

Importing The Libraries

```
In [...]  
import numpy as np  
import pandas as pd  
import pickle  
import seaborn as sns  
import matplotlib.pyplot as plt  
%matplotlib inline  
import sklearn  
from sklearn.preprocessing import LabelEncoder  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model_selection import RandomizedSearchCV  
from xgboost import XGBClassifier  
from sklearn.ensemble import RandomForestClassifier  
import imblearn  
from imblearn.under_sampling import RandomUnderSampler  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import scale  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Reading The Dataset

```
In [4]: df=pd.read_csv('Loan_dataset.csv')  
df
```

```
O...
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Lo
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	
...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	

614 rows × 13 columns

```
In [5]: df.info()
```

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

```
In [6]: df.shape
```

Out[6]:(614, 13)

```
In [7]: df=df.drop(columns=["Loan_ID"],axis=1)
```

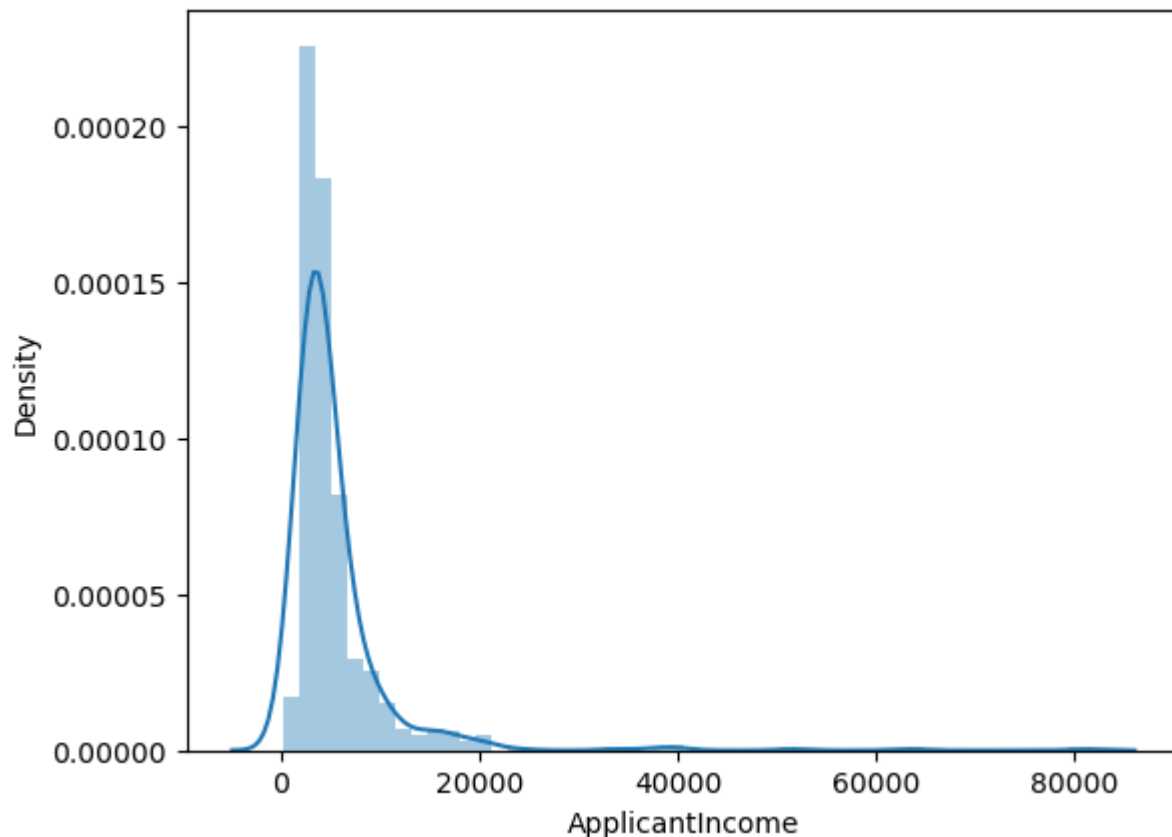
Uni-Variate Analysis

```
In [90]: sns.distplot(df.ApplicantIncome)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `dist plot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an a xes-level function for histograms).

warnings.warn(msg, FutureWarning)

Out[90]:

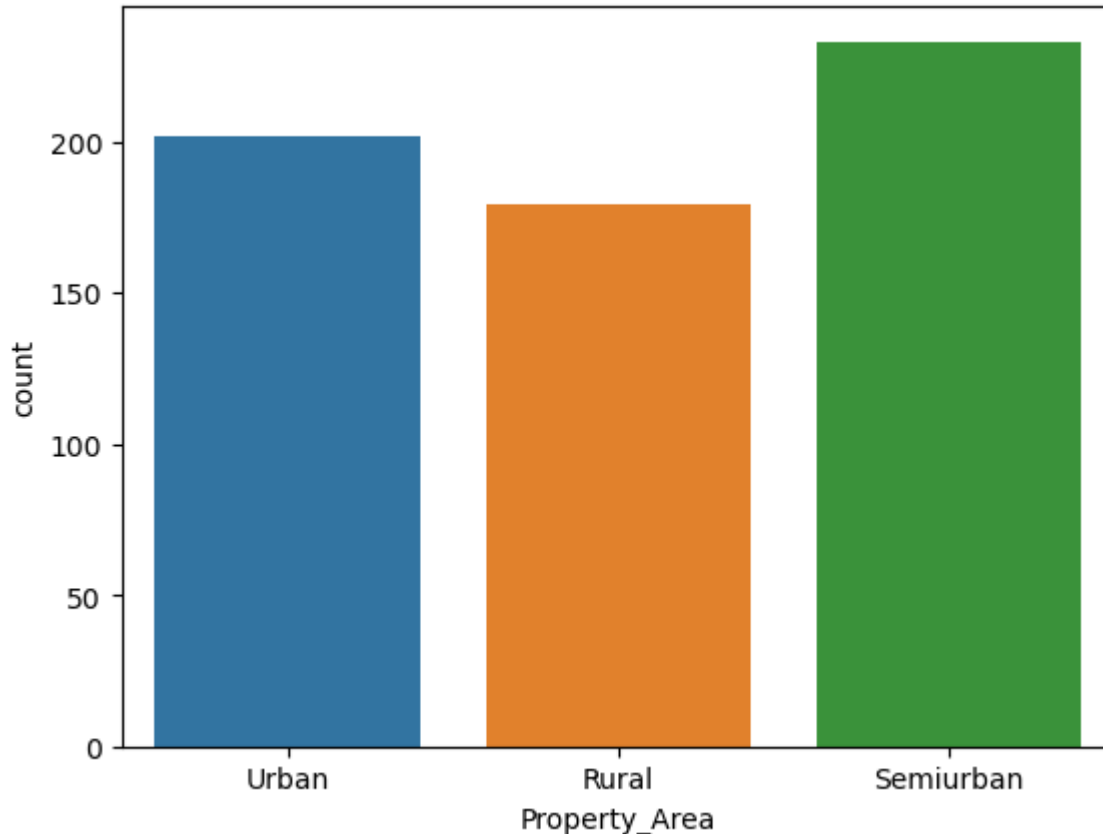


```
In [91]: sns.countplot(df.Property_Area)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[91]:
```

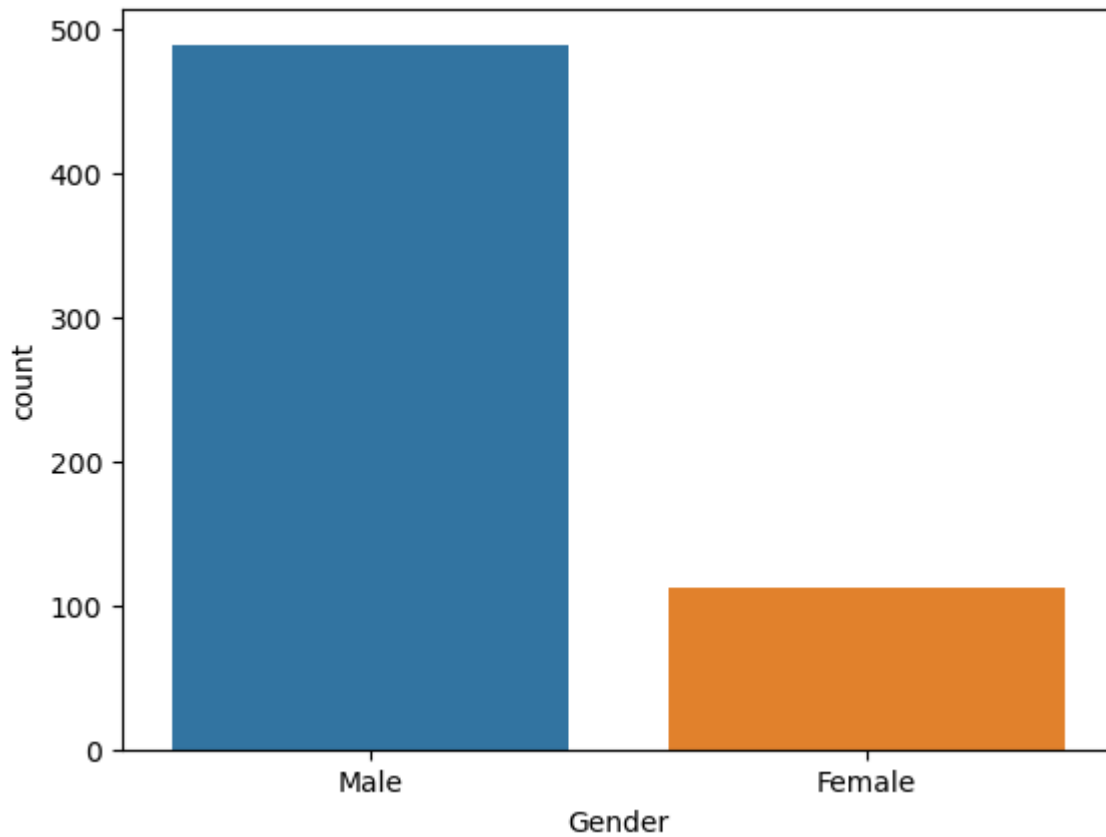


