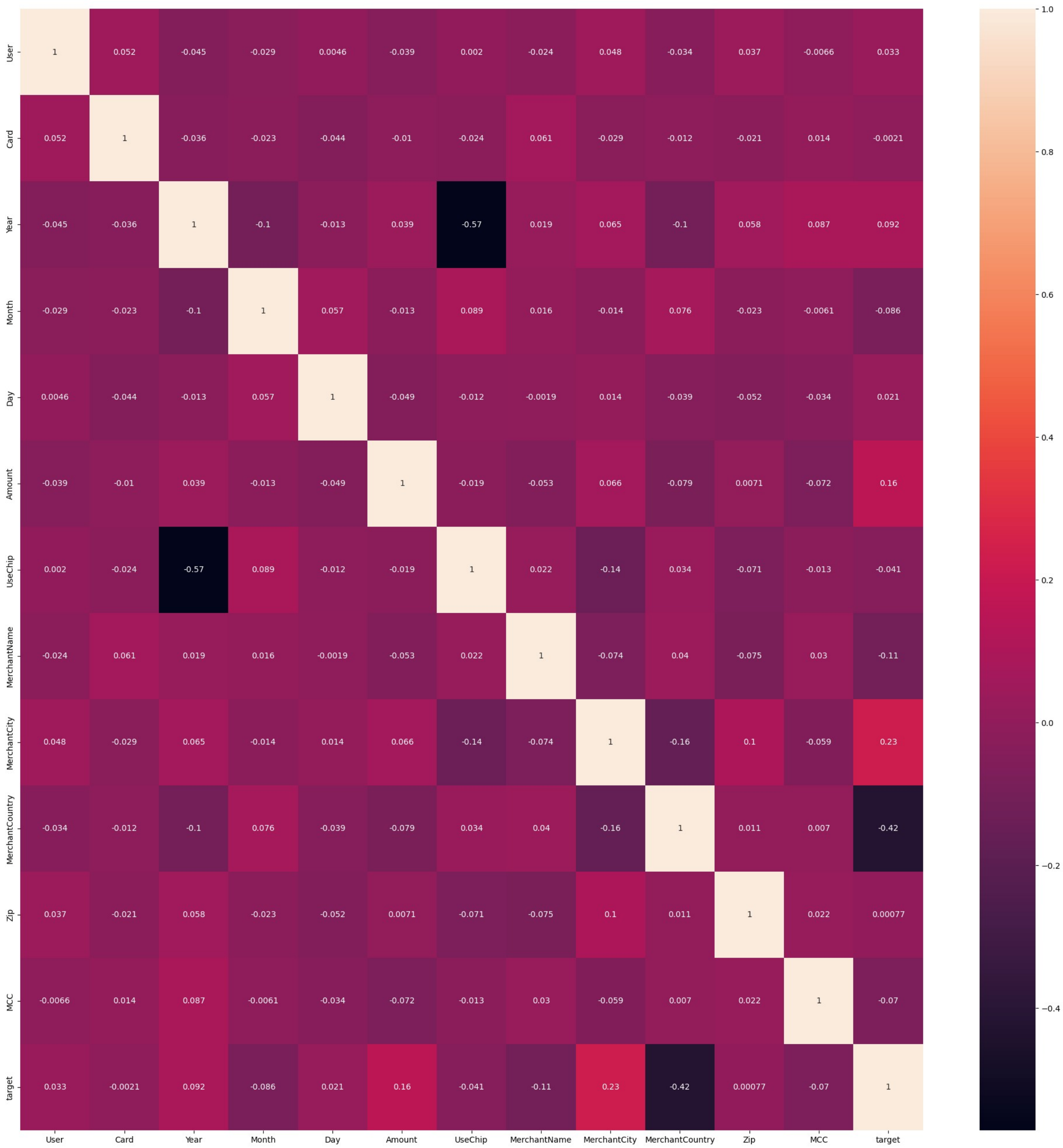
```python
import pandas as pd

df=pd.read_csv("undersampled_data.csv")
df
```

```
        User  Card    Year  Month   Day   Amount  
UseChip  \
0     1750.0   0.0  2015.0    7.0  16.0   $21.42   Swipe Transaction

1     1959.0   1.0  2016.0    5.0   5.0   $76.99    Chip Transaction

2      182.0   2.0  2012.0   11.0  23.0    $2.19   Swipe Transaction

3      458.0   2.0  2019.0    5.0  15.0   $45.73    Chip Transaction

4     1949.0   0.0  2018.0    1.0   4.0    $1.25    Chip Transaction

...      ...   ...     ...    ...   ...      ...                 ...

1739  1998.0   2.0  2013.0    1.0  26.0  $193.24   Swipe Transaction

1740  1999.0   3.0  2020.0    1.0  26.0  $221.96   Swipe Transaction

1741  1999.0   3.0  2020.0    1.0  26.0   $26.69   Swipe Transaction

1742  1999.0   3.0  2020.0    1.0  26.0  $103.95    Chip Transaction

1743  1999.0   3.0  2020.0    1.0  26.0    $0.24  Online Transaction


                               MerchantName MerchantCity 
MerchantCountry  \
0                           Lukass Theaters       Tiffin 
USA
1                         Jadens Wholesale    Lancaster 
USA
2                      Supermarket Chain 3      Houston 
USA
3                      Supermarket Chain 3        Flint 
USA
4                Convenience Store Chain 1     Brooklyn 
USA
...                                     ...          ...             .
..
1739  Neufelder Tegucigalpa Wine and Liquor  Tegucigalpa 
Honduras
1740                       Abrils Wholesale  Saint Louis 
USA
1741                       Abrils Wholesale  Saint Louis 
USA
1742             Cox Saint Louis Restaurant  Saint Louis
```

```
USA
1743                 Digital Content Company 2    San Jose
USA

                  Zip     MCC target
0     44883.000000  7832.0     No
1     93535.000000  5300.0     No
2     77096.000000  5411.0     No
3     48532.000000  5411.0     No
4     11213.000000  5499.0     No
...            ...     ...    ...
1739  51694.676895  5921.0    Yes
1740  63146.000000  5300.0    Yes
1741  63146.000000  5300.0    Yes
1742  63146.000000  5812.0    Yes
1743  51694.676895  5815.0    Yes

[1744 rows x 13 columns]

df['Amount'] = df['Amount'].replace({'\$': '', ',': ''},
regex=True).astype(float)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
col = ['UseChip', 'MerchantName', 'MerchantCity',
'MerchantCountry','target']
for i in col:
    df[i] = le.fit_transform(df[i]).astype(int)

independent=df[['User', 'Card', 'Year', 'Month',
'Day','UseChip','Amount',
         'MerchantName', 'MerchantCity', 'MerchantCountry', 'Zip',
'MCC']]

dependent=df[['target']]
independent

         User  Card    Year  Month   Day  UseChip  Amount  MerchantName
\
0     1750.0   0.0  2015.0    7.0  16.0        2   21.42           383

1     1959.0   1.0  2016.0    5.0   5.0        0   76.99           318

2      182.0   2.0  2012.0   11.0  23.0        2    2.19           529

3      458.0   2.0  2019.0    5.0  15.0        0   45.73           529

4     1949.0   0.0  2018.0    1.0   4.0        0    1.25           153

...      ...   ...     ...    ...   ...      ...     ...           ...
```

```
1739  1998.0   2.0  2013.0    1.0  26.0        2  193.24            422

1740  1999.0   3.0  2020.0    1.0  26.0        2  221.96             10

1741  1999.0   3.0  2020.0    1.0  26.0        2   26.69             10

1742  1999.0   3.0  2020.0    1.0  26.0        0  103.95            160

1743  1999.0   3.0  2020.0    1.0  26.0        1    0.24            193


      MerchantCity  MerchantCountry           Zip     MCC
0              522               15  44883.000000  7832.0
1              270               15  93535.000000  5300.0
2              230               15  77096.000000  5411.0
3              171               15  48532.000000  5411.0
4               68               15  11213.000000  5499.0
...            ...              ...           ...     ...
1739           520                6  51694.676895  5921.0
1740           470               15  63146.000000  5300.0
1741           470               15  63146.000000  5300.0
1742           470               15  63146.000000  5812.0
1743           479               15  51694.676895  5815.0

[1744 rows x 12 columns]
```

```python
import seaborn as sns
import matplotlib.pyplot as plt
fig,ax=plt.subplots(figsize=(25,25))
sns.heatmap(df.corr(),annot=True,ax=ax)
plt.show()
```




```
cor=df.corr()
cor_target=abs(cor["target"])
relevant_features=cor_target[cor_target>0.06]
print(relevant_features)
```

```
Year                   0.091520
Month                  0.086224
Amount                 0.162054
MerchantName           0.107166
```

