6eeonjrsj

February 2, 2025

1 Machine Learning Part 1

[ ]:

[64]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler

data = pd.read_csv('/Users/ratnadeepgurav/Desktop/AIDS/Study for carrer␣
 ↪material/Project/WScube_ML_learning/loan.csv')

print(data.head(5))
```

```
    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
```

2 Find no of NULL values and percentage

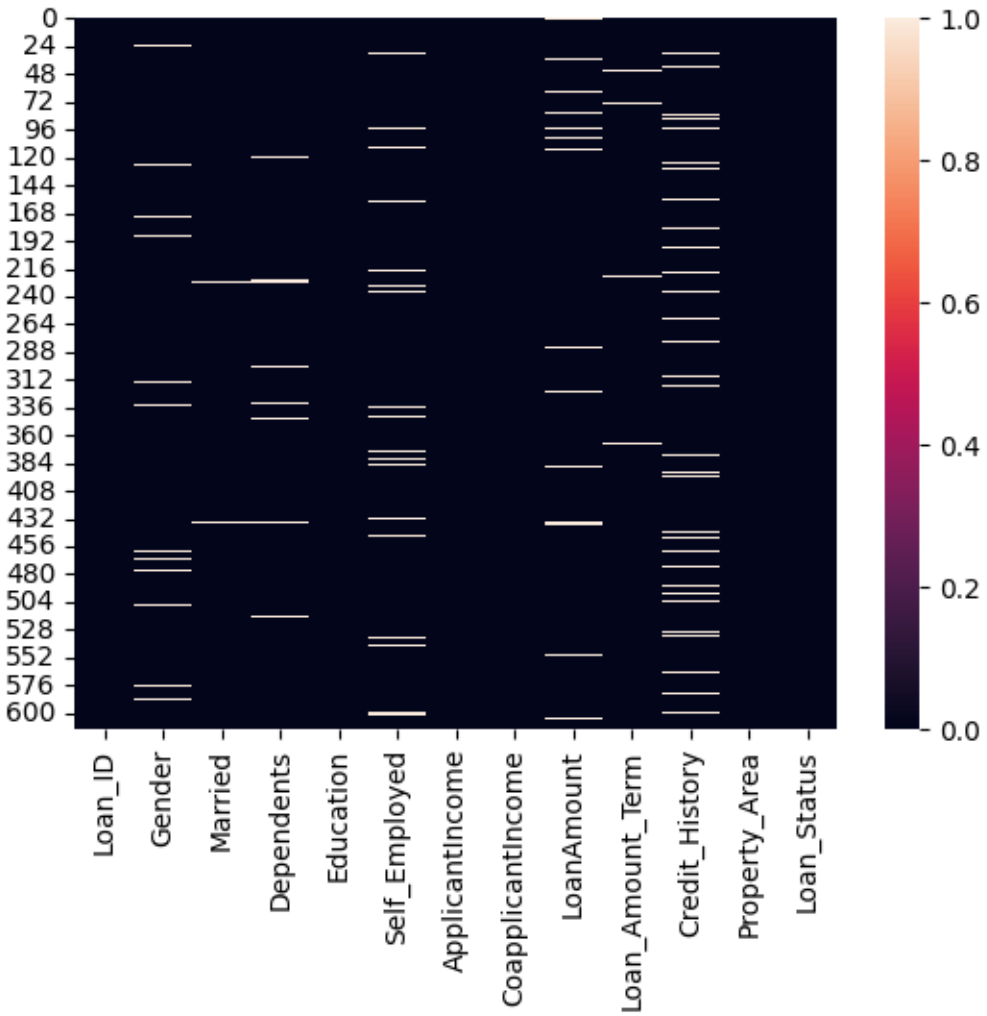
[63]:
```
print("\n\n Total Rows and columns :- ",data.shape,"\n\n")

print(data.isnull().sum())

print("\n\n Null Values in Percentage :- \n\n",(data.isnull().sum()/data.
 ↪shape[0])*100)

print("\n\n Overall dataset Null value Percentage :- \n\n ",(data.isnull().
 ↪sum().sum()/ (data.shape[0] * data.shape[1])) * 100)

print("\n\n Not Null values :- \n\n ",data.notnull().sum())

print("\n\n Sum of Not Null values :- \n\n ",data.notnull().sum().sum())
```

```
 Total Rows and columns :-  (614, 12)


Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Property_Area         0
Loan_Status           0
dtype: int64


 Null Values in Percentage :-

 Loan_ID              0.000000
Gender               2.117264
Married              0.488599
Dependents           2.442997
Education            0.000000
Self_Employed        5.211726
ApplicantIncome      0.000000
```

```
CoapplicantIncome    0.000000
LoanAmount           3.583062
Loan_Amount_Term     2.280130
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64


 Overall dataset Null value Percentage :-

  1.3436482084690555


 Not Null values :-

  Loan_ID              614
Gender                 601
Married                611
Dependents             599
Education              614
Self_Employed          582
ApplicantIncome        614
CoapplicantIncome      614
LoanAmount             592
Loan_Amount_Term       600
Property_Area          614
Loan_Status            614
dtype: int64


 Sum of Not Null values :-

  7269
```

3 Visualize null values

[7]:
```
sns.heatmap(data.isnull())
plt.show()
```



4 Drop columns and change in dataset

[66]:
```
data.drop(columns = ["Credit_History"],inplace=True,errors = 'ignore')
```

[11]:
```
data.fillna(method = 'bfill',axis = 1)
```

```
/var/folders/c_/rbrshmgx64b9ch2skklfhfbw0000gn/T/ipykernel_1475/2508724752.py:1:
FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  data.fillna(method = 'bfill',axis = 1)
```

[11]:

	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      360.0             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]
```

5 Null Value Replace with data

[16]:
```
data['Gender'].fillna(data['Gender'].mode()[0],inplace = True)
```

[19]:
```
for i in data.select_dtypes(include = 'object').columns:
    data[i].fillna(data[i].mode()[0],inplace = True)
```

```
/var/folders/c_/rbrshmgx64b9ch2skklfhfbw0000gn/T/ipykernel_1475/3899513592.py:2:
```

```
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.


  data[i].fillna(data[i].mode()[0],inplace = True)
```

[24]:
```
data.isnull().sum()
```

[24]:
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

6 Fill Null values by SimpleImputer

[37]:
```
si = SimpleImputer(strategy = 'mean')

ar = si.fit_transform(data[['LoanAmount','Loan_Amount_Term','Credit_History']])

new_data = pd.DataFrame(ar,columns =␣
 ↪['LoanAmount','Loan_Amount_Term','Credit_History'])



print(new_data.isnull().sum())
```

```
LoanAmount          0
Loan_Amount_Term    0
Credit_History      0
dtype: int64
```

7 One Hot Encoding

[84]:
```
en_data = data[['Gender','Married']]

aa =pd.get_dummies(en_data).info()


# Initialize the OneHotEncoder
ohe = OneHotEncoder()

# Fit and transform the data
transformed_data = ohe.fit_transform(data[['Gender', 'Married']]).toarray()

# Retrieve the feature names generated by the encoder
feature_names = ohe.get_feature_names_out(['Gender', 'Married'])

# Create the DataFrame with the transformed data
new_data = pd.DataFrame(transformed_data, columns=feature_names)

print(new_data)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Gender_Female  614 non-null    bool
 1   Gender_Male    614 non-null    bool
 2   Married_No     614 non-null    bool
 3   Married_Yes    614 non-null    bool
dtypes: bool(4)
memory usage: 2.5 KB
     Gender_Female  Gender_Male  Gender_nan  Married_No  Married_Yes  \
0              0.0          1.0         0.0         1.0          0.0
1              0.0          1.0         0.0         0.0          1.0
2              0.0          1.0         0.0         0.0          1.0
3              0.0          1.0         0.0         0.0          1.0
4              0.0          1.0         0.0         1.0          0.0
..             …            …           …           …            …
609            1.0          0.0         0.0         1.0          0.0
610            0.0          1.0         0.0         0.0          1.0
611            0.0          1.0         0.0         0.0          1.0
612            0.0          1.0         0.0         0.0          1.0
613            1.0          0.0         0.0         1.0          0.0

     Married_nan
0            0.0
```

```
1            0.0
2            0.0
3            0.0
4            0.0
..           …
609          0.0
610          0.0
611          0.0
612          0.0
613          0.0

[614 rows x 6 columns]
```

