## 1. Import respective modules to python notebook.

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

## 2. Read CSV file and print Initial 5 rows of the same.

```
df = pd.read_csv('C:/Users/visha/Downloads/Loan Prediction
Dataset.csv')
df.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

## 3. Display the number of rows and columns.

```
 print(f"The number of rows and columns are-> {df.shape}")
```

```
The number of rows and columns are-> (614, 13)
```

Inference:The Loan Prediction dataset contains of 614 rows and 12 columns

# 4. Display description of all attributes in given dataset.

```
df.describe()
```

```
       ApplicantIncome  CoapplicantIncome  LoanAmount
Loan_Amount_Term  \
count       614.000000         614.000000  592.000000
600.00000
mean       5403.459283        1621.245798  146.412162
342.00000
std        6109.041673        2926.248369   85.587325
65.12041
min         150.000000           0.000000    9.000000
12.00000
25%        2877.500000           0.000000  100.000000
360.00000
50%        3812.500000        1188.500000  128.000000
360.00000
75%        5795.000000        2297.250000  168.000000
360.00000
max       81000.000000       41667.000000  700.000000
480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

# 5. Displays datatypes of each attributes.

```
df.dtypes
```

```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
```

```
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

# 6. Display datatype and non-null count of the given dataset.

```
df.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
```

Preprocessing the dataset

# 7. Display the Null values in each Attribute.

```
# find the null values
df.isnull().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
```

```
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

# 8. Handle the null values in the dataset.

```
# fill the missing values for numerical terms - mean
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] =
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] =
df['Credit_History'].fillna(df['Credit_History'].mean())

# fill the missing values for categorical terms - mode
df['Gender'] = df["Gender"].fillna(df['Gender'].mode()[0])
df['Married'] = df["Married"].fillna(df['Married'].mode()[0])
df['Dependents'] = df["Dependents"].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] =
df["Self_Employed"].fillna(df['Self_Employed'].mode()[0])

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

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

# 9. Counts the number of unique values in the dataset.

```
df.nunique()
```

```
Loan_ID               614
Gender                  2
Married                 2
Dependents              4
Education               2
Self_Employed           2
ApplicantIncome       505
CoapplicantIncome     287
LoanAmount            204
Loan_Amount_Term       11
Credit_History          3
Property_Area           3
Loan_Status             2
dtype: int64
```

# 10 Display names of all columns in respective dataset.

```
df.columns
```

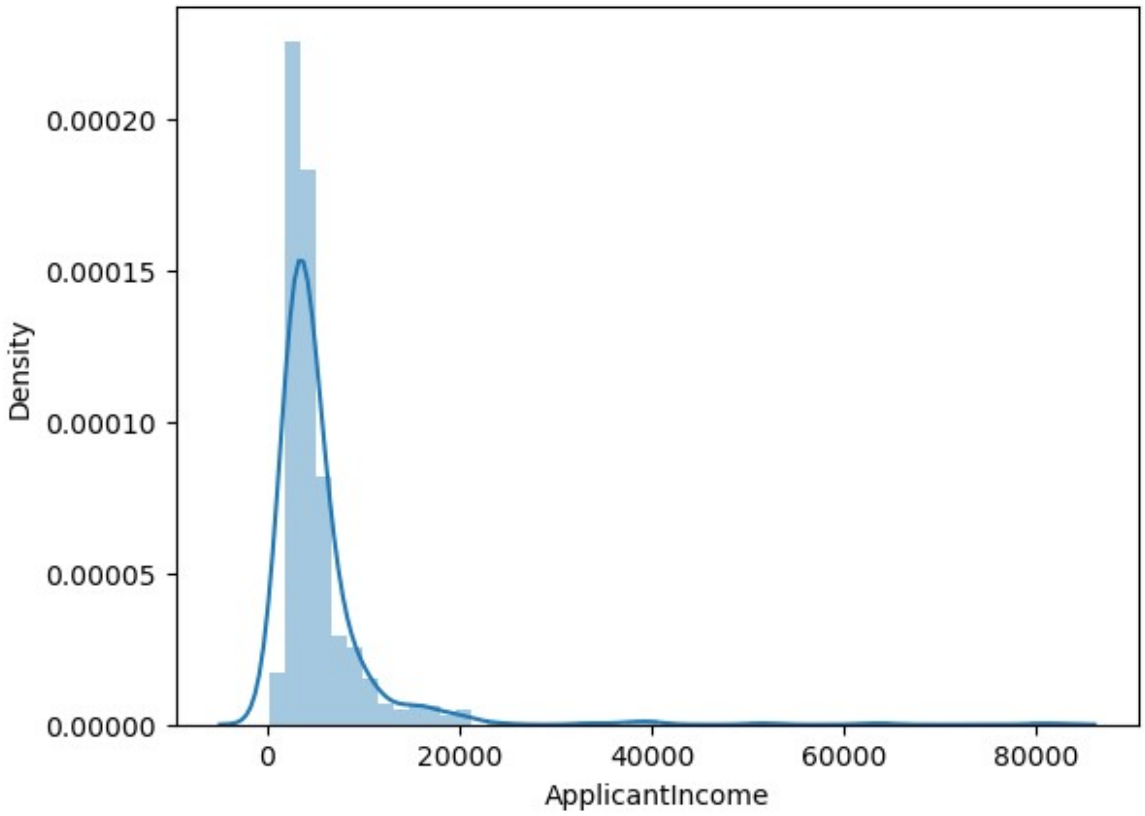
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome',
'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area',
'Loan_Status'],
      dtype='object')
```