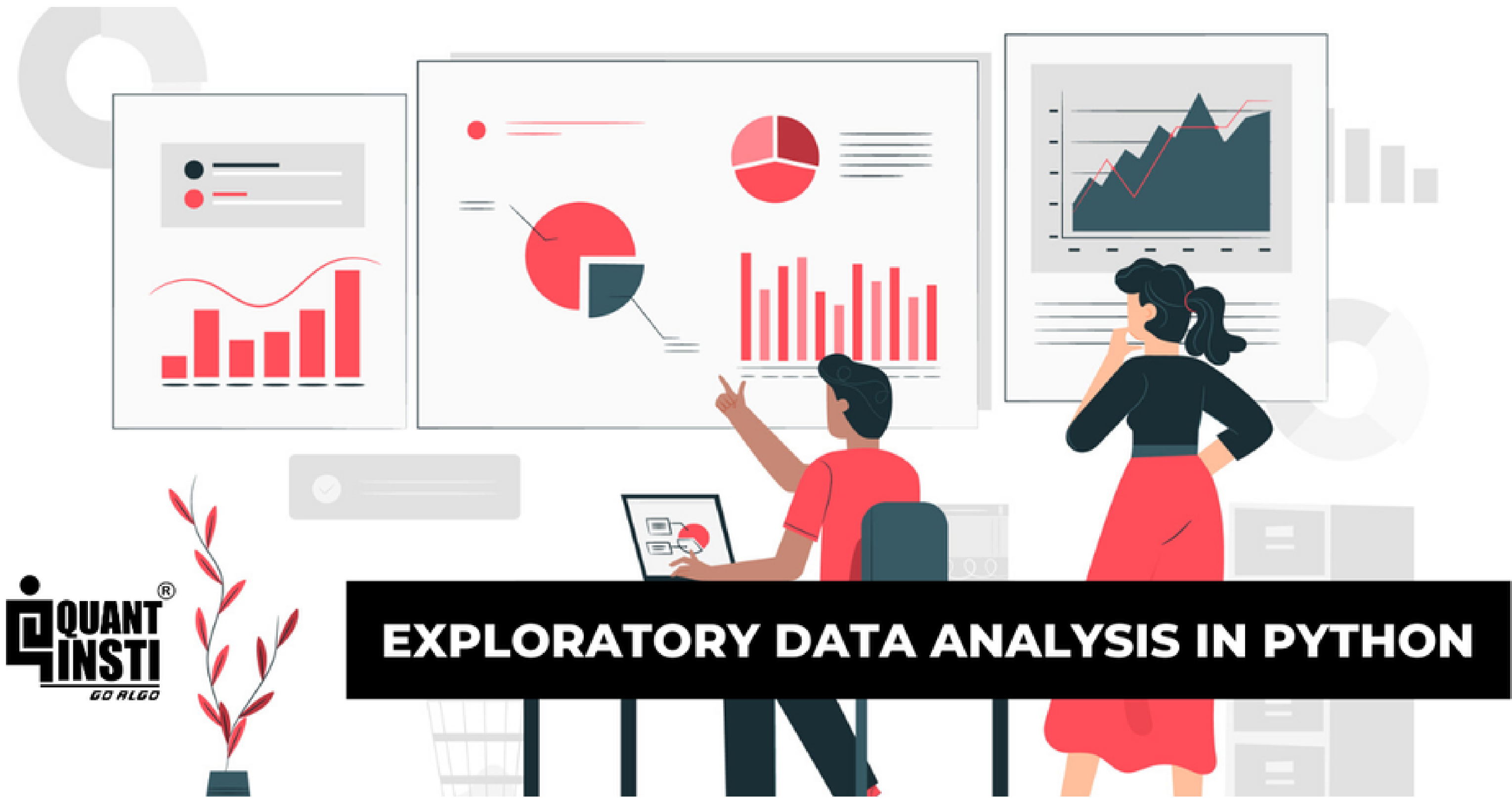
```
MerchantCity         0.225082
MerchantCountry      0.420328
MCC                  0.070231
target               1.000000
Name: target, dtype: float64
```

```python
from sklearn.ensemble import ExtraTreesClassifier
import numpy as np

# Create the ExtraTreesClassifier model
model = ExtraTreesClassifier()

# Fit the model on your data
model.fit(independent, dependent)

# Get feature importances
importances = model.feature_importances_

# Sort the feature importances in descending order
sorted_indices = np.argsort(importances)[::-1]

# Print feature importances
for i, index in enumerate(sorted_indices):
    print(f"{i + 1}. Feature: {independent.columns[index]} - 
Importance: {importances[index]}")
```

```
E:\Anaconda\envs\card\lib\site-packages\sklearn\base.py:1151: 
DataConversionWarning: A column-vector y was passed when a 1d array 
was expected. Please change the shape of y to (n_samples,), for 
example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
1. Feature: MerchantCountry - Importance: 0.22055205114447016
2. Feature: Zip - Importance: 0.17938351894628904
3. Feature: MerchantCity - Importance: 0.11271882120638689
4. Feature: Year - Importance: 0.09060643211823145
5. Feature: MerchantName - Importance: 0.07081793565354642
6. Feature: MCC - Importance: 0.06806538966894225
7. Feature: UseChip - Importance: 0.06695707451490819
8. Feature: Amount - Importance: 0.04481856946478957
9. Feature: Month - Importance: 0.04472053728454379
10. Feature: Day - Importance: 0.03746341392337656
11. Feature: User - Importance: 0.03603032410471112
12. Feature: Card - Importance: 0.02786593196980452
```

