```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
train_data=pd.read_csv("/content/loan-train.csv")
test_data=pd.read_csv("/content/loan-test.csv")
print("Train Data\n")
print(train_data.head())
print("\nTest Data\n")
print(test_data.head())
```

```
Train Data

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

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

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y

Test Data

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

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
```

```
3             NaN         Urban
4             1.0         Urban
```

In [3]:
```
print("Fielda in Train Data:\n")
print(train_data.dtypes)
print("\n",train_data.shape)
print("\nFields in Test Data:\n")
print(test_data.dtypes)
print("\n",test_data.shape)
```

```
Fielda in Train Data:

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

 (614, 13)

Fields in Test Data:

Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome      int64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
dtype: object

 (367, 12)
```

In [4]:
```
train_copy=train_data.copy()
```

In [5]:
```
train_copy.isnull().sum()
```

Out[5]:
```
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 [...

```
train_copy["Gender"].fillna(train_copy["Gender"].mode()[0],inplace=True)
train_copy["Married"].fillna(train_copy["Married"].mode()[0],inplace=True)
train_copy["Dependents"].fillna(train_copy["Dependents"].mode()[0],inplace=True)
train_copy["Self_Employed"].fillna(train_copy["Self_Employed"].mode()[0],inplace=True)
train_copy["Credit_History"].fillna(train_copy["Credit_History"].mode()[0],inplace=True)
train_copy["Loan_Amount_Term"].fillna(train_copy["Loan_Amount_Term"].mode()[0],inplace=Tr
train_copy["LoanAmount"].fillna(train_copy["LoanAmount"].median(), inplace=True)
```

In [7]:

```
train_copy.isnull().sum()
```

Out[7]:

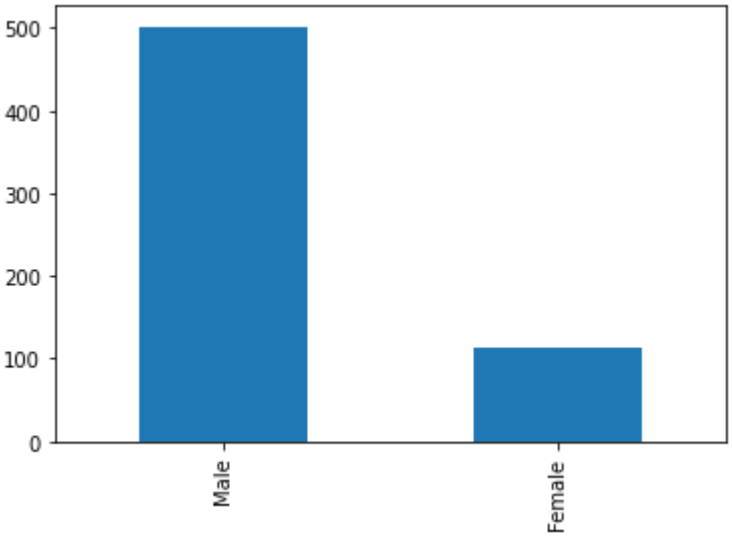
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [8]:

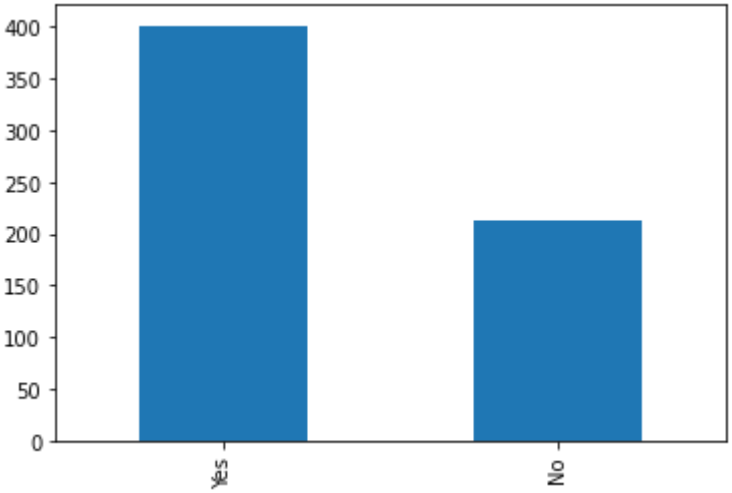
```
for i in train_copy.columns:
  print("\n",i)
  if i=="Loan_ID":
    continue
  if(i=="Credit_History"):
    train_copy[i].value_counts().plot.bar()
    plt.show()
    continue
  if train_copy[i].dtype=="object":
    train_copy[i].value_counts().plot.bar()
    plt.show()
  else:
    sns.distplot(train_copy[i])
    plt.show()
```