```
In [92]: sns.countplot(df.Gender)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[92]:
```

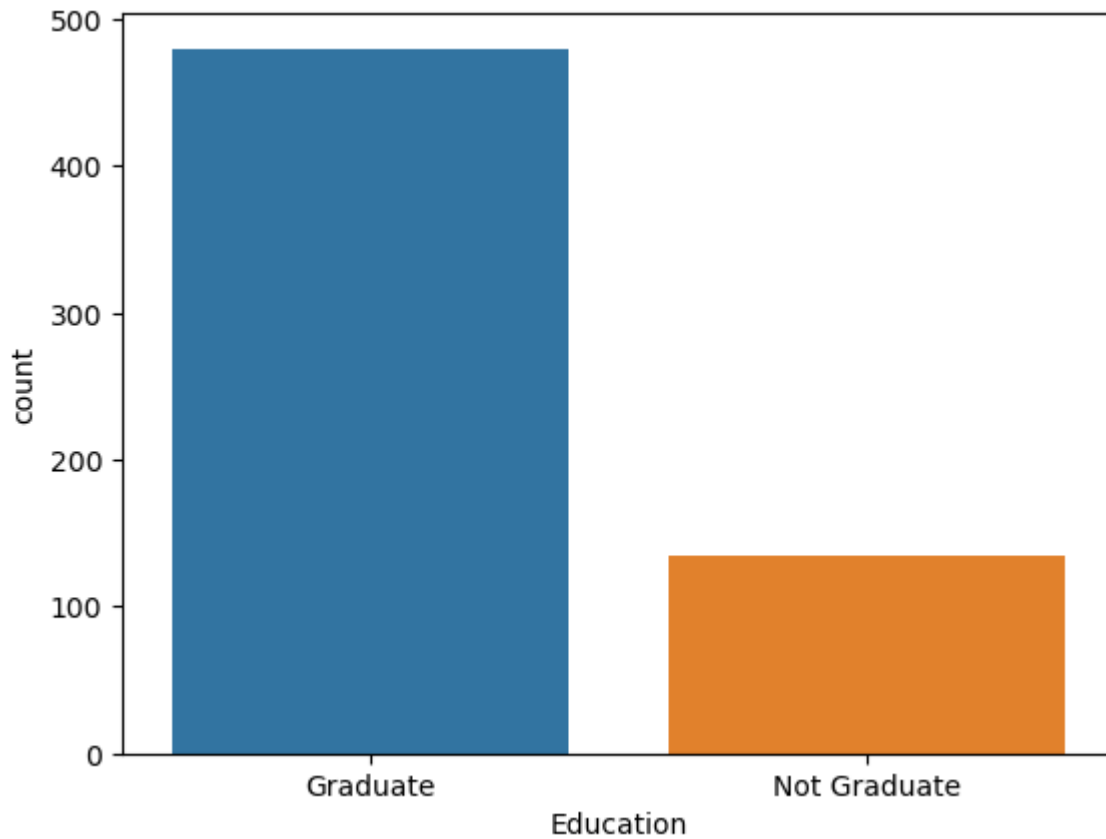


```
In [93]: sns.countplot(df.Education)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[93]:
```

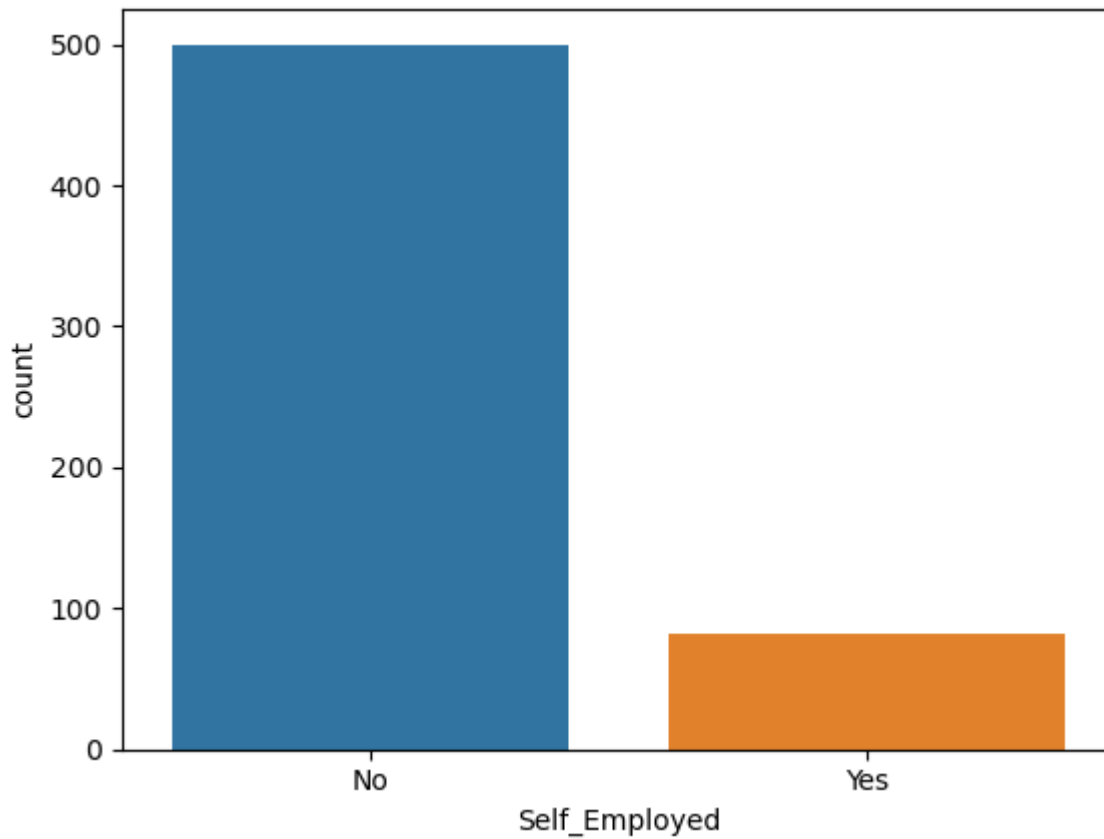


```
In [94]: sns.countplot(df.Self_Employed)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[94]:
```

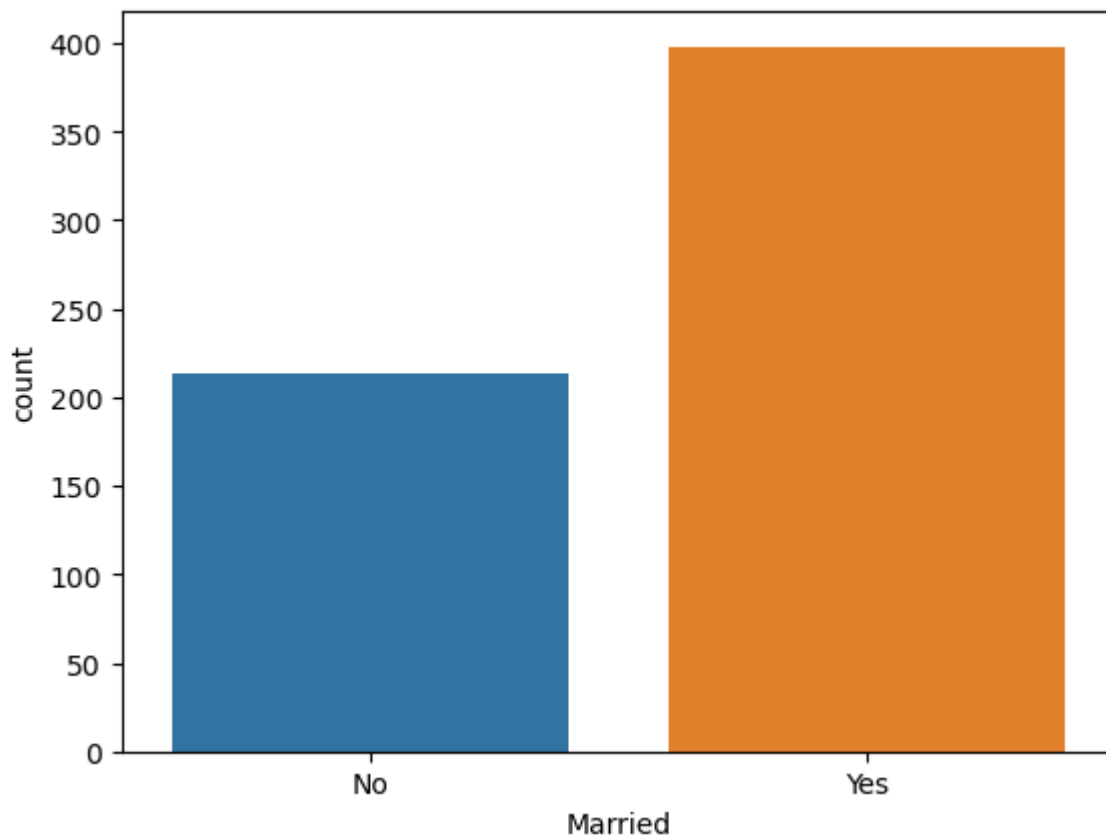


```
In [95]: sns.countplot(df.Married)
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

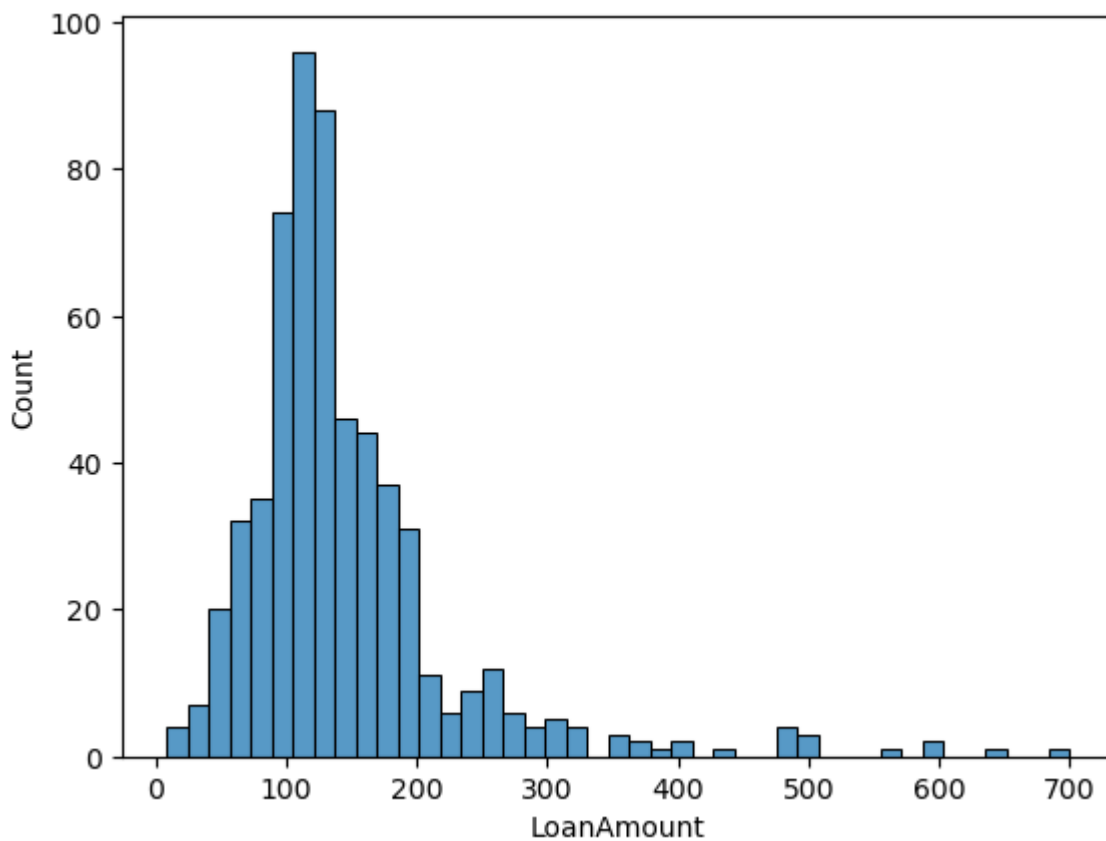
```
warnings.warn(
```

```
Out[95]:
```