4.Exploratory Data Analysis



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')

df=pd.read_csv("undersampled_data.csv")
df
```

```
        User  Card    Year  Month   Day   Amount
UseChip  \
0     1750.0   0.0  2015.0    7.0  16.0   $21.42   Swipe Transaction

1     1959.0   1.0  2016.0    5.0   5.0   $76.99    Chip Transaction

2      182.0   2.0  2012.0   11.0  23.0    $2.19   Swipe Transaction

3      458.0   2.0  2019.0    5.0  15.0   $45.73    Chip Transaction

4     1949.0   0.0  2018.0    1.0   4.0    $1.25    Chip Transaction

...      ...   ...     ...    ...   ...      ...                 ...

1739  1998.0   2.0  2013.0    1.0  26.0  $193.24   Swipe Transaction

1740  1999.0   3.0  2020.0    1.0  26.0  $221.96   Swipe Transaction

1741  1999.0   3.0  2020.0    1.0  26.0   $26.69   Swipe Transaction

1742  1999.0   3.0  2020.0    1.0  26.0  $103.95    Chip Transaction

1743  1999.0   3.0  2020.0    1.0  26.0    $0.24  Online Transaction


                               MerchantName MerchantCity
MerchantCountry  \
0                           Lukass Theaters       Tiffin
USA
1                          Jadens Wholesale    Lancaster
USA
2                       Supermarket Chain 3      Houston
USA
3                       Supermarket Chain 3        Flint
USA
4                 Convenience Store Chain 1     Brooklyn
USA
...                                     ...          ...              .
..
1739  Neufelder Tegucigalpa Wine and Liquor  Tegucigalpa
Honduras
```

```
1740                         Abrils Wholesale  Saint Louis
USA
1741                         Abrils Wholesale  Saint Louis
USA
1742               Cox Saint Louis Restaurant  Saint Louis
USA
1743                Digital Content Company 2    San Jose
USA

                Zip     MCC target
0     44883.000000  7832.0     No
1     93535.000000  5300.0     No
2     77096.000000  5411.0     No
3     48532.000000  5411.0     No
4     11213.000000  5499.0     No
...            ...     ...    ...
1739  51694.676895  5921.0    Yes
1740  63146.000000  5300.0    Yes
1741  63146.000000  5300.0    Yes
1742  63146.000000  5812.0    Yes
1743  51694.676895  5815.0    Yes

[1744 rows x 13 columns]
```

```
unique_country = df['User'].unique()

print("Unique country Names:")
print(unique_country)
```

```
Unique country Names:
[1750. 1959.  182.  458. 1949.  899.  168.  282. 1167.  362.  591.
1113.
  732. 1075. 1216.   95.   56.  141. 1135. 1195. 1129.  220. 1933.
1664.
 1752.  475.   70. 1567.  693.  816.  920. 1804.  706. 1744. 1450.
740.
   47.  358.  323.  401. 1333.  175.  545.  101.   38. 1202.  913.
180.
   35.  839. 1304.  261.  575. 1696.  813.  435.  197.   24.  385.
389.
 1682. 1594.  611. 1458.  705.  648.  574.  974.  244. 1335.  300.
508.
 1474. 1006. 1278.   37.  478. 1138.  751.  956. 1913. 1079. 1547.
1491.
  433. 1196. 1876.  531. 1287.  772. 1247.  864. 1385. 1183.  124.
188.
 1739.   82.  150.  143.  255. 1603. 1783. 1931. 1638.  885. 1610.
132.
 1980. 1721.  311. 1666. 1150.  617.  370.  718. 1325. 1358.  111.
1895.
```