Exploratory Data Analysis

# 11.Display the distribution of ApplicantIncome

```
# numerical attributes visualization
sns.distplot(df["ApplicantIncome"])
```

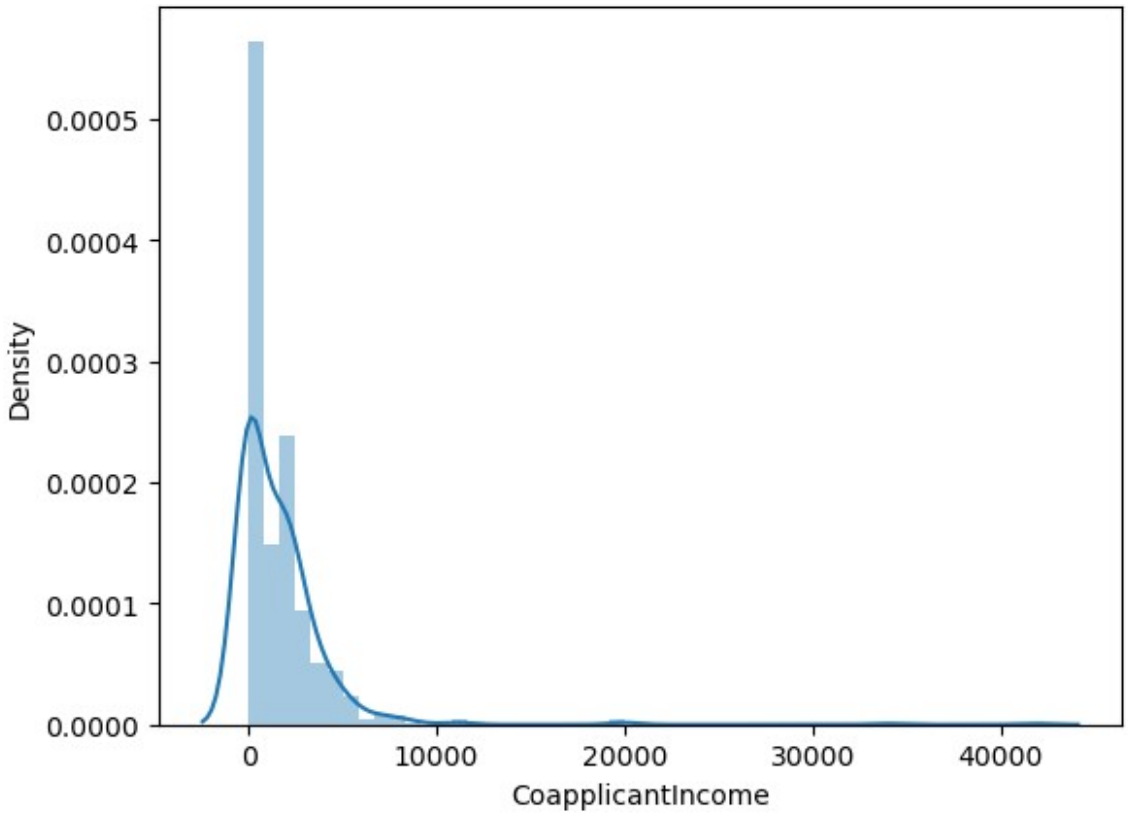
```
<Axes: xlabel='ApplicantIncome', ylabel='Density'>
```




