Loan Status Detection

NAME: **MANASHJYOTI BORO**

CERTIFICATION CODE: **TCRIG02R81**

EMAIL: **manashjyoti533@gmail.com**

BATCH: **MACHINE LEARNING WITH PYTHON**

PROJECT NAME: **PREDICTION OF LOAN STATUS DETECTION**

# Importing Required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

#Importing the Dataset

dataset=pd.read\_csv('/content/train\_u6lujuX\_CVtuZ9i (1).csv')

dataset.head()

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360.0 | 1.0 | Urban | Y |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | N |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | Urban | Y |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | Urban | Y |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360.0 | 1.0 | Urban | Y |

dataset.tail()

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **609** | LP002978 | Female | No | 0 | Graduate | No | 2900 | 0.0 | 71.0 | 360.0 | 1.0 | Rural | Y |
| **610** | LP002979 | Male | Yes | 3+ | Graduate | No | 4106 | 0.0 | 40.0 | 180.0 | 1.0 | Rural | Y |
| **611** | LP002983 | Male | Yes | 1 | Graduate | No | 8072 | 240.0 | 253.0 | 360.0 | 1.0 | Urban | Y |
| **612** | LP002984 | Male | Yes | 2 | Graduate | No | 7583 | 0.0 | 187.0 | 360.0 | 1.0 | Urban | Y |
| **613** | LP002990 | Female | No | 0 | Graduate | Yes | 4583 | 0.0 | 133.0 | 360.0 | 0.0 | Semiurban | N |

dataset.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 614 entries, 0 to 613 Data columns (total 13 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 Loan\_ID 614 non-null object 1 Gender 601 non-null object 2 Married 611 non-null object 3 Dependents 599 non-null object 4 Education 614 non-null object 5 Self\_Employed 582 non-null object 6 ApplicantIncome 614 non-null int64 7 CoapplicantIncome 614 non-null float64 8 LoanAmount 592 non-null float64 9 Loan\_Amount\_Term 600 non-null float64 10 Credit\_History 564 non-null float64 11 Property\_Area 614 non-null object 12 Loan\_Status 614 non-null object dtypes: float64(4), int64(1), object(8) memory usage: 62.5+ KB

dataset.\_\_len\_\_()

614

dataset.shape

(614, 13)

dataset.describe()

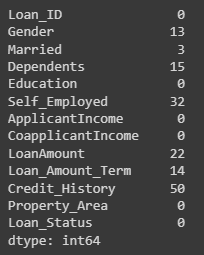
|  | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** |
| --- | --- | --- | --- | --- | --- |
| **count** | 614.000000 | 614.000000 | 592.000000 | 600.00000 | 564.000000 |
| **mean** | 5403.459283 | 1621.245798 | 146.412162 | 342.00000 | 0.842199 |
| **std** | 6109.041673 | 2926.248369 | 85.587325 | 65.12041 | 0.364878 |
| **min** | 150.000000 | 0.000000 | 9.000000 | 12.00000 | 0.000000 |
| **25%** | 2877.500000 | 0.000000 | 100.000000 | 360.00000 | 1.000000 |
| **50%** | 3812.500000 | 1188.500000 | 128.000000 | 360.00000 | 1.000000 |
| **75%** | 5795.000000 | 2297.250000 | 168.000000 | 360.00000 | 1.000000 |
| **max** | 81000.000000 | 41667.000000 | 700.000000 | 480.00000 | 1.000000 |

dataset.isnull()

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False | False | True | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **609** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **610** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **611** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **612** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **613** | False | False | False | False | False | False | False | False | False | False | False | False | False |

# Checking total number of NULL values

dataset.isnull().sum()



#Dropping all null values

dataset=dataset.dropna()

dataset.info()

<class 'pandas.core.frame.DataFrame'> Int64Index: 480 entries, 1 to 613 Data columns (total 13 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 Loan\_ID 480 non-null object 1 Gender 480 non-null object 2 Married 480 non-null object 3 Dependents 480 non-null object 4 Education 480 non-null object 5 Self\_Employed 480 non-null object 6 ApplicantIncome 480 non-null int64 7 CoapplicantIncome 480 non-null float64 8 LoanAmount 480 non-null float64 9 Loan\_Amount\_Term 480 non-null float64 10 Credit\_History 480 non-null float64 11 Property\_Area 480 non-null object 12 Loan\_Status 480 non-null object dtypes: float64(4), int64(1), object(8) memory usage: 52.5+ KB