```
 1579.  662.  630. 1961.  374.  961. 1078. 1763. 1487.  387.  336.
1574.
 1765. 1930.  291. 1382.   21. 1439. 1520. 1611.  292.  440. 1233.
126.
 1746.  279. 1861. 1643. 1633. 1517. 1502.  327. 1046. 1616. 1874.
1925.
 1628. 1395. 1307.  927. 1475. 1306. 1351.  973. 1751. 1560. 1037.
1062.
 1663.  903.  429.  759. 1781. 1040. 1889. 1832.   29.  699.  667.
1034.
 1327.  276.   85.  694. 1028.  477. 1285.  501. 1119. 1418. 1571.
1399.
  985. 1073. 1988.  558. 1676. 1488.  675. 1106. 1417.  436. 1114.
1934.
 1599.  896. 1059.  785. 1007.  689.  193.   50. 1246. 1846. 1707.
1740.
  589. 1725. 1554. 1348. 1483.  324.  765. 1718.  878.   76.   77.
821.
 1656.  238.  502.  755.  870.  658.  941. 1625. 1806. 1108. 1044.
1715.
  684. 1071.  450. 1724.  727. 1501.  254.  783.   25.  714. 1817.
1447.
 1000. 1816. 1294. 1237.  247.   74.  970. 1023.  735.    5. 1137.
804.
 1321.   28. 1972. 1346.  695.  812. 1406. 1672.   69.  259. 1695.
1123.
  348.  624. 1016. 1459.  861. 1842.  392. 1383.  810.  642.  103.
1170.
  713.  618. 1270. 1729.  570. 1772. 1801.  731.  996.  457.  862.
309.
  553.  275.  204. 1983. 1126. 1394. 1956. 1937. 1444. 1837. 1636.
1826.
 1834.  463.  438. 1180.  214. 1253.  397. 1537.  388. 1565. 1243.
472.
 1486. 1629. 1118. 1206.  495. 1808.  583. 1014. 1288.  404. 1070.
757.
  656.  660. 1928. 1156. 1514. 1622.  722. 1478. 1236. 1184. 1557.
167.
 1935.  943. 1026. 1767.  540. 1127. 1974.    2.  408.  750.  550.
1207.
  983. 1702. 1662.  278. 1562.  380. 1039. 1749. 1300. 1690.  371.
474.
 1777.  745.  464. 1815.  517.  733. 1191.  873.  717.   27. 1423.
55.
  846. 1878. 1066.  681.  641. 1163.  106.  314.  882.  485.  687.
1856.
 1542. 1597. 1986. 1665.  847. 1704. 1701. 1124.  493.  281. 1363.
950.
  246. 1297.  547. 1887. 1519.  760.  698. 1015. 1674. 1035. 1096.
```

```
1942.
  376.  483. 1590. 1250. 1541.  778.  747. 1457.  480.  466.  696.
468.
 1190. 1888. 1408.  968.  568.  992. 1640. 1265. 1402. 1606. 1844.
777.
  191. 1737. 1005.  425.  136.  361.  919.  189. 1862. 1263. 1891.
1596.
  122. 1072. 1945. 1426.  290. 1604.  516. 1967. 1645. 1355. 1735.
73.
  886.  482. 1694. 1614.  525. 1464.  936.  270.  993. 1173. 1164.
386.
 1513. 1799.  442. 1429.  288.  954. 1229.  721. 1125.  640.  720.
152.
 1258. 1291.  312. 1580. 1228.  345. 1499.  960.  744.  773.  352.
1132.
  771.  889.  216. 1523. 1659.  417. 1220.  289. 1796.  581.  922.
94.
 1556. 1318. 1907.  677.  526. 1872. 1218. 1098.  339.  533. 1273.
1388.
 1598.  500.  567. 1168.   52.  229.  791. 1769. 1753.  549.  151.
410.
 1723.  486.  945.  131. 1051. 1647.  170. 1998.  454. 1159.  848.
22.
  551. 1479.  161. 1591.   20. 1380. 1211. 1425.  606. 1330.  673.
629.
  840. 1699.  962. 1431. 1578. 1549. 1471. 1372. 1885. 1404. 1635.
409.
 1929. 1840.   58.   97.  195.  253.   11. 1343. 1080.  372. 1239.
90.
   81.  285.  585.  269. 1693. 1821. 1061.  995. 1076.  424.  492.
584.
   80.  597.  488.  586. 1010.  686. 1210.  895. 1688. 1532.  420.
391.
 1054.  133. 1366.  692. 1624. 1602.  555. 1245.  530. 1370.  395.
905.
 1792. 1286.  284. 1377. 1779.  923. 1269.  271.  564. 1583. 1852.
61.
  691. 1867.  171.    3.    6.   13.   14.   16.   17.   18.   39.
44.
   53.   54.   60.   66.  112.  119.  123.  127.  144.  146.  164.
207.
  228.  243.  245.  264.  332.  337.  353.  377.  383.  393.  400.
443.
  453.  462.  481.  496.  498.  503.  506.  510.  524.  536.  544.
559.
  576.  582.  588.  590.  603.  620.  639.  645.  652.  682.  711.
754.
  756.  763.  767.  768.  782.  806.  819.  822.  831.  832.  835.
838.
```