## 12.Display the distribution of CoapplicantIncome.

```
sns.distplot(df["CoapplicantIncome"])
```

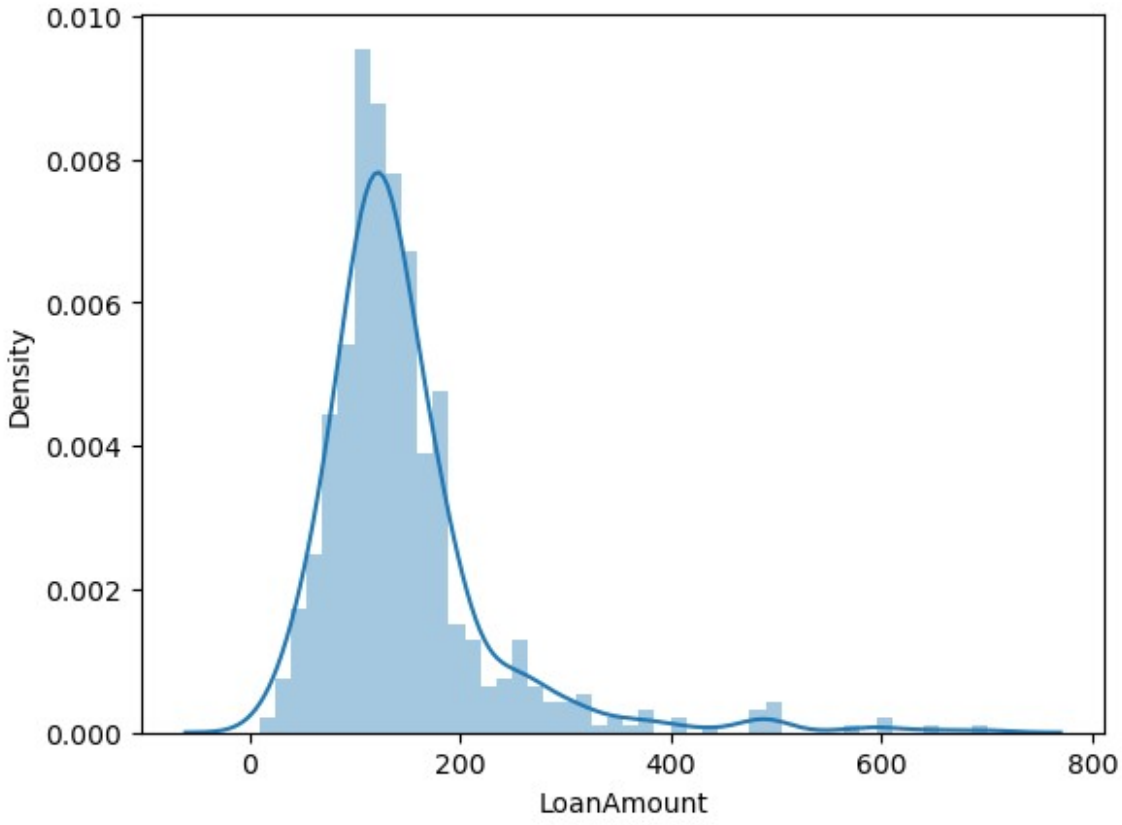
```
<Axes: xlabel='CoapplicantIncome', ylabel='Density'>
```