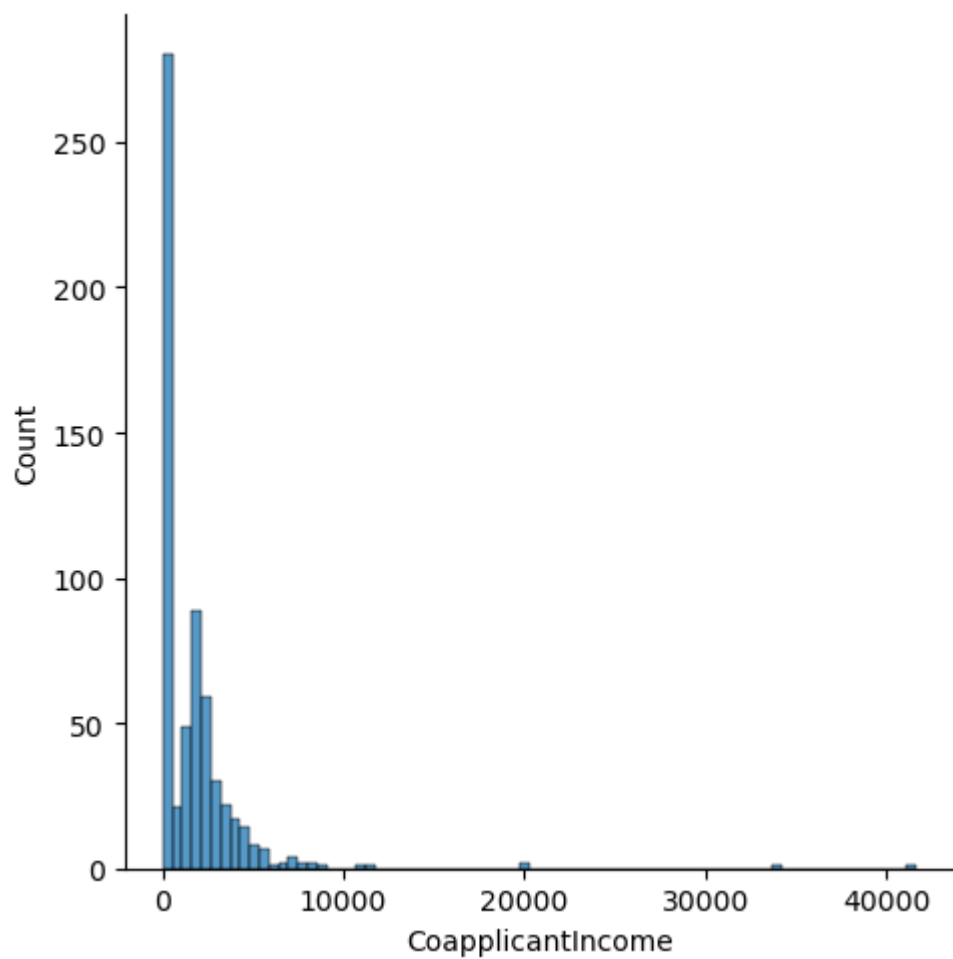
In [96]: `sns.histplot(df.LoanAmount)`

Out[96]:



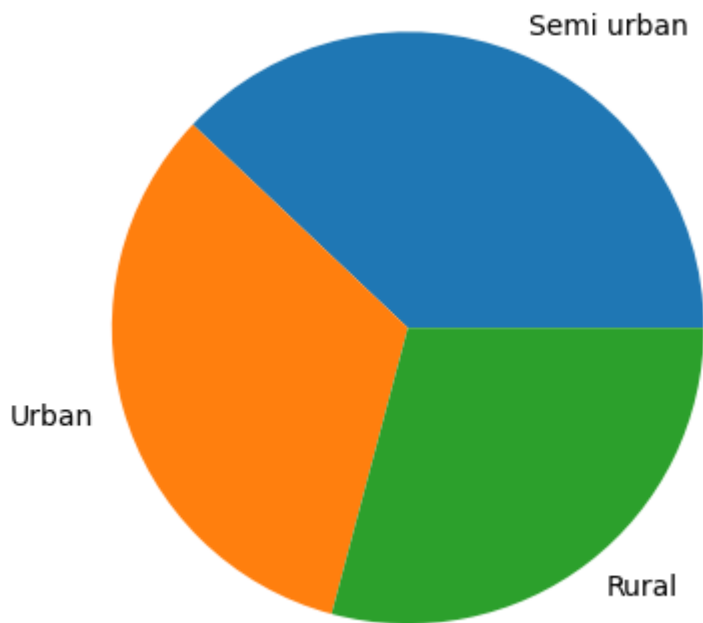
In [97]: `sns.displot(df.CoapplicantIncome)`

Out[97]:



In [98]: `plt.pie(df.Property_Area.value_counts(),[0,0,0],labels=['Semi urban','Urban','Rural'])`

Out[98]: ([,],
[Text(0.40661098511372595, 1.0220897743275028, 'Semi urban'),
Text(-1.0582795633383781, -0.3000739339235115, 'Urban'),
Text(0.67000963198199, -0.8724030565348555, 'Rural')])



```
In [99]: df.head()
```

Out...	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	No	0	Graduate	No	5849	0.0	NaN
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0
4	Male	No	0	Graduate	No	6000	0.0	141.0

```
In [100]: df.Loan_Status.value_counts()
```

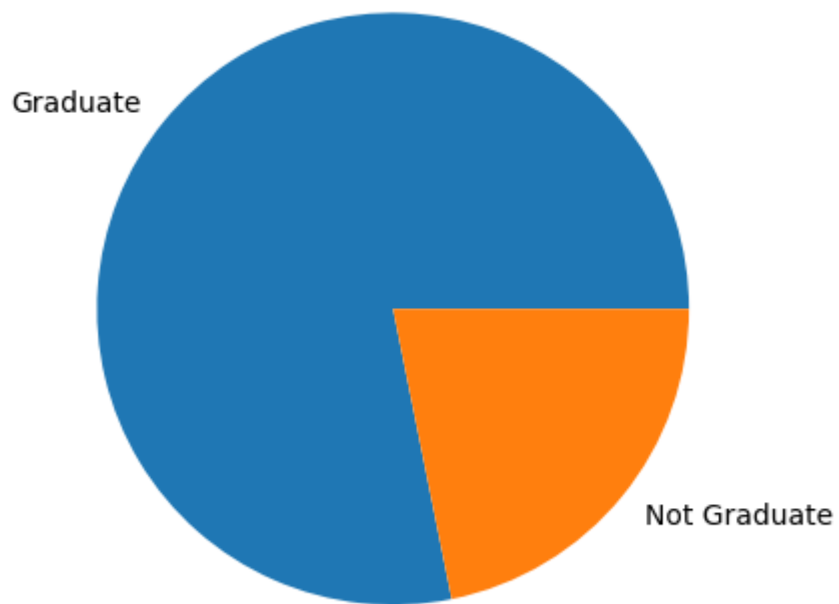
```
Out[100]: Y    422
          N    192
          Name: Loan_Status, dtype: int64
```

```
In [101]: df.Credit_History.value_counts()
```

```
Out[101]: 1.0    475
          0.0    89
          Name: Credit_History, dtype: int64
```

```
In [102]: plt.pie(df.Education.value_counts(),[0,0],labels=['Graduate','Not Graduate'])
```

```
Out[102]: ([
],
[Text(-0.8514262161117528, 0.6964721089301588, 'Graduate'),
Text(0.8514262161117524, -0.6964721089301593, 'Not Graduate')])
```



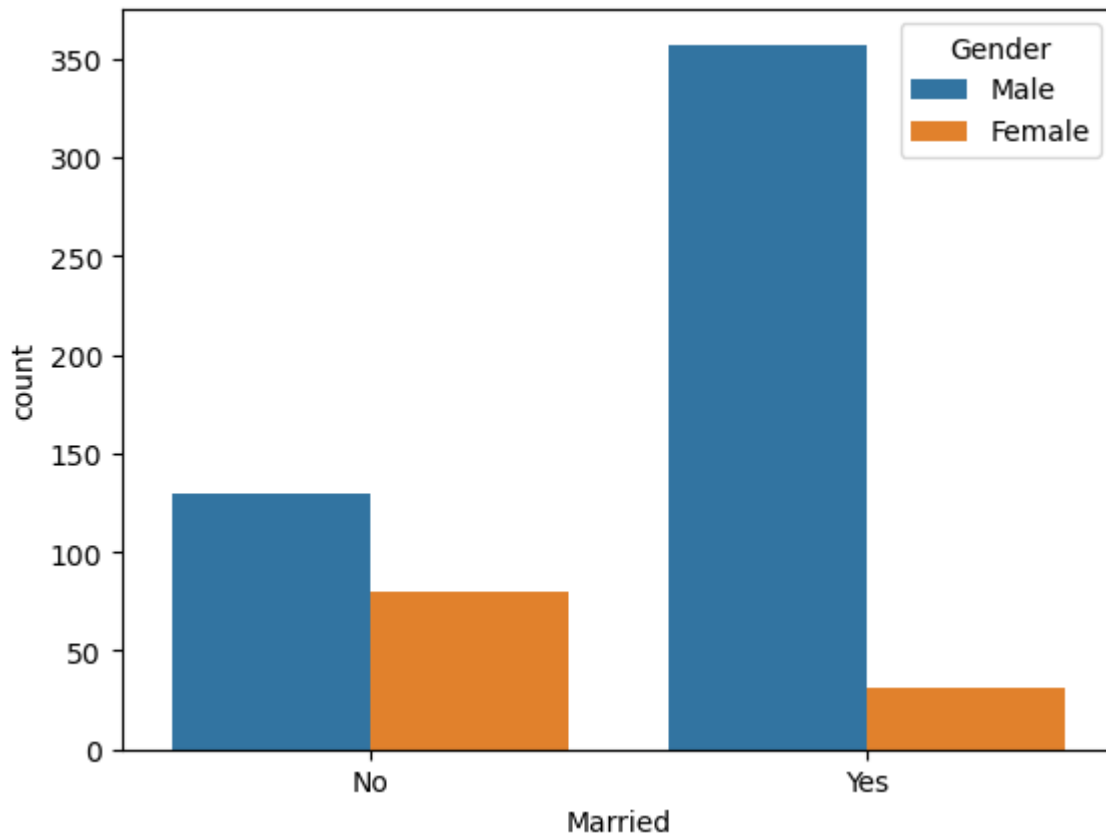
Bivariate Analysis

```
In [103]: sns.countplot(df['Married'],hue=df['Gender'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[103]:
```

