1-loan

September 25, 2024

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

[2]:
```
pwd
```

[2]:
```
'/Users/sourabkumargiri'
```

[5]:
```
df = pd.read_csv("Downloads/loan-train.csv")
```

[7]:
```
df.head()
```

[7]:

	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

[9]:
```
pd.crosstab(df['Credit_History'],df['Loan_Status'],margins=True)
```

[9]:
```
Loan_Status       N    Y  All
Credit_History
0.0              82    7   89
1.0              97  378  475
All             179  385  564
```

[65]:
```
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
```

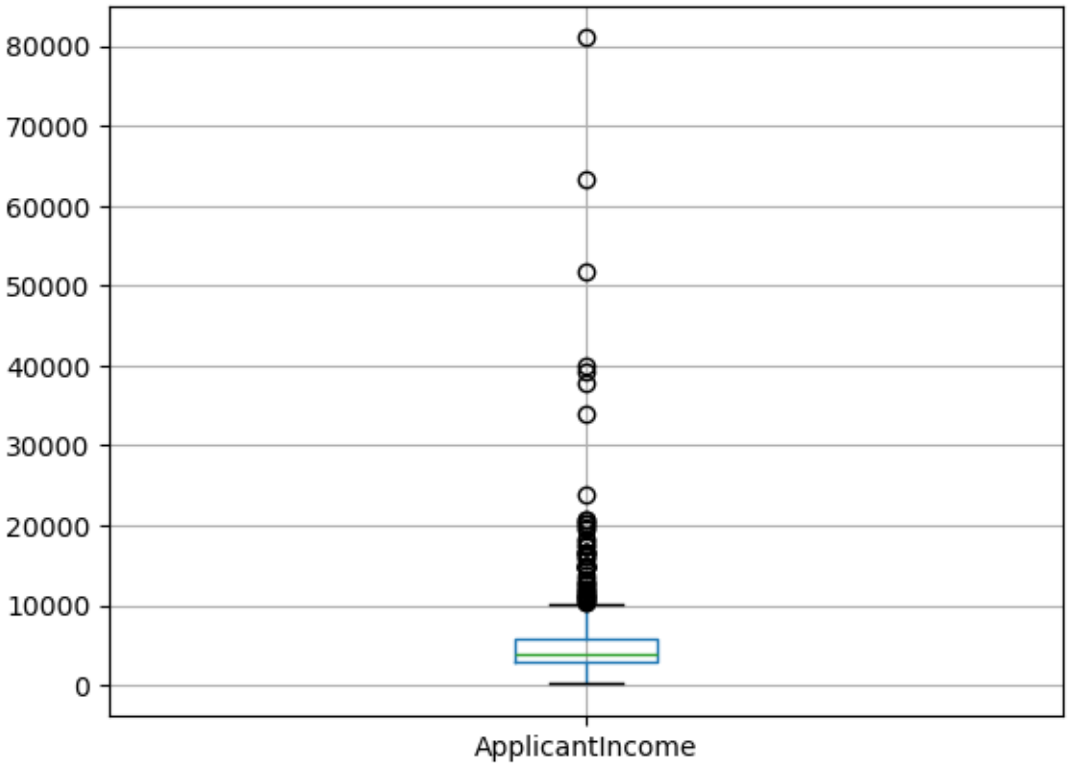
[67]:
```
df.describe()
```

[67]:
```
       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
```

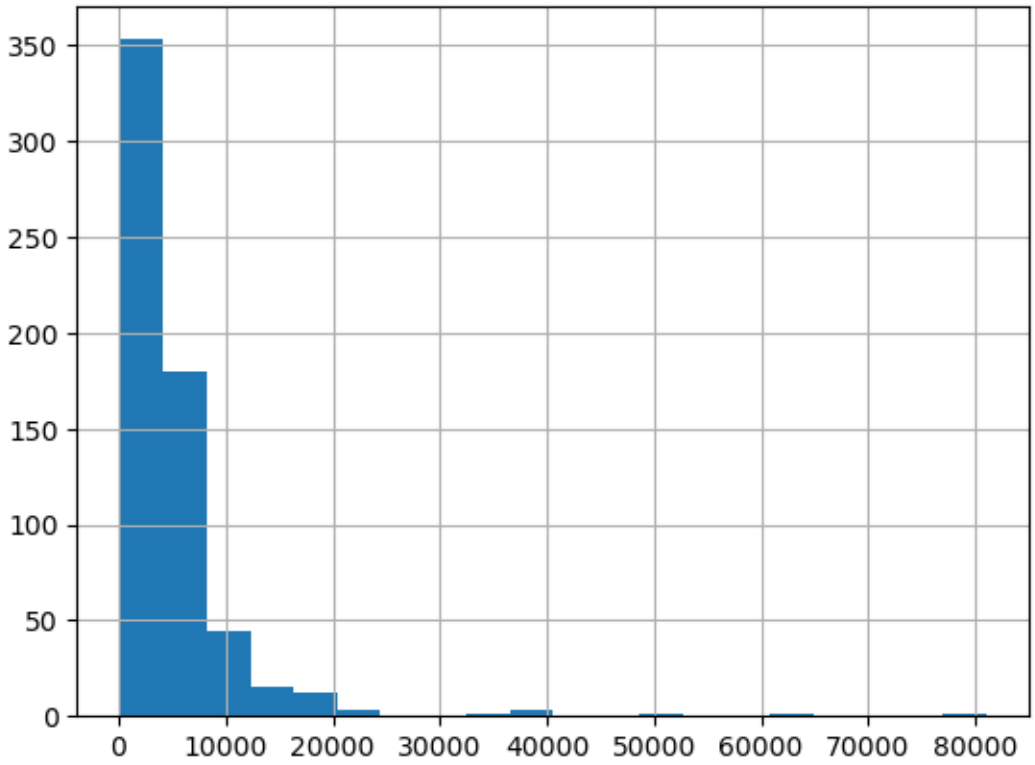
[69]:
```
df.boxplot(column="ApplicantIncome")
```