## 13.Display the distribution of LoanAmount.

```
sns.distplot(df["LoanAmount"])
```

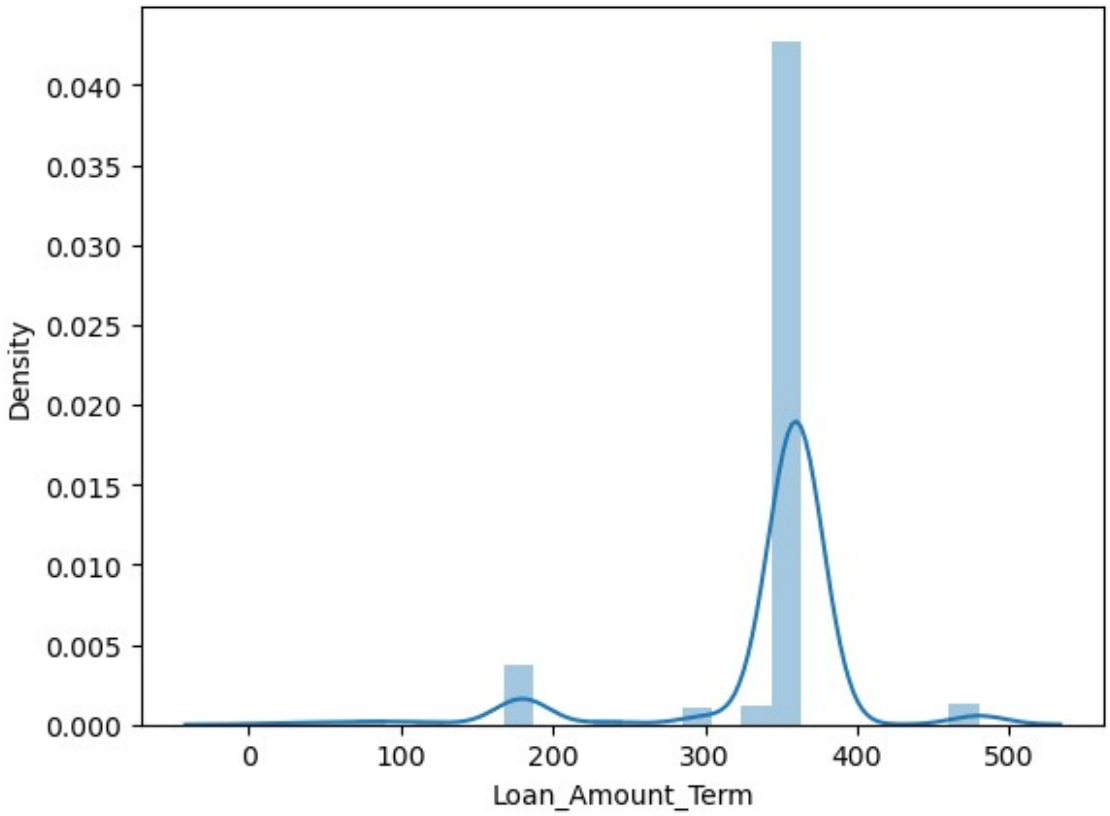
```
<Axes: xlabel='LoanAmount', ylabel='Density'>
```




# 14.Display the distribution of Loan_Amount_Term.

```
sns.distplot(df['Loan_Amount_Term'])
```

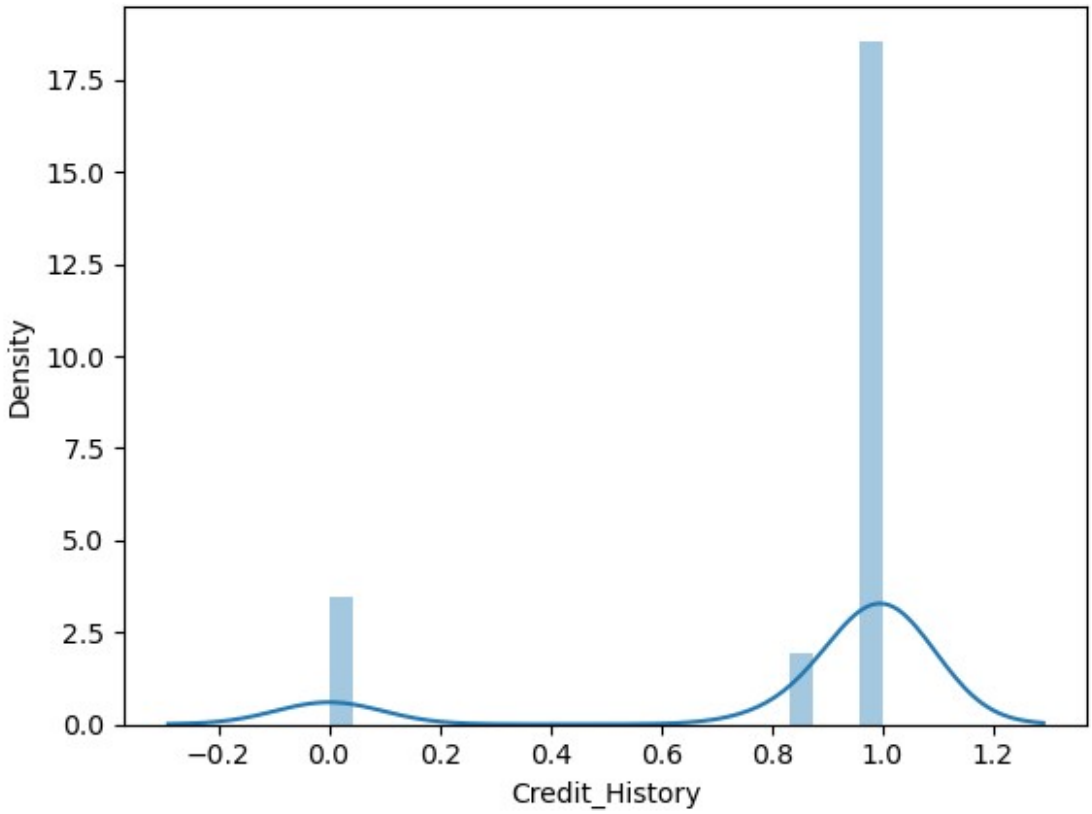
```
<Axes: xlabel='Loan_Amount_Term', ylabel='Density'>
```