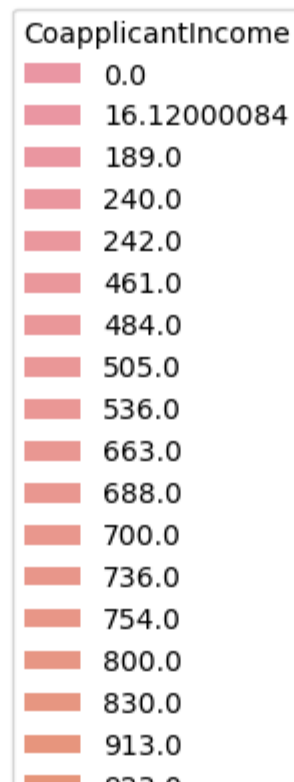


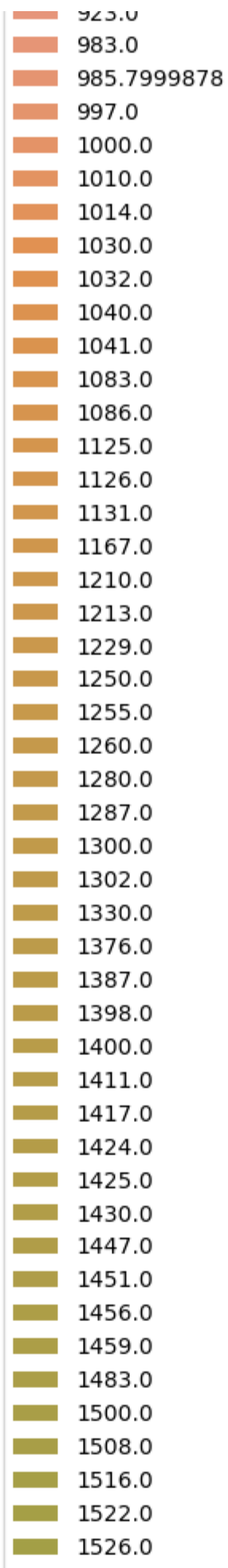
In [104]: `sns.countplot(df['ApplicantIncome'], hue=df['CoapplicantIncome'])`

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[104]:



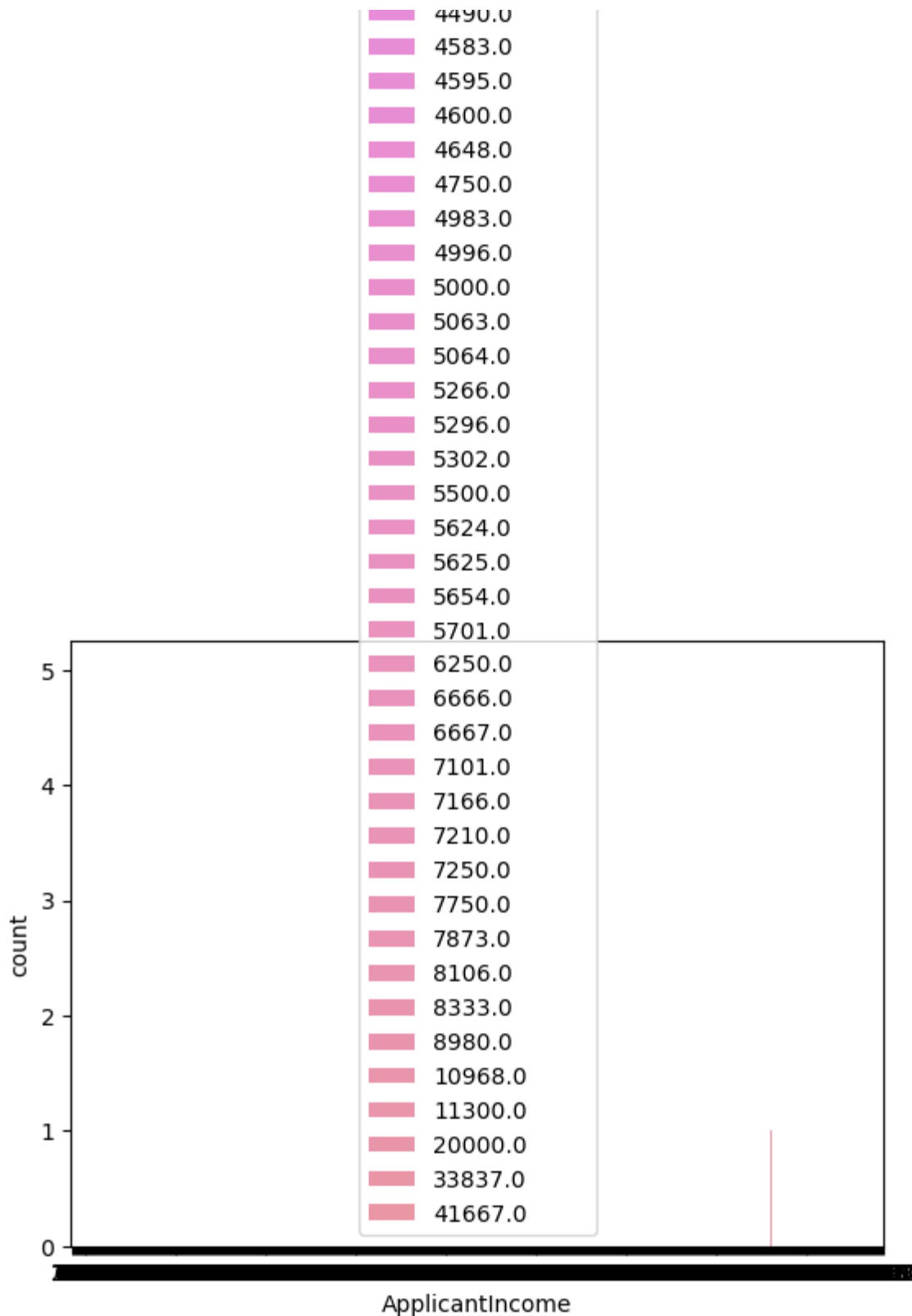


■	1542.0
■	1560.0
■	1587.0
■	1590.0
■	1591.0
■	1600.0
■	1603.0
■	1619.0
■	1625.0
■	1632.0
■	1640.0
■	1644.0
■	1664.0
■	1666.0
■	1667.0
■	1668.0
■	1695.0
■	1700.0
■	1710.0
■	1717.0
■	1719.0
■	1733.0
■	1742.0
■	1750.0
■	1769.0
■	1774.0
■	1775.0
■	1779.0
■	1783.0
■	1793.0
■	1800.0
■	1803.0
■	1811.0
■	1820.0
■	1833.0
■	1840.0
■	1842.0
■	1843.0
■	1851.0
■	1857.0
■	1863.0
■	1868.0
■	1872.0
■	1875.0
■	1881.0
■	1911.0
■	1915.0

1917.0
1929.0
1950.0
1964.0
1983.0
1987.0
1993.0
2000.0
2004.0
2014.0
2016.0
2033.0
2034.0
2035.0
2042.0
2054.0
2064.0
2067.0
2079.0
2083.0
2087.0
2100.0
2115.0
2118.0
2134.0
2138.0
2142.0
2157.0
2160.0
2166.0
2167.0
2168.0
2188.0
2200.0
2209.0
2210.0
2223.0
2232.0
2250.0
2253.0
2254.0
2275.0
2283.0
2302.0
2306.0
2330.0
2333.0

2336.0
2340.0
2358.0
2365.0
2375.0
2383.0
2400.0
2405.0
2416.0
2417.0
2426.0
2436.0
2451.0
2458.0
2466.0
2500.0
2504.0
2524.0
2531.0
2541.0
2569.0
2583.0
2598.0
2667.0
2669.0
2739.0
2773.0
2785.0
2791.0
2792.0
2816.0
2840.0
2845.0
2857.0
2859.0
2900.0
2917.0
2925.0
2934.0
2985.0
3000.0
3013.0
3021.0
3022.0
3033.0
3053.0
3066.0

3088.0
3136.0
3150.0
3166.0
3167.0
3230.0
3237.0
3250.0
3263.0
3274.0
3300.0
3333.0
3334.0
3369.0
3416.0
3428.0
3440.0
3447.0
3449.0
3500.0
3541.0
3583.0
3600.0
3666.0
3667.0
3683.0
3750.0
3796.0
3800.0
3806.0
3850.0
3890.0
3906.0
4000.0
4083.0
4114.0
4167.0
4196.0
4232.0
4250.0
4266.0
4300.0
4301.0
4333.0
4416.0
4417.0
4486.0
4488.0

