In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
train = pd.read_csv("train (1).csv")
test = pd.read_csv("test.csv")
```

In [4]:
```
train_original = train.copy()
test_original = test.copy()
```

In [5]:
```
train.columns
```

Out[5]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmou
nt',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [6]:
```
test.columns
```

Out[6]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmou
nt',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [7]:
```
train.dtypes
```

Out[7]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [8]:
```
print('Training data shape: ', train.shape)
train.head()
```

```
Training data shape:  (614, 13)
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
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 [9]:
```
print('Test data shape: ', test.shape)
test.head()
```

```
Test data shape:  (367, 12)
```

Out[9]:

	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 [10]:
```
train["Loan_Status"].count()
```

Out[10]:
```
614
```

In [11]:
```
train["Loan_Status"].value_counts()
```

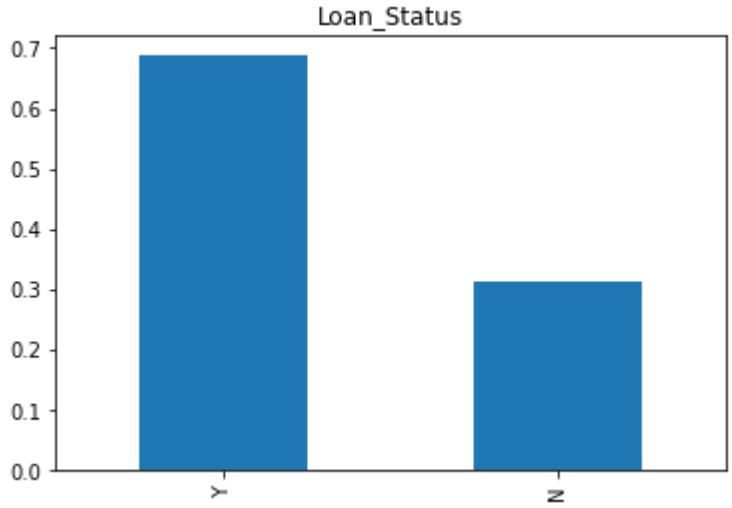
Out[11]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [12]:
```
train["Loan_Status"].value_counts(normalize=True)*100
```

Out[12]:
```
Y    68.729642
N    31.270358
Name: Loan_Status, dtype: float64
```

In [13]:
```
train["Loan_Status"].value_counts(normalize=True).plot.bar(title = 'Loan_St
```

Out[13]: <AxesSubplot:title={'center':'Loan_Status'}>



In [14]:
```
train["Gender"].count()
```

Out[14]: 601

In [15]:
```
train["Gender"].value_counts()
```

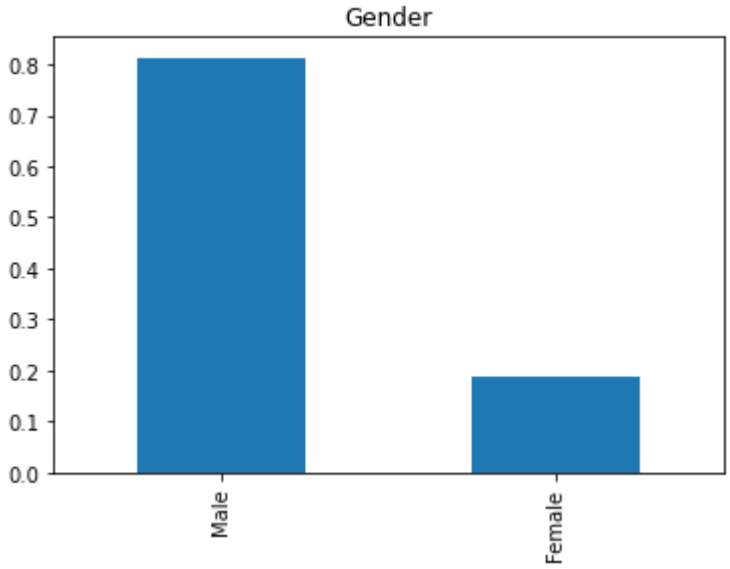
Out[15]:
```
Male      489
Female    112
Name: Gender, dtype: int64
```

In [16]:
```
train['Gender'].value_counts(normalize=True)*100
```

Out[16]:
```
Male      81.364393
Female    18.635607
Name: Gender, dtype: float64
```

In [17]:
```
train['Gender'].value_counts(normalize=True).plot.bar(title= 'Gender')
```

Out[17]: <AxesSubplot:title={'center':'Gender'}>



In [18]:
```
train["Married"].count()
```

Out[18]: 611

In [19]:
```
train["Married"].value_counts()
```

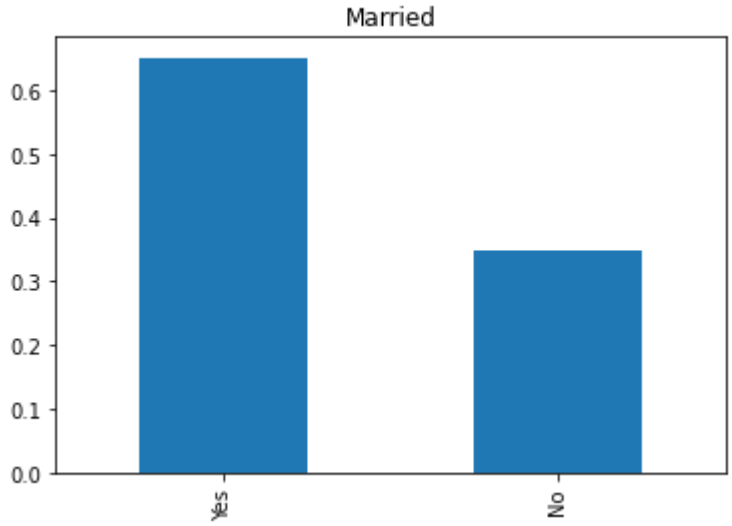
Out[19]:
```
Yes    398
No     213
Name: Married, dtype: int64
```

In [20]:
```
train['Married'].value_counts(normalize=True)*100
```

Out[20]:
```
Yes    65.139116
No     34.860884
Name: Married, dtype: float64
```

In [21]:
```
train['Married'].value_counts(normalize=True).plot.bar(title= 'Married')
```

Out[21]: <AxesSubplot:title={'center':'Married'}>



In [22]:
```
train["Self_Employed"].count()
```

Out[22]: 582

In [23]:
```
train["Self_Employed"].value_counts()
```

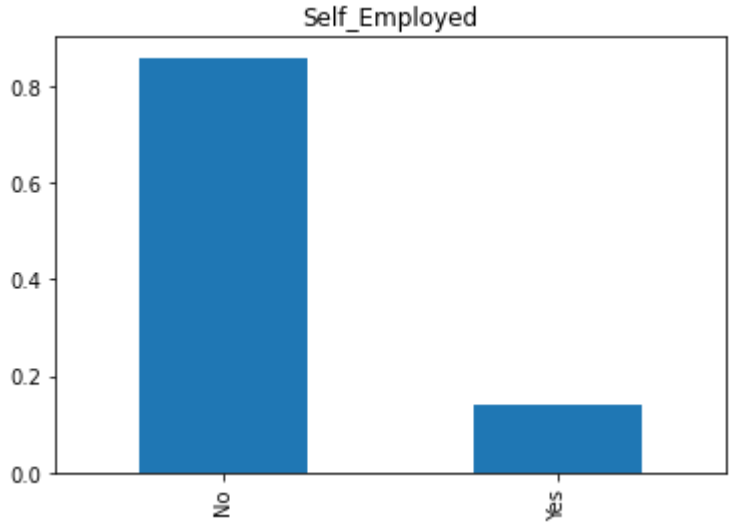
Out[23]:
```
No     500
Yes     82
Name: Self_Employed, dtype: int64
```

In [24]:
```
train['Self_Employed'].value_counts(normalize=True)*100
```

Out[24]:
```
No     85.910653
Yes    14.089347
Name: Self_Employed, dtype: float64
```

In [25]:
```
train['Self_Employed'].value_counts(normalize=True).plot.bar(title='Self_Em
```