```
  845.  860.  865.  874.  875.  879.  880.  921.  925.  928.  953.
969.
  984.  994.  998. 1019. 1027. 1036. 1057. 1105. 1112. 1130. 1134.
1157.
 1175. 1189. 1199. 1201. 1214. 1226. 1240. 1241. 1268. 1289. 1301.
1312.
 1314. 1320. 1331. 1337. 1340. 1344. 1345. 1365. 1369. 1390. 1391.
1405.
 1416. 1421. 1430. 1436. 1445. 1448. 1456. 1472. 1477. 1480. 1505.
1518.
 1528. 1568. 1569. 1612. 1618. 1630. 1634. 1646. 1673. 1683. 1728.
1732.
 1734. 1736. 1742. 1754. 1757. 1770. 1778. 1795. 1803. 1819. 1829.
1833.
 1836. 1851. 1853. 1886. 1910. 1922. 1927. 1936. 1938. 1939. 1953.
1954.
 1958. 1965. 1984. 1990. 1995. 1997. 1999.]
```

```
df['Amount'] = df['Amount'].replace({'\$': '', ',': ''},
regex=True).astype(float)
```

```
df["target"].value_counts()
```

```
target
No     872
Yes    872
Name: count, dtype: int64
```

```
df.isnull().sum()
```

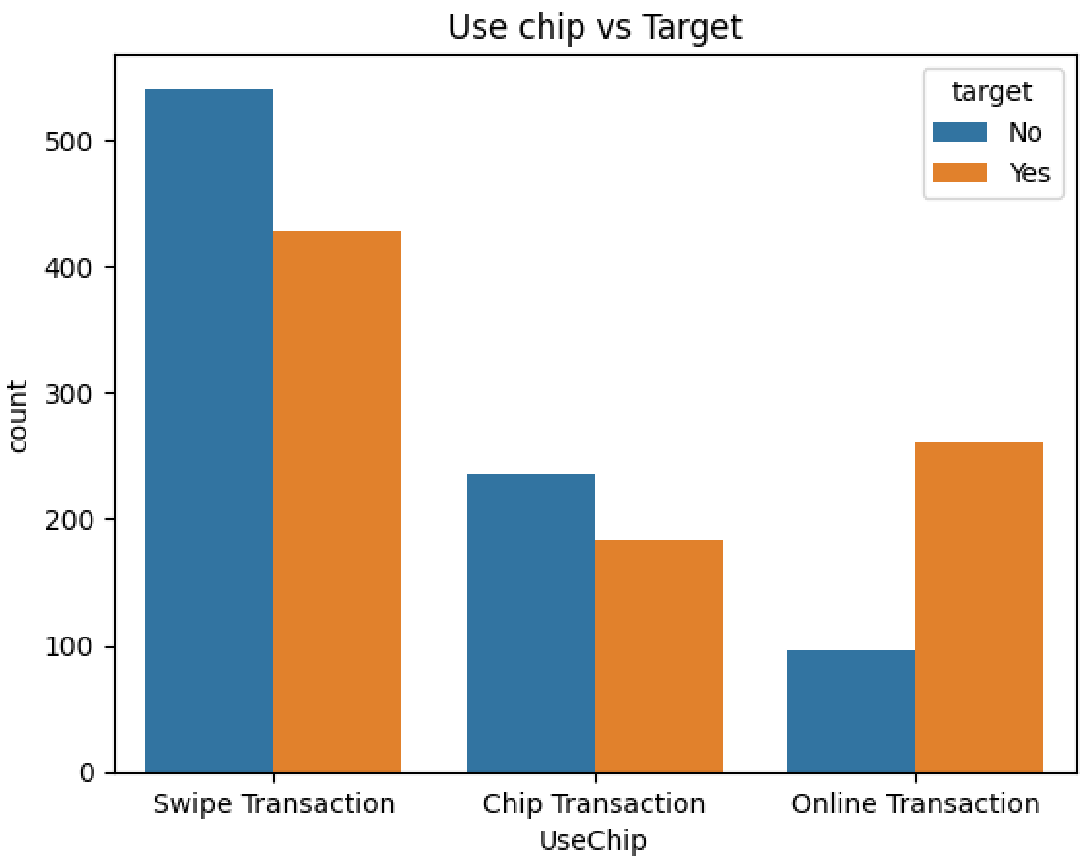
```
User               0
Card               0
Year               0
Month              0
Day                0
Amount             0
UseChip            0
MerchantName       0
MerchantCity       0
MerchantCountry    0
Zip                0
MCC                0
target             0
dtype: int64
```

