For this project I will be using an open-source loan / borrower dataset from Kaggle to explore and train a ML model to predict whether a loan should be accepted towards a certain group of people depending on their income and academics.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
df = pd.read_csv('LoanPrediction.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	15
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	23
4	LP001008	Male	No	0	Graduate	No	6000	

In [4]:

```
df.describe()
```

Out[4]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [5]:

```
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
```

In [6]:

```
# what are the null values?

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

Out[6]:

```
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
```

In [8]:

```
# I will fill the missing values with mean of column

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 [10]:

```
# Fill categorical values with mode of column

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 [11]:

```
# Look at null values again, there are no more!

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

Out[11]:

```
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
```

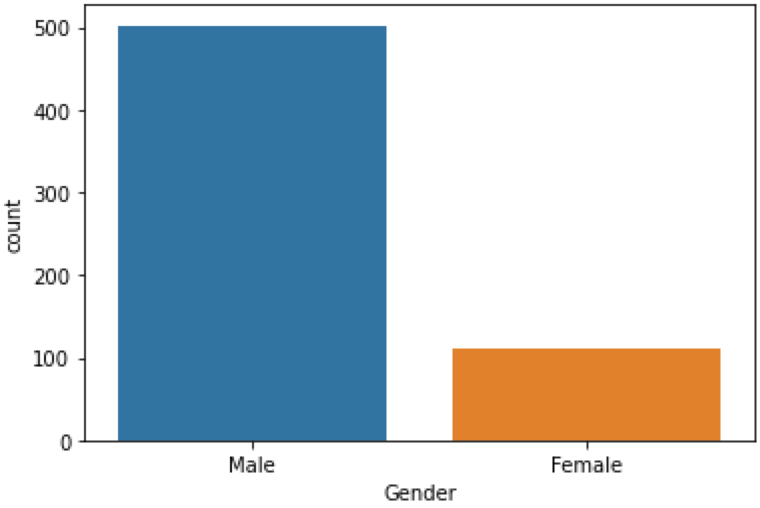
In [25]:

```
# I will now be visualizing categorical attributes.

# How many males vs. females in dataset?

sns.countplot(df['Gender'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[25]:

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



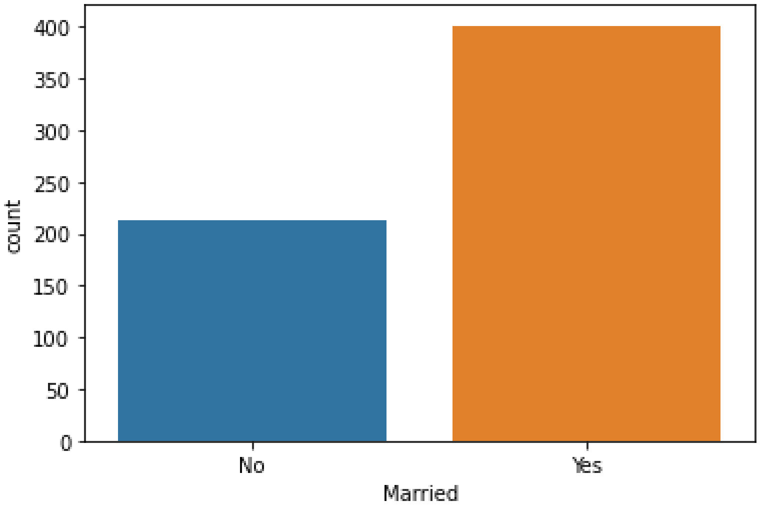
In [13]:

```
# How many are married?

sns.countplot(df['Married'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:

```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



In [14]:

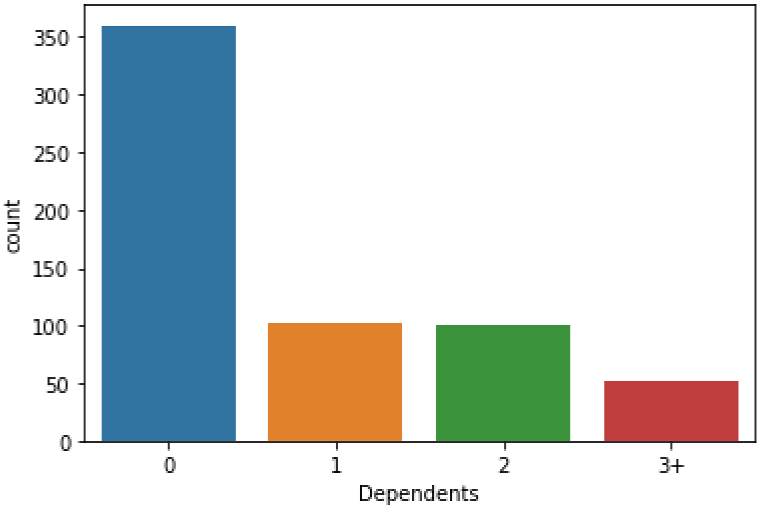
```
# How many people have children?
```