## 15.Display the distribution of Credit_History.

```
sns.distplot(df['Credit_History'])
```

```
<Axes: xlabel='Credit_History', ylabel='Density'>
```




Creation of new attributes

# 16.Calculate the applicant total income including applicant and coapplicant incomes.

```
# total income
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | \ |
|---|---|---|---|---|---|---|---|
| 0 | LP001002 | Male | No | 0 | Graduate | No | |
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | |
| 2 | LP001005 | Male | Yes | 0 | Graduate | Yes | |
| 3 | LP001006 | Male | Yes | 0 | Not Graduate | No | |
| 4 | LP001008 | Male | No | 0 | Graduate | No | |

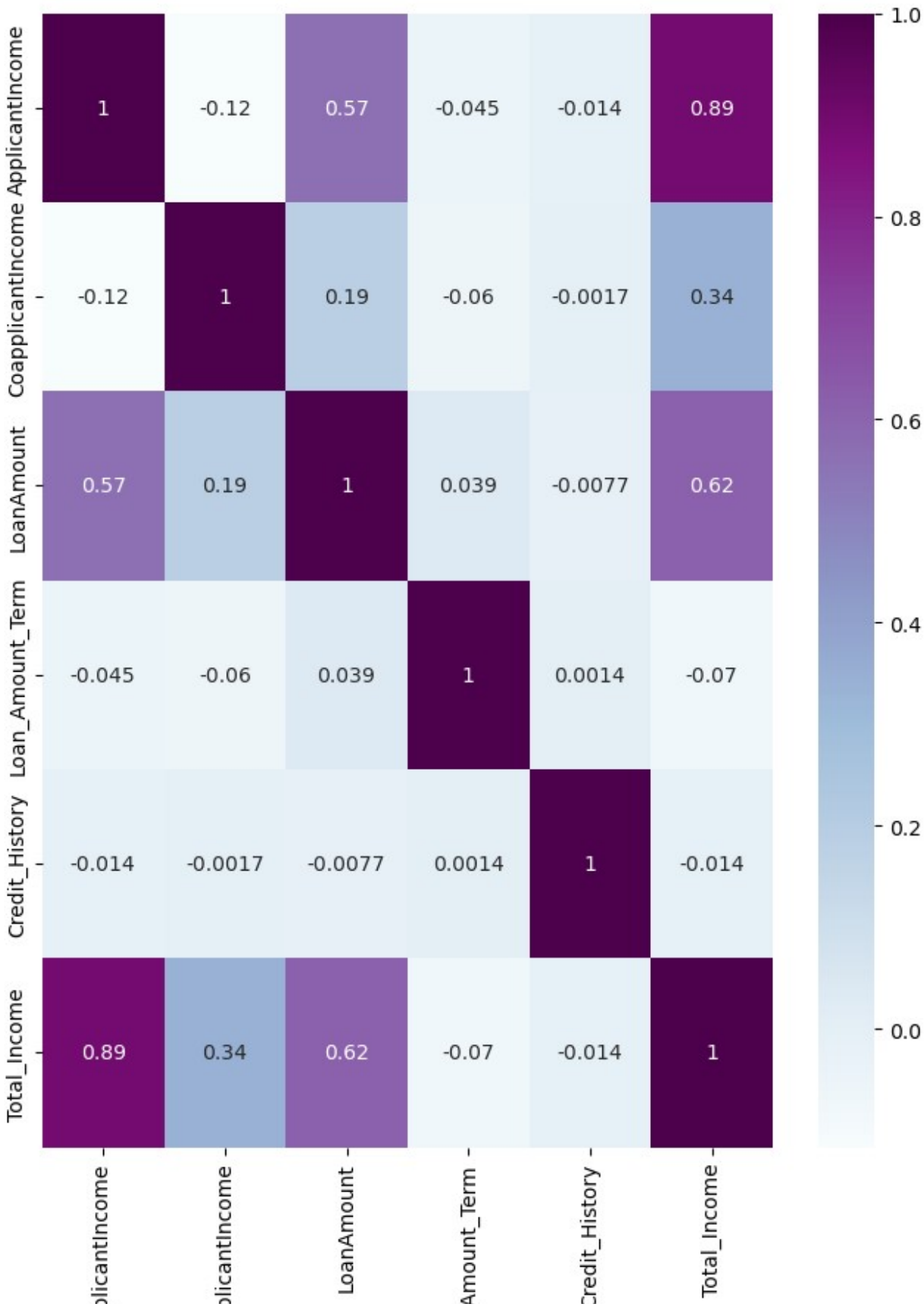
| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | \ |
|---|---|---|---|---|---|
| 0 | 5849 | 0.0 | 146.412162 | 360.0 | |
| 1 | 4583 | 1508.0 | 128.000000 | 360.0 | |
| 2 | 3000 | 0.0 | 66.000000 | 360.0 | |
| 3 | 2583 | 2358.0 | 120.000000 | 360.0 | |
| 4 | 6000 | 0.0 | 141.000000 | 360.0 | |