```
import pandas as pd

# Create a cross-tabulation table
cross_tab = pd.crosstab(df['UseChip'], df['target'])
```

```
print("Cross-tabulation Table:")
print(cross_tab)
```

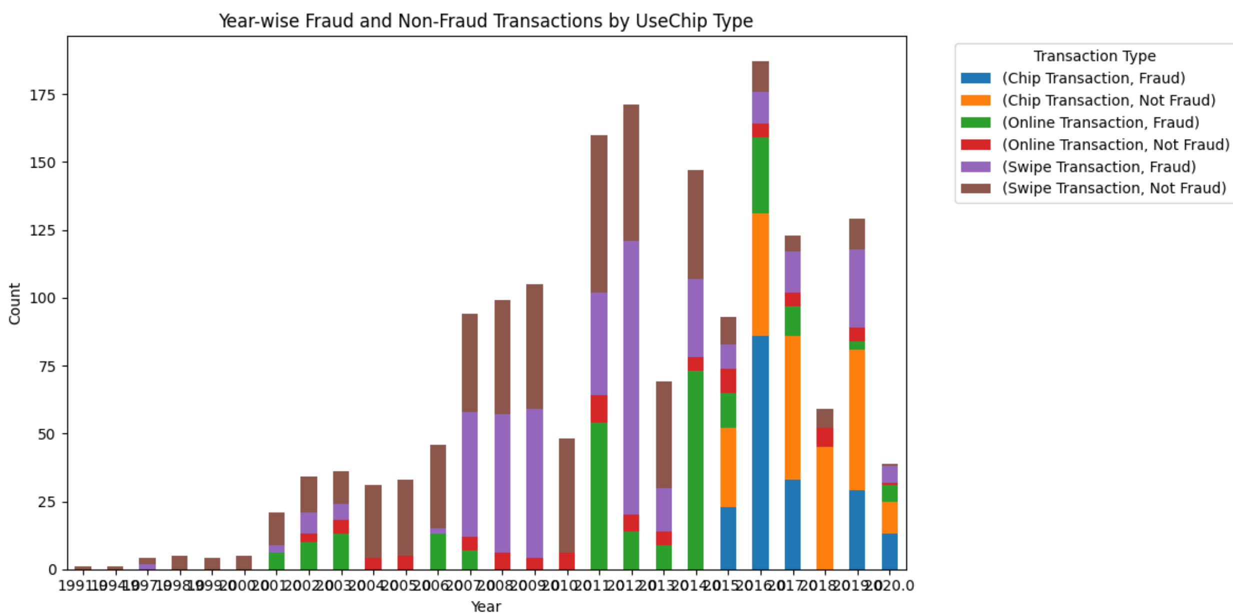
```
Cross-tabulation Table:
target                No  Yes
UseChip
Chip Transaction     236  184
Online Transaction    96  260
Swipe Transaction    540  428
```

```
sns.countplot(x='UseChip', hue='target', data=df)
plt.title("Use chip vs Target")
plt.show()
```

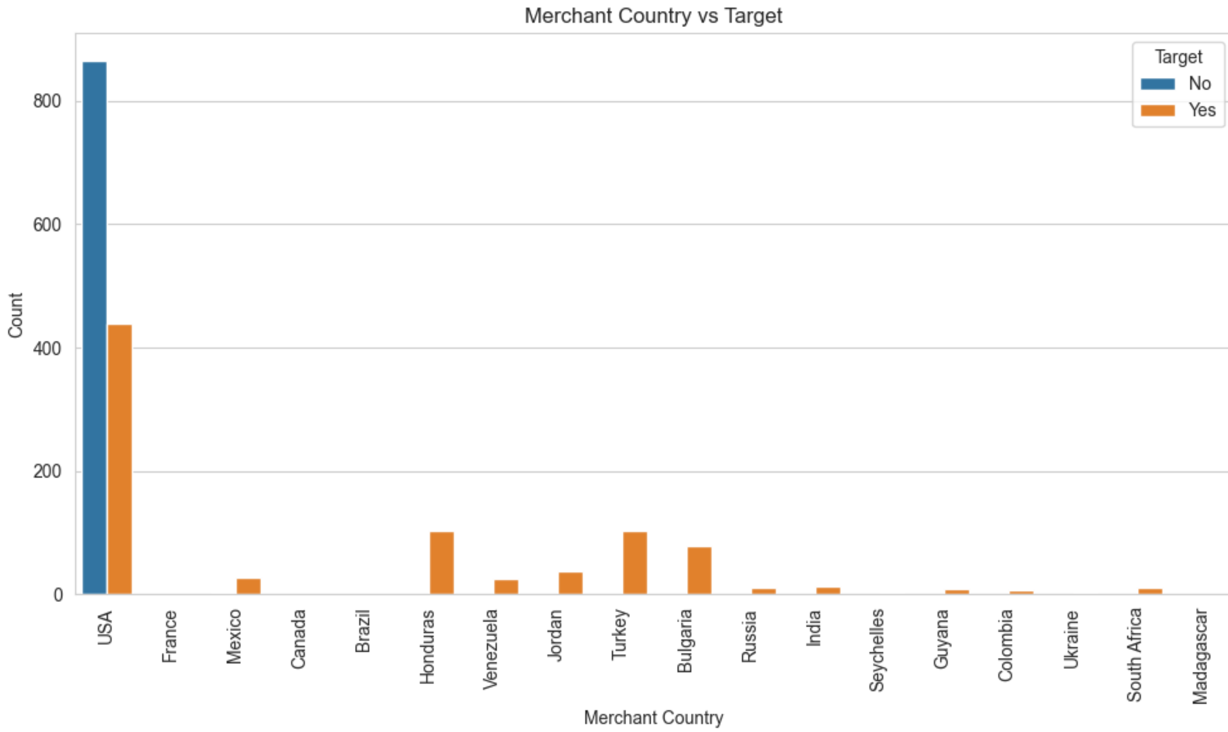



```
df['FraudStatus'] = df['target'].apply(lambda x: 'Fraud' if x == 'Yes' 
else 'Not Fraud')

grouped_data = df.groupby(['Year', 'UseChip', 
'FraudStatus']).size().reset_index(name='Count')