Out[25]: <AxesSubplot:title={'center':'Self_Employed'}>



In [26]:
```
train["Credit_History"].count()
```

Out[26]:
```
564
```

In [27]:
```
train["Credit_History"].value_counts()
```

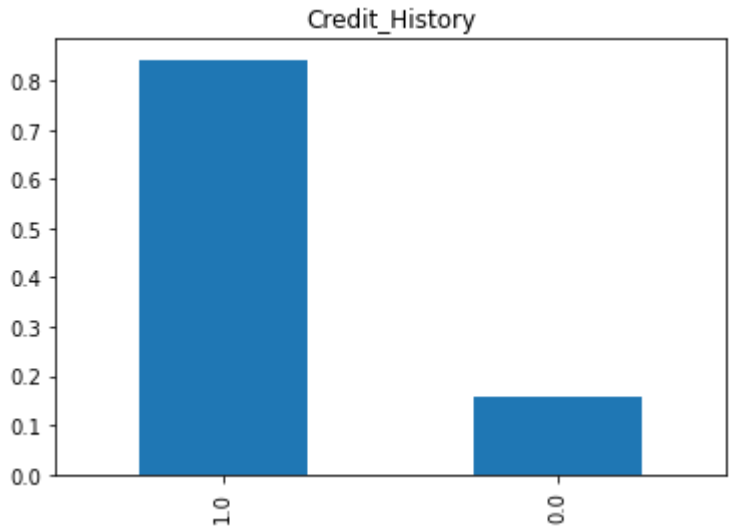
Out[27]:
```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

In [28]:
```
train['Credit_History'].value_counts(normalize=True)*100
```

Out[28]:
```
1.0    84.219858
0.0    15.780142
Name: Credit_History, dtype: float64
```

In [29]:
```
train['Credit_History'].value_counts(normalize=True).plot.bar(title='Credit
```

Out[29]:
```
<AxesSubplot:title={'center':'Credit_History'}>
```



In [30]:
```
train['Dependents'].count()
```

Out[30]:
```
599
```

In [31]:
```
train["Dependents"].value_counts()
```

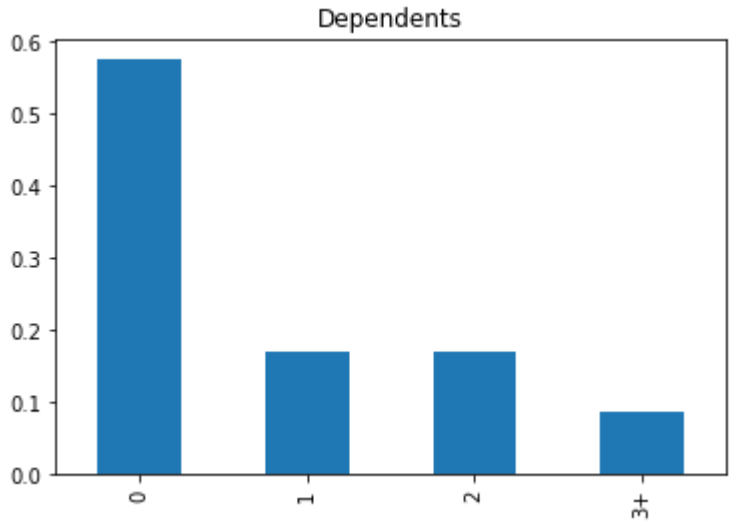
Out[31]:
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [32]:
```
train['Dependents'].value_counts(normalize=True)*100
```

Out[32]:
```
0     57.595993
1     17.028381
2     16.861436
3+     8.514190
Name: Dependents, dtype: float64
```

In [33]:
```
train['Dependents'].value_counts(normalize=True).plot.bar(title="Dependents
```

Out[33]:
```
<AxesSubplot:title={'center':'Dependents'}>
```



In [34]:
```
train["Education"].count()
```

Out[34]:
```
614
```

In [35]:
```
train["Education"].value_counts()
```

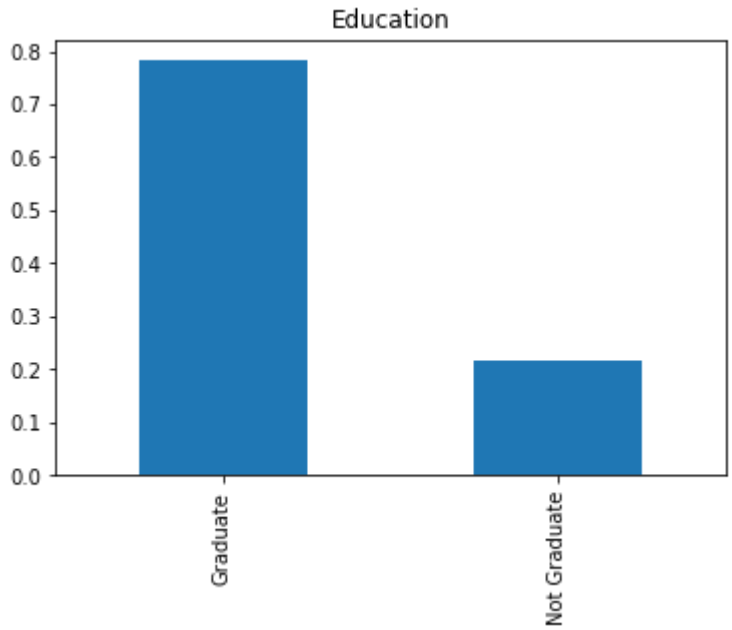
Out[35]:
```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

In [36]:
```
train["Education"].value_counts(normalize=True)*100
```

Out[36]:
```
Graduate        78.175896
Not Graduate    21.824104
Name: Education, dtype: float64
```

In [37]:
```
train["Education"].value_counts(normalize=True).plot.bar(title = "Educatior
```

Out[37]: <AxesSubplot:title={'center':'Education'}>



In [38]:
```
train["Property_Area"].count()
```

Out[38]: 614

In [39]:
```
train["Property_Area"].value_counts()
```

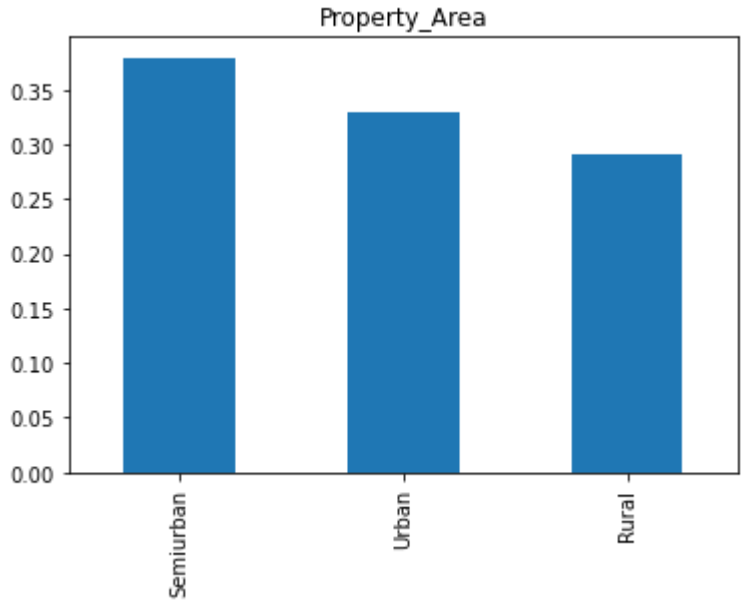
Out[39]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

In [40]:
```
train["Property_Area"].value_counts(normalize=True)*100
```

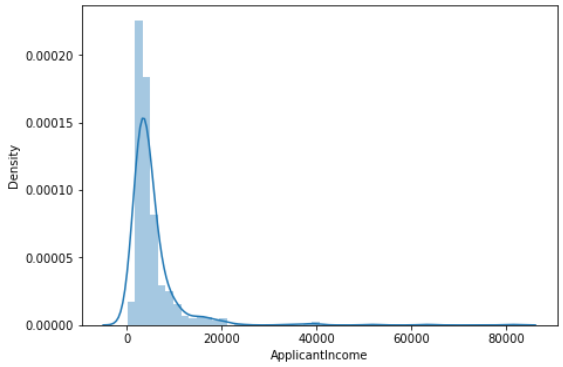
Out[40]:
```
Semiurban    37.947883
Urban        32.899023
Rural        29.153094
Name: Property_Area, dtype: float64
```

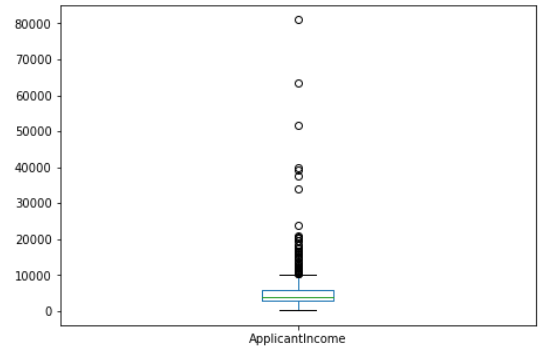
In [41]:
```
train["Property_Area"].value_counts(normalize=True).plot.bar(title="Propert
```