```
sns.countplot(df['Dependents'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:

```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



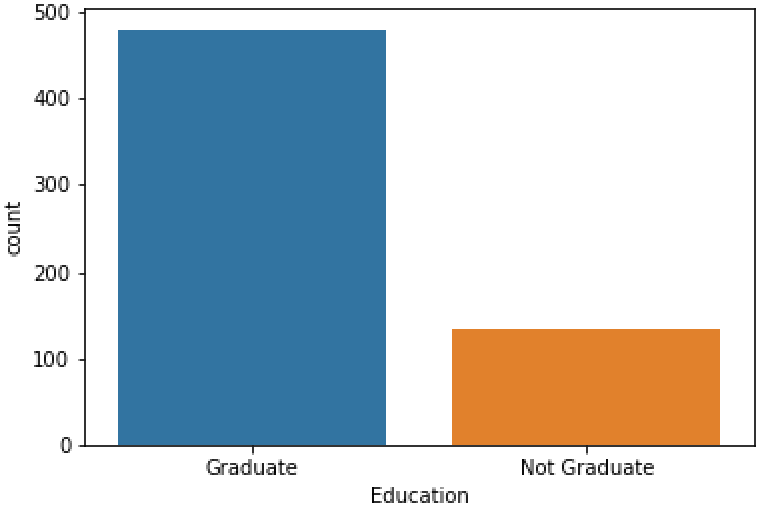
In [15]:

```
# How many are graduates from college or university?

sns.countplot(df['Education'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[15]:

```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



In [16]:

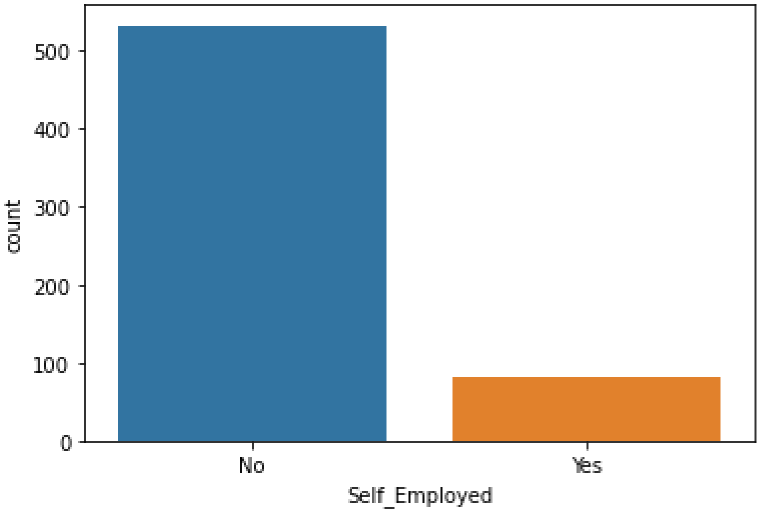
```
# How many have their own business?

sns.countplot(df['Self_Employed'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
```

```
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:

```
<AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```



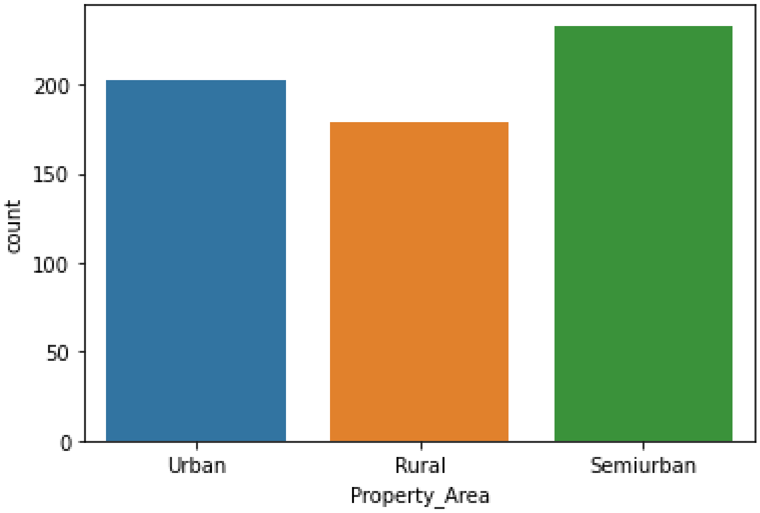
In [17]:

```
# Which area do they commonly reside?

sns.countplot(df['Property_Area'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:

```
<AxesSubplot:xlabel='Property_Area', ylabel='count'>
```