pivot_data = grouped_data.pivot_table(index='Year', 
columns=['UseChip', 'FraudStatus'], values='Count', fill_value=0)
```

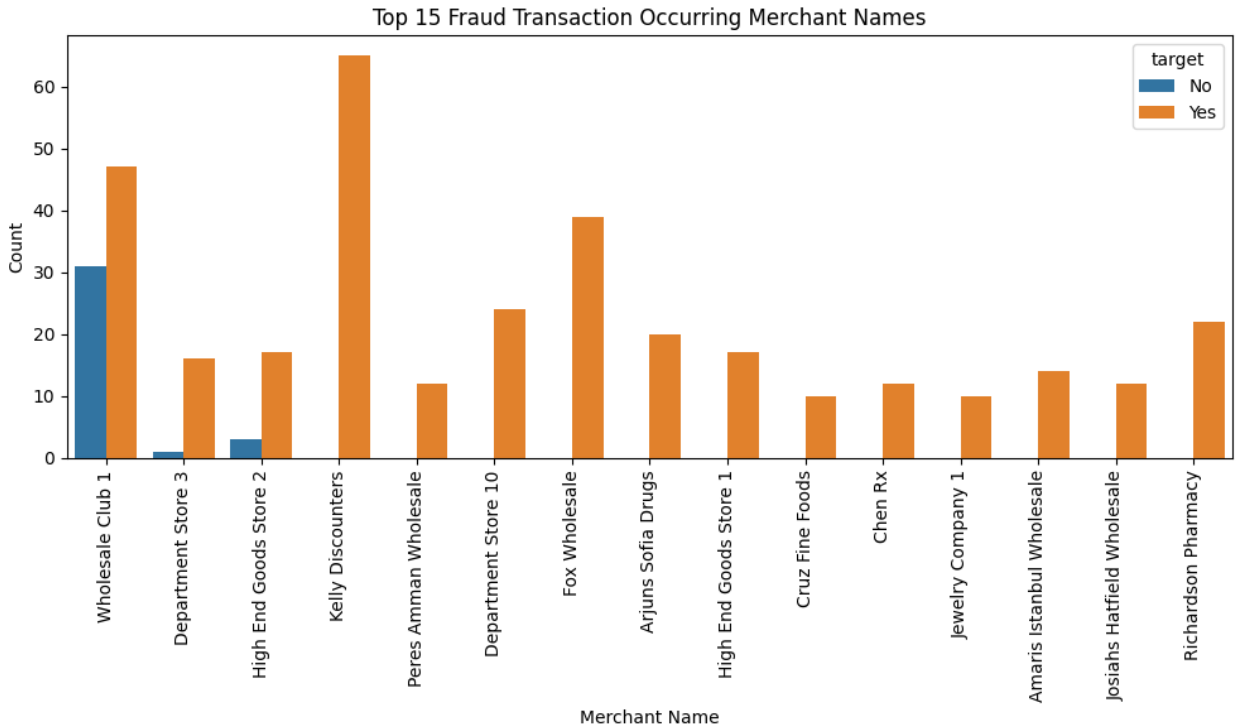
```python
pivot_data.plot(kind='bar', stacked=True, figsize=(12, 6))

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('Year-wise Fraud and Non-Fraud Transactions by UseChip
Type')