Out[41]: <AxesSubplot:title={'center':'Property_Area'}>

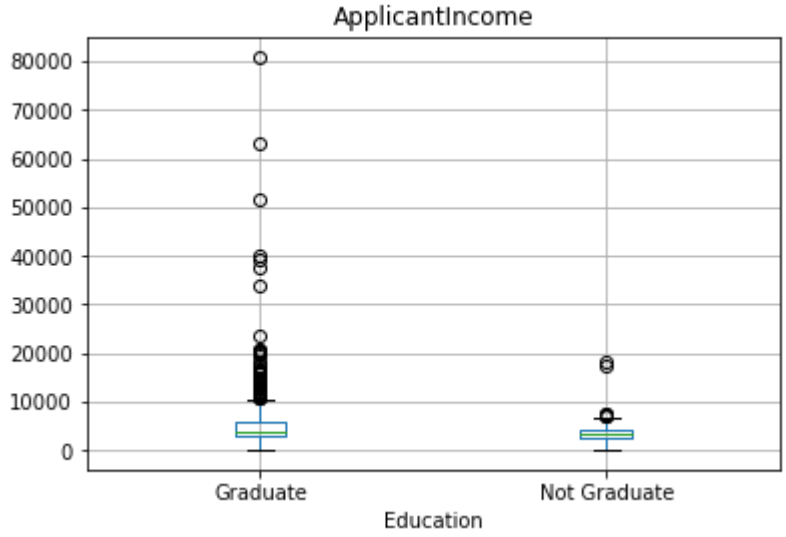


In [42]:
```
plt.figure(1)
plt.subplot(121)
sns.distplot(train["ApplicantIncome"]);

plt.subplot(122)
train["ApplicantIncome"].plot.box(figsize=(16,5))
plt.show()
```

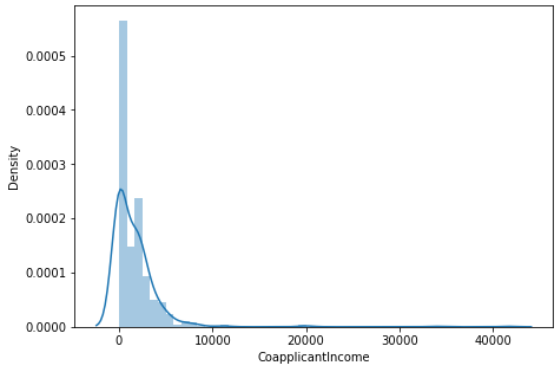


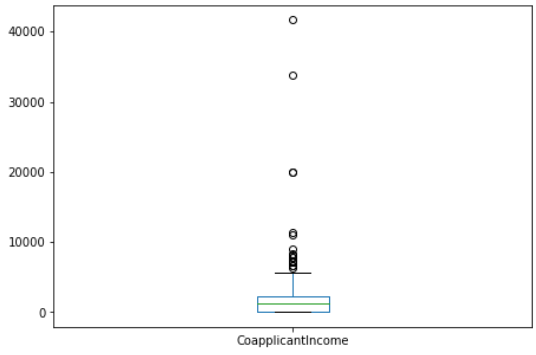


In [43]:
```
train.boxplot(column='ApplicantIncome',by="Education" )
plt.suptitle(" ")
plt.show()
```

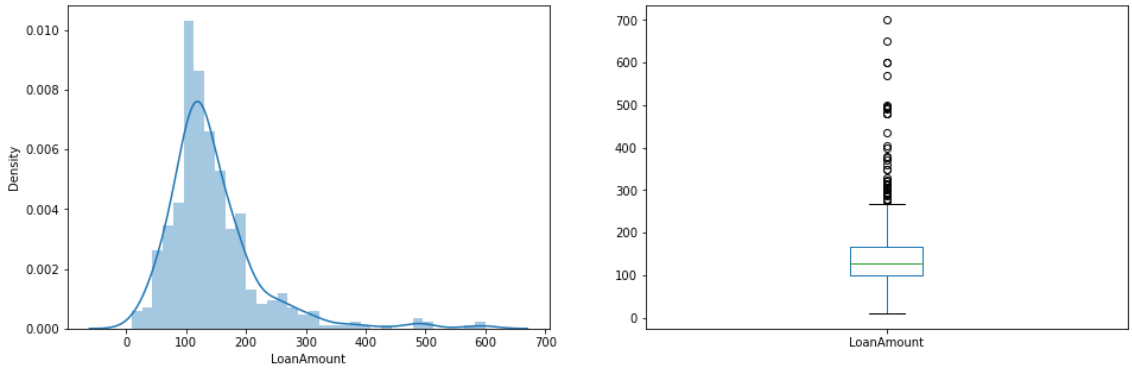


In [44]:
```
plt.figure(1)
plt.subplot(121)
sns.distplot(train["CoapplicantIncome"]);

plt.subplot(122)
train["CoapplicantIncome"].plot.box(figsize=(16,5))
plt.show()
```

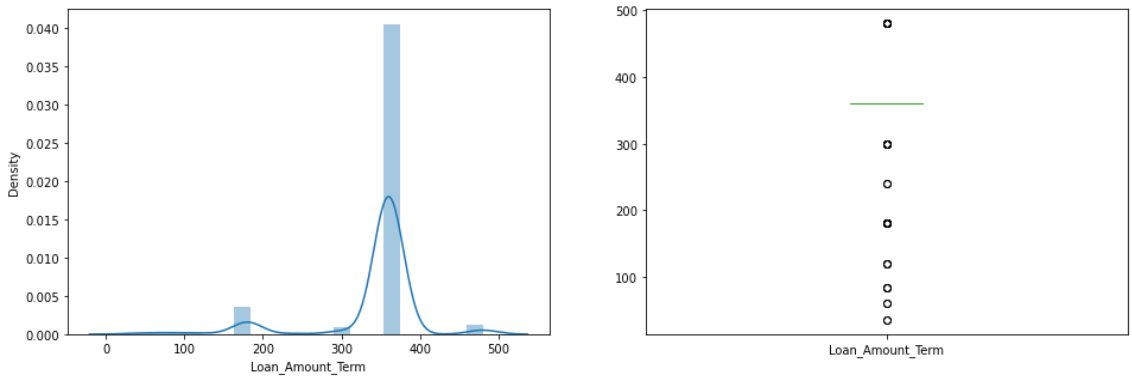




In [45]:
```
plt.figure(1)
plt.subplot(121)
df=train.dropna()
sns.distplot(df['LoanAmount']);

plt.subplot(122)
train['LoanAmount'].plot.box(figsize=(16,5))

plt.show()
```

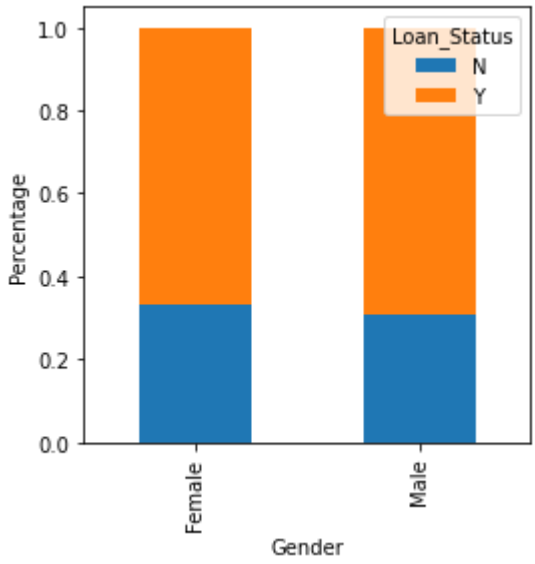


In [46]:
```
plt.figure(1)
plt.subplot(121)
df = train.dropna()
sns.distplot(df["Loan_Amount_Term"]);

plt.subplot(122)
df["Loan_Amount_Term"].plot.box(figsize=(16,5))
plt.show()
```



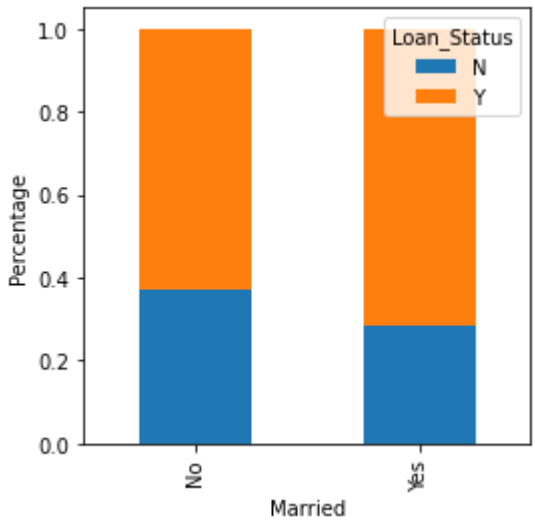
In [47]:
```
print(pd.crosstab(train["Gender"],train["Loan_Status"]))
Gender = pd.crosstab(train["Gender"],train["Loan_Status"])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True
plt.xlabel("Gender")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status    N    Y
Gender
Female        37   75
Male         150  339
```