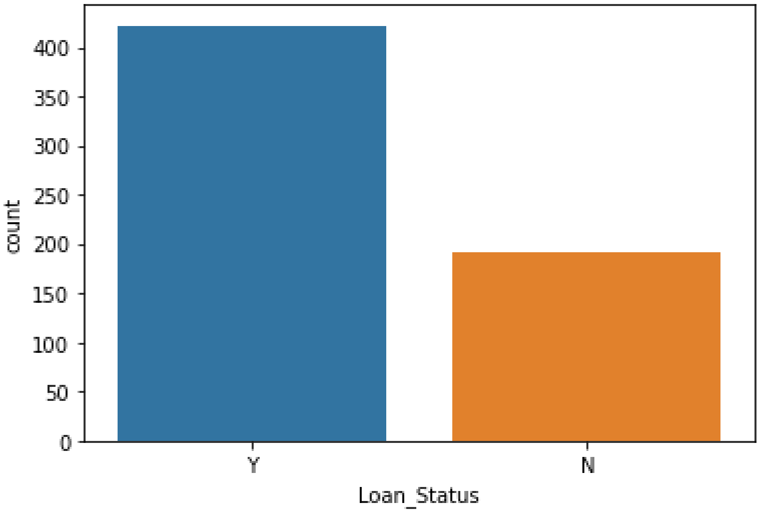
In [20]:

```
# Many people are approved for a loan.

sns.countplot(df['Loan_Status'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

Out[20]:

```
<AxesSubplot:xlabel='Loan_Status', ylabel='count'>
```



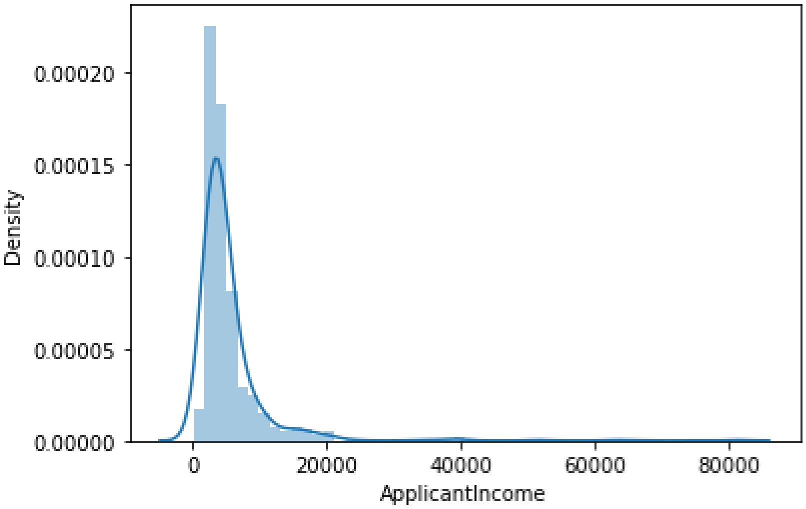
In [21]:

```
# Now I will be visualizing numerical attributes.

# What are the most common average Income?

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

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[21]:

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



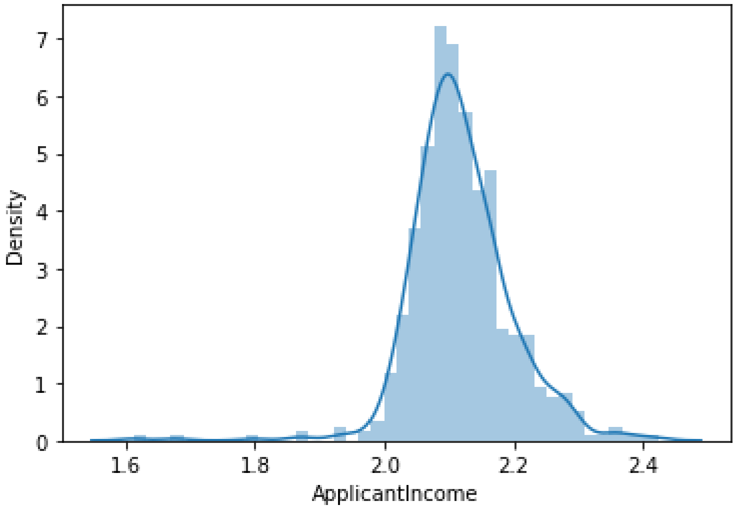
In [24]:

```
# Lets normalize the distplot by applying log transformation.

df['ApplicantIncome'] = np.log(df['ApplicantIncome'])
sns.distplot(df['ApplicantIncome'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[24]:

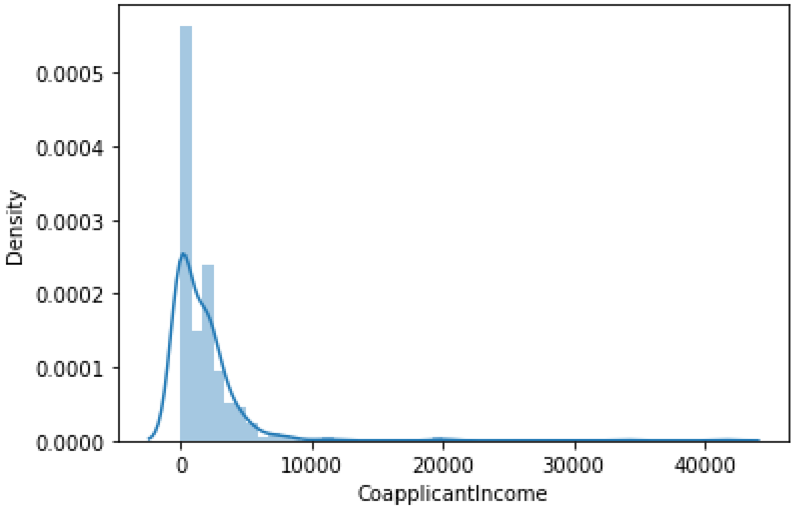
```
<AxesSubplot:xlabel='ApplicantIncome', ylabel='Density'>
```



