**NAME:** DEEPJYOTI DEKA

**CERTIFICATION CODE:** TCRIG02R51

**EMAIL:** deepjyotideka8@gmail.com

**BATCH:** MACHINE LEARNING WITH PYTHON

**PROJECT NAME-** PREDICTION OF LOAN STATUS

**GROUP :** OWN

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

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

train.head()

**Output :**

|  | **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 |

train.describe()

**Output :**

|  | **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 |

# Exploring the null dataset

train.isnull()

**Output :**

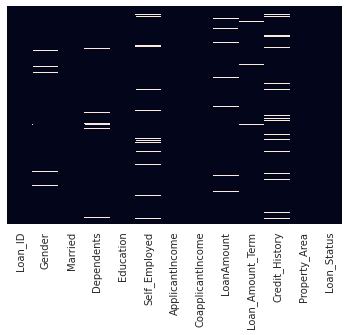
|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

614 rows × 13 columns

sns.heatmap(train.isnull() , yticklabels = False, cbar=False)

**Output :**

<AxesSubplot:>



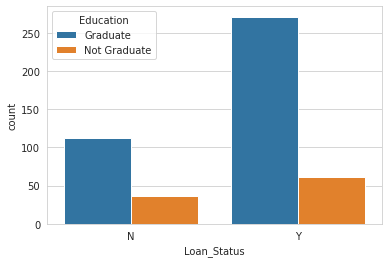
train = train.dropna()

sns.set\_style("whitegrid")

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

**Output :**

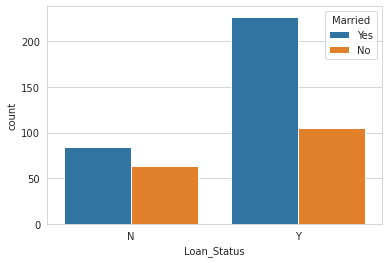
<AxesSubplot:xlabel='Loan\_Status', ylabel='count'>



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

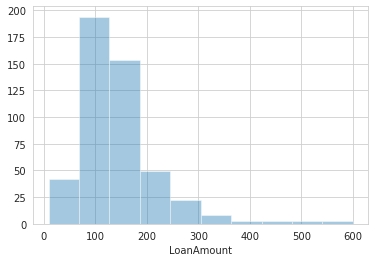
**Output :**

<AxesSubplot:xlabel='Loan\_Status', ylabel='count'>



sns.distplot(train['LoanAmount'].dropna(),kde=False,bins=10)

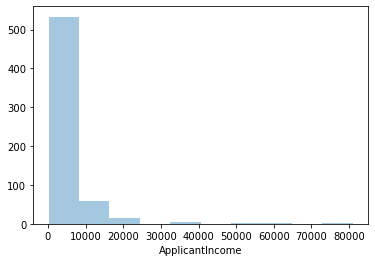
<AxesSubplot:xlabel='LoanAmount'>



sns.distplot(train['ApplicantIncome'].dropna(),kde=False,bins=10)

**Output :**

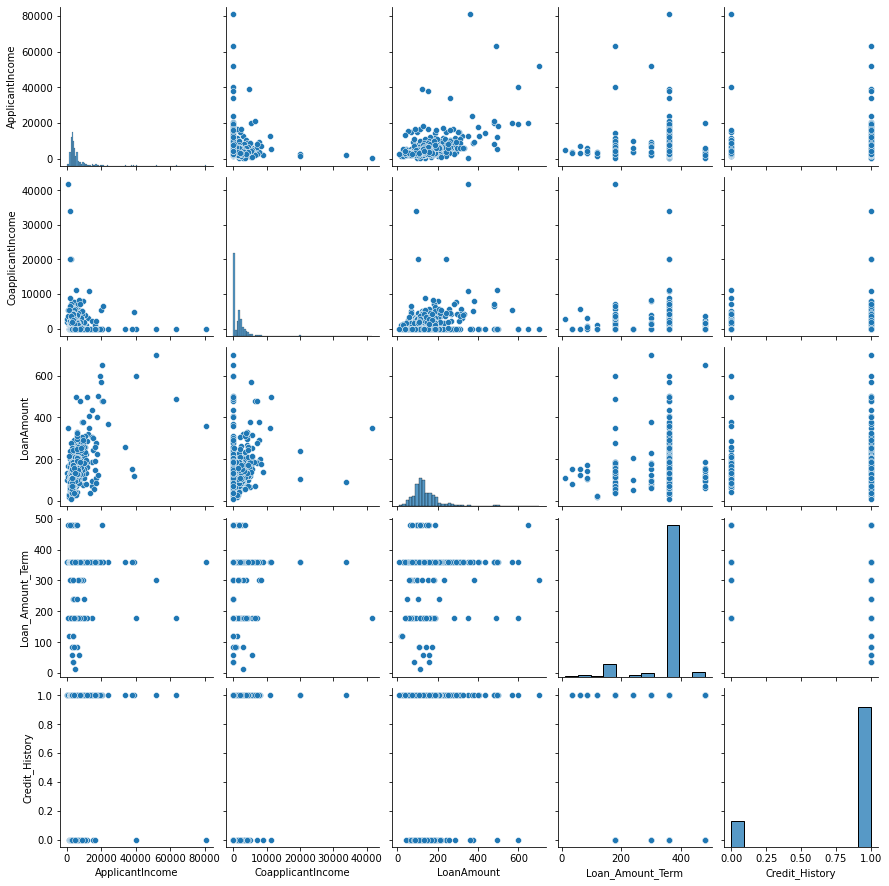
<AxesSubplot:xlabel='ApplicantIncome'>



sns.pairplot(train)

**Output :**

<seaborn.axisgrid.PairGrid at 0x7f063fc8fd60>



train.info()

**Output :**

<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: 68.7+ KB

# converting catergorial features into dummy numbers

train.head()

**Output :**

|  | **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 | 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 |
| **5** | LP001011 | Male | Yes | 2 | Graduate | Yes | 5417 | 4196.0 | 267.0 | 360.0 | 1.0 | Urban | Y |

train['Self\_Employed'].replace(to\_replace = ['Yes' , 'No'] , value=[1,0] , inplace=True)

train['Married'].replace(to\_replace = ['Yes' , 'No'] , value=[1,0] , inplace=True)

train['Loan\_Status'].replace(to\_replace = ['Y' , 'N'] , value=[1,0] , inplace=True)

education = pd.get\_dummies(train['Education'] , drop\_first=True)

gender = pd.get\_dummies(train['Gender'],drop\_first=True)

property\_area = pd.get\_dummies(train['Property\_Area'],drop\_first=True)

# self\_employed = pd.get\_dummies(train['Self\_Employed'] , drop\_first=True)

train = pd.concat([train , gender , education , property\_area] , axis = 1 )

train.drop(['Gender' , 'Education' , 'Property\_Area' ] , axis = 1 , inplace = True)

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

y=train['Loan\_Status']

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.33, random\_state=42)

from sklearn.linear\_model import LogisticRegression

logmodel = LogisticRegression()

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

LogisticRegression()

predictions = logmodel.predict(X\_test)

from sklearn.metrics import classification\_report

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

**Output :**

precision recall f1-score support

0 0.89 0.36 0.52 47

1 0.79 0.98 0.87 112

accuracy 0.80 159

macro avg 0.84 0.67 0.69 159

weighted avg 0.82 0.80 0.77 159

from sklearn.metrics import confusion\_matrix

confusion\_matrix(y\_test , predictions)

**Output :**

array([[ 17, 30],

[ 2, 110]])

# The accuracy of the logistic regression is about 80%