8 Lable Encoding

[76]:
```
df = pd.DataFrame({"Name" : ['Cow','Rat','Dog','Cat','Black']})

le = LabelEncoder()
df["Encoding"] = le.fit_transform(df['Name'])

print(df)
```

```
    Name  Encoding
0    Cow         2
1    Rat         4
2    Dog         3
3    Cat         1
4  Black         0
```

9 Ordinal Encoding using Sklearn

[94]:
```
df = pd.DataFrame({ "Size" :␣
 ↪['m','xl','s','xxl','l','s','l','xl','m','s','xxl'] })

df.head(3)

ord_data = [['s','m','l','xl','xxl']]

oe = OrdinalEncoder(categories = ord_data)

df["Encoding"] = oe.fit_transform(df[["Size"]])


print(df)
```

```
   Size  Encoding
```

```
0     m       1.0
1    xl       3.0
2     s       0.0
3   xxl       4.0
4     l       2.0
5     s       0.0
6     l       2.0
7    xl       3.0
8     m       1.0
9     s       0.0
10  xxl       4.0
```

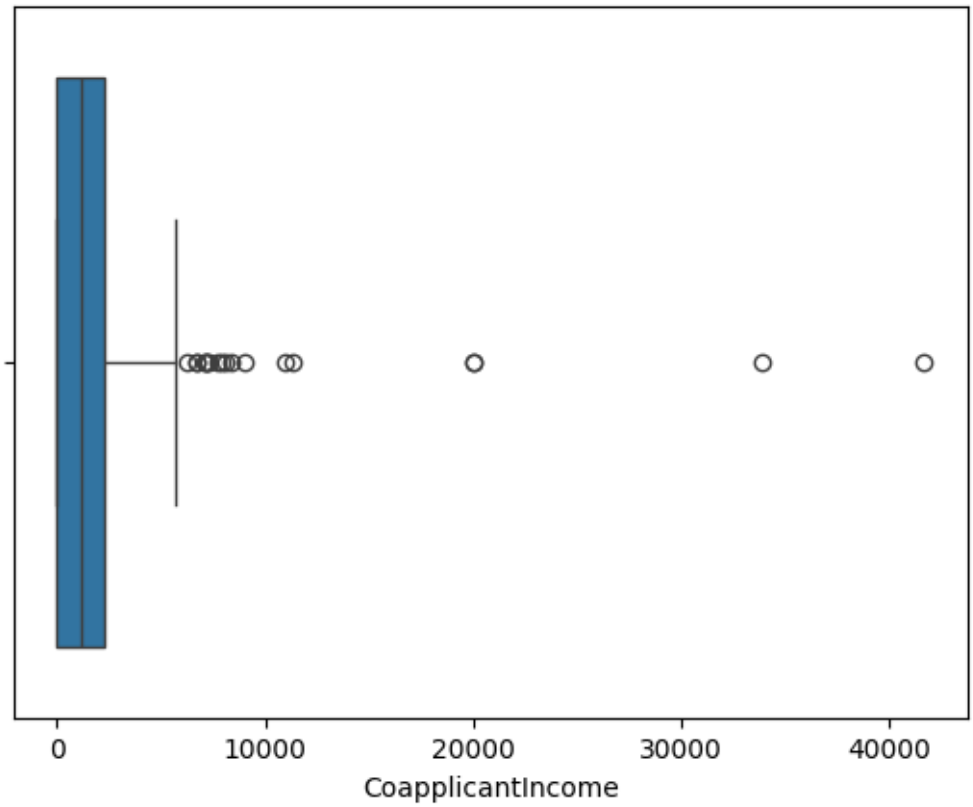
10 Ordinal Encoding using map function

[3]:
```
df = pd.DataFrame({ "Size" :␣
 ↪['m','xl','s','xxl','l','s','l','xl','m','s','xxl'] })

ord_data1 = {"s":0,"m":1,"l":2,"xl":3,"xxl":4}

df["Map_Encoding"] = df["Size"].map(ord_data1)

print(df)
```

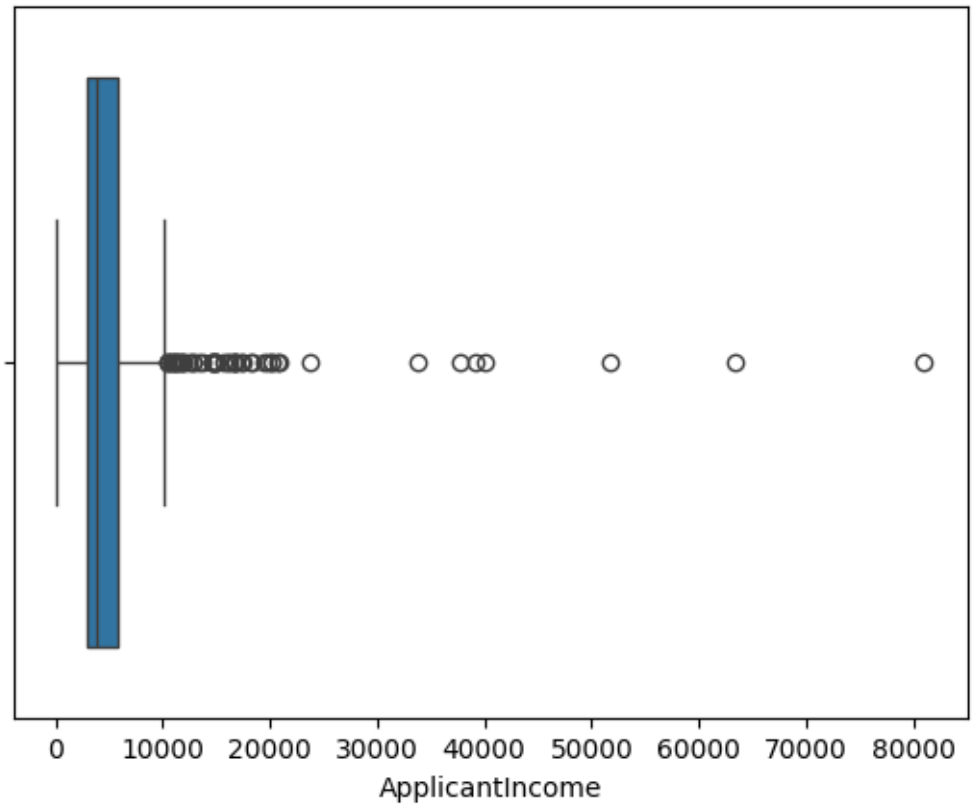
```
   Size  Map_Encoding
0     m             1
1    xl             3
2     s             0
3   xxl             4
4     l             2
5     s             0
6     l             2
7    xl             3
8     m             1
9     s             0
10  xxl             4
```