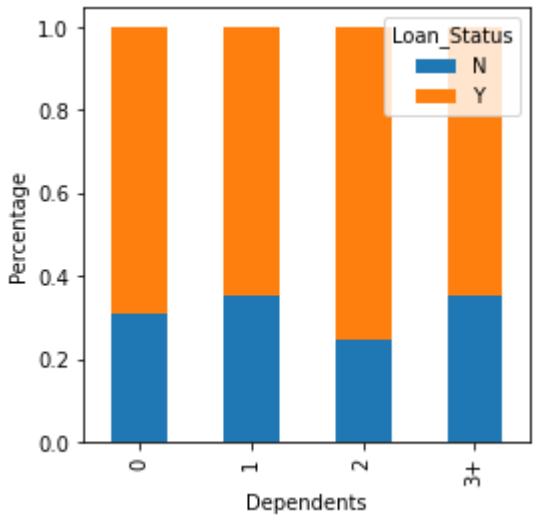
In [48]:
```
print(pd.crosstab(train["Married"],train["Loan_Status"]))
Married=pd.crosstab(train["Married"],train["Loan_Status"])
Married.div(Married.sum(1).astype(float),axis=0).plot(kind="bar",stacked=Tr
plt.xlabel("Married")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status    N    Y
Married
No            79  134
Yes          113  285
```



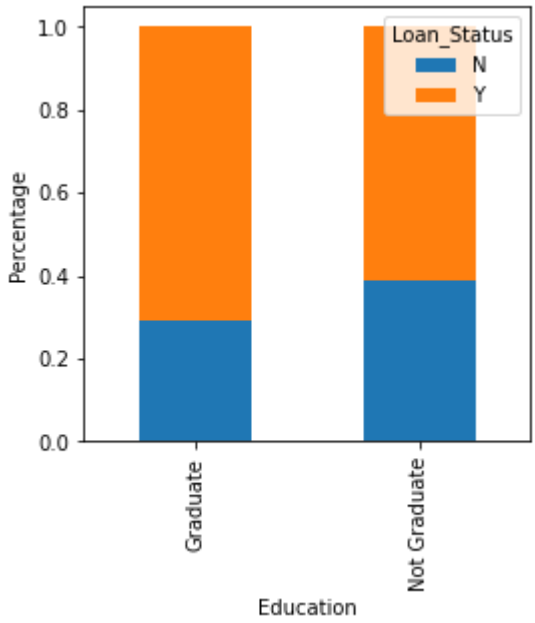
In [49]:
```
print(pd.crosstab(train['Dependents'],train["Loan_Status"]))
Dependents = pd.crosstab(train['Dependents'],train["Loan_Status"])
Dependents.div(Dependents.sum(1).astype(float),axis=0).plot(kind="bar",stac
plt.xlabel("Dependents")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status    N    Y
Dependents
0            107  238
1             36   66
2             25   76
3+            18   33
```



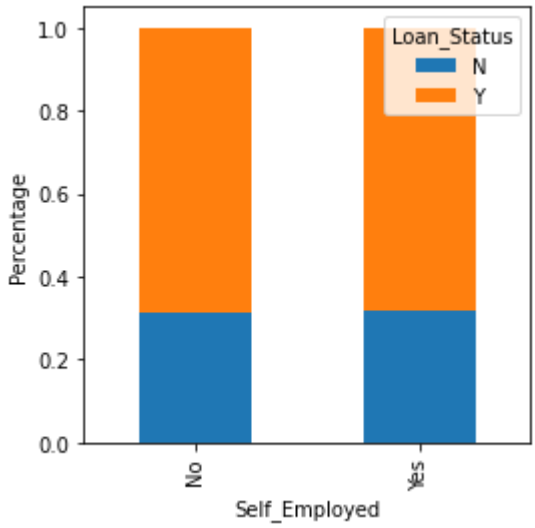
In [50]:
```
print(pd.crosstab(train["Education"],train["Loan_Status"]))
Education = pd.crosstab(train["Education"],train["Loan_Status"])
Education.div(Education.sum(1).astype(float),axis=0).plot(kind="bar",stacke
plt.xlabel("Education")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status     N    Y
Education
Graduate      140  340
Not Graduate   52   82
```



In [52]:
```
print(pd.crosstab(train["Self_Employed"],train["Loan_Status"]))
SelfEmployed = pd.crosstab(train["Self_Employed"],train["Loan_Status"])
SelfEmployed.div(SelfEmployed.sum(1).astype(float),axis=0).plot(kind="bar",
plt.xlabel("Self_Employed")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status      N    Y
Self_Employed
No             157  343
Yes             26   56
```

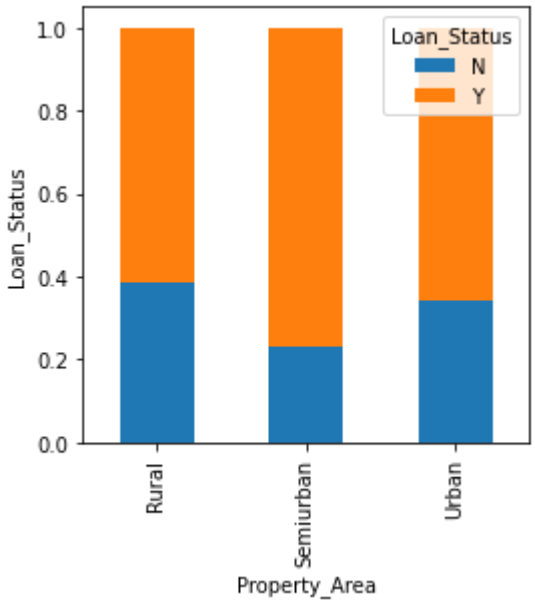


In [53]:
```
print(pd.crosstab(train["Credit_History"],train["Loan_Status"]))
CreditHistory = pd.crosstab(train["Credit_History"],train["Loan_Status"])
CreditHistory.div(CreditHistory.sum(1).astype(float),axis=0).plot(kind="bar
plt.xlabel("Credit_History")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status      N    Y
Credit_History
0.0             82    7
1.0             97  378
```

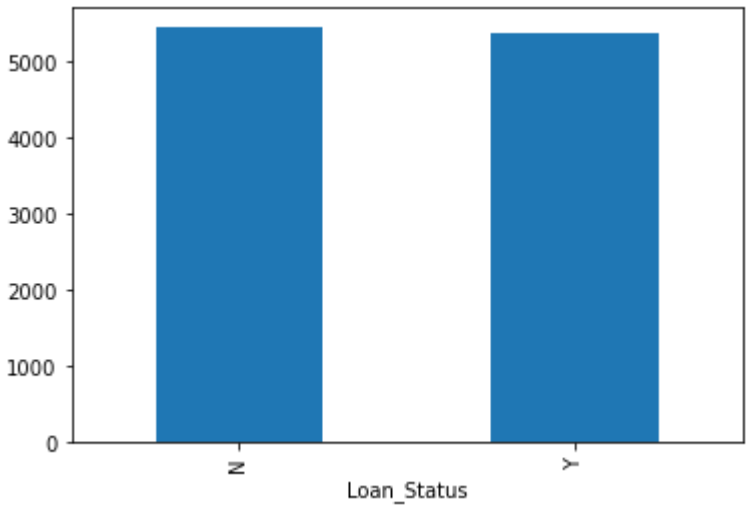
In [54]:
```
print(pd.crosstab(train["Property_Area"],train["Loan_Status"]))
PropertyArea = pd.crosstab(train["Property_Area"],train["Loan_Status"])
PropertyArea.div(PropertyArea.sum(1).astype(float),axis=0).plot(kind="bar",
plt.xlabel("Property_Area")
plt.ylabel("Loan_Status")
plt.show()
```

```
Loan_Status     N    Y
Property_Area
Rural          69  110
Semiurban      54  179
Urban          69  133
```



In [55]:
```
train.groupby("Loan_Status")['ApplicantIncome'].mean().plot.bar()
```

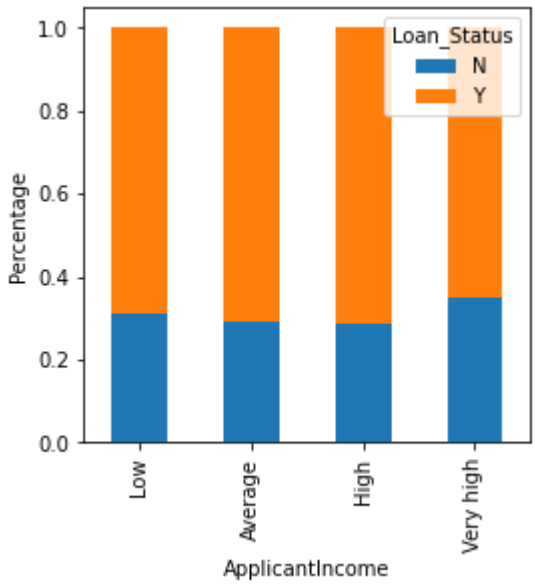
Out[55]:
```
<AxesSubplot:xlabel='Loan_Status'>
```



In [56]:
```
bins=[0,2500,4000,6000,81000]
group=['Low','Average','High', 'Very high']
train['Income_bin']=pd.cut(df['ApplicantIncome'],bins,labels=group)
```

In [57]:
```
print(pd.crosstab(train["Income_bin"],train["Loan_Status"]))
Income_bin = pd.crosstab(train["Income_bin"],train["Loan_Status"])
Income_bin.div(Income_bin.sum(1).astype(float),axis=0).plot(kind="bar",sta
plt.xlabel("ApplicantIncome")
plt.ylabel("Percentage")
plt.show()
```

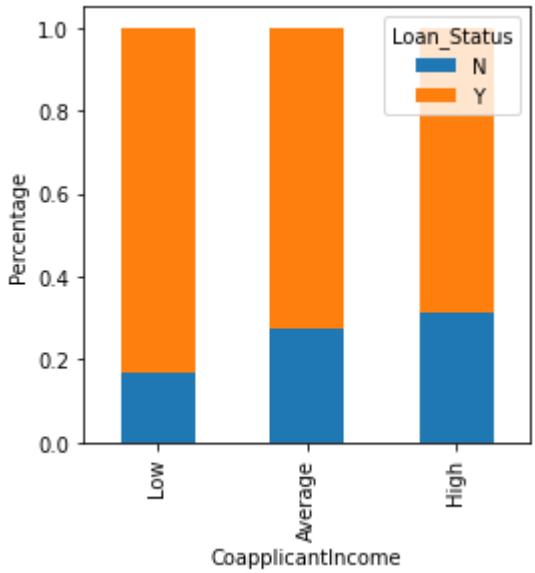
```
Loan_Status    N    Y
Income_bin
Low           26   57
Average       51  123
High          32   79
Very high     39   73
```



In [58]:
```
bins=[0,1000,3000,42000]
group =['Low','Average','High']
train['CoapplicantIncome_bin']=pd.cut(df["CoapplicantIncome"],bins,labels=g
```

In [59]:
```
print(pd.crosstab(train["CoapplicantIncome_bin"],train["Loan_Status"]))
CoapplicantIncome_Bin = pd.crosstab(train["CoapplicantIncome_bin"],train["L
CoapplicantIncome_Bin.div(CoapplicantIncome_Bin.sum(1).astype(float),axis=0
plt.xlabel("CoapplicantIncome")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status              N    Y
CoapplicantIncome_bin
Low                      3   15
Average                 46  123
High                    24   53
```

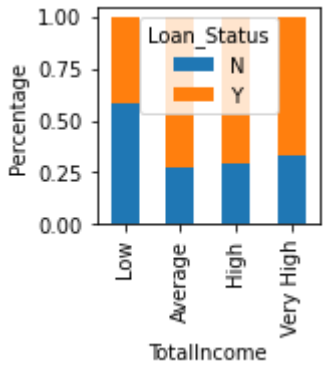


In [60]:
```
train["TotalIncome"]=train["ApplicantIncome"]+train["CoapplicantIncome"]
```

In [61]:
```
bins =[0,2500,4000,6000,81000]
group=['Low','Average','High','Very High']
train["TotalIncome_bin"]=pd.cut(train["TotalIncome"],bins,labels=group)
```

In [62]:
```
print(pd.crosstab(train["TotalIncome_bin"],train["Loan_Status"]))
TotalIncome = pd.crosstab(train["TotalIncome_bin"],train["Loan_Status"])
TotalIncome.div(TotalIncome.sum(1).astype(float),axis=0).plot(kind='bar',st
plt.xlabel("TotalIncome")
plt.ylabel("Percentage")
plt.show()
```

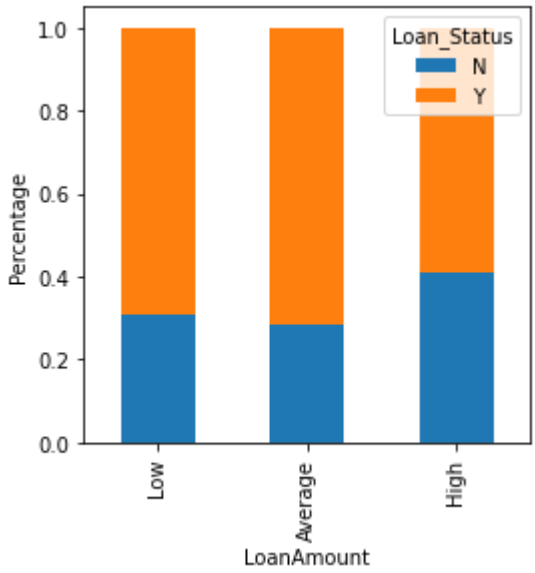
Loan_Status	N	Y
TotalIncome_bin		
Low	14	10
Average	32	87
High	65	159
Very High	81	166



In [63]:
```
bins = [0,100,200,700]
group=['Low','Average','High']
train["LoanAmount_bin"]=pd.cut(df["LoanAmount"],bins,labels=group)
```

In [64]:
```
print(pd.crosstab(train["LoanAmount_bin"],train["Loan_Status"]))
LoanAmount=pd.crosstab(train["LoanAmount_bin"],train["Loan_Status"])
LoanAmount.div(LoanAmount.sum(1).astype(float),axis=0).plot(kind='bar',stac
plt.xlabel("LoanAmount")
plt.ylabel("Percentage")
plt.show()
```

```
Loan_Status       N    Y
LoanAmount_bin
Low              38   86
Average          83  207
High             27   39
```



In [65]:
```
train=train.drop(["Income_bin","CoapplicantIncome_bin","LoanAmount_bin","To
```

In [66]:
```
train['Dependents'].replace('3+',3,inplace=True)
test['Dependents'].replace('3+',3,inplace=True)
train['Loan_Status'].replace('N', 0,inplace=True)
train['Loan_Status'].replace('Y', 1,inplace=True)
```

In [67]:
```
matrix = train.corr()
f, ax = plt.subplots(figsize=(10, 12))
sns.heatmap(matrix, vmax=.8, square=True, cmap="BuPu",annot=True);
```

In [68]:
```
train.isnull().sum()
```

Out[68]:
```
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 [69]:
```
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=Tr
```

In [70]:
```
train["Loan_Amount_Term"].value_counts()
```

Out[70]:
```
360.0    512
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 [71]:
```
train["Loan_Amount_Term"].fillna(train["Loan_Amount_Term"].mode()[0],inplac
```

In [72]:
```
train["Loan_Amount_Term"].value_counts()
```

Out[72]:
```
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 [73]:
```
train["LoanAmount"].fillna(train["LoanAmount"].median(),inplace=True)
```

In [74]:
```
train.isnull().sum()
```

Out[74]:
```
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 [75]:
```
test.isnull().sum()
```

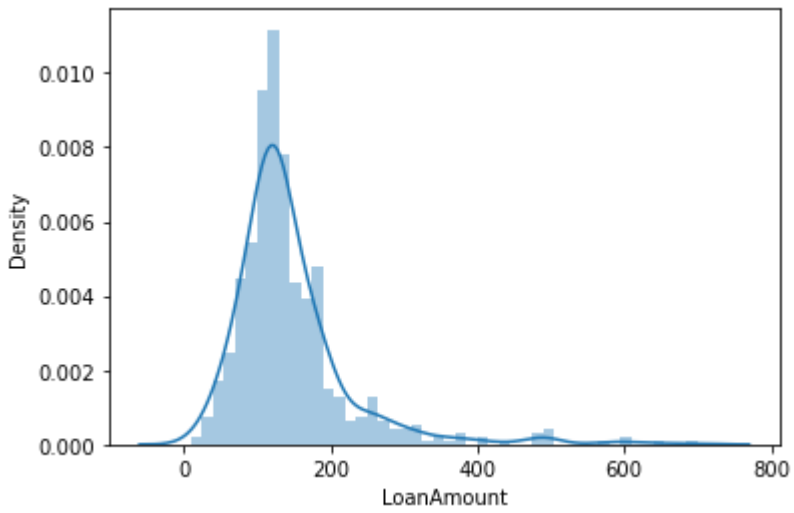
Out[75]:
```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

In [76]:
```
test["Gender"].fillna(test["Gender"].mode()[0],inplace=True)
test['Dependents'].fillna(test["Dependents"].mode()[0],inplace=True)
test["Self_Employed"].fillna(test["Self_Employed"].mode()[0],inplace=True)
test["Loan_Amount_Term"].fillna(test["Loan_Amount_Term"].mode()[0],inplace=
test["Credit_History"].fillna(test["Credit_History"].mode()[0],inplace=True
test["LoanAmount"].fillna(test["LoanAmount"].median(),inplace=True)
```

In [77]:
```
test.isnull().sum()
```

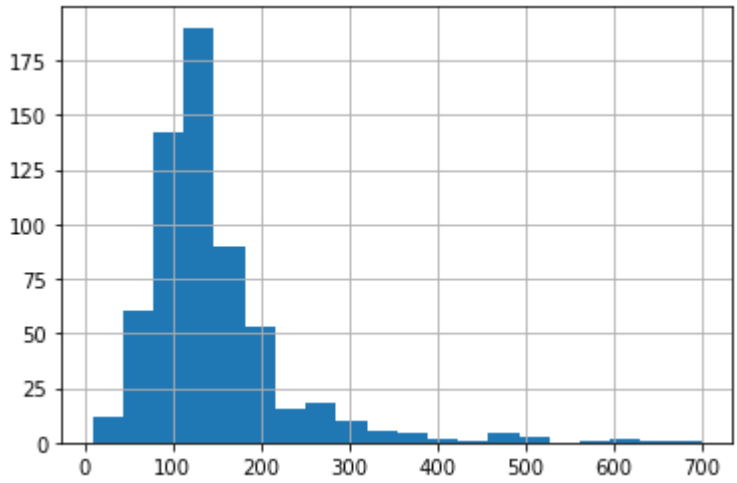
Out[77]:
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

In [78]:
```
sns.distplot(train["LoanAmount"]);
```



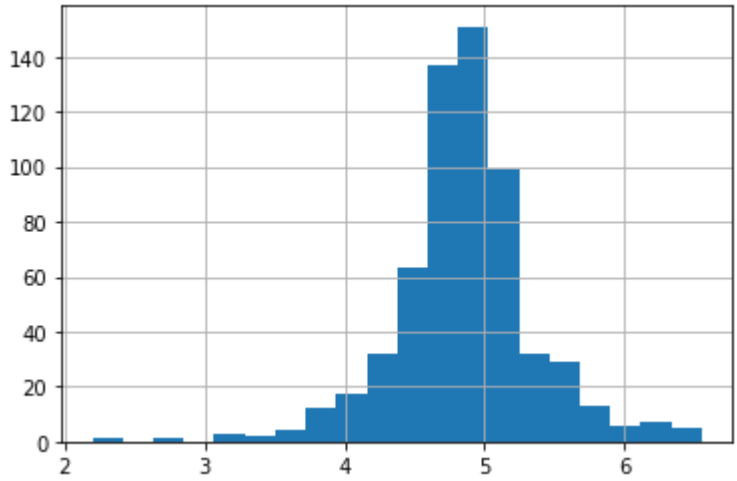
In [79]:
```
train['LoanAmount'].hist(bins=20)
```

Out[79]: <AxesSubplot:>



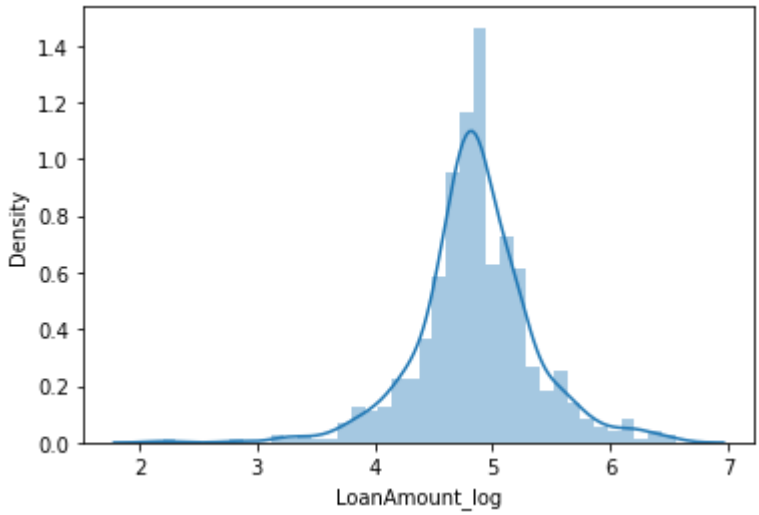
In [80]:
```
train['LoanAmount_log'] = np.log(train['LoanAmount'])
train['LoanAmount_log'].hist(bins=20)
```

Out[80]: <AxesSubplot:>



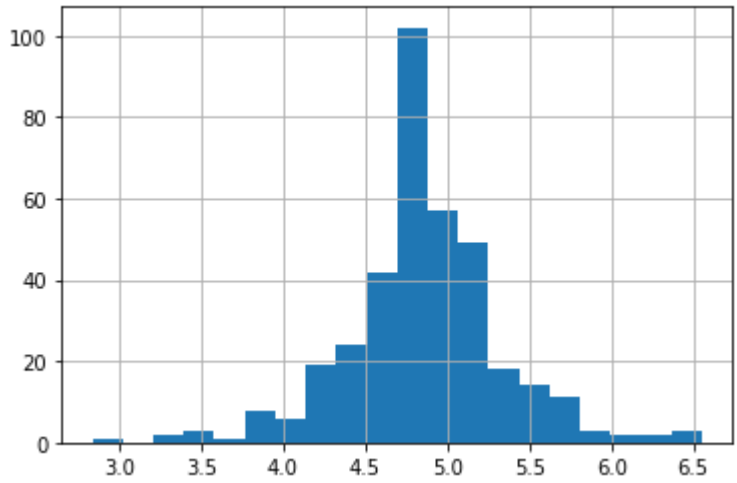
In [81]:
```
sns.distplot(train["LoanAmount_log"])
```

Out[81]: <AxesSubplot:xlabel='LoanAmount_log', ylabel='Density'>



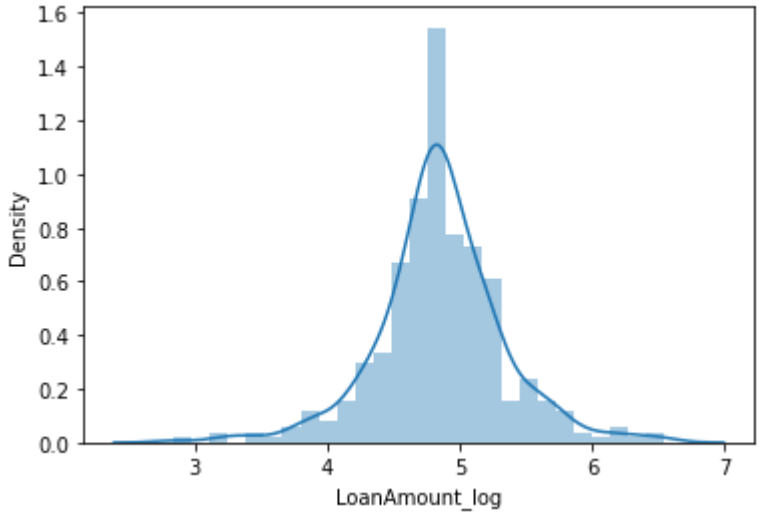
In [82]:
```
test["LoanAmount_log"]=np.log(train["LoanAmount"])
test['LoanAmount_log'].hist(bins=20)
```

Out[82]: <AxesSubplot:>



In [83]:
```
sns.distplot(test["LoanAmount_log"])
```

Out[83]:
```
<AxesSubplot:xlabel='LoanAmount_log', ylabel='Density'>
```



In [84]:
```
train["TotalIncome"]=train["ApplicantIncome"]+train["CoapplicantIncome"]
```

In [85]:
```
train[["TotalIncome"]].head()
```

Out[85]:

	TotalIncome
0	5849.0
1	6091.0
2	3000.0
3	4941.0
4	6000.0

In [86]:
```
test["TotalIncome"]=test["ApplicantIncome"]+test["CoapplicantIncome"]
```

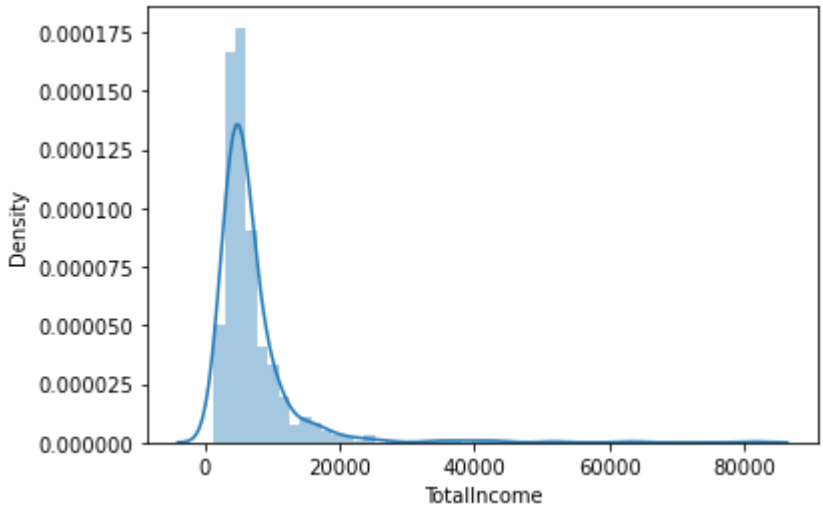
In [87]:
```
test[["TotalIncome"]].head()
```

Out[87]:

	TotalIncome
0	5720
1	4576
2	6800
3	4886
4	3276

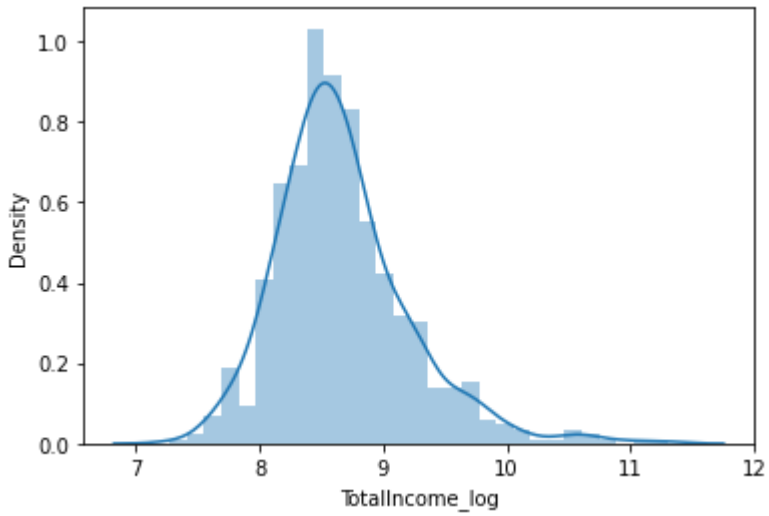
In [88]:
```
sns.distplot(train["TotalIncome"])
```

Out[88]: <AxesSubplot:xlabel='TotalIncome', ylabel='Density'>



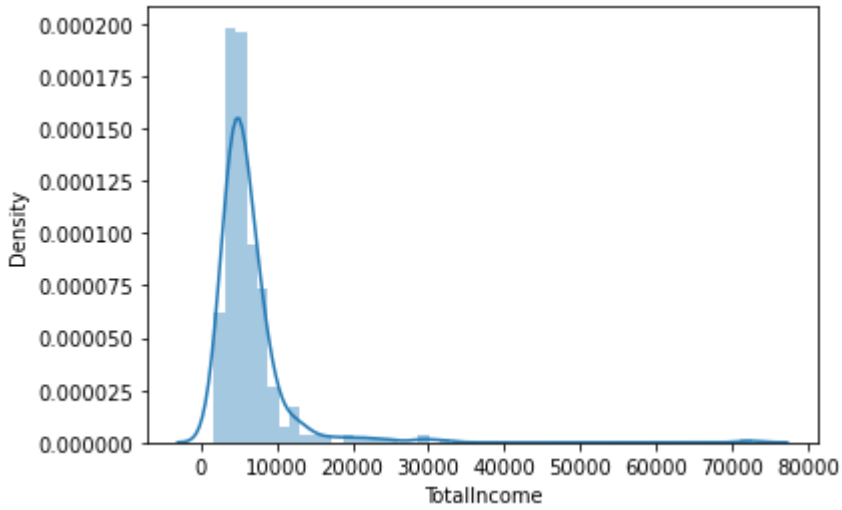
In [89]:
```
train["TotalIncome_log"]=np.log(train["TotalIncome"])
sns.distplot(train["TotalIncome_log"])
```

Out[89]: <AxesSubplot:xlabel='TotalIncome_log', ylabel='Density'>



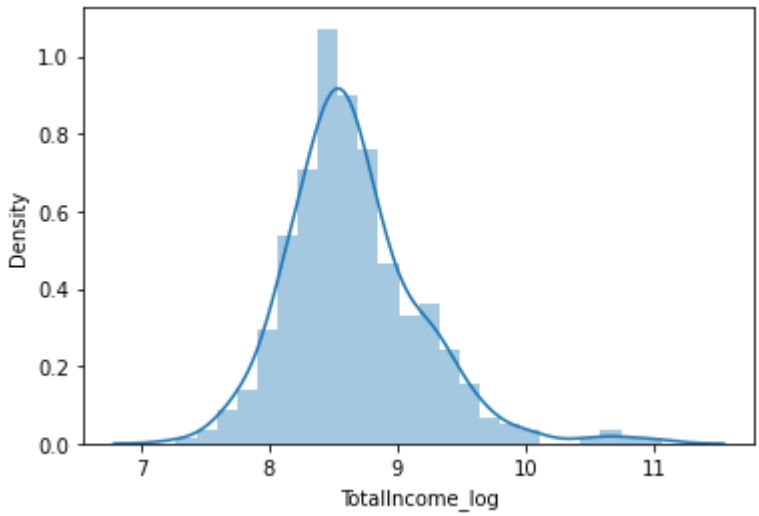
In [90]:
```
sns.distplot(test["TotalIncome"])
```

Out[90]:
```
<AxesSubplot:xlabel='TotalIncome', ylabel='Density'>
```



In [91]:
```
test["TotalIncome_log"] = np.log(train["TotalIncome"])
sns.distplot(test["TotalIncome_log"])
```

Out[91]:
```
<AxesSubplot:xlabel='TotalIncome_log', ylabel='Density'>
```



In [92]:
```
train["EMI"]=train["LoanAmount"]/train["Loan_Amount_Term"]
test["EMI"]=test["LoanAmount"]/test["Loan_Amount_Term"]
```

In [93]:
```
train[["EMI"]].head()
```

Out[93]:

	EMI
0	0.355556
1	0.355556
2	0.183333
3	0.333333
4	0.391667

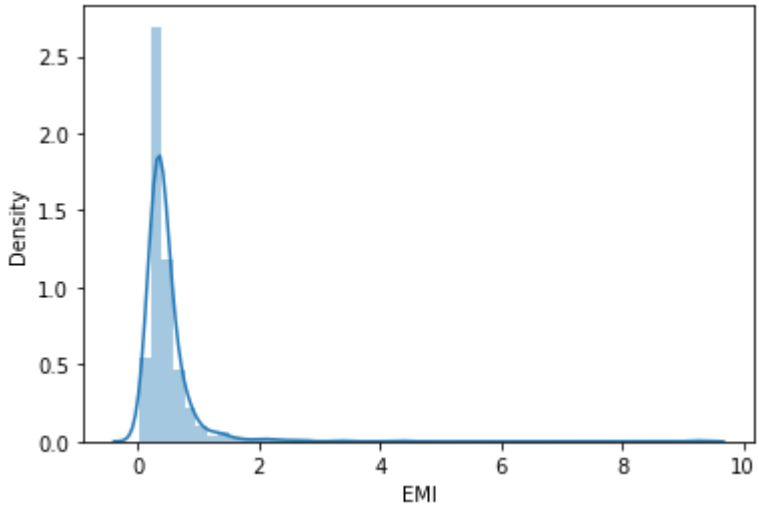
In [94]:
```
test[["EMI"]].head()
```

Out[94]:

	EMI
0	0.305556
1	0.350000
2	0.577778
3	0.277778
4	0.216667

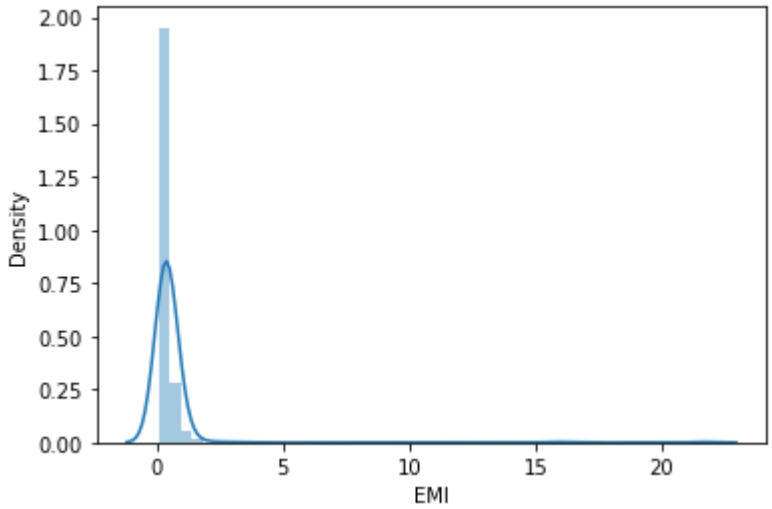
In [95]:
```
sns.distplot(train["EMI"])
```

Out[95]: <AxesSubplot:xlabel='EMI', ylabel='Density'>



In [96]:
```
sns.distplot(test["EMI"])
```

Out[96]: <AxesSubplot:xlabel='EMI', ylabel='Density'>



In [97]:
```
train["Balance_Income"] = train["TotalIncome"]-train["EMI"]*1000
test["Balance_Income"] = test["TotalIncome"]-test["EMI"]
```

In [98]:
```
train[["Balance_Income"]].head()
```

Out[98]:

	Balance_Income
0	5493.444444
1	5735.444444
2	2816.666667
3	4607.666667
4	5608.333333

In [99]:
```
test[["Balance_Income"]].head()
```

Out[99]:

	Balance_Income
0	5719.694444
1	4575.650000
2	6799.422222
3	4885.722222
4	3275.783333

In [100]:
```
train=train.drop(["ApplicantIncome","CoapplicantIncome","LoanAmount","Loan_
```

In [101]:
```
train.head()
```

Out[101]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Prope
0	LP001002	Male	No	0	Graduate	No	1.0	
1	LP001003	Male	Yes	1	Graduate	No	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	1.0	
4	LP001008	Male	No	0	Graduate	No	1.0	

In [102]:
```
test = test.drop(["ApplicantIncome","CoapplicantIncome","LoanAmount","Loan_
```

In [103]:
```
test.head()
```

Out[103]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Prope
0	LP001015	Male	Yes	0	Graduate	No	1.0	
1	LP001022	Male	Yes	1	Graduate	No	1.0	
2	LP001031	Male	Yes	2	Graduate	No	1.0	
3	LP001035	Male	Yes	2	Graduate	No	1.0	
4	LP001051	Male	No	0	Not Graduate	No	1.0	

In [104]:
```
train=train.drop("Loan_ID",axis=1)
test=test.drop("Loan_ID",axis=1)
```

In [105]:
```
train.head(3)
```

Out[105]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	L
0	Male	No	0	Graduate	No	1.0	Urban	
1	Male	Yes	1	Graduate	No	1.0	Rural	
2	Male	Yes	0	Graduate	Yes	1.0	Urban	

In [106]:
```
test.head(3)
```

Out[106]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	L
0	Male	Yes	0	Graduate	No	1.0	Urban	
1	Male	Yes	1	Graduate	No	1.0	Urban	
2	Male	Yes	2	Graduate	No	1.0	Urban	

In [107]:
```
X=train.drop("Loan_Status",1)
```

In [108]:
```
X.head(2)
```

Out[108]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	L
0	Male	No	0	Graduate	No	1.0	Urban	
1	Male	Yes	1	Graduate	No	1.0	Rural	

In [109]:
```
y=train[["Loan_Status"]]
```

In [110]:
```
y.head(2)
```

Out[110]:

	Loan_Status
0	1
1	0

In [111]:
```
X = pd.get_dummies(X)
```

In [112]:
```
X.head(3)
```

Out[112]:

	Credit_History	LoanAmount_log	TotalIncome	TotalIncome_log	EMI	Balance_Income
0	1.0	4.852030	5849.0	8.674026	0.355556	5493.444444
1	1.0	4.852030	6091.0	8.714568	0.355556	5735.444444
2	1.0	4.189655	3000.0	8.006368	0.183333	2816.666667

3 rows × 21 columns

In [113]:
```
train=pd.get_dummies(train)
test=pd.get_dummies(test)
```

In [114]:
```
train.head(3)
```

Out[114]:

	Credit_History	Loan_Status	LoanAmount_log	TotalIncome	TotalIncome_log	EMI	B
0	1.0	1	4.852030	5849.0	8.674026	0.355556	
1	1.0	0	4.852030	6091.0	8.714568	0.355556	
2	1.0	1	4.189655	3000.0	8.006368	0.183333	

3 rows × 22 columns

In [115]:
```
test.head(3)
```

Out[115]:

	Credit_History	LoanAmount_log	TotalIncome	TotalIncome_log	EMI	Balance_Income
0	1.0	4.852030	5720	8.674026	0.305556	5719.694444
1	1.0	4.852030	4576	8.714568	0.350000	4575.650000
2	1.0	4.189655	6800	8.006368	0.577778	6799.422222

3 rows × 21 columns

In [116]:
```
from sklearn.model_selection import train_test_split