[69]: <Axes: >



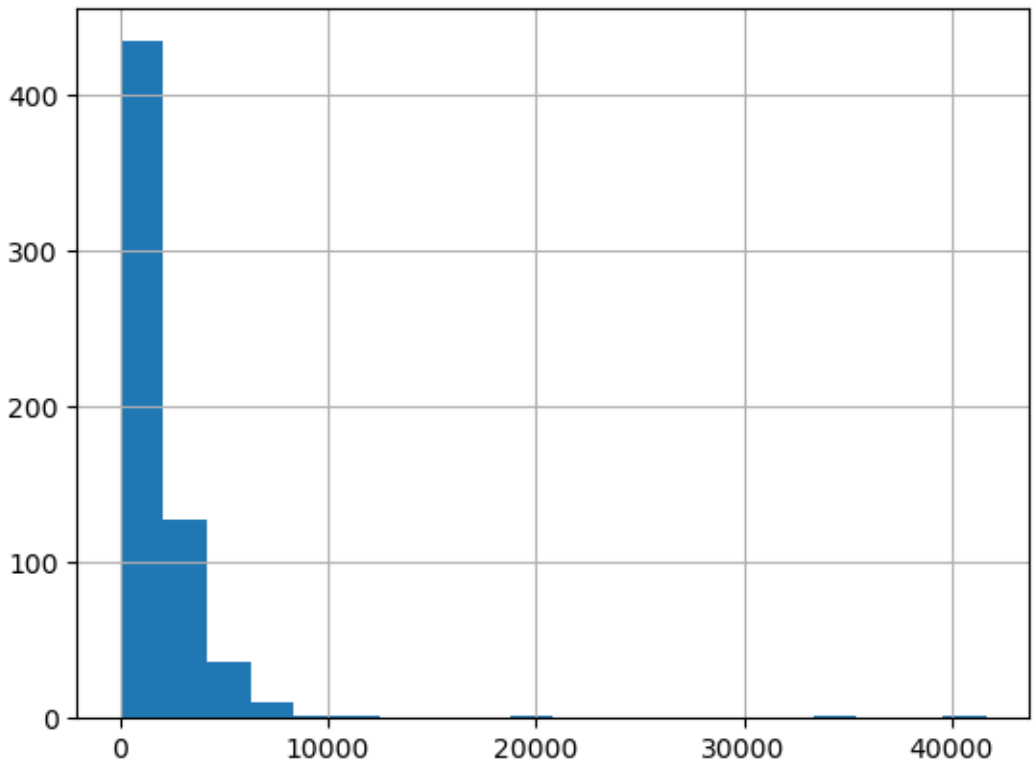
[71]:
```
df["ApplicantIncome"].hist(bins=20)
```

[71]: <Axes: >



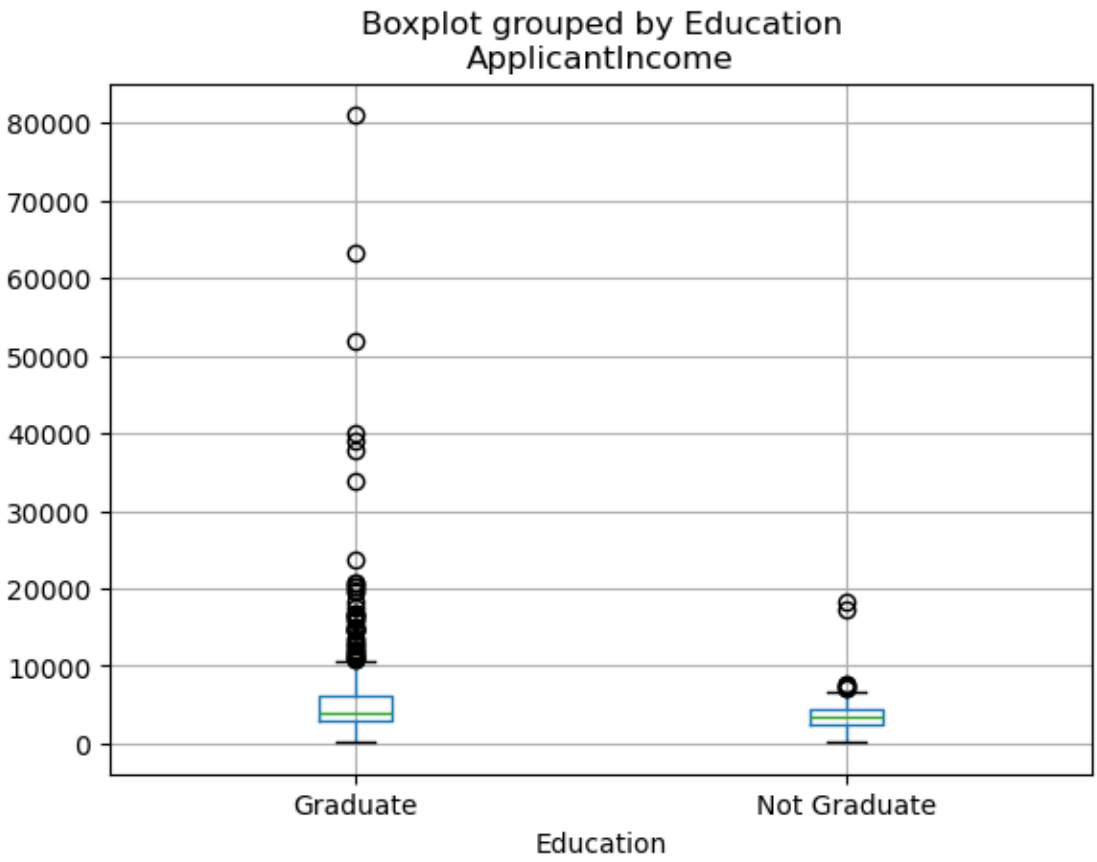
[73]:
```
df["CoapplicantIncome"].hist(bins=20)
```