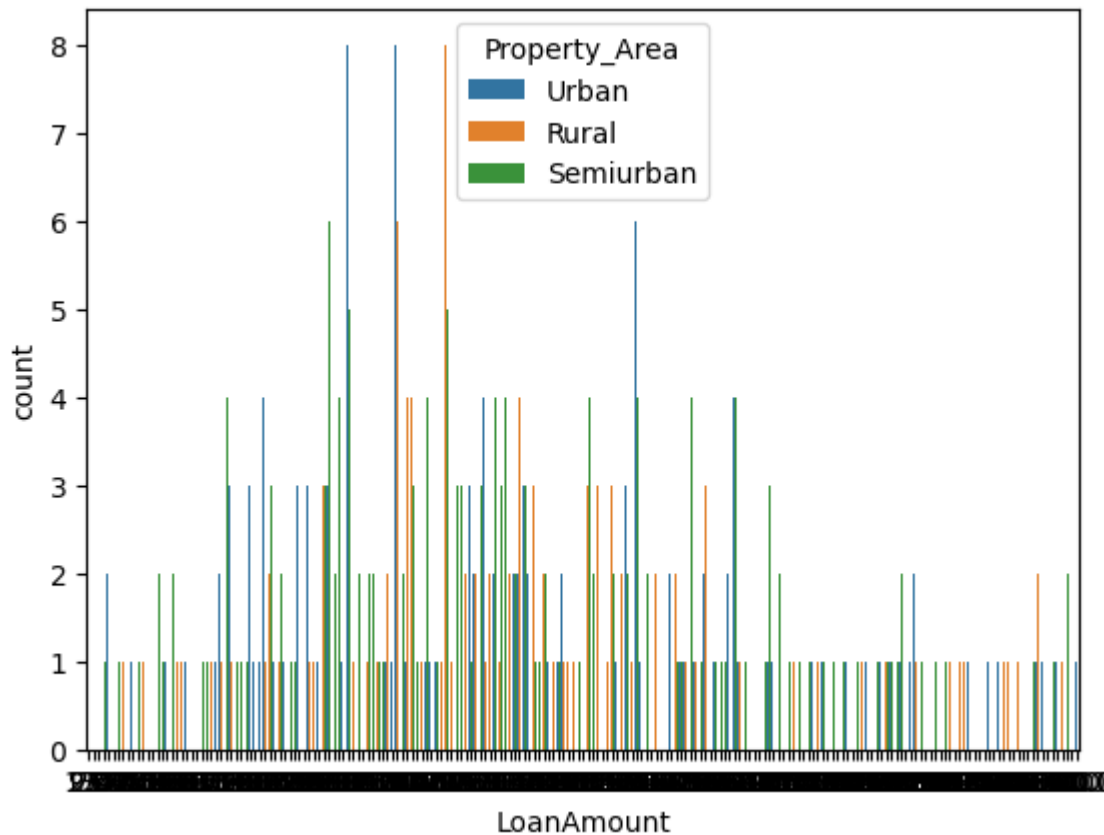


```
In [105]: sns.countplot(df['LoanAmount'], hue=df['Property_Area'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[105]:
```



```
In [ ]: sns.countplot(df['LoanAmount'],hue=df['Loan_Amount_Term'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

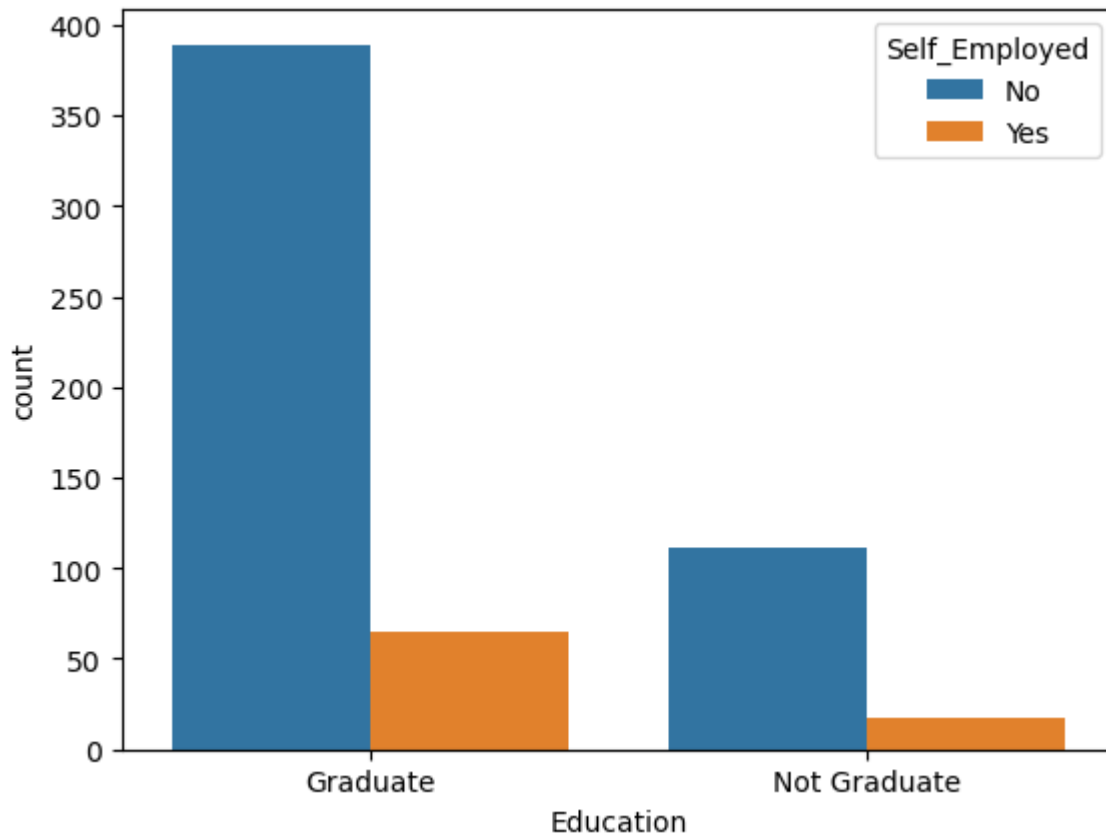
```
warnings.warn(
```