11 Visualize Data

[5]:
```
sns.boxplot(x = 'CoapplicantIncome', data =data )
plt.show()
```

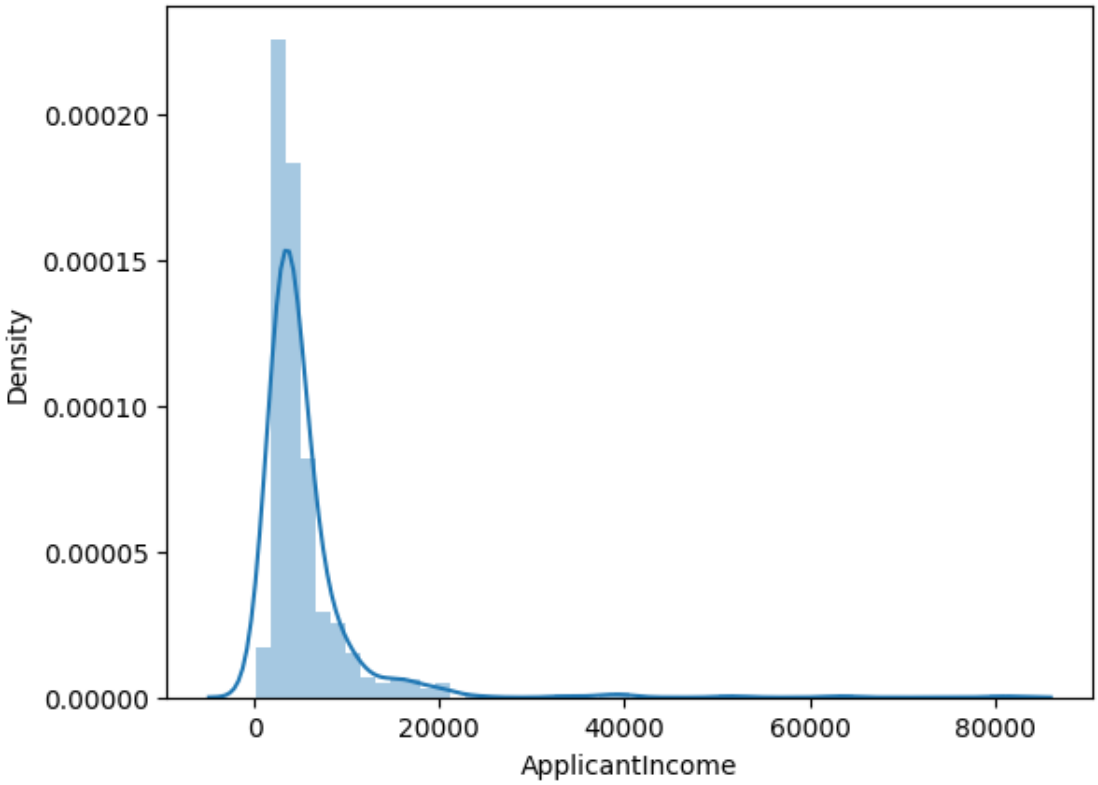


[7]:
```
sns.boxplot(x = 'ApplicantIncome', data =data )
plt.show()
```



[15]:
```
sns.distplot(data['ApplicantIncome'])
plt.show()
```

```
/var/folders/c_/rbrshmgx64b9ch2skklfhfbw0000gn/T/ipykernel_1260/2841061266.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['ApplicantIncome'])
```



12 Remove Outlier

[18]:
```
q1 = data['CoapplicantIncome'].quantile(0.25)
q1
```

[18]: np.float64(0.0)

[22]:
```
q3 = data['CoapplicantIncome'].quantile(0.75)
q3
```

[22]: np.float64(2297.25)

[25]:
```
IQR = q3 - q1
```

[34]:
```
min_range = q1 - (1.5 * IQR)
max_range = q3 + (1.5 * IQR)

min_range , max_range
```

[34]: (np.float64(-3445.875), np.float64(5743.125))

[36]:
```
data.describe()
```

[36]:

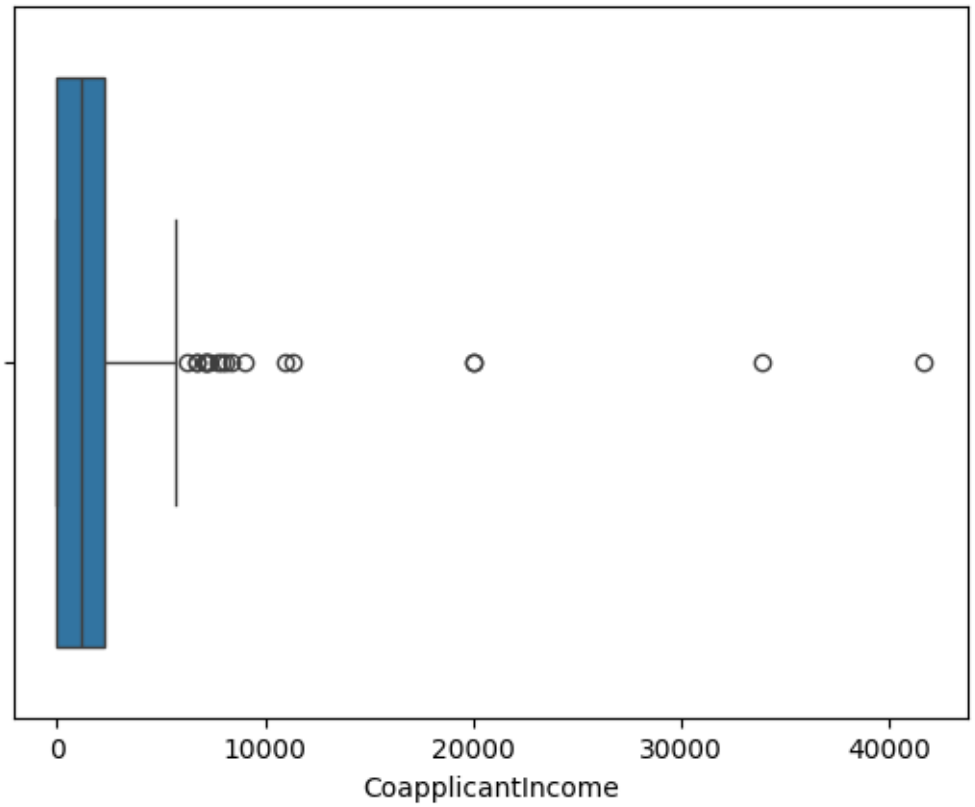
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
count	614.000000	614.000000	592.000000	600.00000	
mean	5403.459283	1621.245798	146.412162	342.00000	
std	6109.041673	2926.248369	85.587325	65.12041	
min	150.000000	0.000000	9.000000	12.00000	
25%	2877.500000	0.000000	100.000000	360.00000	
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

[41]:
```
new_data = data[data['CoapplicantIncome']<=max_range]
new_data.shape
```

[41]:
```
(596, 13)
```

[44]:
```
sns.boxplot(x = 'CoapplicantIncome', data =data )
plt.show()
```



[47]:
```
z_force = (data['CoapplicantIncome'] - data['CoapplicantIncome'].mean())/
  ↪(data['CoapplicantIncome'].std())
z_force
```

[47]:
```
0     -0.554036
1     -0.038700
2     -0.554036
3      0.251774
4     -0.554036
         …
609   -0.554036
610   -0.554036
611   -0.472019
612   -0.554036
613   -0.554036
Name: CoapplicantIncome, Length: 614, dtype: float64
```