[73]: <Axes: >



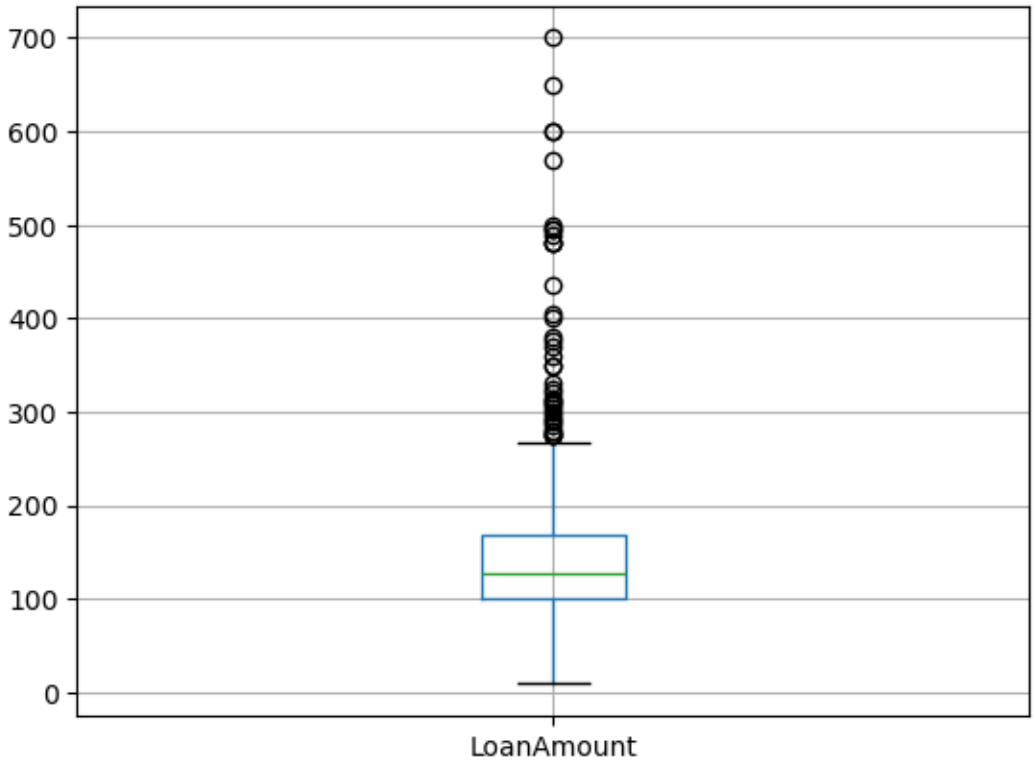
[75]:
```
df.boxplot(column="ApplicantIncome",by= "Education")
```

[75]:
```
<Axes: title={'center': 'ApplicantIncome'}, xlabel='Education'>
```



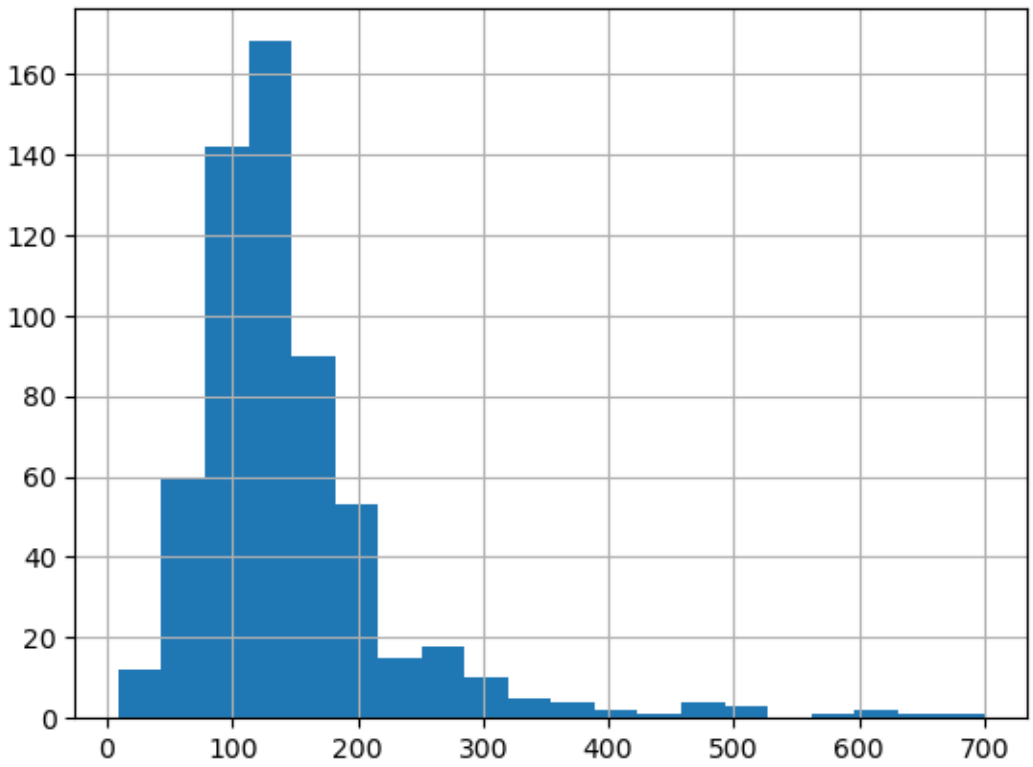
[77]:
```
df.boxplot(column="LoanAmount")
```