```
In [5]: sns.countplot(df['Education'],hue=df['Self_Employed'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[5]:
```

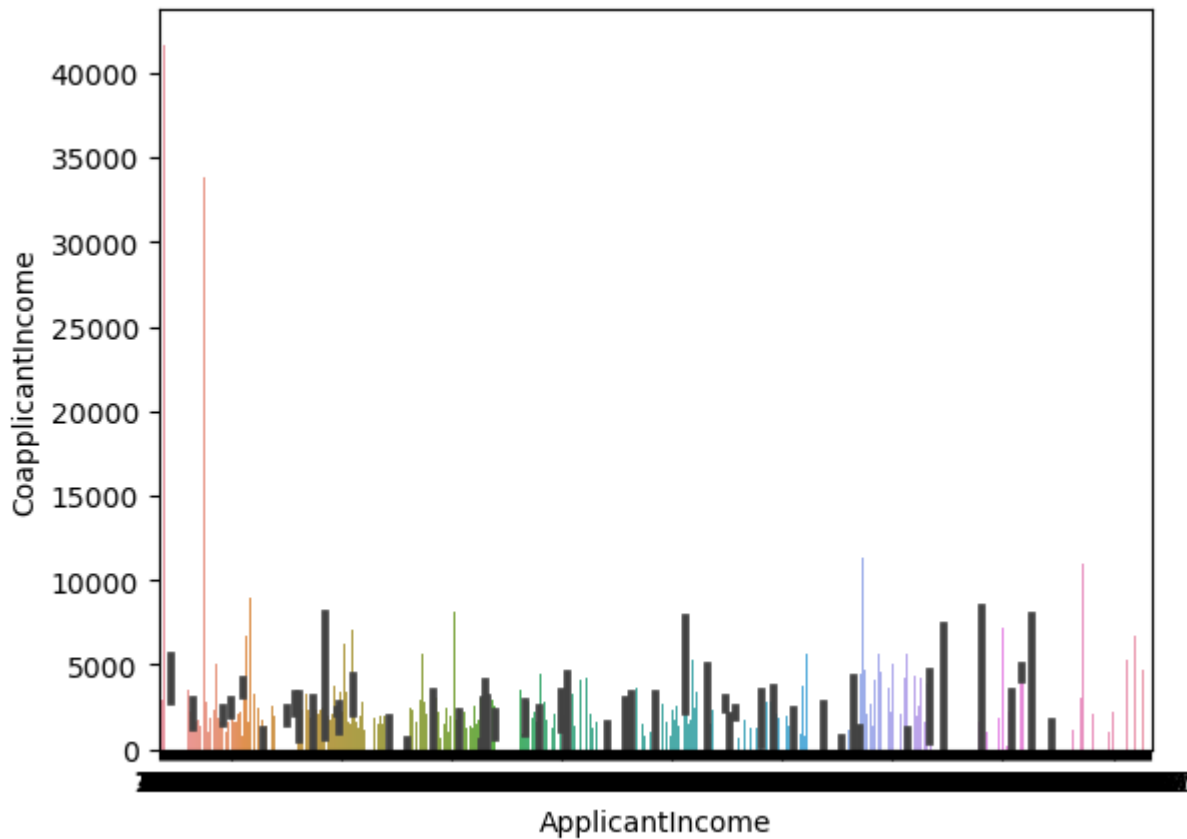


In [6]: `sns.barplot(df.ApplicantIncome,df.CoapplicantIncome)`

C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

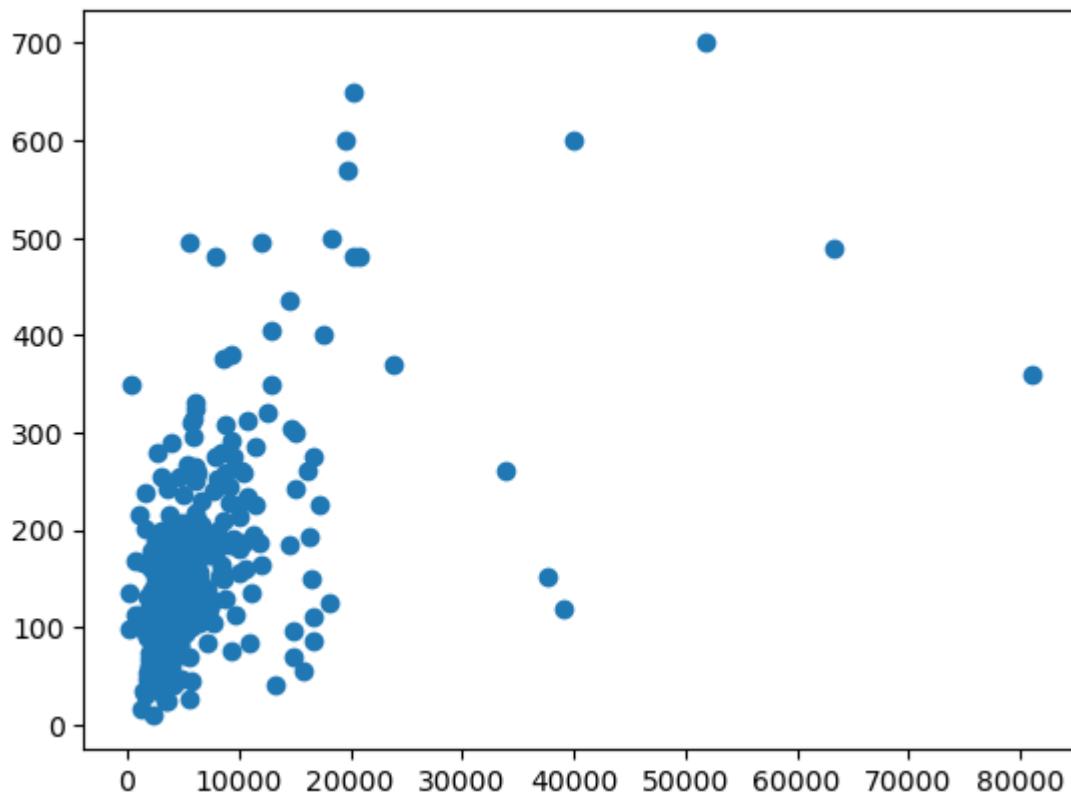
warnings.warn(

Out[6]:



In [7]: `plt.scatter(df.ApplicantIncome,df.LoanAmount)`

Out[7]:

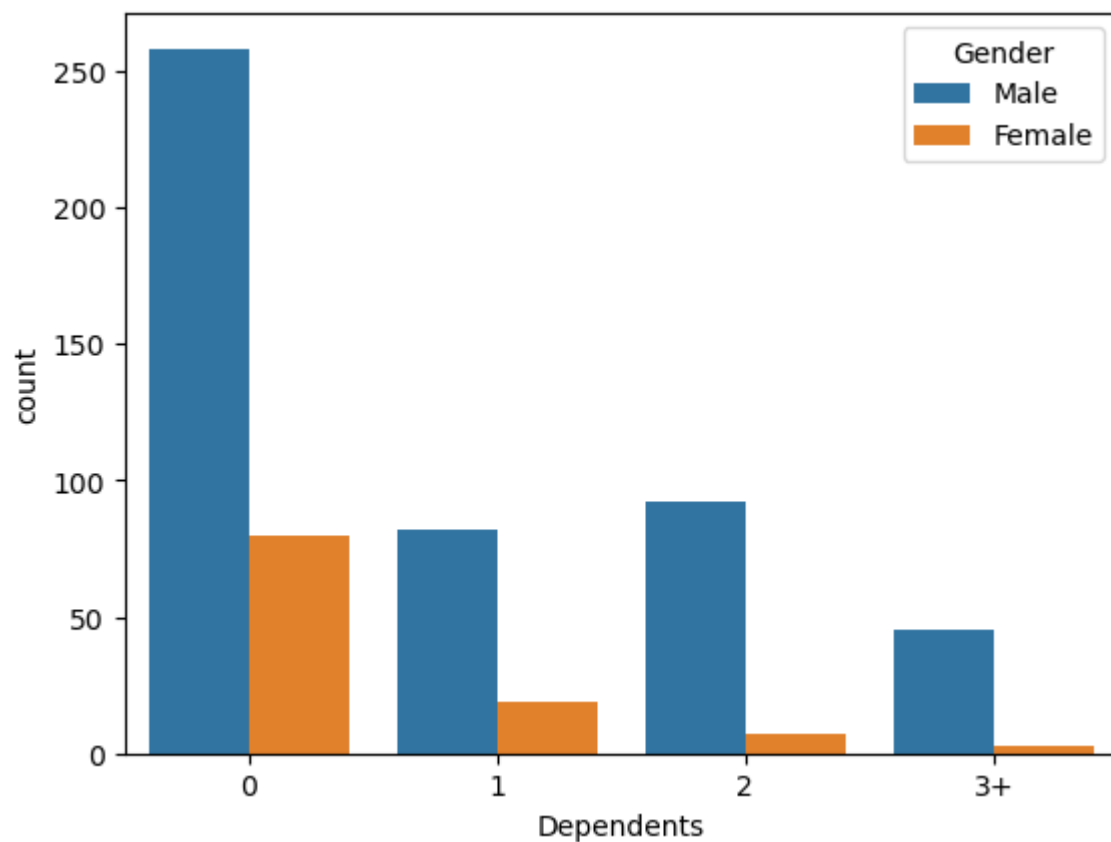


In [8]: `sns.countplot(df['Dependents'],hue=df['Gender'])`

following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

Out[8]:

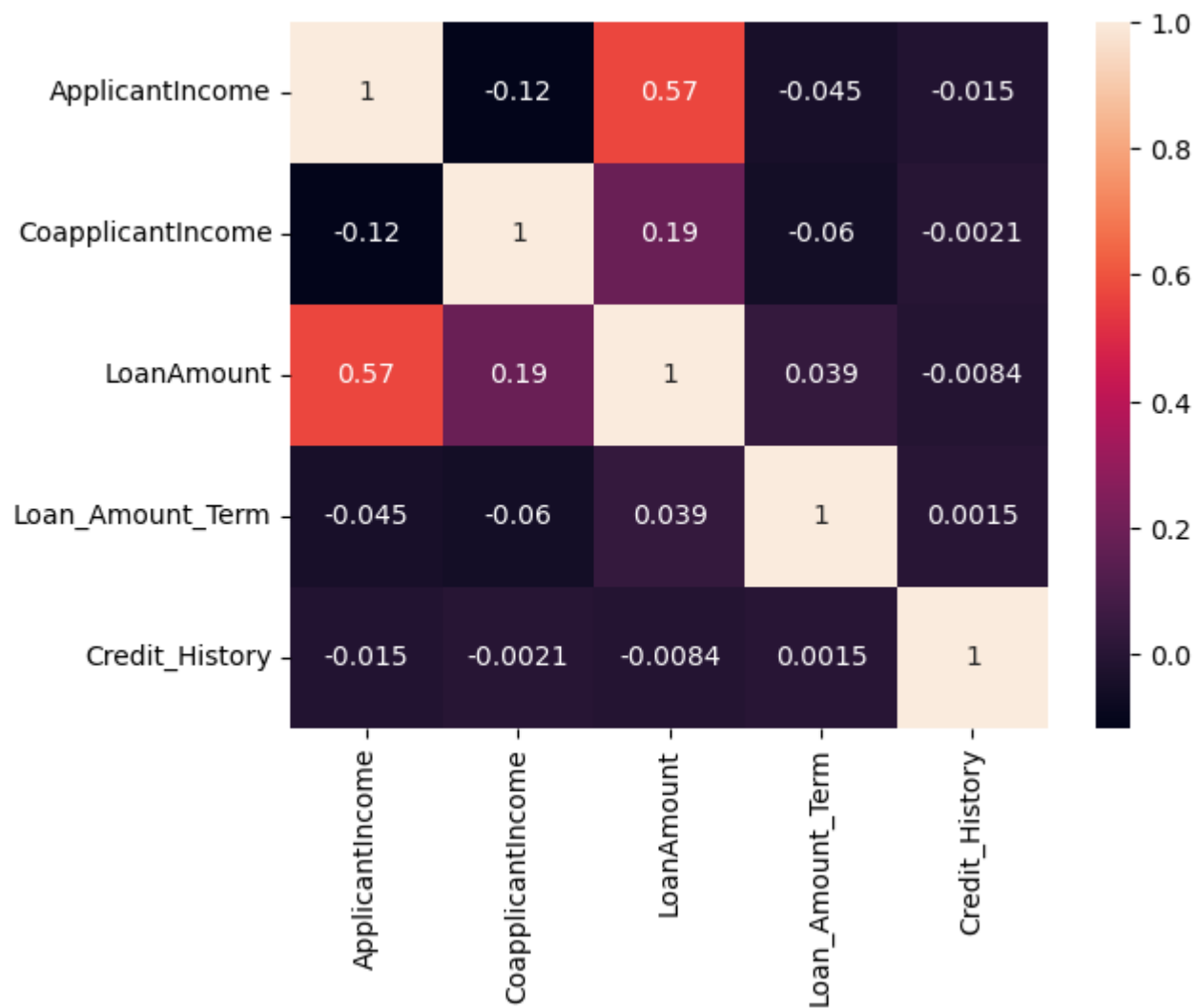


Multivariate Analysis

In [9]:

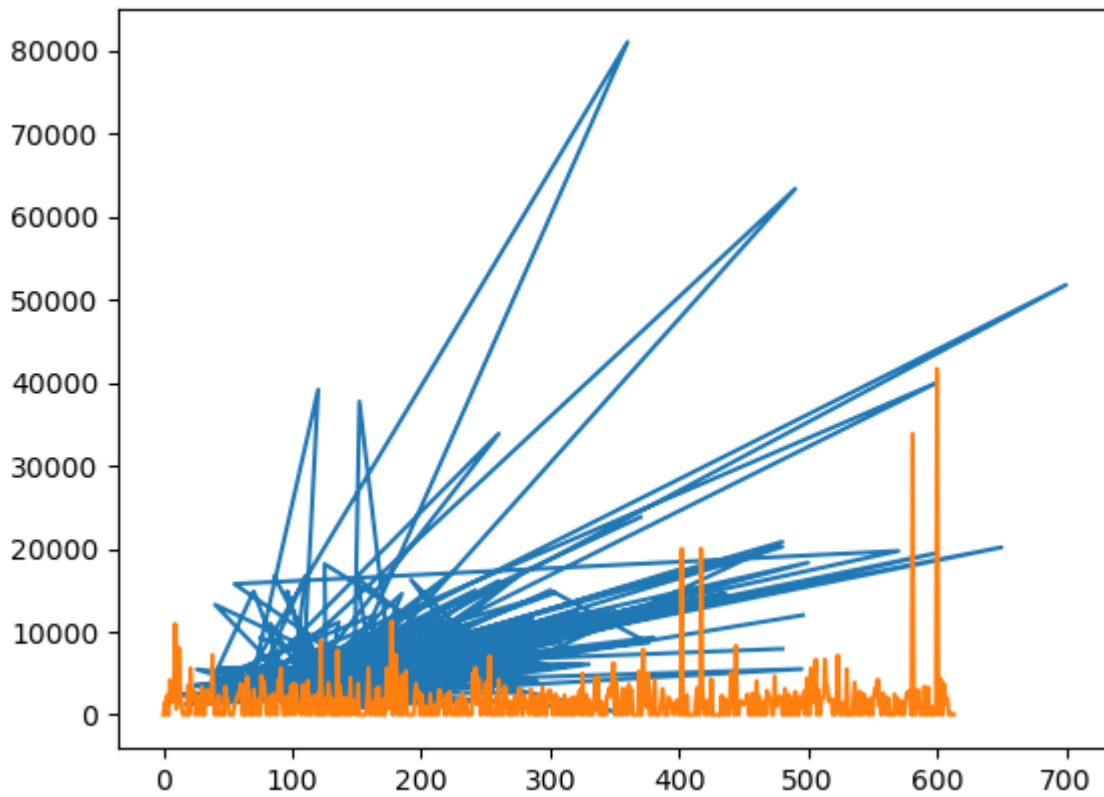
```
sns.heatmap(df.corr(),annot=True)
```

Out[9]:



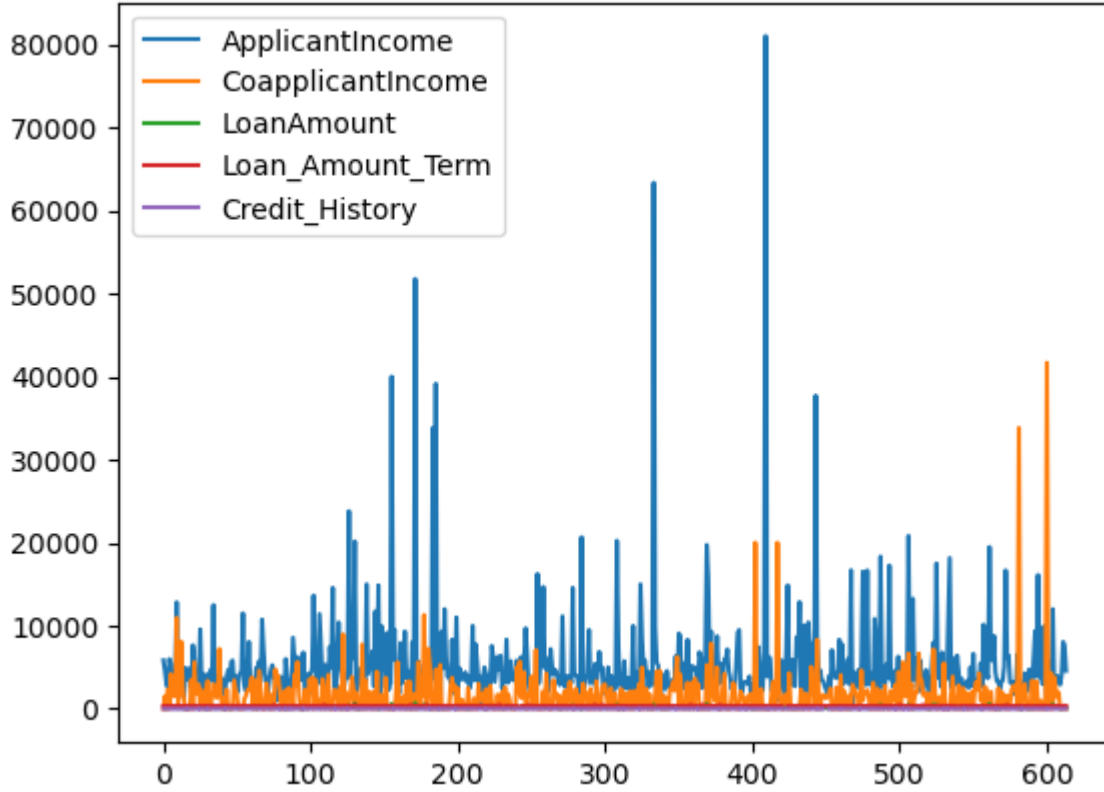
In [10]: `plt.plot(df.LoanAmount,df.ApplicantIncome,df.CoapplicantIncome)`

Out[10]:
[,
]



In [11]: `df.plot.line()`

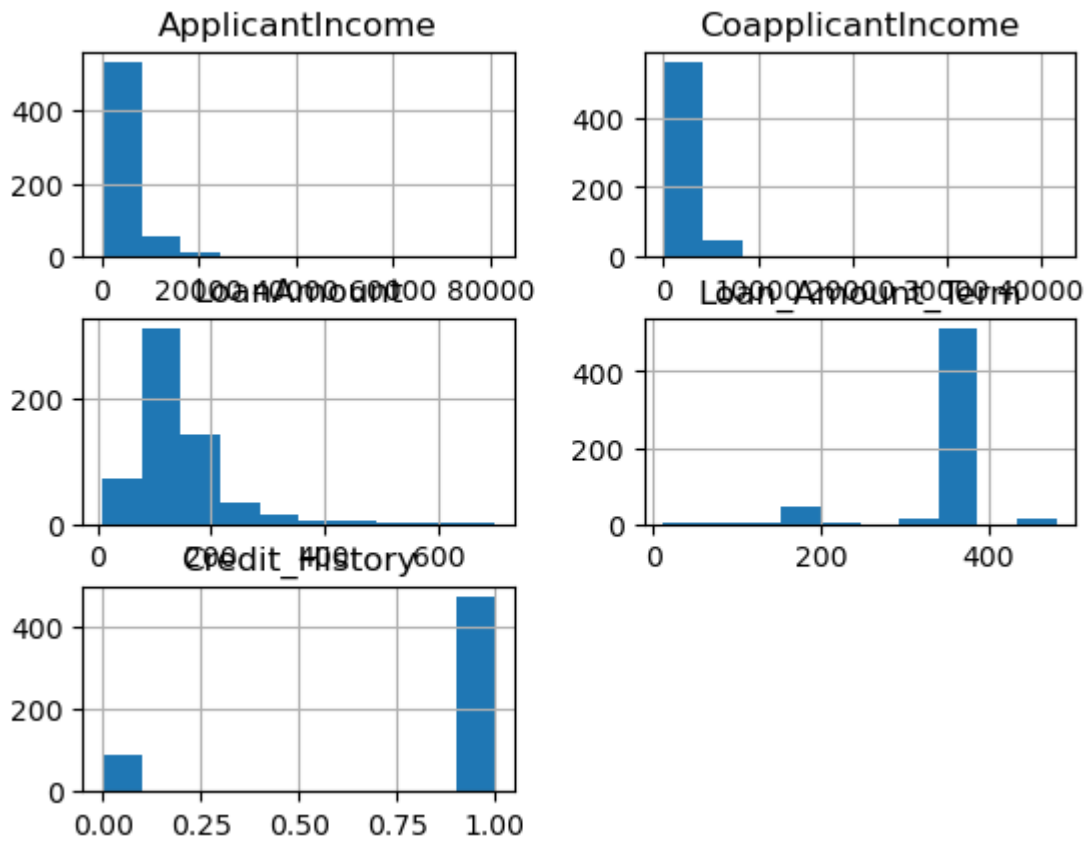
Out[11]:



In [12]: `df.hist()`

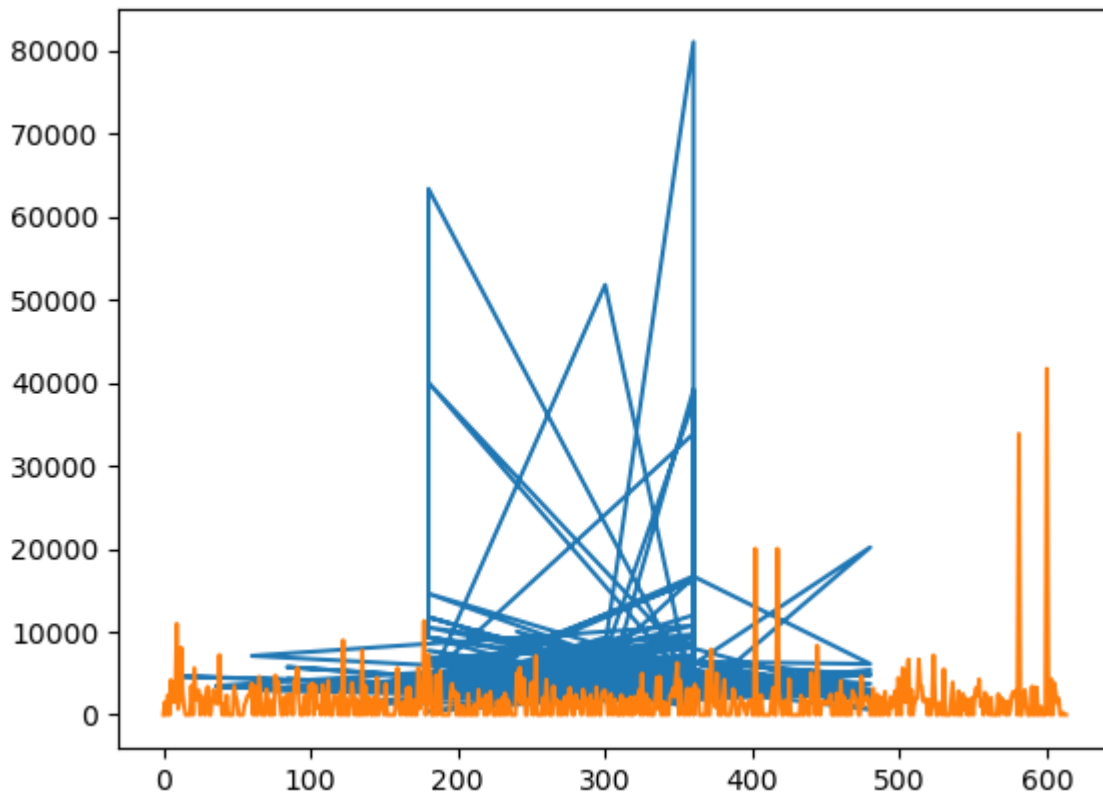
Out[12]:array([[,
],