# Show the plot
plt.legend(title='Transaction Type', bbox_to_anchor=(1.05, 1),
loc='upper left')
plt.xticks(rotation=0)

# Display the plot
plt.tight_layout()
plt.show()
```




```python
plt.figure(figsize=(10, 6))
sns.countplot(x='MerchantCountry', hue='target', data=df)
plt.title("Merchant Country vs Target")
plt.xlabel("Merchant Country")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.legend(title='Target', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```




```
fraud_df = df[df['target'] == 'Yes']

# Get the top 10 fraud transaction-occurring merchant names
top_10_fraud_merchants =
fraud_df['MerchantName'].value_counts().head(15).index.tolist()

# Filter the original DataFrame to include only the top 10 fraud
merchants
df_top_10_fraud = df[df['MerchantName'].isin(top_10_fraud_merchants)]

plt.figure(figsize=(10, 6))
sns.countplot(x='MerchantName', hue='target', data=df_top_10_fraud)
plt.title("Top 15 Fraud Transaction Occurring Merchant Names")
plt.xlabel("Merchant Name")
plt.ylabel("Count")
plt.xticks(rotation=90)
#plt.legend(title='Target', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```



Top 15 Fraud Transaction Occurring Merchant Names
target
No
Yes
Count
0
10
20
30
40
50
60
Wholesale Club 1
Department Store 3
High End Goods Store 2
Kelly Discounters
Peres Amman Wholesale
Department Store 10
Fox Wholesale
Arjuns Sofia Drugs
High End Goods Store 1
Cruz Fine Foods
Chen Rx
Jewelry Company 1
Amaris Istanbul Wholesale
Josiahs Hatfield Wholesale
Richardson Pharmacy
Merchant Name

5. Model Creation and Model Evaluation

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

import warnings
warnings.filterwarnings('ignore')

df=pd.read_csv("undersampled_data.csv")
df
```

```
        User  Card    Year  Month   Day   Amount
UseChip  \
0     1750.0   0.0  2015.0    7.0  16.0   $21.42   Swipe Transaction

1     1959.0   1.0  2016.0    5.0   5.0   $76.99    Chip Transaction

2      182.0   2.0  2012.0   11.0  23.0    $2.19   Swipe Transaction

3      458.0   2.0  2019.0    5.0  15.0   $45.73    Chip Transaction

4     1949.0   0.0  2018.0    1.0   4.0    $1.25    Chip Transaction

...      ...   ...     ...    ...   ...      ...                 ...

1739  1998.0   2.0  2013.0    1.0  26.0  $193.24   Swipe Transaction

1740  1999.0   3.0  2020.0    1.0  26.0  $221.96   Swipe Transaction

1741  1999.0   3.0  2020.0    1.0  26.0   $26.69   Swipe Transaction

1742  1999.0   3.0  2020.0    1.0  26.0  $103.95    Chip Transaction

1743  1999.0   3.0  2020.0    1.0  26.0    $0.24  Online Transaction


                               MerchantName MerchantCity
MerchantCountry  \
0                           Lukass Theaters       Tiffin
USA
1                          Jadens Wholesale    Lancaster
USA
2                      Supermarket Chain 3      Houston
USA
3                      Supermarket Chain 3        Flint
USA
4                Convenience Store Chain 1     Brooklyn
USA
...                                     ...          ...             .
..
1739  Neufelder Tegucigalpa Wine and Liquor  Tegucigalpa
```

```
Honduras
1740                       Abrils Wholesale  Saint Louis
USA
1741                       Abrils Wholesale  Saint Louis
USA
1742             Cox Saint Louis Restaurant  Saint Louis
USA
1743              Digital Content Company 2    San Jose
USA

               Zip     MCC target
0     44883.000000  7832.0     No
1     93535.000000  5300.0     No
2     77096.000000  5411.0     No
3     48532.000000  5411.0     No
4     11213.000000  5499.0     No
...            ...     ...    ...
1739  51694.676895  5921.0    Yes
1740  63146.000000  5300.0    Yes
1741  63146.000000  5300.0    Yes
1742  63146.000000  5812.0    Yes
1743  51694.676895  5815.0    Yes

[1744 rows x 13 columns]
```

```
df["target"].value_counts()
```

```
target
No     872
Yes    872
Name: count, dtype: int64
```

```
df.isnull().sum()
```

```
User              0
Card              0
Year              0
Month             0
Day               0
Amount            0
UseChip           0
MerchantName      0
MerchantCity      0
MerchantCountry   0
Zip               0
MCC               0
target            0
dtype: int64
```

```
df['Amount'] = df['Amount'].replace({'\$': '', ',': ''},
regex=True).astype(float)
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
col = ['UseChip', 'MerchantName', 'MerchantCity',
'MerchantCountry','target']
for i in col:
    df[i] = le.fit_transform(df[i]).astype(int)

independent=df[['Year', 'Month', 'UseChip','Amount',
       'MerchantName', 'MerchantCity', 'MerchantCountry','MCC']]

dependent=df[['target']]
independent
```

```
        Year  Month  UseChip  Amount  MerchantName  MerchantCity  \
0     2015.0    7.0        2   21.42           383           522
1     2016.0    5.0        0   76.99           318           270
2     2012.0   11.0        2    2.19           529           230
3     2019.0    5.0        0   45.73           529           171
4     2018.0    1.0        0    1.25           153            68
...      ...    ...      ...     ...           ...           ...
1739  2013.0    1.0        2  193.24           422           520
1740  2020.0    1.0        2  221.96            10           470
1741  2020.0    1.0        2   26.69            10           470
1742  2020.0    1.0        0  103.95           160           470
1743  2020.0    1.0        1    0.24           193           479

      MerchantCountry     MCC
0                  15  7832.0
1                  15  5300.0
2                  15  5411.0
3                  15  5411.0
4                  15  5499.0
...               ...     ...
1739                6  5921.0
1740               15  5300.0
1741               15  5300.0
1742               15  5812.0
1743               15  5815.0

[1744 rows x 8 columns]
```

```
#split into training set and test
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(independent,dependent,t
est_size=1/3,random_state=42)

X_test.shape
```

```
(582, 8)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators= 100,
criterion="entropy")
classifier=classifier.fit(X_train,Y_train.values.ravel())

y_pred=classifier.predict(X_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
0,
       0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
1,
       1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
1,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
0,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
1,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
0,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0,
0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
1,
       0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0,
1,
```

```
       0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
1,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
1,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0])
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test.values.ravel(),y_pred)

cm
```

```
array([[276,  14],
       [ 24, 268]], dtype=int64)
```

```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test.values.ravel(),y_pred)

print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       290
           1       0.95      0.92      0.93       292

    accuracy                           0.93       582
   macro avg       0.94      0.93      0.93       582
weighted avg       0.94      0.93      0.93       582
```

```
Year=int(input())
Month=int(input())
UseChip=int(input())
Amount=int(input())
MerchantName=int(input())
MerchantCity=int(input())
MerchantCountry=int(input())
mcc=int(input())
```

```
 67
 80
 8
 79
 8096
 7889
```

```
 8900
 5637
```

```
future_prediction=classifier.predict([[Year,Month,UseChip,Amount,MerchantName,MerchantCity,MerchantCountry,mcc]])
future_prediction
```

```
array([0])
```

```
import joblib

# Assuming you have already trained a model named 'model'
# Save the model to a file
joblib.dump(classifier, 'frauddetection.pkl')
```

```
['frauddetection.pkl']
```

6.Model Deployment

```python
import joblib
import gradio as gr
from pydantic import BaseModel

# 1. Load the trained model
model = joblib.load('frauddetection.pkl')

# 2. Define the input data schema using Pydantic BaseModel
class InputData(BaseModel):
    Year: int
    Month: int
    UseChip: int
    Amount: int
    MerchantName: int
    MerchantCity: int
    MerchantCountry: int
    mcc: int
    # Add the rest of the input features (feature4, feature5, ..., feature12)

# 3. Define the prediction function
def predict(year, month, use_chip, amount, merchant_name, merchant_city,
merchant_country, mcc):
    # Perform the prediction using the loaded model
    prediction = model.predict([[year, month, use_chip, amount, merchant_name,
merchant_city, merchant_country, mcc]])[0]  # Replace ... with the rest of the
features

    # Convert the prediction to a string (or any other format you prefer)
    result = "Fraud" if prediction == 1 else "Not a Fraud"

    return result

# 4. Create a Gradio interface
iface = gr.Interface(
    fn=predict,
    inputs=[
        gr.inputs.Number(label="Year"),
        gr.inputs.Number(label="Month"),
        gr.inputs.Number(label="UseChip"),
        gr.inputs.Number(label="Amount"),
        gr.inputs.Number(label="MerchantName"),
        gr.inputs.Number(label="MerchantCity"),
        gr.inputs.Number(label="MerchantCountry"),
        gr.inputs.Number(label="mcc"),
        # Add the rest of the input features as individual Gradio input components
    ],
    outputs=gr.outputs.Textbox(),
)

# 5. Launch the Gradio interface
iface.launch()
```

Want to Know more about Machine learning Project .

Don't forget to follow me on Linkedin