[77]: <Axes: >



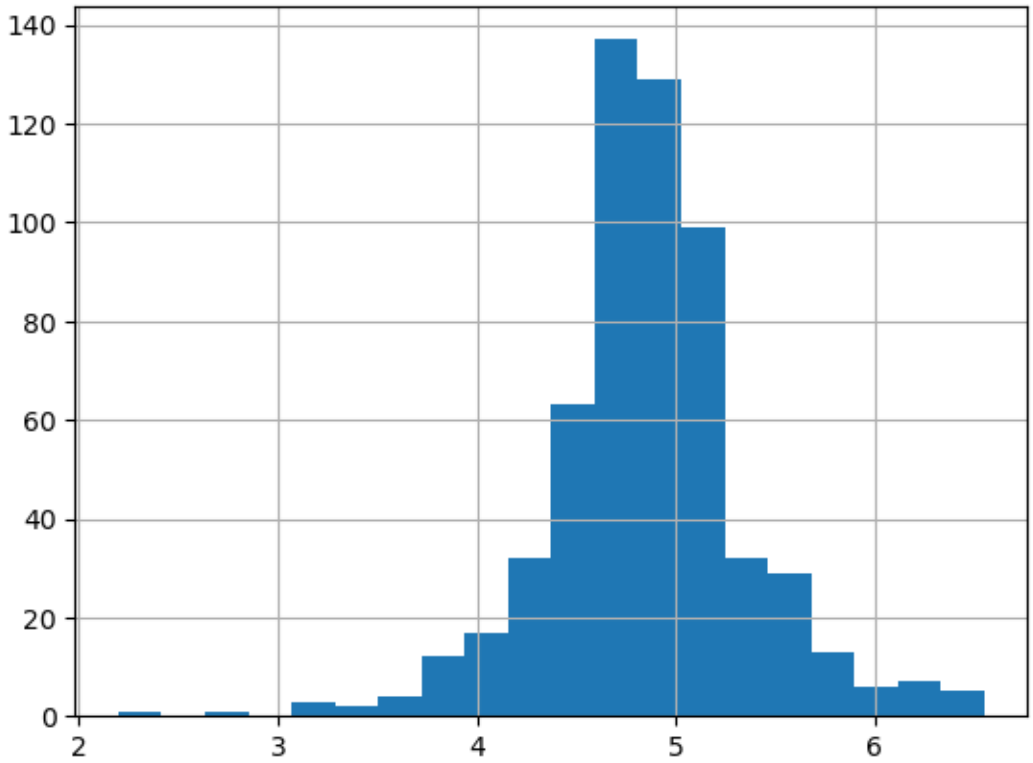
[79]:
```
df["LoanAmount"].hist(bins=20)
```

[79]: <Axes: >



[81]:
```
df["LoanAmount_log"]=np.log(df["LoanAmount"])
df["LoanAmount_log"].hist(bins=20)
```

[81]: <Axes: >



[83]:
```
df.isnull().sum()
```

[83]:
```
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
LoanAmount_log       22
dtype: int64
```

[103]:
```
df["Gender"].fillna(df["Gender"].mode(),inplace=True)
```

[107]:
```
df["Married"].fillna(df["Married"].mode(),inplace=True)
```

[113]:
```
df["Dependents"].fillna(df["Dependents"].mode(),inplace=True)
```

[119]:
```
df["Self_Employed"].fillna(df["Self_Employed"].mode(),inplace=True)
```

[123]:
```
df.LoanAmount=df.LoanAmount.fillna(df.LoanAmount.mean())
df.LoanAmount_log=df.LoanAmount_log.fillna(df.LoanAmount_log.mean())
```

[11]:
```
df["Loan_Amount_Term"].fillna(df["Loan_Amount_Term"].mode(),inplace=True)
```

```
/var/folders/3s/vmbvn6xj4ld_5hnrk6lcfxkh0000gn/T/ipykernel_977/3062782434.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Loan_Amount_Term"].fillna(df["Loan_Amount_Term"].mode(),inplace=True)
```

[147]:
```
df["Credit_History"].fillna(df["Credit_History"].mode(),inplace=True)
```

```
/var/folders/3s/vmbvn6xj4ld_5hnrk6lcfxkh0000gn/T/ipykernel_1398/461245335.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Credit_History"].fillna(df["Credit_History"].mode(),inplace=True)
```

[149]:
```
df.head()
```

[149]:

	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  LoanAmount_log
0             1.0         Urban           Y        4.857444
1             1.0         Rural           N        4.852030
2             1.0         Urban           Y        4.189655
3             1.0         Urban           Y        4.787492
4             1.0         Urban           Y        4.948760
```

[151]:
```
x=df.iloc[:,np.r_[1:2,2:3,3:4]].values
y=df.iloc[:,12].values
```

[91]:
```
x
```

[91]:
```
array([['Male', 'No', '0'],
       ['Male', 'Yes', '1'],
       ['Male', 'Yes', '0'],
       …,
       ['Male', 'Yes', '1'],
       ['Male', 'Yes', '2'],
       ['Female', 'No', '0']], dtype=object)
```

[93]:
```
y
```

[93]:
```
array(['Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'N', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N',
       'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'N',
       'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'N',
       'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N',
```

```
       'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'N', 'N',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'N', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'N', 'Y',
       'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'N', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'N', 'N',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'N',
       'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N'], dtype=object)
```

[153]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)
```

[157]:
```
print(x_train)
```

```
[['Male' 'Yes' '0']
 ['Male' 'No' '1']
 ['Male' 'Yes' '0']
 …
 ['Male' 'Yes' '3+']
 ['Male' 'Yes' '0']
 ['Female' 'Yes' '0']]
```

[159]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder_x =LabelEncoder()
```

[184]:
```
for i in range (0,5):
    x_train [:,i]=labelencoder_x.fit_transform(x_train[:,i])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[184], line 2
      1 for i in range (0,5):
----> 2     x_train [:,i]=labelencoder_x. fit_transform(x_train[:,i])

IndexError: index 3 is out of bounds for axis 1 with size 3
```

[ ]:
```
x_train[:,7]=labelencoder_x.fit_transform(x_train[:,7]
```

[180]:
```
x_train
```

[180]:
```
array([[1, 1, 0],
       [1, 0, 1],
       [1, 1, 0],
       …,
       [1, 1, 3],
       [1, 1, 0],
       [0, 1, 0]], dtype=object)
```

[188]:
```
labelencoder_y=LabelEncoder()
y_train=labelencoder_y.fit_transform(y_train)
```

[190]:
```
y_train
```

[190]:
```
array([1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
```

```
       1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 1])
```

[198]:
```
for i in range (0,5):
    x_test[:,i]= labelencoder_x.fit_transform(x_test[:,i])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[198], line 2
      1 for i in range (0,5):
----> 2     x_test[:,i]= labelencoder_x.fit_transform(x_test[:,i])

IndexError: index 3 is out of bounds for axis 1 with size 3
```

[200]:
```
df.describe()
```

[200]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
count	614.000000	614.000000	614.000000	600.00000	
mean	5403.459283	1621.245798	146.412162	342.00000	
std	6109.041673	2926.248369	84.037468	65.12041	
min	150.000000	0.000000	9.000000	12.00000	
25%	2877.500000	0.000000	100.250000	360.00000	
50%	3812.500000	1188.500000	129.000000	360.00000	
75%	5795.000000	2297.250000	164.750000	360.00000	
max	81000.000000	41667.000000	700.000000	480.00000	

	Credit_History	LoanAmount_log
count	564.000000	614.000000
mean	0.842199	4.857444
std	0.364878	0.495995
min	0.000000	2.197225
25%	1.000000	4.607658
50%	1.000000	4.857444
75%	1.000000	5.104426
max	1.000000	6.551080

[204]:
```
df.head()
```

[204]:

	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  LoanAmount_log
0             1.0         Urban           Y        4.857444
1             1.0         Rural           N        4.852030
2             1.0         Urban           Y        4.189655
3             1.0         Urban           Y        4.787492
4             1.0         Urban           Y        4.948760
```

[208]:
```
df.
```

```
  Cell In[208], line 1
    df.how many person are graduate
           ^
SyntaxError: invalid syntax
```

