**Credit Card Default Prediction**

The data set consists of 2000 samples from each of two categories. Five variables are

1. Income
2. Age
3. Loan
4. Loan to Income (engineered feature)
5. Default

*# Step 1 : import library*

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

In [2]:

*# Step 2 : import data*

default = pd.read\_csv('https://github.com/ybifoundation/Dataset/raw/main/Credit%20Default.csv')

In [3]:

default.head()

Out[3]:

|  | **Income** | **Age** | **Loan** | **Loan to Income** | **Default** |
| --- | --- | --- | --- | --- | --- |
| **0** | 66155.92510 | 59.017015 | 8106.532131 | 0.122537 | 0 |
| **1** | 34415.15397 | 48.117153 | 6564.745018 | 0.190752 | 0 |
| **2** | 57317.17006 | 63.108049 | 8020.953296 | 0.139940 | 0 |
| **3** | 42709.53420 | 45.751972 | 6103.642260 | 0.142911 | 0 |
| **4** | 66952.68885 | 18.584336 | 8770.099235 | 0.130990 | 1 |

In [4]:

default.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2000 entries, 0 to 1999

Data columns (total 5 columns):

# Column Non-Null Count Dtype

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

0 Income 2000 non-null float64

1 Age 2000 non-null float64

2 Loan 2000 non-null float64

3 Loan to Income 2000 non-null float64

4 Default 2000 non-null int64

dtypes: float64(4), int64(1)

memory usage: 78.2 KB

In [5]:

default.describe()

Out[5]:

|  | **Income** | **Age** | **Loan** | **Loan to Income** | **Default** |
| --- | --- | --- | --- | --- | --- |
| **count** | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 | 2000.000000 |
| **mean** | 45331.600018 | 40.927143 | 4444.369695 | 0.098403 | 0.141500 |
| **std** | 14326.327119 | 13.262450 | 3045.410024 | 0.057620 | 0.348624 |
| **min** | 20014.489470 | 18.055189 | 1.377630 | 0.000049 | 0.000000 |
| **25%** | 32796.459720 | 29.062492 | 1939.708847 | 0.047903 | 0.000000 |
| **50%** | 45789.117310 | 41.382673 | 3974.719418 | 0.099437 | 0.000000 |
| **75%** | 57791.281670 | 52.596993 | 6432.410625 | 0.147585 | 0.000000 |
| **max** | 69995.685580 | 63.971796 | 13766.051240 | 0.199938 | 1.000000 |

In [22]:

*# Count of each category*

default['Default'].value\_counts()

Out[22]:

0 1717

1 283

Name: Default, dtype: int64

In [6]:

*# Step 3 : define target (y) and features (X)*

In [7]:

default.columns

Out[7]:

Index(['Income', 'Age', 'Loan', 'Loan to Income', 'Default'], dtype='object')

In [8]:

y = default['Default']

In [9]:

X = default.drop(['Default'],axis=1)

In [10]:

*# Step 4 : train test split*

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, train\_size=0.7, random\_state=2529)

In [11]:

*# check shape of train and test sample*

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

Out[11]:

((1400, 4), (600, 4), (1400,), (600,))

In [12]:

*# Step 5 : select model*

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

model = LogisticRegression()

In [13]:

*# Step 6 : train or fit model*

model.fit(X\_train,y\_train)

Out[13]:

LogisticRegression

LogisticRegression()

In [14]:

model.intercept\_

Out[14]:

array([9.39569095])

In [15]:

model.coef\_

Out[15]:

array([[-2.31410016e-04, -3.43062682e-01, 1.67863323e-03,

1.51188530e+00]])

In [16]:

*# Step 7 : predict model*

y\_pred = model.predict(X\_test)

In [17]:

y\_pred

Out[17]:

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

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,

1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,

1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0])

In [18]:

*# Step 8 : model accuracy*

**from** **sklearn.metrics** **import** confusion\_matrix, accuracy\_score, classification\_report

In [19]:

confusion\_matrix(y\_test,y\_pred)

Out[19]:

array([[506, 13],

[ 17, 64]])

In [20]:

accuracy\_score(y\_test,y\_pred)

Out[20]:

0.95

In [21]:

print(classification\_report(y\_test,y\_pred))

precision recall f1-score support

0 0.97 0.97 0.97 519

1 0.83 0.79 0.81 81

accuracy 0.95 600

macro avg 0.90 0.88 0.89 600

weighted avg 0.95 0.95 0.95 600