```
   Credit_History Property_Area Loan_Status  Total_Income
0             1.0         Urban           Y        5849.0
1             1.0         Rural           N        6091.0
2             1.0         Urban           Y        3000.0
3             1.0         Urban           Y        4941.0
4             1.0         Urban           Y        6000.0
```

## 17.Check the correlation between dataset attributes using heatmap.

```
corr = df.corr()
plt.figure(figsize=(8,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
```

```
<Axes: >
```


ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
Total_Income
1 -0.12 0.57 -0.045 -0.014 0.89
-0.12 1 0.19 -0.06 -0.0017 0.34
0.57 0.19 1 0.039 -0.0077 0.62
-0.045 -0.06 0.039 1 0.0014 -0.07
-0.014 -0.0017 -0.0077 0.0014 1 -0.014
0.89 0.34 0.62 -0.07 -0.014 1
LoanAmount
Total_Income
1.0
0.8
0.6
0.4
0.2
0.0

# 18. Drop the columns

```
# drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount",
"Loan_Amount_Term", "Total_Income", 'Loan_ID']
df = df.drop(columns=cols, axis=1)
df.head()
```

```
  Gender Married Dependents     Education Self_Employed
Credit_History  \
0   Male      No          0      Graduate            No
1.0
1   Male     Yes          1      Graduate            No
1.0
2   Male     Yes          0      Graduate           Yes
1.0
3   Male     Yes          0  Not Graduate            No
1.0
4   Male      No          0      Graduate            No
1.0

  Property_Area Loan_Status
0         Urban           Y
1         Rural           N
2         Urban           Y
3         Urban           Y
4         Urban           Y
```

# 19.Label Encoding.

```
from sklearn.preprocessing import LabelEncoder
cols =
['Gender',"Married","Education",'Self_Employed',"Property_Area","Loan_
Status","Dependents"]
le = LabelEncoder()
for col in cols:
    df[col] = le.fit_transform(df[col])

df.head()
```

```
   Gender  Married  Dependents  Education  Self_Employed
Credit_History  \
0       1        0           0          0              0
1.0
1       1        1           1          0              0
1.0
2       1        1           0          0              1
1.0
```

```
3       1         1            0           1               0
1.0
4       1         0            0           0               0
1.0

   Property_Area  Loan_Status
0              2            1
1              0            0
2              2            1
3              2            1
4              2            1
```

## 20.Split the Train-Test .

```
# specify input and output attributes
X = df.drop(columns=['Loan_Status'], axis=1)
y = df['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=0.25, random_state=42)
```

## 21.Model Training.

```
# classify function
from sklearn.model_selection import cross_val_score
def classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy is", model.score(x_test, y_test)*100)
    # cross validation - it is used for better validation of model
    # eg: cv-5, train-4, test-1
    score = cross_val_score(model, x, y, cv=5)
    print("Cross validation is",np.mean(score)*100)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X, y)

Accuracy is 77.27272727272727
Cross validation is 80.9462881514061

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, X, y)

Accuracy is 72.72727272727273
Cross validation is 73.94908703185392
```

```
from sklearn.ensemble import
RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
classify(model, X, y)
```

```
Accuracy is 75.97402597402598
Cross validation is 76.8812475009996
```

```
model = ExtraTreesClassifier()
classify(model, X, y)
```

```
Accuracy is 72.07792207792207
Cross validation is 74.59816073570572
```