```
In [28]: sns.distplot(df['CoapplicantIncome'])
```

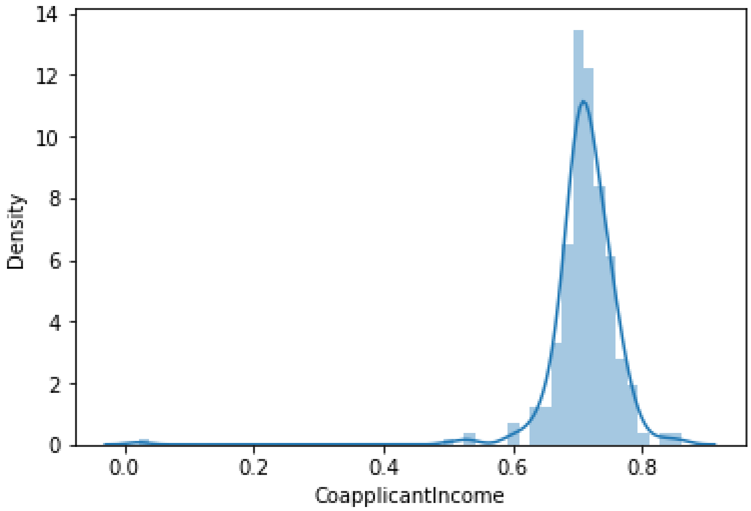
```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[28]: <AxesSubplot:xlabel='CoapplicantIncome', ylabel='Density'>
```



```
In [31]: df['CoapplicantIncome'] = np.log(df['CoapplicantIncome'])
         sns.distplot(df['CoapplicantIncome'])
```

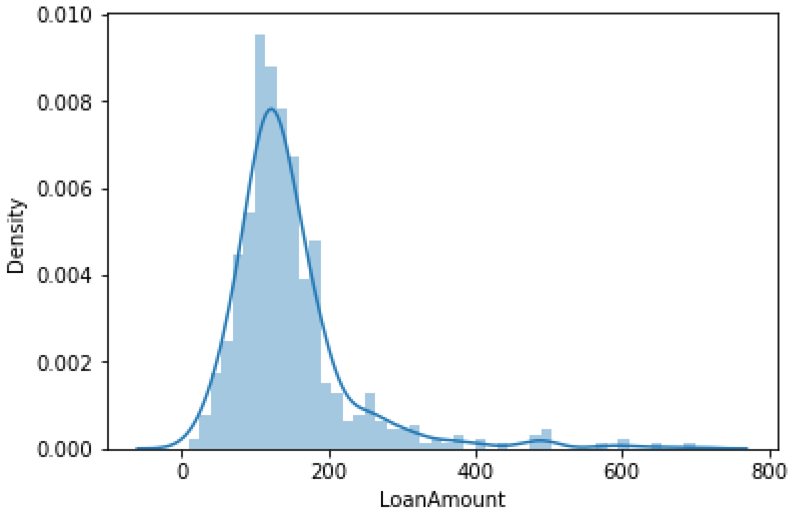
```
Out[31]: <AxesSubplot:xlabel='CoapplicantIncome', ylabel='Density'>
```



In [32]:
```
# What is the average Loan amount?

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

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

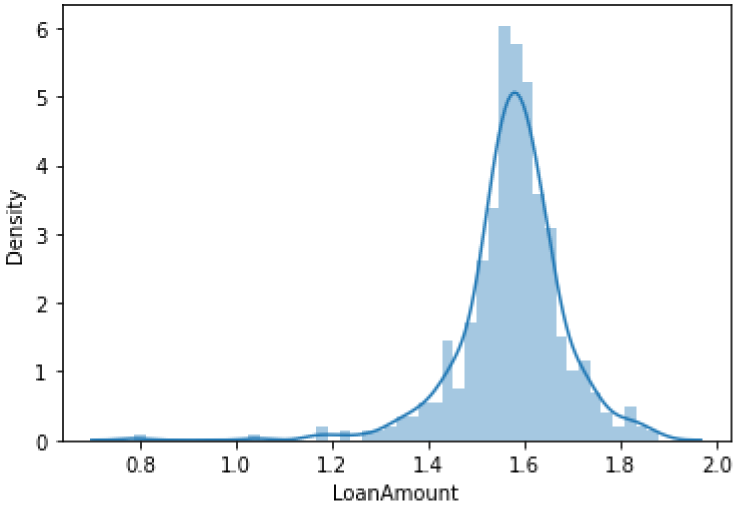
Out[32]:
```
<AxesSubplot:xlabel='LoanAmount', ylabel='Density'>
```



In [34]:
```
df['LoanAmount'] = np.log(df['LoanAmount'])
sns.distplot(df['LoanAmount'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

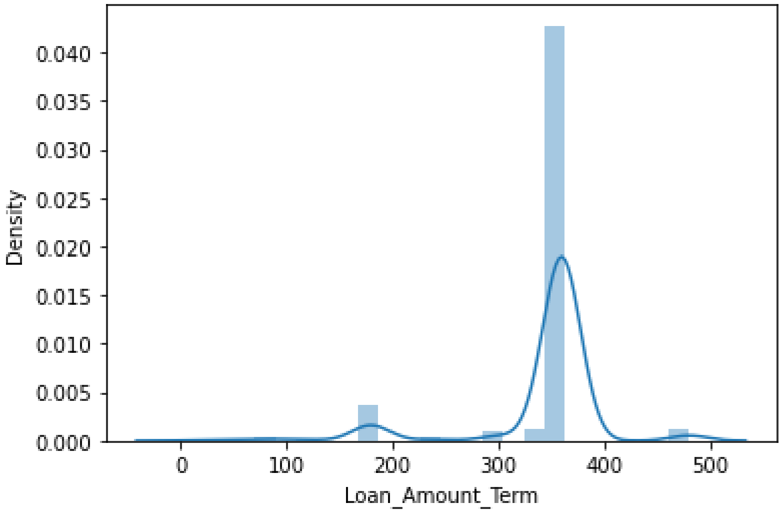
Out[34]:
```
<AxesSubplot:xlabel='LoanAmount', ylabel='Density'>
```



In [35]:
```
sns.distplot(df['Loan_Amount_Term'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

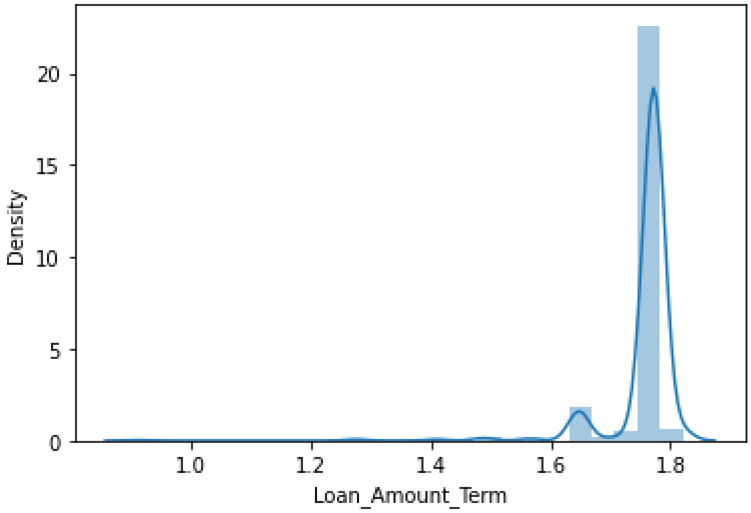
Out[35]:
```
<AxesSubplot:xlabel='Loan_Amount_Term', ylabel='Density'>
```



In [37]:
```
df['Loan_Amount_Term'] = np.log(df['Loan_Amount_Term'])
sns.distplot(df['Loan_Amount_Term'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

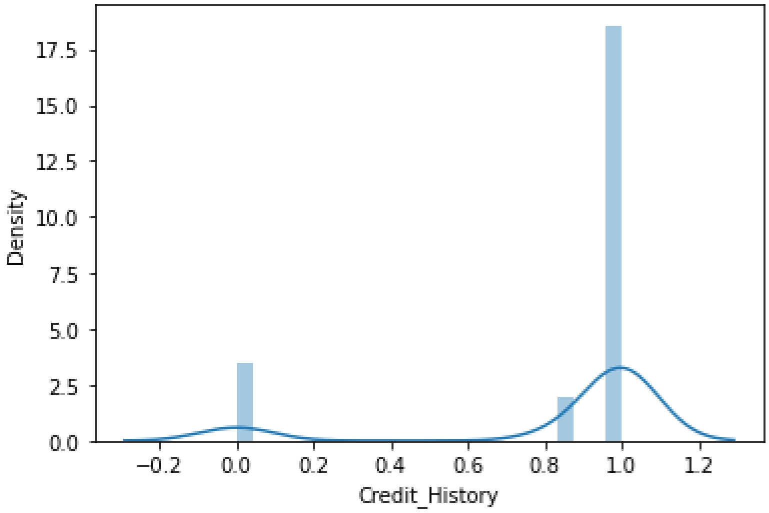
Out[37]:
```
<AxesSubplot:xlabel='Loan_Amount_Term', ylabel='Density'>
```



In [38]:
```
sns.distplot(df['Credit_History'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[38]:
```
<AxesSubplot:xlabel='Credit_History', ylabel='Density'>
```



In [ ]:
```
# I will create a new 'total income' column into the data, adding applicant and coappli
```

In [39]:
```
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

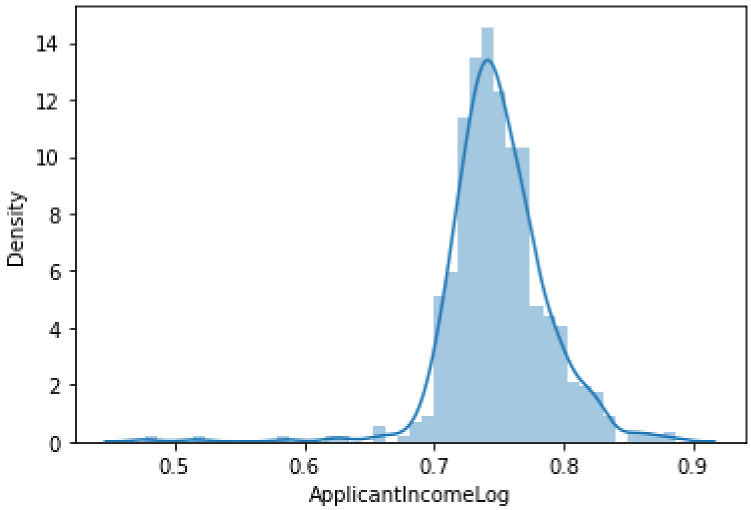
Out[39]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	2.160333	
1	LP001003	Male	Yes	1	Graduate	No	2.131810	0.68
2	LP001005	Male	Yes	0	Graduate	Yes	2.080237	
3	LP001006	Male	Yes	0	Not Graduate	No	2.061368	0.71
4	LP001008	Male	No	0	Graduate	No	2.163267	

In [47]:
```
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
sns.distplot(df['ApplicantIncomeLog'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

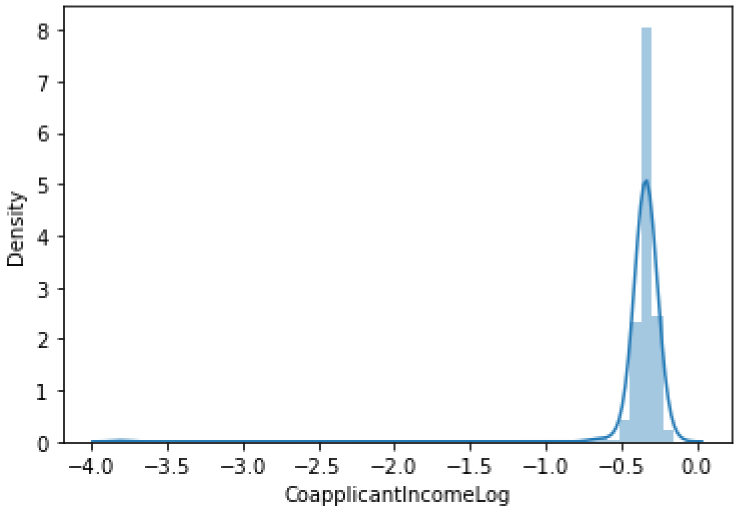
Out[47]:
```
<AxesSubplot:xlabel='ApplicantIncomeLog', ylabel='Density'>
```



In [46]:
```
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
sns.distplot(df['CoapplicantIncomeLog'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[46]:
```
<AxesSubplot:xlabel='CoapplicantIncomeLog', ylabel='Density'>
```



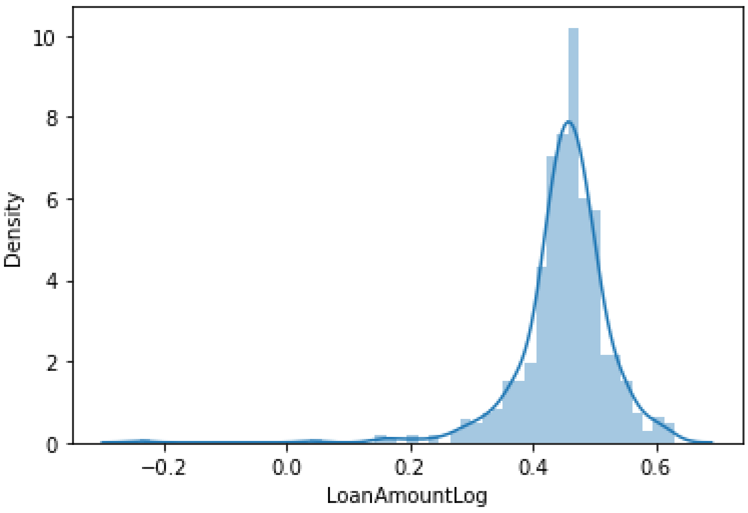
In [45]:
```
df['LoanAmountLog'] = np.log(df['LoanAmount'])
sns.distplot(df['LoanAmountLog'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
```