[155]:
```
import numpy as np
import pandas as pd
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.svm import SVC
from imblearn.over_sampling import RandomOverSampler
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

[157]:
```
pwd
```

[157]:
```
'/Users/sourabkumargiri'
```

[159]:
```
df = pd.read_csv("Downloads/loan-train.csv")
```

[7]:
```
df
```

[7]:

	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	
..	…	…	…	…	…	…	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	3+	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

	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	
..	…	…	…	…	
609	2900	0.0	71.0	360.0	
610	4106	0.0	40.0	180.0	
611	8072	240.0	253.0	360.0	
612	7583	0.0	187.0	360.0	
613	4583	0.0	133.0	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
..	…	…	
609	1.0	Rural	Y
610	1.0	Rural	Y
611	1.0	Urban	Y
612	1.0	Urban	Y
613	0.0	Semiurban	N

[614 rows x 13 columns]

1 see the shape of dataset, we can use shape method

[161]:
```
df.shape
```

[161]: (614, 13)

[27]:
```
df.info
```

[27]:
```
<bound method DataFrame.info of       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
..        …     …     …         …           …           …
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     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
..               …                …         …                 …
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             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
..              …           …         …
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]>
```

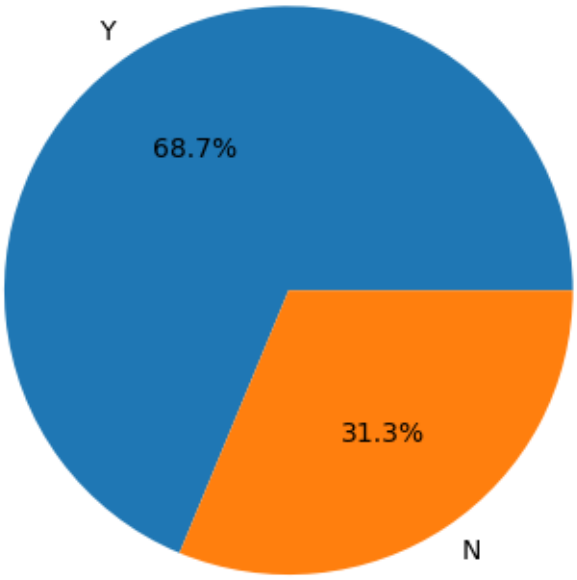
2 To get value like the mean, count and min of the column we can use describe() method

[163]:
```
df.describe
```

[163]:
```
<bound method NDFrame.describe of       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
..        …     …     …         …            …           …
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     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
..               …                  …         …                 …