[50]:
```
data["z_force"] = z_force

data[data['z_force']<3]
z_force.shape
```

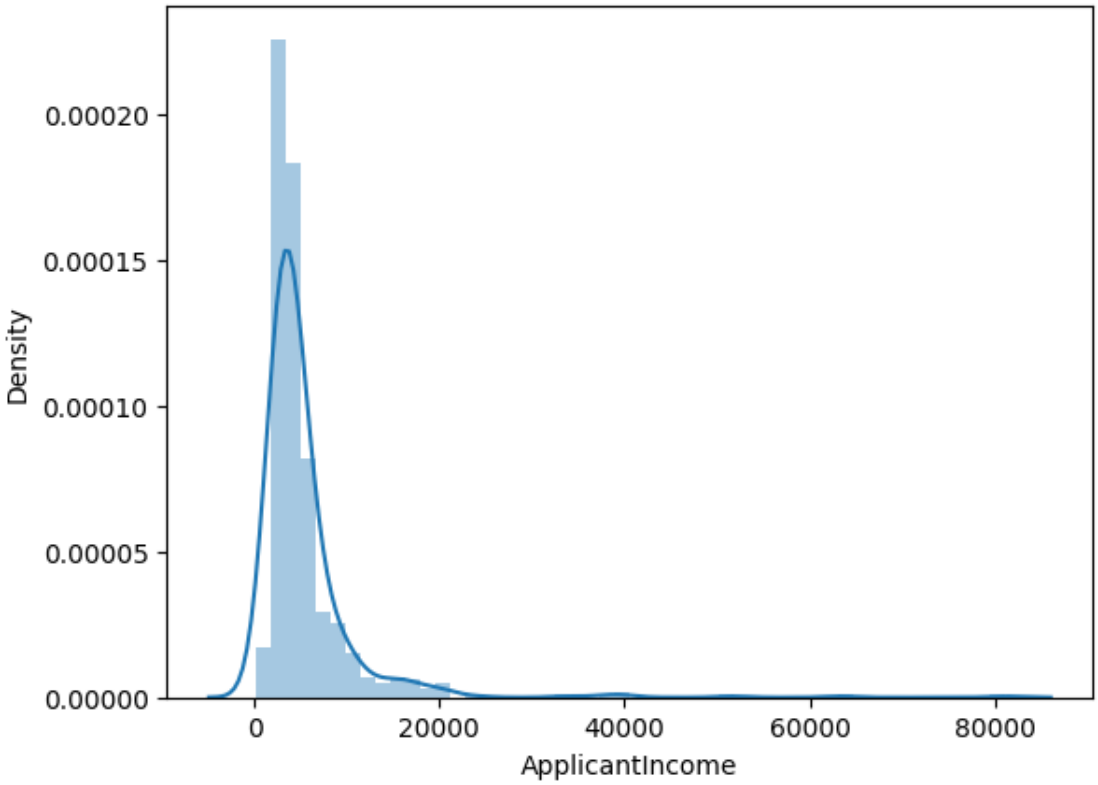
[50]: (614,)

[60]:
```
data["ApplicantIncome"].fillna(data["ApplicantIncome"].mean())
```

[60]:
```
0      5849
1      4583
2      3000
3      2583
4      6000
       …
609    2900
610    4106
611    8072
612    7583
613    4583
Name: ApplicantIncome, Length: 614, dtype: int64
```

[65]:
```
sns.distplot(data['ApplicantIncome'])
plt.show()
```

```
/var/folders/c_/rbrshmgx64b9ch2skklfhfbw0000gn/T/ipykernel_1260/2841061266.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['ApplicantIncome'])
```



[70]:
```
ss = StandardScaler()
ss.fit(data[['ApplicantIncome']])

data['ApplicantIncome ss'] = pd.DataFrame(ss.
  ↪transform(data[['ApplicantIncome']]),columns = ["x"])

print(data.head(3))

sns.histplot(data['ApplicantIncome'])
plt.show()
```

```
    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

   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

   Credit_History Property_Area Loan_Status  ApplicantIncome ss
```

0	1.0	Urban	Y	0.072991
1	1.0	Rural	N	-0.134412
2	1.0	Urban	Y	-0.393747