```
[,
 ],
[, ]],
dtype=object)
```



```
In [13]: plt.plot(df.Loan_Amount_Term,df.ApplicantIncome,df.CoapplicantIncome)
```

```
Out[13]:[,
 ]
```

Descriptive Analysis

In [8]:
df.describe()

Out[8]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

In [9]:
df.mean()

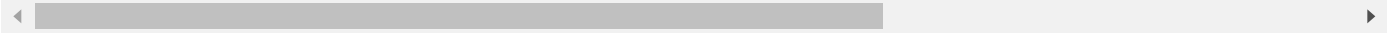
C:\Users\Aishwarya\AppData\Local\Temp\ipykernel_7884\3698961737.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
df.mean()
Out[9]: ApplicantIncome    5403.459283
 CoapplicantIncome      1621.245798
 LoanAmount             146.412162
 Loan_Amount_Term       342.000000
 Credit_History          0.842199
 dtype: float64
```

```
In [10]: df.mode()
```

```
Out[10]:
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	Yes	0	Graduate	No	2500	0.0	120.0



```
In [11]: df.std()
```

C:\Users\Aishwarya\AppData\Local\Temp\ipykernel_7884\3390915376.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
df.std()
Out[11]:
```

ApplicantIncome	6109.041673
CoapplicantIncome	2926.248369
LoanAmount	85.587325
Loan_Amount_Term	65.120410
Credit_History	0.364878

dtype: float64

```
In [12]: df.count()
```

```
Out[12]:
```

Gender	601
Married	611
Dependents	599
Education	614
Self_Employed	582
ApplicantIncome	614
CoapplicantIncome	614
LoanAmount	592
Loan_Amount_Term	600
Credit_History	564
Property_Area	614
Loan_Status	614

dtype: int64

Checking For Null Values

```
In [13]: df.isnull().any()
```

```
Out[13]:
```

Gender	True
Married	True
Dependents	True
Education	False
Self_Employed	True
ApplicantIncome	False
CoapplicantIncome	False
LoanAmount	True
Loan_Amount_Term	True
Credit_History	True
Property_Area	False
Loan_Status	False

dtype: bool

```
In [14]: df.isnull().sum()
```

```
Out[14]:
```

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
```

```
In [15]: 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())
```

```
In [16]: 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])
```

```
In [17]: df.isnull().any()
```

```
Out[17]:Gender      False
Married            False
Dependents         False
Education          False
Self_Employed      False
ApplicantIncome    False
CoapplicantIncome  False
LoanAmount         False
Loan_Amount_Term   False
Credit_History     False
Property_Area      False
Loan_Status        False
dtype: bool
```

```
In [18]: df.isnull().sum()
```

```
Out[18]: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
```

Handling Categorical Values

```
In [19]: df.head()
```

```
Out...   Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmount
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	Male	No	0	Graduate	No	5849	0.0	146.412162
1	Male	Yes	1	Graduate	No	4583	1508.0	128.000000
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000
4	Male	No	0	Graduate	No	6000	0.0	141.000000

In [20]:
le=LabelEncoder()

In [21]:
df.Gender=le.fit_transform(df.Gender)
df.Married=le.fit_transform(df.Married)
df.Education=le.fit_transform(df.Education)
df.Self_Employed=le.fit_transform(df.Self_Employed)
df.Property_Area=le.fit_transform(df.Property_Area)
df.Loan_Status=le.fit_transform(df.Loan_Status)
df.Dependents=le.fit_transform(df.Dependents)

In [22]:
df.head()

Ou...	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0	0	0	5849	0.0	146.412162
1	1	1	1	0	0	4583	1508.0	128.000000
2	1	1	0	0	1	3000	0.0	66.000000
3	1	1	0	1	0	2583	2358.0	120.000000
4	1	0	0	0	0	6000	0.0	141.000000

Splitting into dependent and independent data

In [23]:
df.head()

Ou...	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0	0	0	5849	0.0	146.412162
1	1	1	1	0	0	4583	1508.0	128.000000
2	1	1	0	0	1	3000	0.0	66.000000
3	1	1	0	1	0	2583	2358.0	120.000000
4	1	0	0	0	0	6000	0.0	141.000000

In [24]:
x=df.iloc[:, :-1]
y=df.Loan_Status

```
In [25]: x.head()
```

```
Out...
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0	0	0	5849	0.0	146.412162
1	1	1	1	0	0	4583	1508.0	128.000000
2	1	1	0	0	1	3000	0.0	66.000000
3	1	1	0	1	0	2583	2358.0	120.000000
4	1	0	0	0	0	6000	0.0	141.000000

```
In [26]: y.head()
```

```
Out[26]:0    1
1    0
2    1
3    1
4    1
Name: Loan_Status, dtype: int32
```

Scaling The Data

```
In [27]: x_scale=pd.DataFrame(scale(x),columns=x.columns)
x_scale.head()
```

```
Out...
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	0.472343	-1.372089	-0.737806	-0.528362	-0.392601	0.072991	-0.554487	0.000000
1	0.472343	0.728816	0.253470	-0.528362	-0.392601	-0.134412	-0.038732	-0.219270
2	0.472343	0.728816	-0.737806	-0.528362	2.547117	-0.393747	-0.554487	-0.957640
3	0.472343	0.728816	-0.737806	1.892641	-0.392601	-0.462062	0.251980	-0.314540
4	0.472343	-1.372089	-0.737806	-0.528362	-0.392601	0.097728	-0.554487	-0.064450

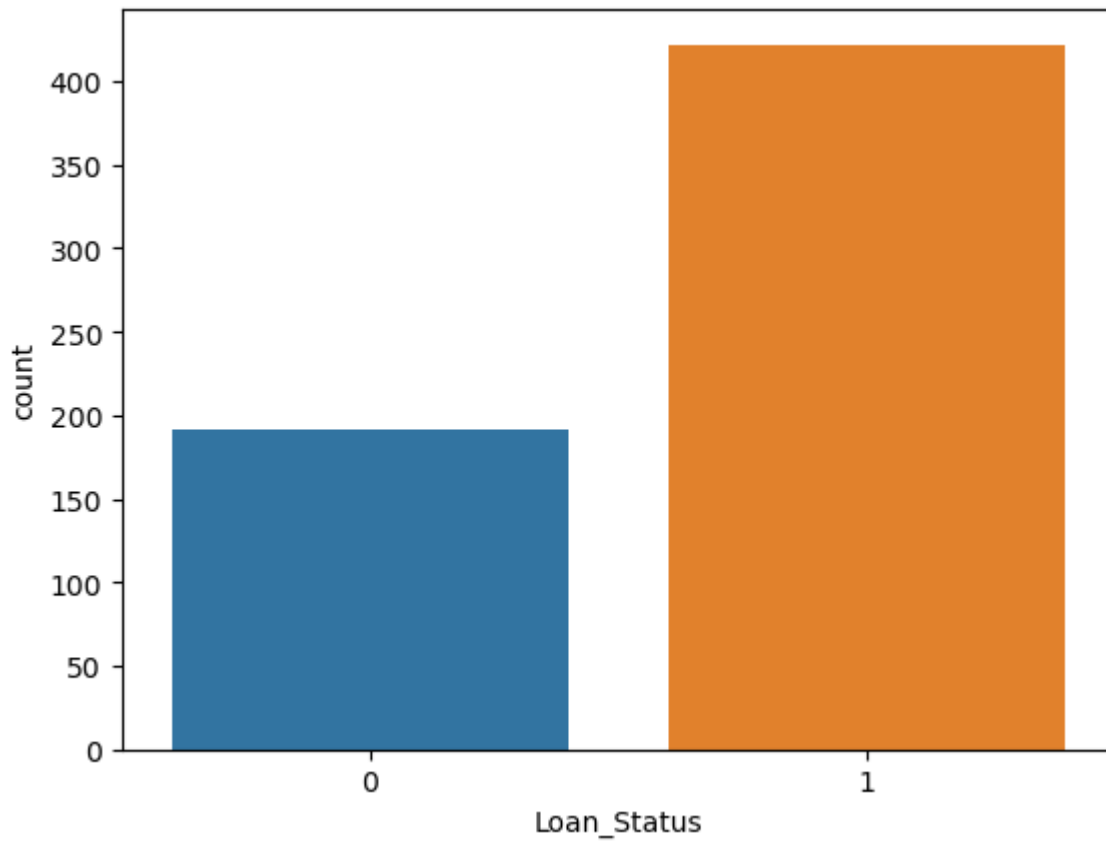
Balancing The Dataset

```
In [131]: sns.countplot(df.Loan_Status)
```

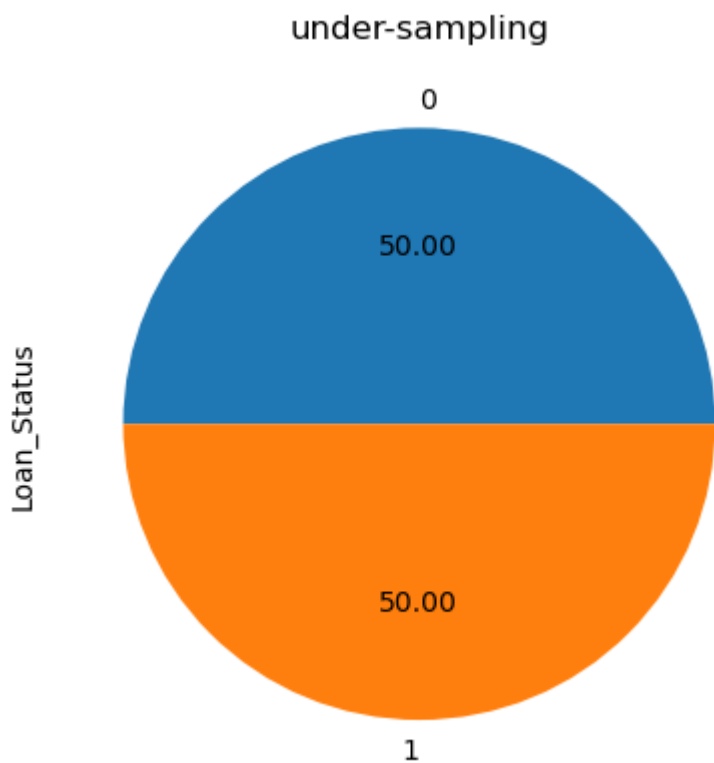
C:\ProgramData\Anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[131]:
```



```
In [132]: rus=RandomUnderSampler(sampling_strategy=1)
x_res,y_res=rus.fit_resample(x,y)
ax=y_res.value_counts().plot.pie(autopct='%.2f')
_ =ax.set_title("under-sampling")
```



Splitting Data Into Train And Test

```
In [133]:
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=10)
```

```
In [134]: xtrain.head()
```

```
Out[...      Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmour
```

245	1	0	0	0	1	6050	4333.0	120
413	1	1	0	1	0	2253	2033.0	110
126	1	1	3	0	0	23803	0.0	370
531	1	1	3	0	0	4281	0.0	100
188	1	1	0	0	1	674	5296.0	168

```
In [135]: xtest.head()
```

```
Out[...      Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  CoapplicantIncome  LoanAmour
```

285	1	0	0	0	0	3158	3053.0	89
323	0	0	0	0	0	3166	2985.0	132
482	1	1	0	0	0	2083	3150.0	128
173	1	1	0	0	0	5708	5625.0	187
518	1	0	0	0	0	4683	1915.0	185

```
In [136]: ytrain.head()
```

```
Out[136]: 245    0
          413    1
          126    1
          531    1
          188    1
          Name: Loan_Status, dtype: int32
```

```
In [137]: ytest.head()
```

```
Out[137]: 285    1
          323    1
          482    1
          173    1
          518    0
          Name: Loan_Status, dtype: int32
```

```
In [138]: xtrain.shape
```

```
Out[138]: (429, 11)
```

```
In [139]: xtest.shape
```

```
Out[139]: (185, 11)
```

```
In [140]:
```

```
ytrain.shape
```

```
Out[140]:(429,)
```

```
In [141]: ytest.shape
```

```
Out[141]:(185,)
```

Model Building

Xgboost Model

```
In [257]: xmodel=XGBClassifier(eval_metric='mlogloss',n_estimators=100,random_state=100)
```

```
In [258]: xmodel.fit(x_res,y_res)
```

```
Out[25...XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
                        colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                        early_stopping_rounds=None, enable_categorical=False,
                        eval_metric='mlogloss', gamma=0, gpu_id=-1,
                        grow_policy='depthwise', importance_type=None,
                        interaction_constraints="", learning_rate=0.300000012,
                        max_bin=256, max_cat_to_onehot=4, max_delta_step=0, max_depth=6,
                        max_leaves=0, min_child_weight=1, missing=nan,
                        monotone_constraints=()), n_estimators=100, n_jobs=0,
                        num_parallel_tree=1, predictor='auto', random_state=100,
                        reg_alpha=0, reg_lambda=1, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [259]: ypredx=xmodel.predict(xtest)
```

```
In [260]: ypred2x=xmodel.predict(xtrain)
```

```
In [261]: print("Xgboost Model Testing Accuracy")
           print(accuracy_score(ytest,ypredx))
           print("Xgboost Model Training Accuracy")
           print(accuracy_score(ytrain,ypred2x))
```

Xgboost Model Testing Accuracy

0.8540540540540541

Xgboost Model Training Accuracy

0.9067599067599068