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             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
..              …           …         …
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]>
```

3 piechart for loanstatus column

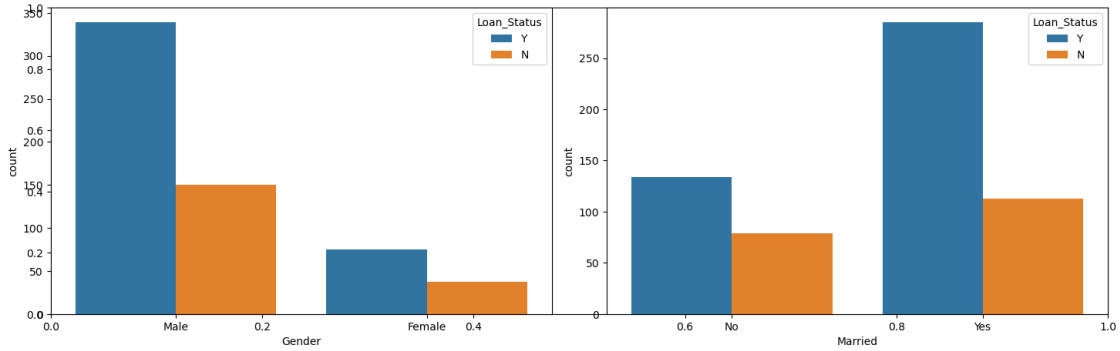
[165]:
```
temp = df ['Loan_Status'].value_counts()
plt.pie(temp.values,
        labels=temp.index,
        autopct='%1.1f%%')
plt.show()
```



4 The x parameter is set to the column name from which the count plot is to be created, and hue is set to “Loan_Startus” to create count bars based on the “Loan_Status” categopries

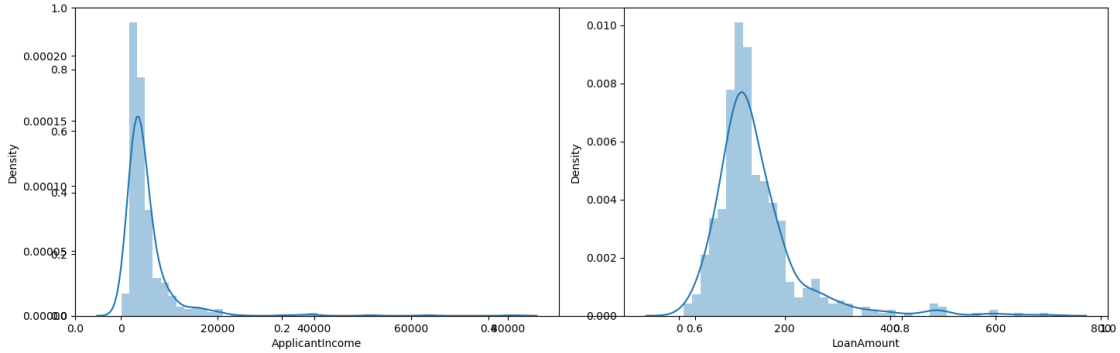
[167]:
```
plt.subplots(figsize=(15 ,5))
for i, col in enumerate(["Gender","Married"]):
    plt.subplot(1,2,i+1)
    sb.countplot(data=df, x=col, hue="Loan_Status")
plt.tight_layout()
plt.show
```

[167]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[169]:
```
plt.subplots(figsize=(15 ,5))
for i, col in enumerate(["ApplicantIncome","LoanAmount"]):
    plt.subplot(1,2,i+1)
    sb.distplot(df[col])
plt.tight_layout()
plt.show
```

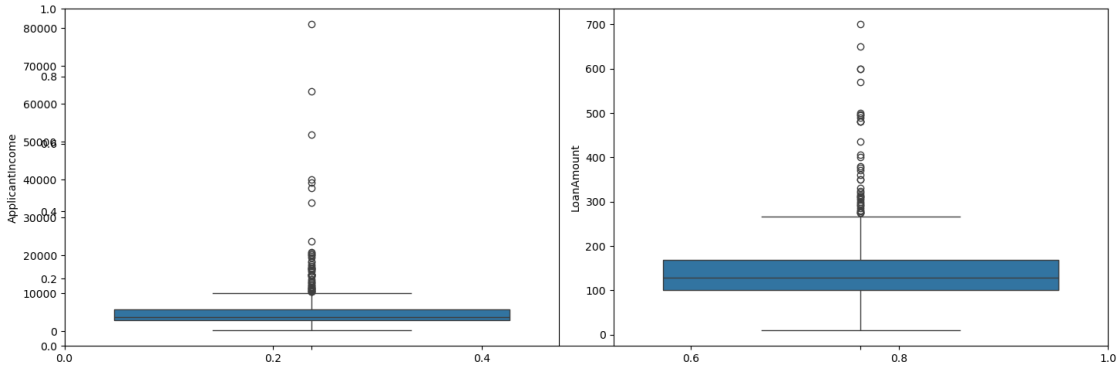
[169]: <function matplotlib.pyplot.show(close=None, block=None)>



5 find out the outliers in the columns we can use boxplot

[171]:
```
plt.subplots(figsize=(15 ,5))
for i, col in enumerate(["ApplicantIncome","LoanAmount"]):
    plt.subplot(1,2,i+1)
    sb.boxplot(df[col])
plt.tight_layout()
plt.show
```

[171]: <function matplotlib.pyplot.show(close=None, block=None)>



6 There are some extreme outlier's in the data we need to remove them

[173]:
```
df = df[df["ApplicantIncome"]<25000]
df = df[df["LoanAmount"]<40000]
```

7 mean of the loan granted to male as well as females.for that, we will use groupby(). method

[175]:
```
df.groupby("Gender") .mean(["LoanAmount"])
```

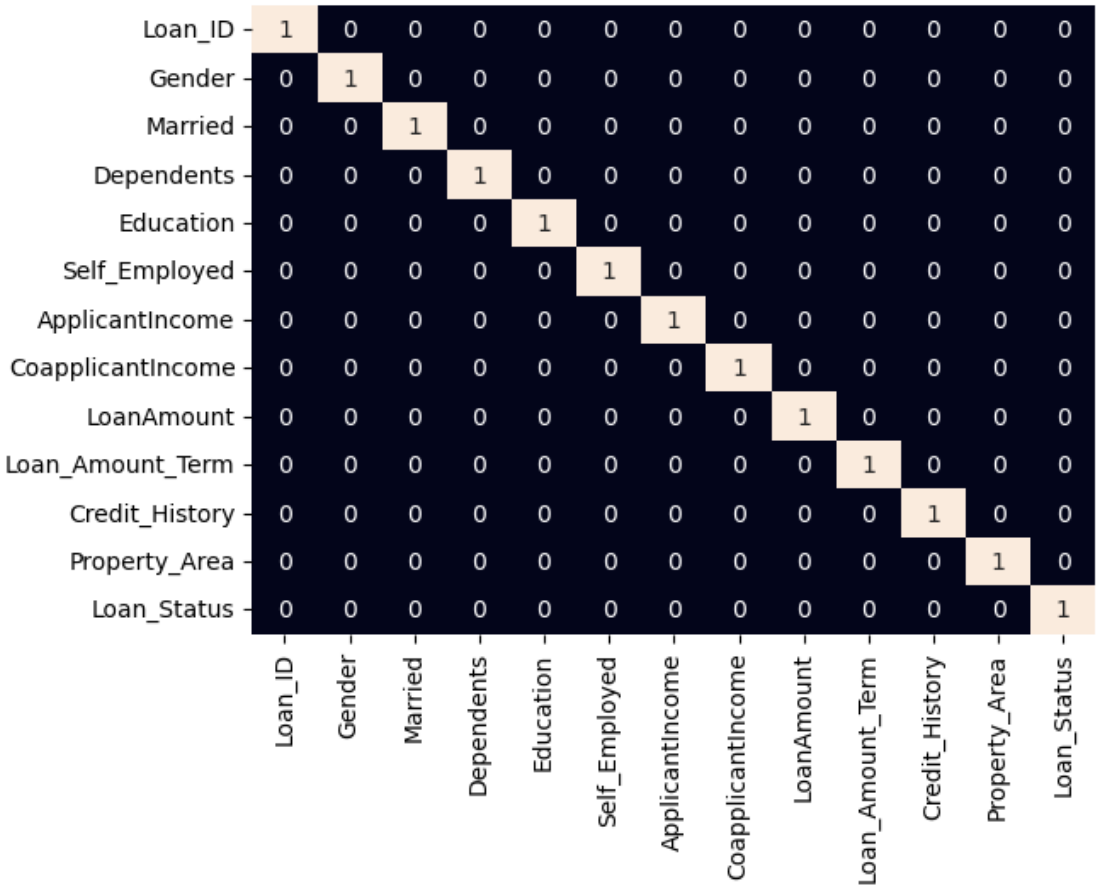
[175]:
```
        ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
Gender
Female      4593.954128        1138.504587  126.697248        353.207547
Male        4890.566810        1777.021379  146.924569        340.132450