```
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[45]:

```
<AxesSubplot:xlabel='LoanAmountLog', ylabel='Density'>
```



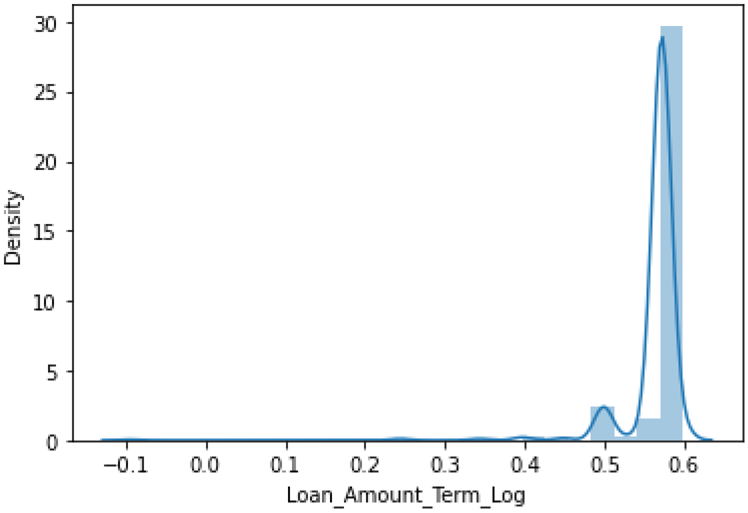
In [44]:

```python
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
sns.distplot(df['Loan_Amount_Term_Log'])
```

```
C:\Users\16193\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[44]:

```
<AxesSubplot:xlabel='Loan_Amount_Term_Log', ylabel='Density'>
```

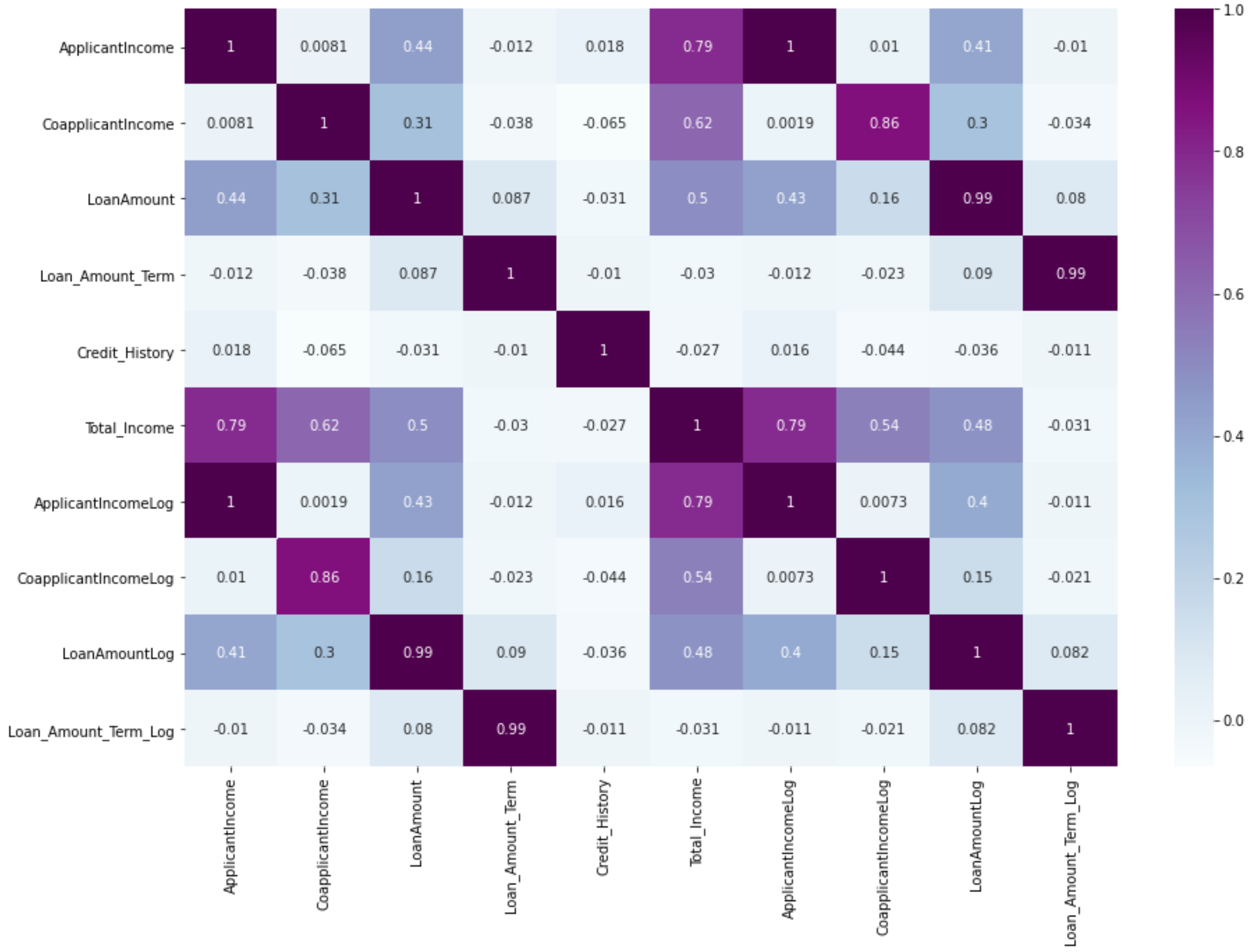


In [55]:

```python
# What are the correlations between all numerical attributes?

corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr,annot=True,cmap='BuPu')
```

Out[55]:

```
<AxesSubplot:>
```



In [56]:

```
# I will now drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Tota
df = df.drop(columns=cols, axis=1)
```

In [57]:

```
df.head()
```

Out[57]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	1.0	Urban	Y
4	Male	No	0	Graduate	No	1.0	Urban	Y

In [64]:

```
# Turn categorical attributes into numerical by label encoding with sklearn

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

In [65]:

```
df.head()
```

Out[65]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status
0	1	0	0	0	0	1.0	2	1
1	1	1	1	0	0	1.0	0	0
2	1	1	0	0	1	1.0	2	1
3	1	1	0	1	0	1.0	2	1
4	1	0	0	0	0	1.0	2	1

In [66]:

```
# Now I will begin the training, what are the input and output attributes?

X = df.drop(columns=['Loan_Status'],axis=1)

y = df['Loan_Status']
```

In [67]:

```
from sklearn.model_selection import train_test_split
```

In [68]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=
```

In [73]:

```
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_st
    model.fit(X_train,y_train)
    print('Accuracy is', model.score(X_test, y_test)*100)

    # cross validation for better validation of model

    score = cross_val_score(model, x, y, cv=5)
    print('Cross validation is,', np.mean(score)*100)
```

In [74]:

```
# First model I will try is Logistic Regression.

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

classify(model, X, y)
```

```
Accuracy is 77.27272727272727
Cross validation is, 80.9462881514061
```

In [76]:

```
# Second model I will try is Decision Tree.

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()

classify(model, X, y)
```

```
Accuracy is 74.02597402597402
Cross validation is, 71.17686258829802
```

In [79]:

```
# I will now try the Random Forest model.
```

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

model = RandomForestClassifier()

classify(model, X, y)
```

```
Accuracy is 77.92207792207793
Cross validation is, 79.15367186458748
```

In [80]:

```
# Extra Trees Classifier?

model = ExtraTreesClassifier()

classify(model, X, y)
```

```
Accuracy is 75.32467532467533
Cross validation is, 75.41116886578703
```

In [85]:

```
# I will now change the hyper parameters to see if I can improve the models

model = RandomForestClassifier(n_estimators=100, min_samples_split=25, max_depth=7, max
classify(model, X, y)
```

```
Accuracy is 76.62337662337663
Cross validation is, 80.61975209916034
```

In [88]:

```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[88]:

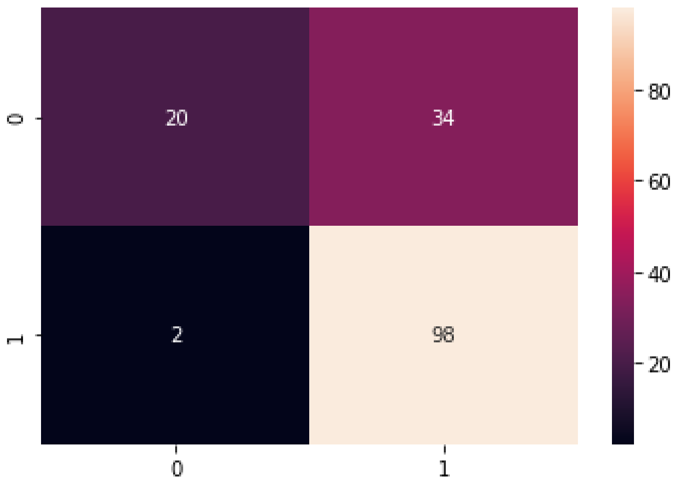
```
array([[20, 34],
       [ 2, 98]], dtype=int64)
```

In [89]:

```
sns.heatmap(cm, annot=True)
```

Out[89]:

```
<AxesSubplot:>
```



In conclusion, the Random Forest model was best to train data on as the accuracy and cross validation was at the highest level of percentage, compared to other models such as logistic regression and decision tree.

After training the model and changing the model's hyperparameters, it calculated 118 correctly and 36 incorrectly.

After exploring the data and learning more about it, I observe that it seems to be more densly populated by those within a married family and with either no kids or one kid. They also are mostly graduates from a college / university and live in a semiurban community.

After learning about these patterns in data, I suggest loaners should focus more on promoting more benefits and advertising amongst young families and independent college students who want to move out or have a car.