In [1]:

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

In [56]:

```
train=pd.read_csv(r"C:\Users\nitya\Downloads\Loan-Approval-Prediction-master\Loan-Appro
val-Prediction-master\train.csv")
test=pd.read_csv(r"C:\Users\nitya\Downloads\Loan-Approval-Prediction-master\Loan-Approv
al-Prediction-master\test.csv")
```

In [3]:

```
train.head()
```

Out[3]:

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

In [4]:

```
train.shape
```

Out[4]:

```
(614, 13)
```

In [6]:

```
train.columns
```

Out[6]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [5]:

```
train["Gender"].value_counts()
```

Out[5]:

```
Male      489
Female    112
Name: Gender, dtype: int64
```

In [6]:

```
train["Married"].value_counts()
```

Out[6]:

```
Yes    398
No     213
Name: Married, dtype: int64
```

In [6]:

```
train["Education"].value_counts()
```

Out[6]:

```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

In [9]:

```
train["Loan_Status"].value_counts()
```

Out[9]:

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [11]:

```
train["Dependents"].value_counts()
```

Out[11]:

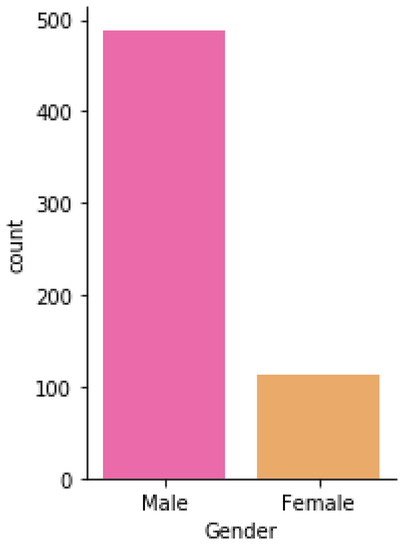
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [21]:

```
sns.catplot(x="Gender",data=train,kind="count",palette="spring",height=4,aspect=0.7)
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x13b5b8bdd68>
```

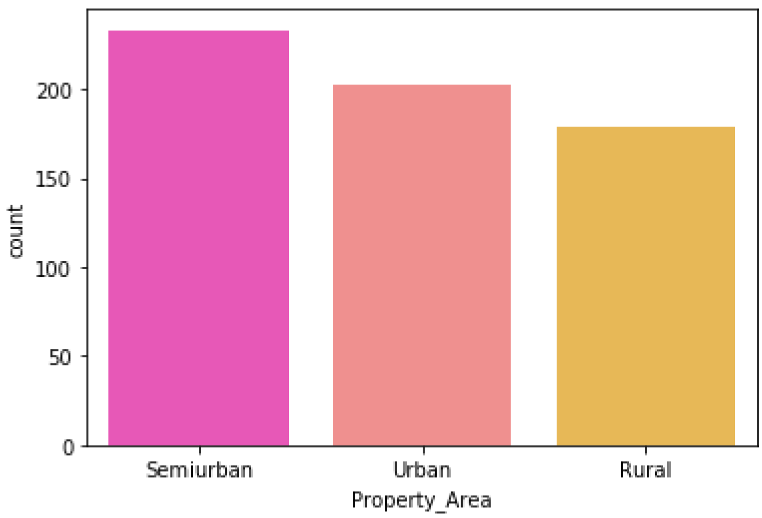


In [9]:

```
sns.countplot(x="Property_Area",data=train,palette="spring",order=['Semiurban','Urban',
'Rural'])
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2116b4325c0>
```



In [8]:

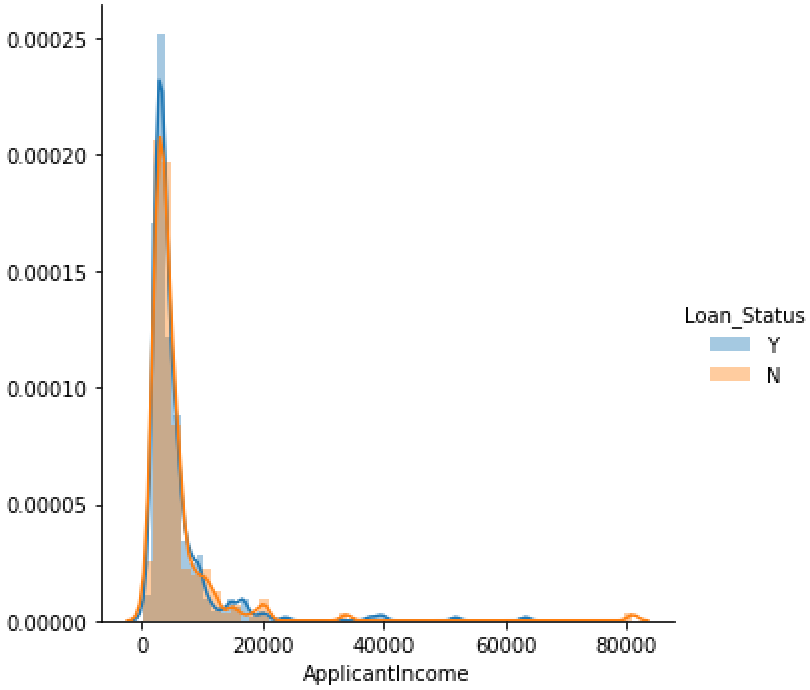
```
train["Property_Area"].value_counts()
```

Out[8]:

```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

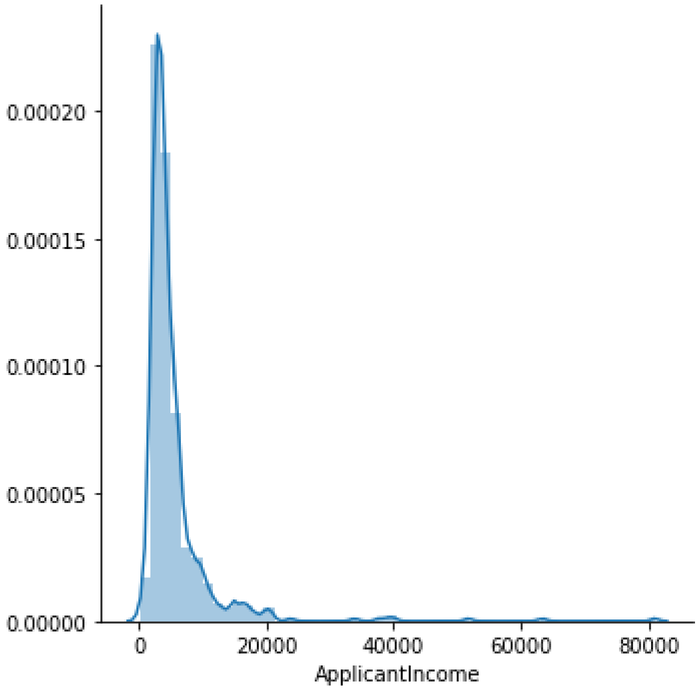
In [16]:

```
#to check that is there any limit of income after which we can say that loan_status is
 aproved but here we cant do that
sns.set_style={"ticks"};
sns.FacetGrid(train,hue="Loan_Status",height=5)\
    .map(sns.distplot,"ApplicantIncome")\
    .add_legend();
plt.show()
```



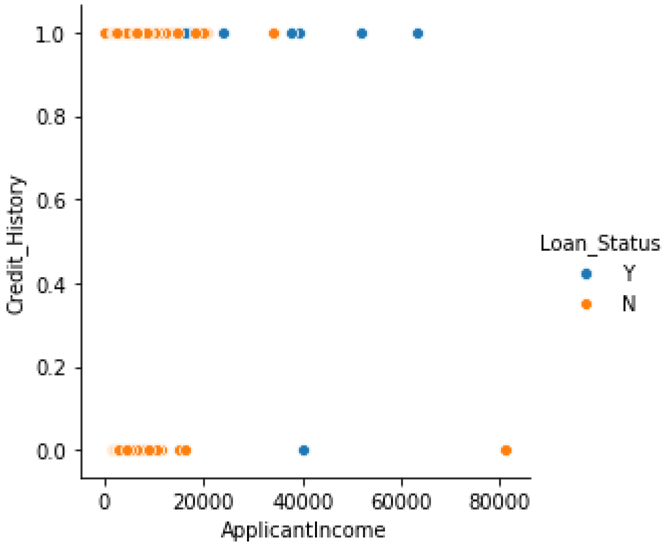
In [15]:

```
sns.set_style={"whitegrid"};
sns.FacetGrid(train,height=5)\
    .map(sns.distplot,"ApplicantIncome")\
    .add_legend();
plt.show()          '''The basic workflow is to initialize the FacetGrid object with th
e dataset and the variables
                         that are used to structure the grid. Then one or more plotting
functions
                      can be applied to each subset by calling FacetGrid.map() or Facet
Grid.map_dataframe()'''
```



In [17]:

```
#to check  that loan_status is dependant on applicant_income and credit history
sns.FacetGrid(train,hue="Loan_Status",height=4)\
    .map(sns.scatterplot,"ApplicantIncome","Credit_History")\
    .add_legend()
plt.show()
```



DATA CLEANING

In [78]:

```
train.isnull().sum()
```

Out[78]:

```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [79]:

```
#every tthing which is missing is categorical data except for LoanAmount and Loan amoun
t terms and so replacing missing
#most occuring element ie.mode of that column
train['Gender'].fillna(train['Gender'].mode()[0],inplace=True)
train['Married'].fillna(train['Married'].mode()[0],inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0],inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0],inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0],inplace=True)
```

In [80]:

```
#for continouous data-we are using median
train['LoanAmount'].median()
```

Out[80]:

```
128.0
```

In [81]:

```
train['LoanAmount'].fillna(128,inplace=True)
```

In [82]:

```
train['Loan_Amount_Term'].value_counts()
```

Out[82]:

```
360.0    526
180.0     44
480.0     15
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

In [94]:

```
train['Loan_Amount_Term'].fillna(360,inplace=True)
```

In [84]:

```
train.isnull().sum()
```

Out[84]:

```
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 [30]:

```
train.apply(lambda x:len(x.unique()))    #to check that loadid is unique for everyone
```

Out[30]:

```
Loan_ID              614
Gender                 2
Married                2
Dependents             4
Education              2
Self_Employed          2
ApplicantIncome      505
CoapplicantIncome    287
LoanAmount           203
Loan_Amount_Term      10
Credit_History         2
Property_Area          3
Loan_Status            2
dtype: int64
```

In [31]:

```
train.shape              #since no.of row is same then loanid is unique
```

Out[31]:

```
(614, 13)
```

DEALING CATEGORIAL DATA

In [38]:

```
test.head()
```

Out[38]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
0	LP001015	Male	Yes	0	Graduate	No	5720	
1	LP001022	Male	Yes	1	Graduate	No	3076	
2	LP001031	Male	Yes	2	Graduate	No	5000	
3	LP001035	Male	Yes	2	Graduate	No	2340	
4	LP001051	Male	No	0	Not Graduate	No	3276	

In [39]:

```
test.isnull().sum()
```

Out[39]:

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

In [85]:

```
test.dropna(inplace=True)
```

In [86]:

```
test.isnull().sum()
```

Out[86]:

```
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
dtype: int64
```

In [43]:

```
test.shape
```

Out[43]:

```
(289, 12)
```

In [91]:

```
train.columns
```

Out[91]:

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

In [ ]:

```

```

In [90]:

```
test.columns
```

Out[90]:

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

In [95]:

```
train.head()
```

Out[95]:

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

In [63]:

```
sex=pd.get_dummies(train['Gender'],drop_first=True)
```

In [64]:

```
sex.head()
```

Out[64]:

	Male
0	1
1	1
2	1
3	1
4	1

In [73]:

```
marry=pd.get_dummies(train['Married'],drop_first=True)
marry.columns=['Married_yes']
marry.head()
```

Out[73]:

	Married_yes
0	0
1	1
2	1
3	1
4	0

In [93]:

```
X=pd.get_dummies(train)
X.head()
```

Out[93]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Ge
0	5849	0.0	128.0	360.0	1.0	
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	

5 rows × 22 columns

In [99]:

```
X.columns                                              #train data
```

Out[99]:

```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Gender_Male', 'Married_Yes',
       'Dependents_1', 'Dependents_2', 'Dependents_3+',
       'Education_Not Graduate', 'Self_Employed_No', 'Property_Area_Semiur
ban',
       'Property_Area_Urban', 'Loan_Status_N'],
      dtype='object')
```

In [92]:

```
train.isnull().sum()
```

Out[92]:

```
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 [101]:

```
X.columns
```

Out[101]:

```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Gender_Male', 'Married_Yes',
       'Dependents_1', 'Dependents_2', 'Dependents_3+',
       'Education_Not Graduate', 'Self_Employed_No', 'Property_Area_Semiur
ban',
       'Property_Area_Urban', 'Loan_Status_N'],
      dtype='object')
```

In [102]:

```
test.head()
```

Out[102]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	Male	Yes	0	Graduate	No	5720	
1	Male	Yes	1	Graduate	No	3076	
2	Male	Yes	2	Graduate	No	5000	
4	Male	No	0	Not Graduate	No	3276	
5	Male	Yes	0	Not Graduate	Yes	2165	

In [103]:

```
test=pd.get_dummies(test)
test.head()                                       #test data
```

Out[103]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Ge
0	5720	0	110.0	360.0	1.0	
1	3076	1500	126.0	360.0	1.0	
2	5000	1800	208.0	360.0	1.0	
4	3276	0	78.0	360.0	1.0	
5	2165	3422	152.0	360.0	1.0	

In [105]:

```
test.drop(['Gender_Female','Married_No','Dependents_0','Education_Graduate','Self_Emplo
yed_Yes','Property_Area_Rural'],axis=1,inplace=True)
test.head()
```

Out[105]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Ge
0	5720	0	110.0	360.0	1.0	
1	3076	1500	126.0	360.0	1.0	
2	5000	1800	208.0	360.0	1.0	
4	3276	0	78.0	360.0	1.0	
5	2165	3422	152.0	360.0	1.0	

In [107]:

```
X.columns
```

Out[107]:

```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Gender_Male', 'Married_Yes',
       'Dependents_1', 'Dependents_2', 'Dependents_3+',
       'Education_Not Graduate', 'Self_Employed_No', 'Property_Area_Semiur
ban',
       'Property_Area_Urban', 'Loan_Status_N'],
      dtype='object')
```

In [108]:

```
X1=X.drop('Loan_Status_N',axis=1)
y=X.Loan_Status_N
```

In [109]:

```
from sklearn.linear_model import LogisticRegression
```

In [110]:

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

In [117]:

```
X1_train, X1_test, y_train, y_test = train_test_split(X1, y, test_size=0.2, random_stat
e=1)
```

In [112]:

```
logmodel=LogisticRegression()
```

In [118]:

```
logmodel.fit(X1_train,y_train)
```

```
C:\Users\nitya\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. S
pecify a solver to silence this warning.
  FutureWarning)
```

Out[118]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [119]:

```
prediction=logmodel.predict(X1_test)
```

In [115]:

```
from sklearn.metrics import accuracy_score
```

In [120]:

```
accuracy_score(y_test,prediction)
```

Out[120]:

```
0.8048780487804879
```

In [ ]:

```

```