        Credit_History
Gender
Female        0.836735
Male          0.850117
```

[177]:
```
df.groupby(["Married","Gender"])
```

[177]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x16a14c5f0>
```

[131]:
```
# Funcctions to apply label encoding
def encode_labels(data):
    for col in data.columns:
        if data[col].dtype == 'object':
            le = LabelEncoder()
            data[col] = le.fit_transform(data[col])
```

```
    return data

# Applying function in whole column
df = encode_labels(df)

# Generating Heatmap
sb.heatmap(df.corr() > 0.8, annot=True, cbar=False)
plt.show()
```



[225]:
```
features = df.drop('Loan_Status', axis=1)
target = df['Loan_Status'].values

X_train, X_val,\
    Y_train, Y_val = train_test_split(features, target, test_size=0.2,␣
 ↪random_state=10)

# As the data was highly imbalanced we will balance
# it by adding repetitive rows of minority class.
```

```
ros = RandomOverSampler(sampling_strategy='minority',
                        random_state=0)
X, Y = ros.fit_resample(X_train, Y_train)

X_train.shape, X.shape
```

[225]:
```
((468, 12), (666, 12))
```

[239]:
```
from sklearn.metrics import roc_auc_score
model = SVC(kernel='rbf')
model .fit(X.Y)

print('Training Accuracy : ', metrics.roc_auc_score(Y, model.predict(X)))
print('Validation Accuracy : ', metrics.roc_auc_score(Y_val, model.
 ↪predict(X_val)))
print()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/3s/vmbvn6xj4ld_5hnrk6lcfxkh0000gn/T/ipykernel_756/708598151.py in
 ↪()
      1 from sklearn.metrics import roc_auc_score
      2 model = SVC(kernel='rbf')
----> 3 model .fit(X.Y)
      4
      5 print('Training Accuracy : ', metrics.roc_auc_score(Y, model.predict(X)) )
      6 print('Validation Accuracy : ', metrics.roc_auc_score(Y_val, model.
 ↪predict(X_val)))

/opt/anaconda3/lib/python3.12/site-packages/pandas/core/generic.py in ?(self,␣
 ↪name)
   6295             and name not in self._accessors
   6296             and self._info_axis.
 ↪_can_hold_identifiers_and_holds_name(name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'Y'
```

[149]:
```
plt.figure(figsize=(6, 6))
sb.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[149], line 2
      1 plt.figure(figsize=(6, 6))
----> 2 sb.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
      3 plt.title('Confusion Matrix')
      4 plt.xlabel('Predicted Label')

NameError: name 'cm' is not defined
```

```
<Figure size 600x600 with 0 Axes>
```

[217]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_val , model.predict(X_val)))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/3s/vmbvn6xj4ld_5hnrk6lcfxkh0000gn/T/ipykernel_756/1183266439.py in ?
 ↪()
      1 from sklearn.metrics import classification_report
----> 2 print(classification_report(Y_val , model.predict(X_val)))

/opt/anaconda3/lib/python3.12/site-packages/sklearn/svm/_base.py in ?(self, X)
    810             and len(self.classes_) > 2
    811         ):
    812             y = np.argmax(self.decision_function(X), axis=1)
    813         else:
--> 814             y = super().predict(X)
    815         return self.classes_.take(np.asarray(y, dtype=np.intp))

/opt/anaconda3/lib/python3.12/site-packages/sklearn/svm/_base.py in ?(self, X)
    425         -------
    426         y_pred : ndarray of shape (n_samples,)
    427             The predicted values.
    428         """
--> 429         X = self._validate_for_predict(X)
    430         predict = self._sparse_predict if self._sparse else self.
 ↪_dense_predict
    431         return predict(X)

/opt/anaconda3/lib/python3.12/site-packages/sklearn/svm/_base.py in ?(self, X)
    603     def _validate_for_predict(self, X):
    604         check_is_fitted(self)
    605
    606         if not callable(self.kernel):
```

```
--> 607             X = self._validate_data(
    608                 X,
    609                 accept_sparse="csr",
    610                 dtype=np.float64,

/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py in ?(self, X, y,␣
 ↪reset, validate_separately, cast_to_ndarray, **check_params)
    629                 out = y
    630             else:
    631                 out = X, y
    632         elif not no_val_X and no_val_y:
--> 633             out = check_array(X, input_name="X", **check_params)
    634         elif no_val_X and not no_val_y:
    635             out = _check_y(y, **check_params)
    636         else:

/opt/anaconda3/lib/python3.12/site-packages/sklearn/utils/validation.py in ?
 ↪(array, accept_sparse, accept_large_sparse, dtype, order, copy,␣
 ↪force_all_finite, ensure_2d, allow_nd, ensure_min_samples,␣
 ↪ensure_min_features, estimator, input_name)
    994                         )
    995                     array = xp.astype(array, dtype, copy=False)
    996                 else:
    997                     array = _asarray_with_order(array, order=order,␣
 ↪dtype=dtype, xp=xp)
--> 998             except ComplexWarning as complex_warning:
    999                 raise ValueError(
   1000                     "Complex data not supported\n{}\n".format(array)
   1001                 ) from complex_warning

/opt/anaconda3/lib/python3.12/site-packages/sklearn/utils/_array_api.py in ?
 ↪(array, dtype, order, copy, xp)
    517         # Use NumPy API to support order
    518         if copy is True:
    519             array = numpy.array(array, order=order, dtype=dtype)
    520         else:
--> 521             array = numpy.asarray(array, order=order, dtype=dtype)
    522
    523         # At this point array is a NumPy ndarray. We convert it to an␣
 ↪array
    524         # container that is consistent with the input's namespace.

/opt/anaconda3/lib/python3.12/site-packages/pandas/core/generic.py in ?(self,␣
 ↪dtype, copy)
   2149     def __array__(
   2150         self, dtype: npt.DTypeLike | None = None, copy: bool_t | None =␣
 ↪None
   2151     ) -> np.ndarray:
```

```
   2152         values = self._values
-> 2153         arr = np.asarray(values, dtype=dtype)
   2154         if (
   2155             astype_is_view(values.dtype, arr.dtype)
   2156             and using_copy_on_write()

ValueError: could not convert string to float: 'LP002435'
```

[13]:
```
df = pd.read_csv("Downloads/loan-test.csv")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[13], line 1
----> 1 df = pd.read_csv("Downloads/loan-test.csv")

File /opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪1026, in read_csv(filepath_or_buffer, sep, delimiter, header, names,␣
 ↪index_col, usecols, dtype, engine, converters, true_values, false_values,␣
 ↪skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,␣
 ↪na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format,␣
 ↪keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator,␣
 ↪chunksize, compression, thousands, decimal, lineterminator, quotechar,␣
 ↪quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect␣
 ↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
 ↪storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (…)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File /opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File /opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)
```

```
File /opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.py:
 ↪1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File /opt/anaconda3/lib/python3.12/site-packages/pandas/io/common.py:873, in␣
 ↪get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'Downloads/loan-test.cs '
```

[ ]: