# Bank_Personal_Loan_Modelling

## 1. Import the required libraries and read the dataset.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()

from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
```

In [2]:
```
df = pd.read_csv('Bank_Personal_Loan_Modelling.csv')
```

## 2. Check the first few samples, shape, info of the data and try to familiarize yourself with different features.

In [3]:
```
df.sample(5)
```

Out[3]:

| | ID | Age | Experience | Income | ZIP Code | Family | CCAvg | Education | Mortgage | Personal Loan | S |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **2290** | 2291 | 38 | 13 | 78 | 91942 | 4 | 0.7 | 3 | 0 | 0 | |
| **1435** | 1436 | 43 | 17 | 55 | 90266 | 1 | 0.2 | 1 | 0 | 0 | |
| **1126** | 1127 | 32 | 8 | 104 | 95192 | 2 | 3.7 | 1 | 0 | 1 | |
| **2726** | 2727 | 62 | 37 | 18 | 92028 | 1 | 1.5 | 2 | 127 | 0 | |
| **3977** | 3978 | 54 | 27 | 51 | 94309 | 3 | 1.0 | 2 | 113 | 0 | |

In [4]:
```
df.shape
```

Out[4]:
```
(5000, 14)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   ID                  5000 non-null   int64  
 1   Age                 5000 non-null   int64  
 2   Experience          5000 non-null   int64  
 3   Income              5000 non-null   int64  
 4   ZIP Code            5000 non-null   int64  
 5   Family              5000 non-null   int64  
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   int64  
 8   Mortgage            5000 non-null   int64  
 9   Personal Loan       5000 non-null   int64  
 10  Securities Account  5000 non-null   int64  
 11  CD Account          5000 non-null   int64  
 12  Online              5000 non-null   int64  
 13  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

## 3. Check if there are any duplicate records present in the dataset? If yes, drop them. and Drop the columns which you feel are redundant.

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
ID                    0
Age                   0
Experience            0
Income                0
ZIP Code              0
Family                0
CCAvg                 0
Education             0
Mortgage              0
Personal Loan         0
Securities Account    0
CD Account            0
Online                0
CreditCard            0
dtype: int64
```

## 4. Display the Five Point Summary and write your key findings.

In [8]:
```
df.describe(include="all")
```

Out[8]:

| | ID | Age | Experience | Income | ZIP Code | Family | CCA |
|---|---|---|---|---|---|---|---|
| **count** | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000000 | 5000.000( |
| **mean** | 2500.500000 | 45.338400 | 20.104600 | 73.774200 | 93152.503000 | 2.396400 | 1.937( |
| **std** | 1443.520003 | 11.463166 | 11.467954 | 46.033729 | 2121.852197 | 1.147663 | 1.747( |
| **min** | 1.000000 | 23.000000 | -3.000000 | 8.000000 | 9307.000000 | 1.000000 | 0.000( |
| **25%** | 1250.750000 | 35.000000 | 10.000000 | 39.000000 | 91911.000000 | 1.000000 | 0.700( |
| **50%** | 2500.500000 | 45.000000 | 20.000000 | 64.000000 | 93437.000000 | 2.000000 | 1.500( |
| **75%** | 3750.250000 | 55.000000 | 30.000000 | 98.000000 | 94608.000000 | 3.000000 | 2.500( |
| **max** | 5000.000000 | 67.000000 | 43.000000 | 224.000000 | 96651.000000 | 4.000000 | 10.000( |

## 5. There are negative values in the variable 'Experience'. Convert them to non-negative values. (Hint:.abs function)

In [9]:
```
df['Experience'].unique()
```

Out[9]:
```
array([ 1, 19, 15,  9,  8, 13, 27, 24, 10, 39,  5, 23, 32, 41, 30, 14, 18,
       21, 28, 31, 11, 16, 20, 35,  6, 25,  7, 12, 26, 37, 17,  2, 36, 29,
        3, 22, -1, 34,  0, 38, 40, 33,  4, -2, 42, -3, 43], dtype=int64)
```

In [10]:
```
df['Experience'] = df['Experience'].abs()
```

In [11]:
```
df['Experience'].unique()
```

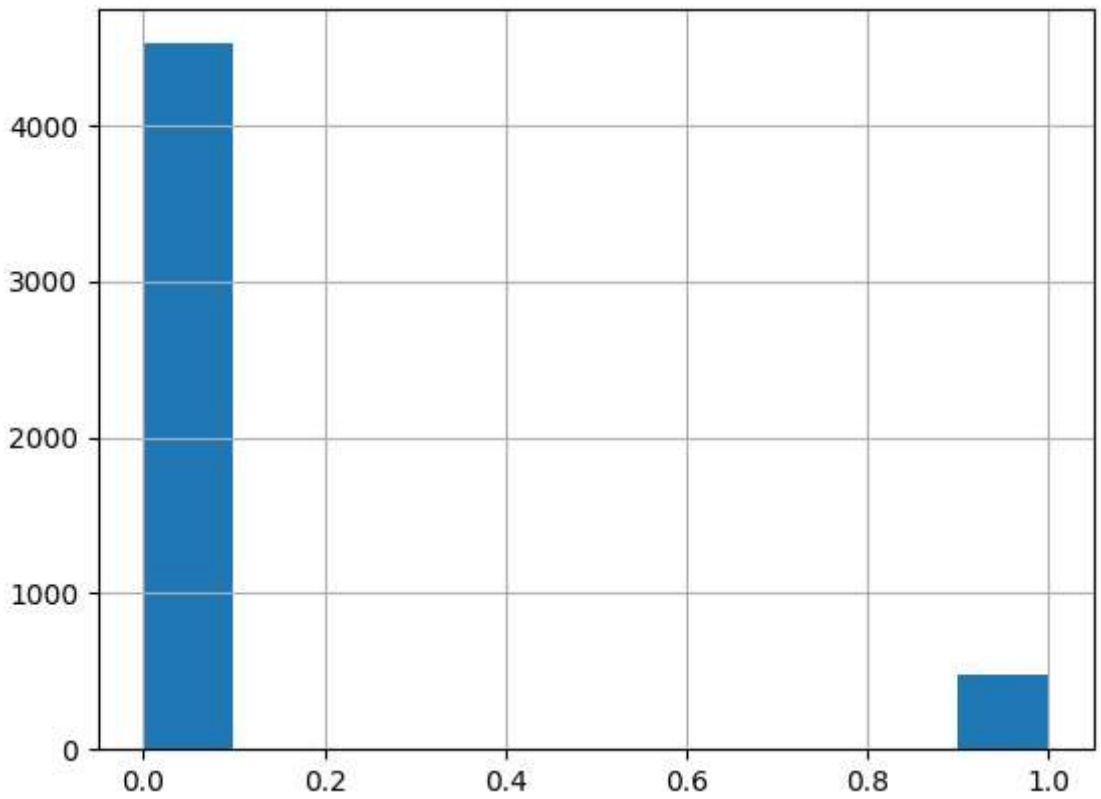
Out[11]:
```
array([ 1, 19, 15,  9,  8, 13, 27, 24, 10, 39,  5, 23, 32, 41, 30, 14, 18,
       21, 28, 31, 11, 16, 20, 35,  6, 25,  7, 12, 26, 37, 17,  2, 36, 29,
        3, 22, 34,  0, 38, 40, 33,  4, 42, 43], dtype=int64)
```

## 6. Get the target column distribution and comment on the class distribution.

In [12]:
```
df['Personal Loan'].unique()
```

Out[12]:
```
array([0, 1], dtype=int64)
```

In [13]:
```
df['Personal Loan'].hist()
plt.show()
```




## 7. Store the target column (i.e.Personal Loan) in the y variable and the rest of the columns in the X variable.

In [14]:
```
df['CCAvg']= le.fit_transform(df['CCAvg'])
```

In [15]:
```
y = df[['Personal Loan']]
X = df.drop('Personal Loan',axis = 1)
```

In [16]:
```
y.shape
```

Out[16]: (5000, 1)

In [17]:
```
X.shape
```

Out[17]: (5000, 13)

## 8. Split the dataset into two parts (i.e. 70% train and 30% test). and standardize the columns using the z-score scaling approach.

In [18]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size= 0.3, random_st
```

In [19]:
```
X_train = ss.fit_transform(X_train)
```

In [20]:
```
X_test = ss.fit_transform(X_test)
```

In [21]:
```
X_test
```

Out[21]:
```
array([[ 0.6807001 ,  1.01476306,  0.94613508, ..., -0.26438635,
         0.81083819, -0.62257857],
       [-0.5466719 ,  0.57924243,  0.50918846, ..., -0.26438635,
         0.81083819, -0.62257857],
       [ 1.0905094 , -0.98863183, -0.97643004, ...,  3.78234351,
         0.81083819, -0.62257857],
       ...,
       [ 1.52636009, -0.5531112 , -0.53948342, ...,  3.78234351,
         0.81083819,  1.60622298],
       [ 1.60448427, -1.33704833, -1.32598734, ..., -0.26438635,
         0.81083819, -0.62257857],
       [ 1.36874113,  1.45028369,  1.38308169, ..., -0.26438635,
        -1.23329169, -0.62257857]])
```

## 9. Train and test a Logistic Regression model to predict the likelihood of a liability customer buying personal loans. Display the train and test accuracy scores.

In [22]:
```
lr.fit(X_train,y_train)
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python311\Lib\site-packages\skle
arn\utils\validation.py:1143: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[22]:
```
LogisticRegression
LogisticRegression()
```

In [23]:
```
y_pre= pd.DataFrame(lr.predict(X_train))
```

In [24]:
```
accuracy_score(y_train,y_pre)
```

Out[24]:
```
0.952
```

In [25]:
```
y_test_pre = pd.DataFrame(lr.predict(X_test))
```

In [26]:
```
accuracy_score(y_test,y_test_pre)
```

Out[26]:
```
0.948
```

## 10. Print the confusion matrix and classification report for the model

In [27]:
```
confusion_matrix(y_train,y_pre)
```

Out[27]:
```
array([[3127,   45],
       [ 123,  205]], dtype=int64)
```

In [28]:
```
confusion_matrix(y_test,y_test_pre)
```

Out[28]:
```
array([[1326,   22],
       [  56,   96]], dtype=int64)
```

In [29]:
```
classification_report(y_train,y_pre)
```

Out[29]:
```
'              precision    recall  f1-score   support\n\n           0
0.96      0.99      0.97      3172\n           1       0.82      0.62      0.
71       328\n\n    accuracy                           0.95      3500\n   mac
ro avg       0.89      0.81      0.84      3500\nweighted avg       0.95
0.95      0.95      3500\n'
```

In [30]:
```
classification_report(y_test,y_test_pre)
```

Out[30]:
```
'              precision    recall  f1-score   support\n\n           0
0.96      0.98      0.97      1348\n           1       0.81      0.63      0.
71       152\n\n    accuracy                           0.95      1500\n   mac
ro avg       0.89      0.81      0.84      1500\nweighted avg       0.94
0.95      0.95      1500\n'
```