# Label encoding

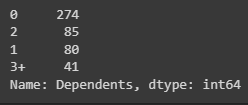
dataset.replace({"Loan\_Status":{'N':0,'Y':1}},inplace=True)

dataset.head()

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | 0 |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | Urban | 1 |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | Urban | 1 |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360.0 | 1.0 | Urban | 1 |
| **5** | LP001011 | Male | Yes | 2 | Graduate | Yes | 5417 | 4196.0 | 267.0 | 360.0 | 1.0 | Urban | 1 |

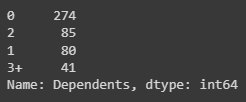
# Dependent column values

dataset['Dependents'].value\_counts()



# Dependent Values

dataset['Dependents'].value\_counts()



# Replacing the values of 3+ to 4

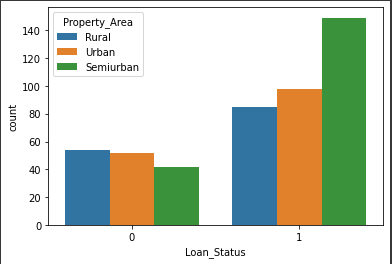
dataset = dataset.replace(to\_replace='3+', value=4)

# Exploratory Data Analysis

#Analysing the variables

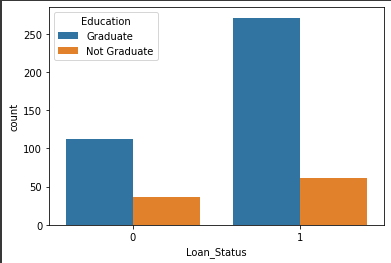
#Property Area and Loan Status

sns.countplot(x="Loan\_Status", hue="Property\_Area", data=dataset)



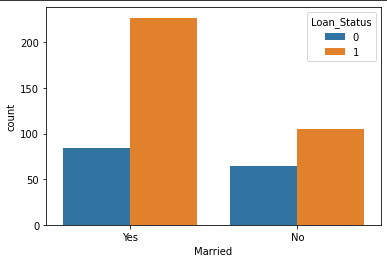
# Education and Loan Status

sns.countplot(x="Loan\_Status", hue="Education",data=dataset)