```

In [117]:
```
x_train,x_cv,y_train,y_cv=train_test_split(X,y,test_size=0.3,random_state=1
```

In [118]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [119]:
```
logistic_model = LogisticRegression(random_state=1)
```

In [120]:
```
logistic_model.fit(x_train,y_train)
```

Out[120]:
```
LogisticRegression(random_state=1)
```

In [121]:
```
pred_cv_logistic=logistic_model.predict(x_cv)
```

In [122]:
```
score_logistic =accuracy_score(pred_cv_logistic,y_cv)*100
```

In [123]:
```
score_logistic
```

Out[123]:
```
75.67567567567568
```

In [124]:
```
pred_test_logistic = logistic_model.predict(test)
```

In [125]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [126]:
```
tree_model = DecisionTreeClassifier(random_state=1)
```

In [127]:
```
tree_model.fit(x_train,y_train)
```

Out[127]:
```
DecisionTreeClassifier(random_state=1)
```

In [128]:
```
pred_cv_tree=tree_model.predict(x_cv)
```

In [129]:
```
score_tree =accuracy_score(pred_cv_tree,y_cv)*100
```

In [130]:
```
score_tree
```

Out[130]:
```
71.35135135135135
```

In [131]:
```
pred_test_tree = tree_model.predict(test)
```

In [132]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [133]:
```
forest_model = RandomForestClassifier(random_state=1,max_depth=10,n_estimat
```

In [134]:
```
forest_model.fit(x_train,y_train)
```

Out[134]:
```
RandomForestClassifier(max_depth=10, n_estimators=50, random_state=1)
```

In [135]:
```
pred_cv_forest=forest_model.predict(x_cv)
```

In [136]:
```
score_forest = accuracy_score(pred_cv_forest,y_cv)*100
```

In [137]:
```
score_forest
```

Out[137]:
```
77.83783783783784
```

In [138]:
```
pred_test_forest=forest_model.predict(test)
```

In [139]:
```
from sklearn.model_selection import GridSearchCV
```

In [140]:
```
paramgrid = {'max_depth': list(range(1,20,2)),'n_estimators':list(range(1,2
```

In [141]:
```
grid_search = GridSearchCV(RandomForestClassifier(random_state=1),paramgric
```

In [142]:
```
grid_search.fit(x_train,y_train)
```

Out[142]:
```
GridSearchCV(estimator=RandomForestClassifier(random_state=1),
             param_grid={'max_depth': [1, 3, 5, 7, 9, 11, 13, 15, 17, 1
9],
                         'n_estimators': [1, 21, 41, 61, 81, 101, 121, 14
1, 161,
                                          181]})
```

In [143]:
```
grid_search.best_estimator_
```

Out[143]:
```
RandomForestClassifier(max_depth=7, n_estimators=41, random_state=1)
```

In [144]:
```
grid_forest_model = RandomForestClassifier(random_state=1,max_depth=3,n_est
```

In [145]:
```
grid_forest_model.fit(x_train,y_train)
```

Out[145]:
```
RandomForestClassifier(max_depth=3, n_estimators=101, random_state=1)
```

In [146]:
```
pred_grid_forest = grid_forest_model.predict(x_cv)
```

In [147]:
```
score_grid_forest = accuracy_score(pred_grid_forest,y_cv)*100
```

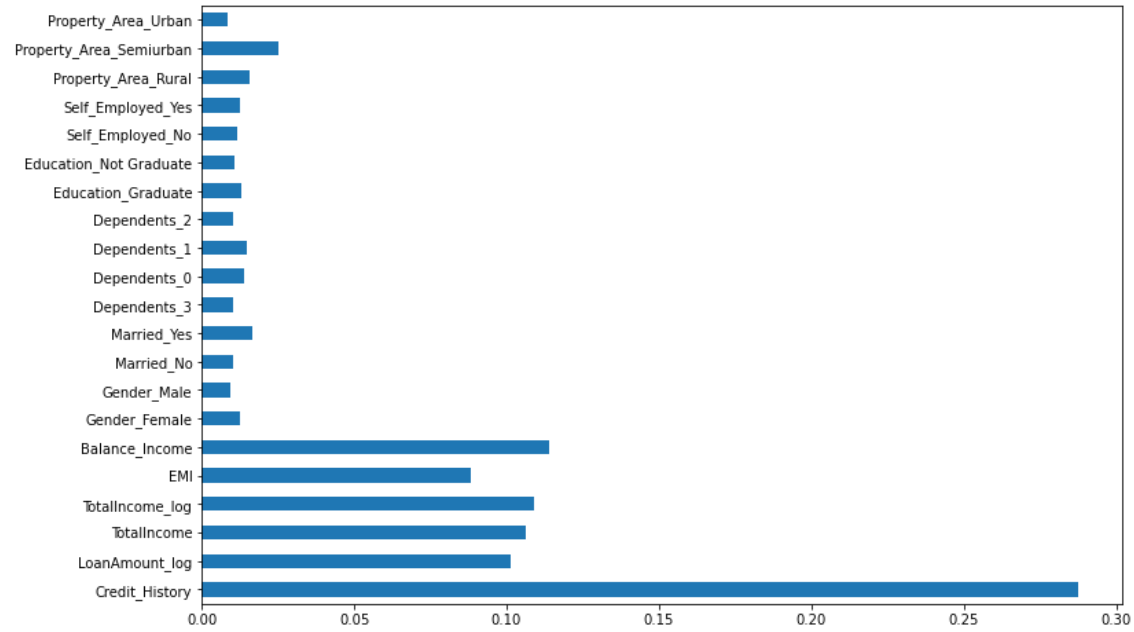
In [148]:
```
score_grid_forest
```

Out[148]:
```
76.75675675675676
```

In [149]:
```
pred_grid_forest_test = grid_forest_model.predict(test)
```

In [152]:
```
importances = pd.Series(forest_model.feature_importances_,index=X.columns)
importances.plot(kind='barh', figsize=(12,8))
```

Out[152]:
```
<AxesSubplot:>
```



In [ ]:
```

```