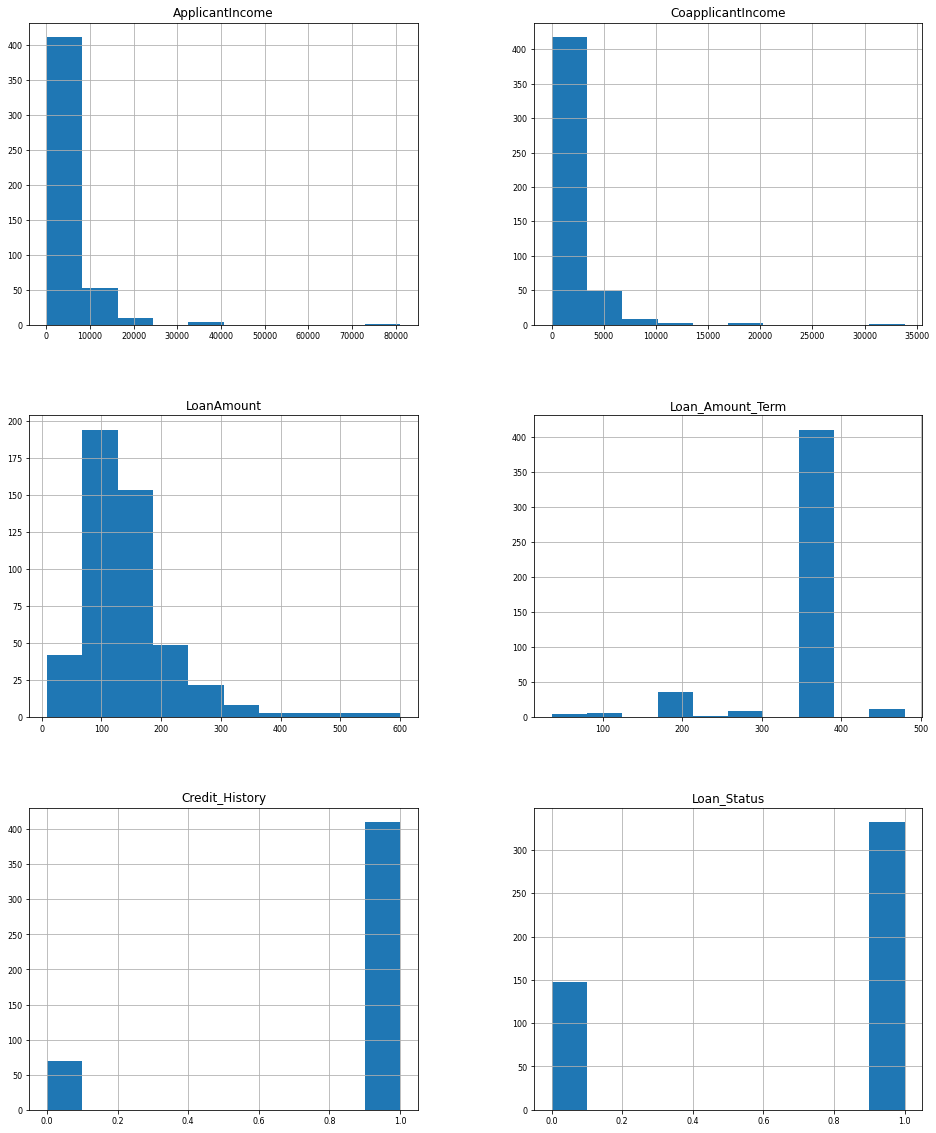
# Marital status & Loan Status

sns.countplot(x='Married',hue='Loan\_Status',data=dataset)



#Overview distribution of each column

dataset.hist(figsize=(16, 20), xlabelsize=8, ylabelsize=8)



# Converting categorical features to numerical values

dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self\_Employed':{'No':0,'Yes':1},

                      'Property\_Area':{'Rural':0,'Semiurban':1,'Urban':2},

                      'Education':{'Graduate':1,'Not Graduate':0}},inplace=True)

dataset.head()

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | LP001003 | 1 | 1 | 1 | 1 | 0 | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | 0 | 0 |
| **2** | LP001005 | 1 | 1 | 0 | 1 | 1 | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | 2 | 1 |
| **3** | LP001006 | 1 | 1 | 0 | 0 | 0 | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | 2 | 1 |
| **4** | LP001008 | 1 | 0 | 0 | 1 | 0 | 6000 | 0.0 | 141.0 | 360.0 | 1.0 | 2 | 1 |
| **5** | LP001011 | 1 | 1 | 2 | 1 | 1 | 5417 | 4196.0 | 267.0 | 360.0 | 1.0 | 2 | 1 |

# Separating the data and label

X = dataset.drop(['Loan\_Status' , 'Loan\_ID' ] , axis =1 )

Y = dataset['Loan\_Status']

#Splitting the dataset into Train and Test

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.30,random\_state=42)

print(X\_train.shape, Y\_train.shape)

(336, 11) (336,)

print(X\_test.shape, Y\_test.shape)

(144, 11) (144,)

# Fitting Logistic Regression to the Training set

from sklearn.linear\_model import LogisticRegression

loan\_model=LogisticRegression()

loan\_model.fit(X\_train,Y\_train)

LogisticRegression()

y\_pred=loan\_model.predict(X\_test)

y\_pred

array([1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1])

# Measuring Accuracy

from sklearn import metrics

print('The accuracy achieved using Logistic Regression is: ', metrics.accuracy\_score(y\_pred, Y\_test))

The accuracy achieved using Logistic Regression is: 0.7847222222222222

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

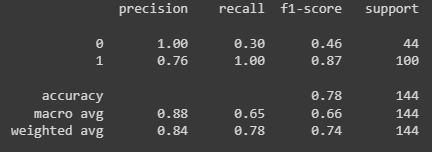
confusion\_matrix(Y\_test,y\_pred)

array([[ 13, 31], [ 0, 100]])

#Classification Report

from sklearn.metrics import classification\_report

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



#The accuracy achieved by using Logistic Regression is around 80.55 %.