```
Loan_ID
```

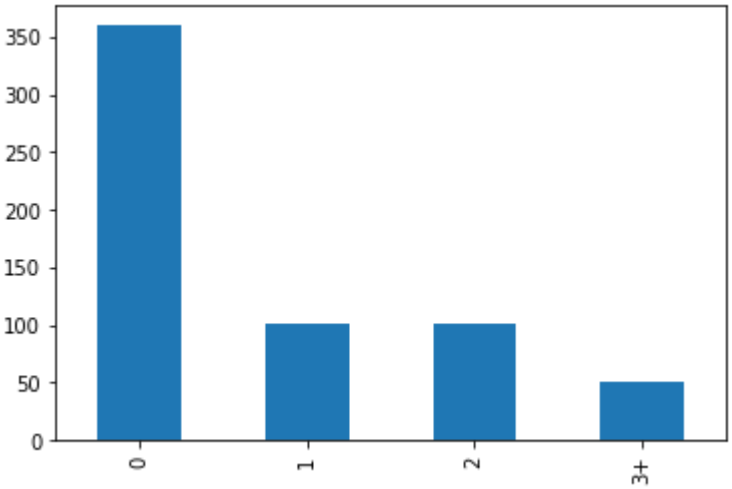
```
Gender
```



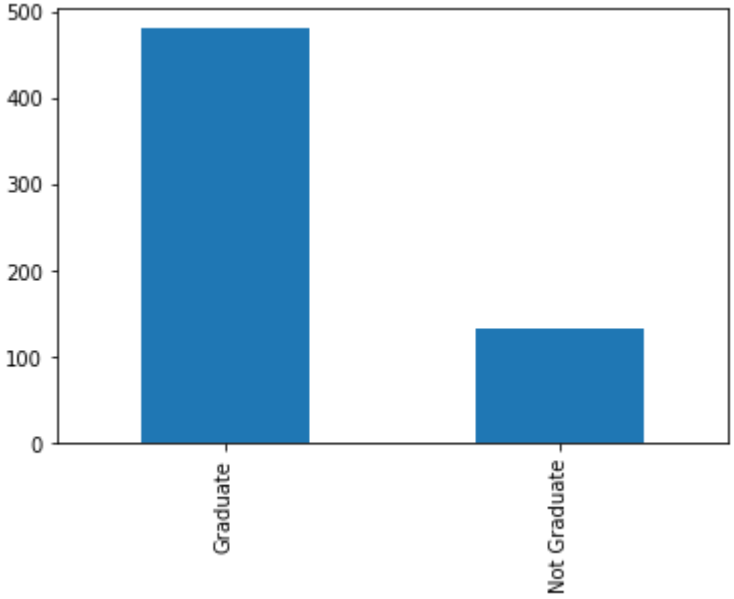
Married



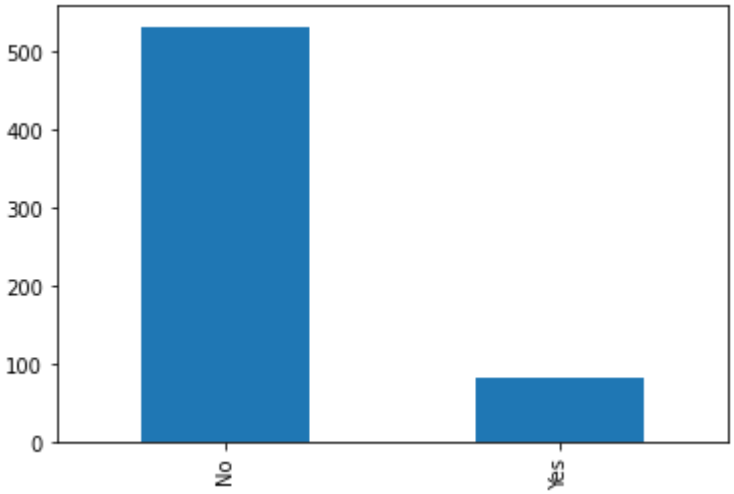
Dependents



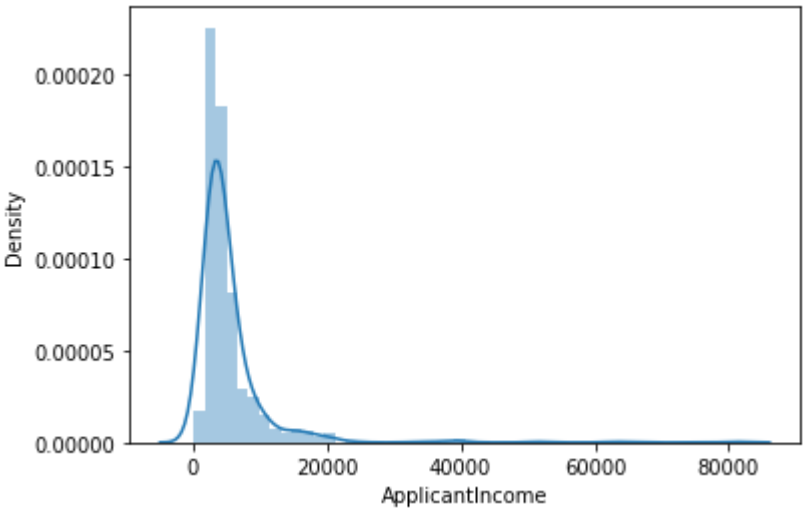
Education



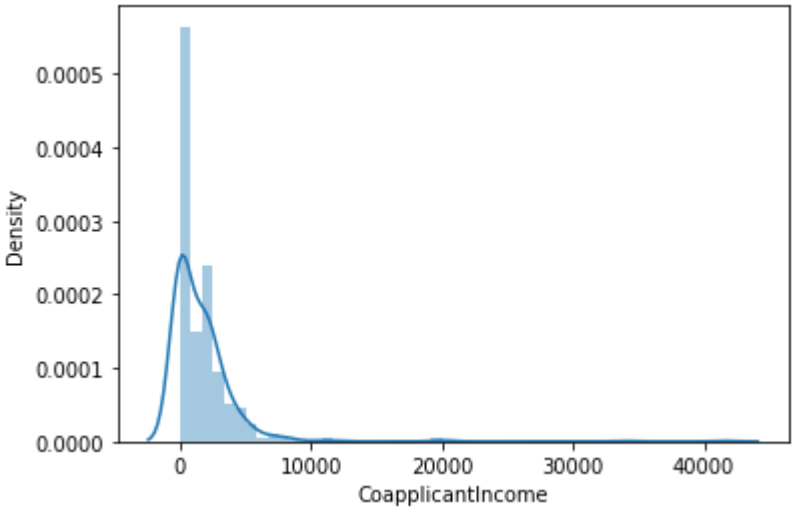
Self_Employed



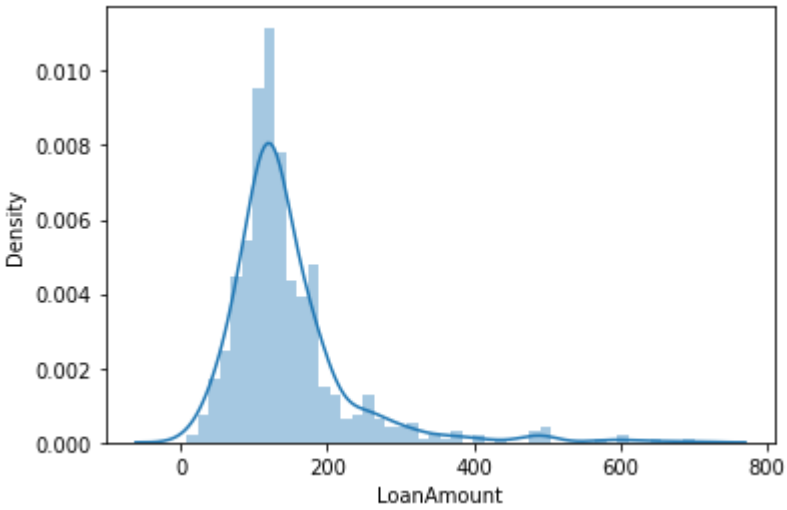
ApplicantIncome



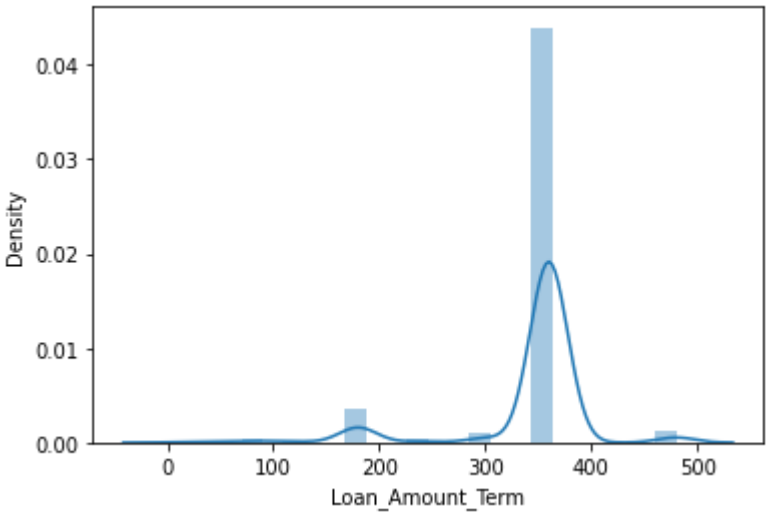
CoapplicantIncome



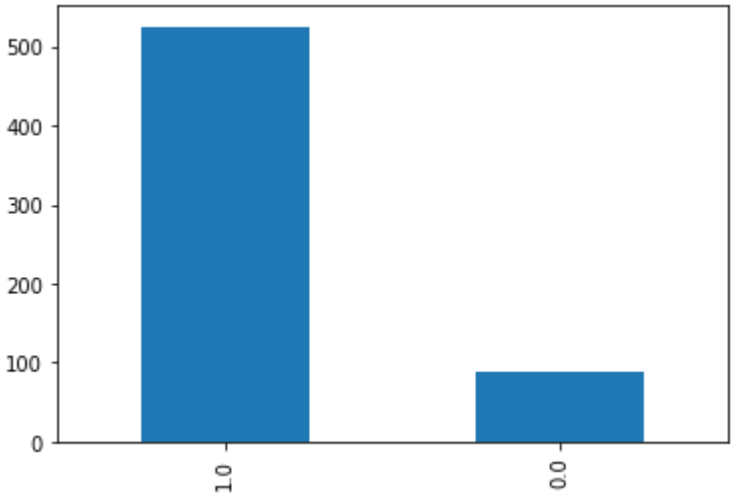
LoanAmount



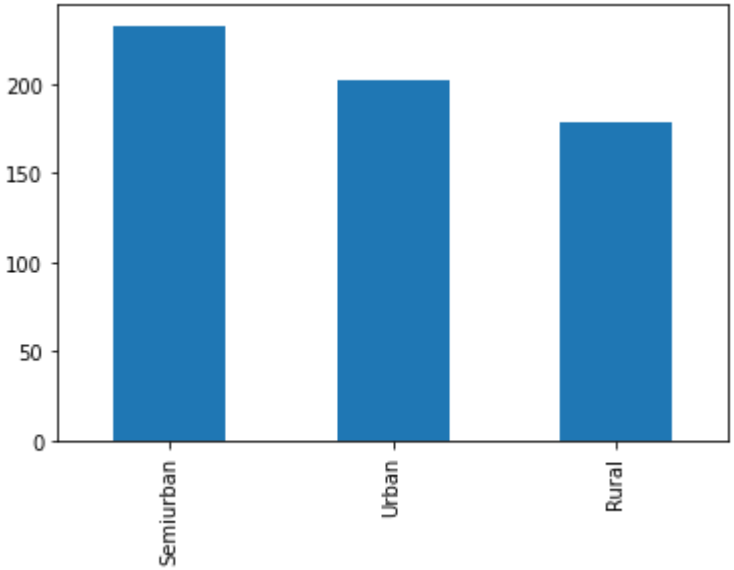
Loan_Amount_Term



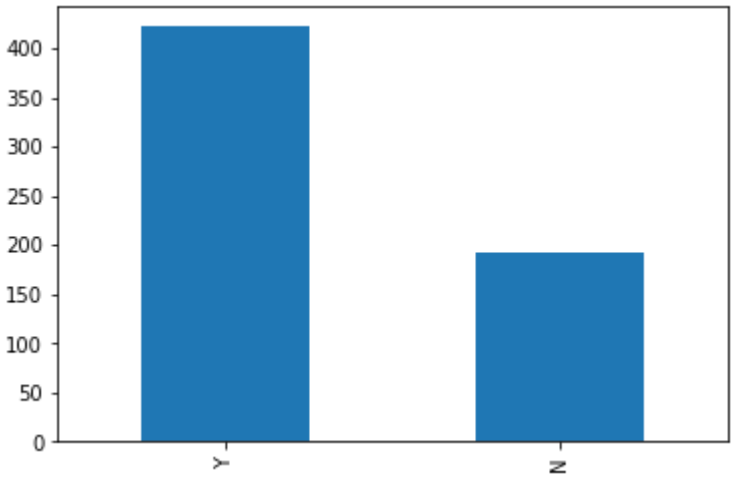
Credit_History



Property_Area



Loan_Status

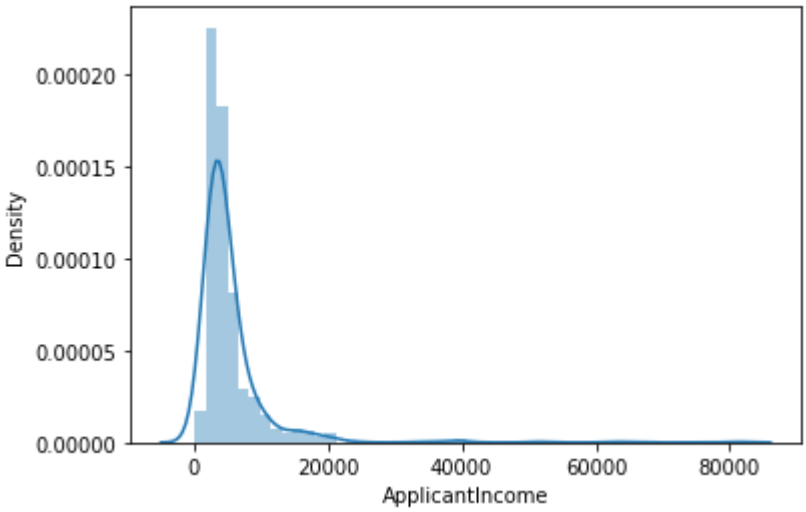


In [ ]:
```
train_copy2=train_copy.copy()
train_copy2["LoanAmount"]=np.log(train_copy["LoanAmount"])
train_copy2["Loan_Amount_Term"]=np.log(train_copy["Loan_Amount_Term"])
```

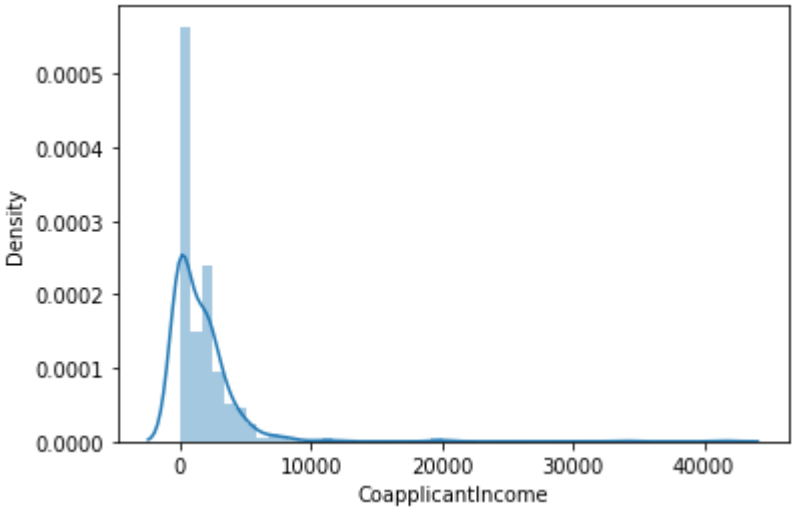
In [ ]:
```
for i in train_copy2.columns:

  if train_copy2[i].dtype!="object":
```

```
        print("\n",i)
        sns.distplot(train_copy2[i])
        plt.show()
```

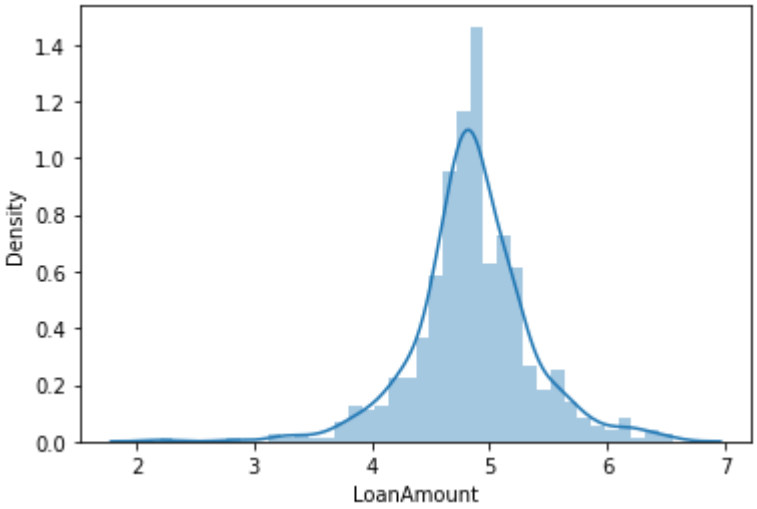
ApplicantIncome



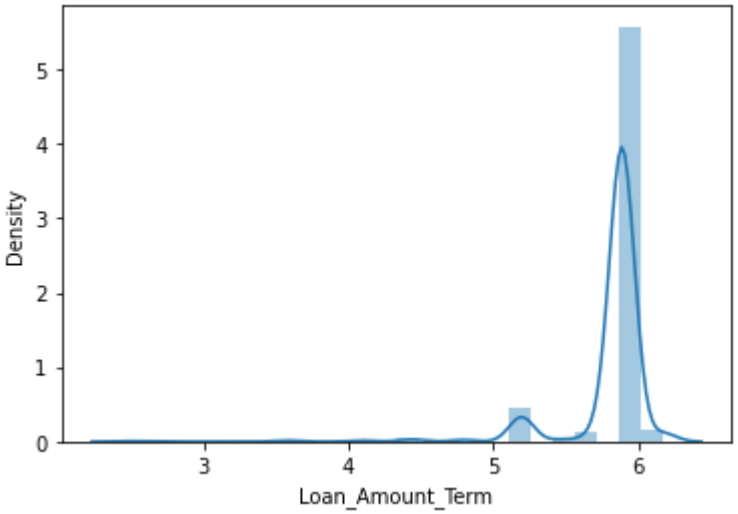
CoapplicantIncome



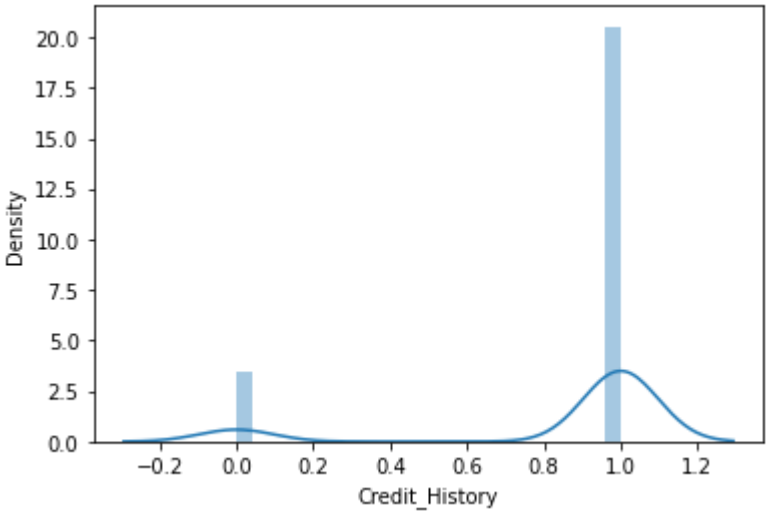
LoanAmount



Loan_Amount_Term



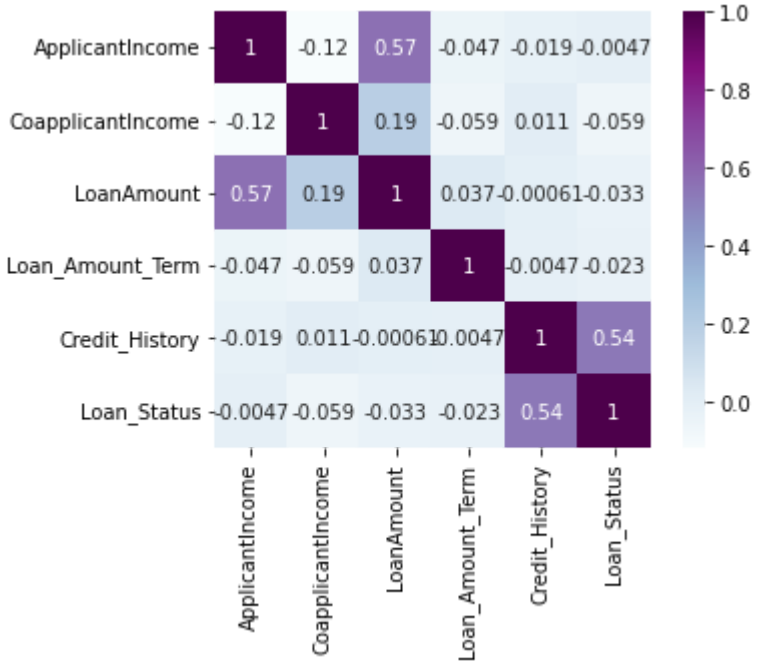
Credit_History



In [10]:
```
train_copy["Dependents"].replace('3+',3,inplace=True)
train_copy["Loan_Status"].replace('Y',1,inplace=True)
train_copy["Loan_Status"].replace('N',0,inplace=True)
```

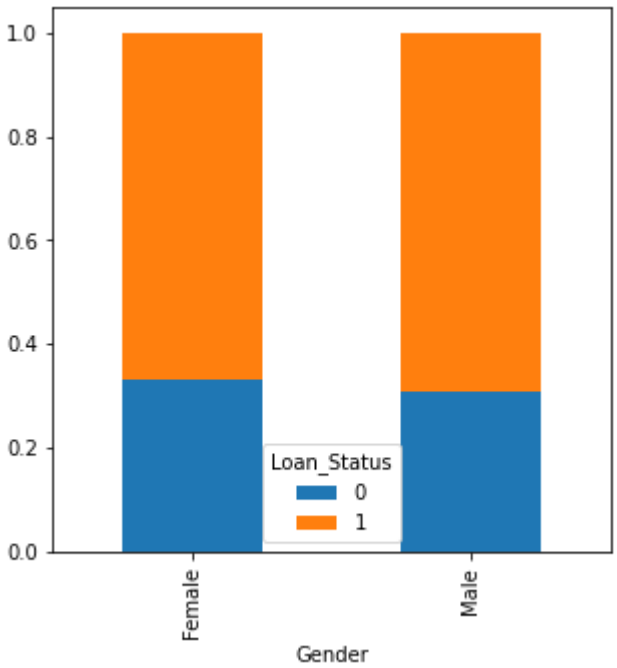
In [11]:
```
matrix=train_copy.corr()
sns.heatmap(matrix,vmax=1,square=True,cmap="BuPu",annot=True)
```

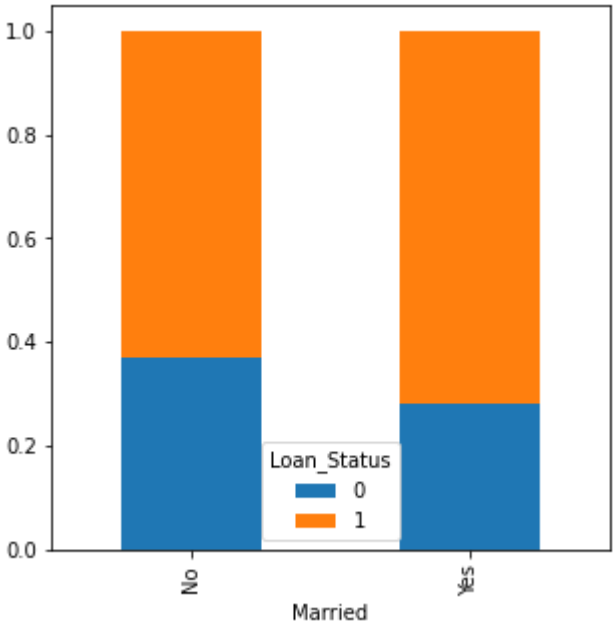
Out[11]:

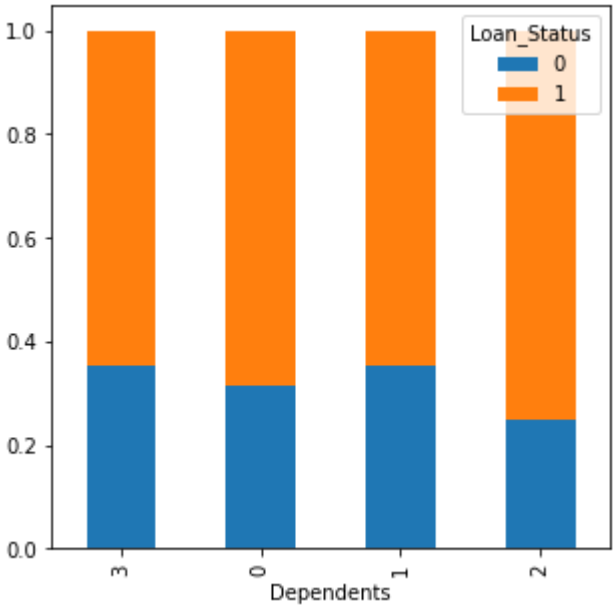


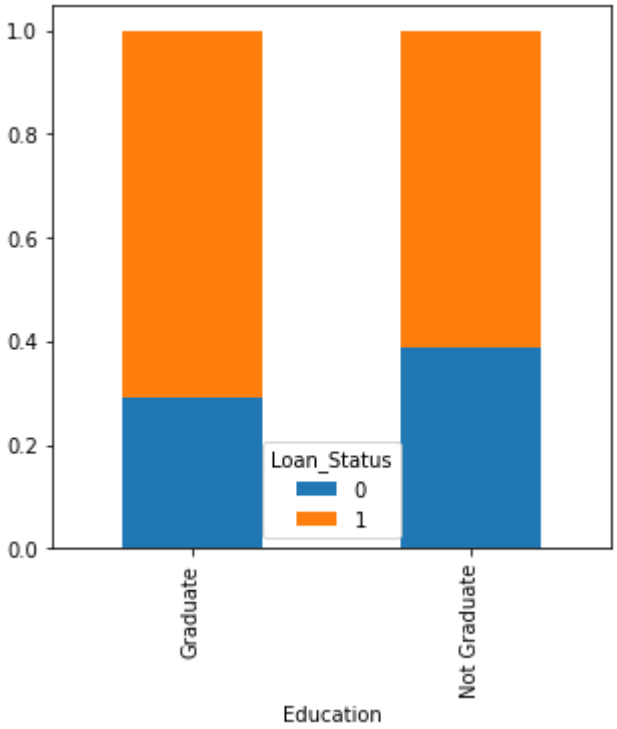
In [14]:

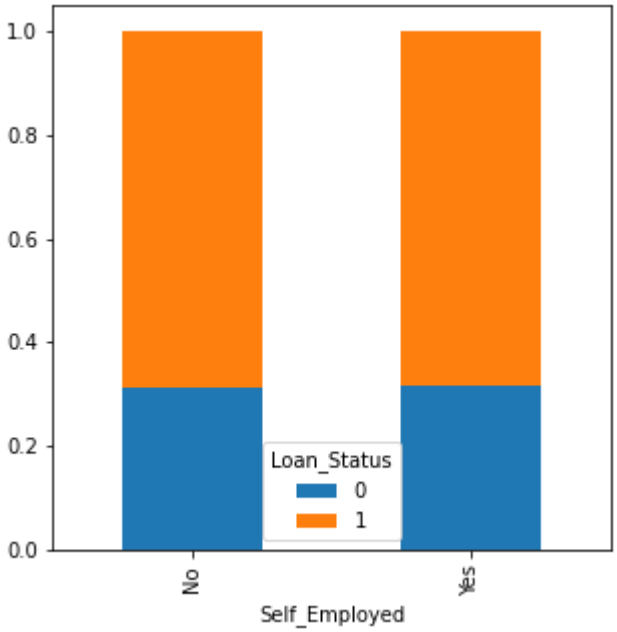
```
for i in train_copy.columns:
  if i=="Loan_Status" or i=="Loan_ID":
    continue
  if i=="Credit_History":
    ob=pd.crosstab(train_copy[i],train_copy["Loan_Status"])
    ob.div(ob.sum(1).astype(float), axis=0).plot(kind="bar",stacked=True,figsize=(5,5))
    plt.show
  if train_copy[i].dtype=="object":
    ob=pd.crosstab(train_copy[i],train_copy["Loan_Status"])
    ob.div(ob.sum(1).astype(float), axis=0).plot(kind="bar",stacked=True,figsize=(5,5))
    plt.show
```

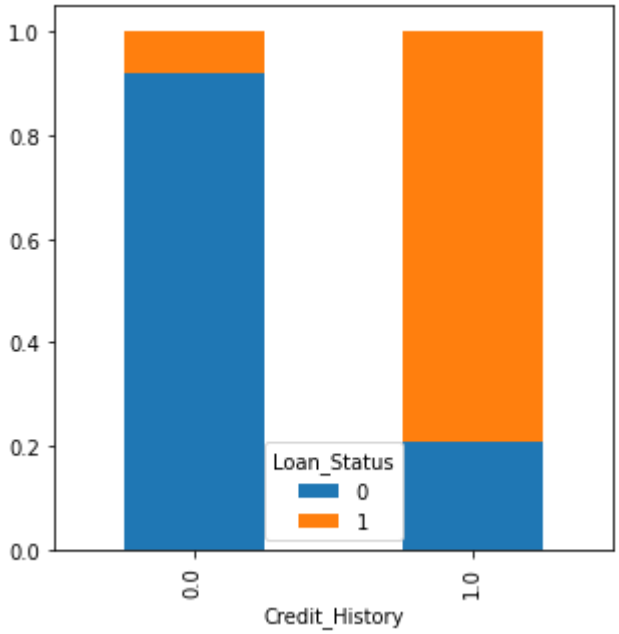


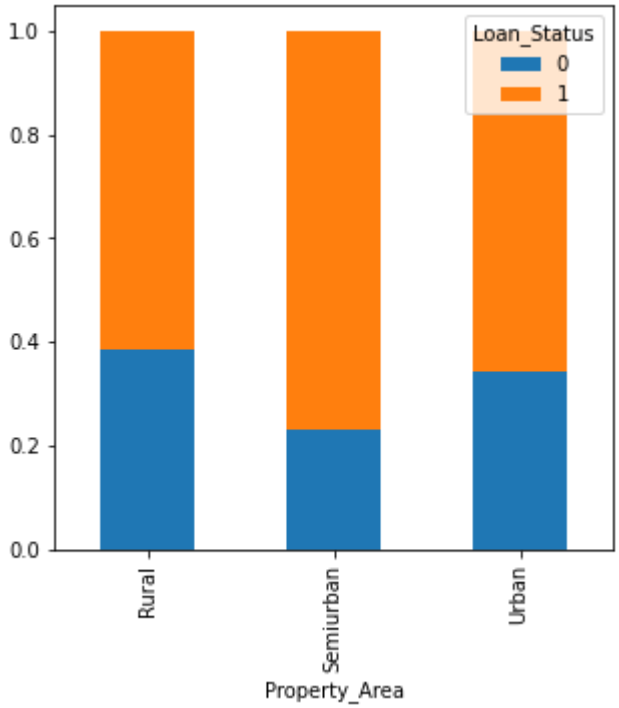

1.0
0.8
0.6
0.4
0.2
0.0
Loan_Status
0
1
No
Yes
Married


1.0
0.8
0.6
0.4
0.2
0.0
Loan_Status
0
1
3
0
1
2
Dependents


1.0
0.8
0.6
0.4
0.2
0.0
Loan_Status
0
1
Graduate
Not Graduate
Education

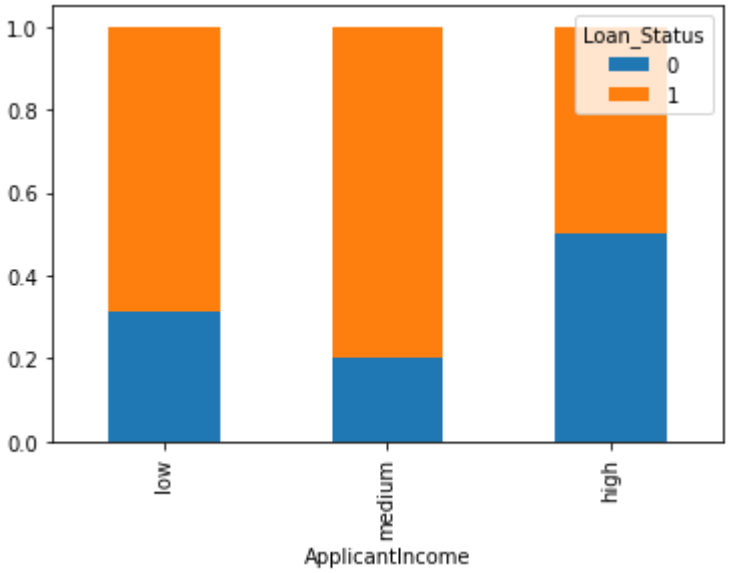

1.0
0.8
0.6
0.4
0.2
0.0
Loan_Status
0
1
No
Yes
Self_Employed





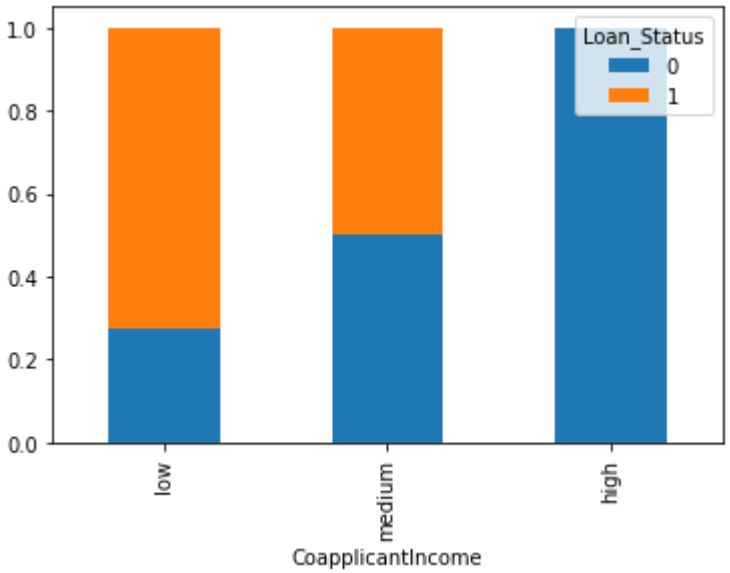
In [12]:

```
range=[0,30000,60000,90000]
label=["low","medium","high"]
income=pd.cut(train_copy["ApplicantIncome"],range,labels=label)
income_bin=pd.crosstab(income,train_copy["Loan_Status"])
income_bin.div(income_bin.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True)
plt.show()
```



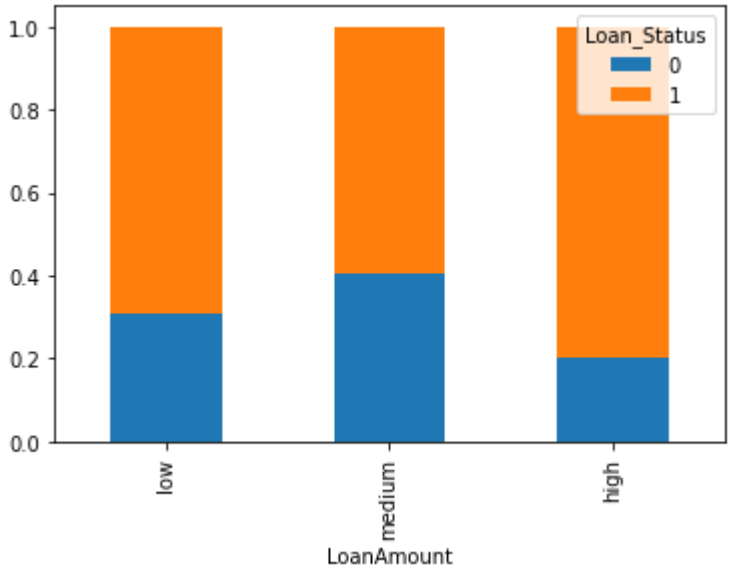
In [13]:

```
range=[0,15000,25000,45000]
label=["low","medium","high",]
income=pd.cut(train_copy["CoapplicantIncome"],range,labels=label)
income_bin=pd.crosstab(income,train_copy["Loan_Status"])
income_bin.div(income_bin.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True)
plt.show()
```



In [15]:

```
range=[0,250,500,750]
label=["low","medium","high",]
income=pd.cut(train_copy["LoanAmount"],range,labels=label)
income_bin=pd.crosstab(income,train_copy["Loan_Status"])
income_bin.div(income_bin.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True)
plt.show()
```



In [16]:
```
test_copy=test_data.copy()
```

In [17]:
```
test_copy.isnull().sum()
```

Out[17]:
```
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 [1...
```
test_copy["Gender"].fillna(test_copy["Gender"].mode()[0],inplace=True)
test_copy["Married"].fillna(test_copy["Married"].mode()[0],inplace=True)
test_copy["Dependents"].fillna(test_copy["Dependents"].mode()[0],inplace=True)
test_copy["Self_Employed"].fillna(test_copy["Self_Employed"].mode()[0],inplace=True)
test_copy["Credit_History"].fillna(test_copy["Credit_History"].mode()[0],inplace=True)
test_copy["Loan_Amount_Term"].fillna(test_copy["Loan_Amount_Term"].mode()[0],inplace=Tru
test_copy["LoanAmount"].fillna(test_copy["LoanAmount"].median(), inplace=True)
```

In [19]:
```
test_copy.isnull().sum()
```

Out[19]:
```
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 [20]:
```
test_copy["Dependents"].replace('3+',3,inplace=True)
```

In [21]:
```
test_copy=test_copy.drop("Loan_ID",axis=1)
train_copy=train_copy.drop("Loan_ID",axis=1)
```

In [22]:
```
x = train_copy.drop("Loan_Status",axis=1)
y = train_copy["Loan_Status"]
```

In [23]:
```
x=pd.get_dummies(x)
train_copy1=pd.get_dummies(train_copy)
test_copy1=pd.get_dummies(test_copy)
x.head()
```

...

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

In [24]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2)
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
LRM=LogisticRegression()
LRM.fit(xtrain,ytrain)
```

Out[25]:
```
LogisticRegression()
```

In [26]:
```
pred=LRM.predict(xtest)
accuracy_score(ytest,pred)
```

Out[26]:
```
0.8292682926829268
```

In [27]:
```
test_pred=LRM.predict(test_copy1)
result=test_copy.copy()
result["Loan_Status"]=test_pred
result["Loan_ID"]=test_data["Loan_ID"]
result["Loan_Status"].replace(1,'Y',inplace=True)
result["Loan_Status"].replace(0,'N',inplace=True)
result[["Loan_ID","Loan_Status"]]
```

Out[27]:

	Loan_ID	Loan_Status
0	LP001015	Y

	Loan_ID	Loan_Status
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y
...	...	...
362	LP002971	Y
363	LP002975	Y
364	LP002980	Y
365	LP002986	Y
366	LP002989	Y

367 rows × 2 columns

In [2...
```
train_copy1["TOtal_Income"]=train_copy1["ApplicantIncome"]+train_copy1["CoapplicantIncom
test_copy1["TOtal_Income"]=test_copy1["ApplicantIncome"]+test_copy1["CoapplicantIncome"]
train_copy1["EMI"]=train_copy1["LoanAmount"]/train_copy1["Loan_Amount_Term"]
test_copy1["EMI"]=test_copy1["LoanAmount"]/test_copy1["Loan_Amount_Term"]
```

In [29]:
```
x=train_copy1.drop("Loan_Status",axis=1)
y=train_copy1["Loan_Status"]
x.head()
```

...

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

5 rows × 22 columns

In [30]:
```
from sklearn.model_selection import StratifiedKFold
```

In [54]:
```
i=1
mean = 0
kf = StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
for train_index,test_index in kf.split(x,y):
    print ('\n{} of kfold {} '.format(i,kf.n_splits))
    xtr,xvl = x.loc[train_index],x.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    LRM = LogisticRegression(random_state=1)
    LRM.fit(xtr,ytr)
```

```
            pred_test=LRM.predict(xvl)
            score=accuracy_score(yvl,pred_test)
            mean=mean+ score

            print ('accuracy score is ',score)
            i=i+1
            if i>kf.n_splits:
              pred_test=LRM.predict(test_copy1)
        print ('\nMean Validation Accuracy',mean/(i-1),'\n')
        result=test_copy.copy()
        result["Loan_Status"]=pred_test
        result["Loan_ID"]=test_data["Loan_ID"]
        result["Loan_Status"].replace(1,'Y',inplace=True)
        result["Loan_Status"].replace(0,'N',inplace=True)
        result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7954545454545454

2 of kfold 7
accuracy score is  0.8068181818181818

3 of kfold 7
accuracy score is  0.8181818181818182

4 of kfold 7
accuracy score is  0.8068181818181818

5 of kfold 7
accuracy score is  0.7954545454545454

6 of kfold 7
accuracy score is  0.8390804597701149

7 of kfold 7
accuracy score is  0.7816091954022989

Mean Validation Accuracy 0.8062024182713838
```

Out[54]:

	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

In [58]:

```
        from sklearn import tree
        i=1
        mean=0
        kf=StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
        for train_index,test_index in kf.split(x,y):
          print('\n{} of kfold {} '.format(i,kf.n_splits))
          xtr,xvl=x.loc[train_index],x.loc[test_index]
          ytr,yvl=y.loc[train_index],y.loc[test_index]
          DT=tree.DecisionTreeClassifier(random_state=1)
```

```
        DT.fit(xtr,ytr)
        pred_test=DT.predict(xvl)
        score=accuracy_score(yvl,pred_test)
        mean=mean+score
        print('accuracy score is ',score)
        i=i+1
        if i>kf.n_splits:
          pred_test=LRM.predict(test_copy1)
      print('\nMean Validation Accuracy ',mean/(i-1),'\n')
      result=test_copy.copy()
      result["Loan_Status"]=pred_test
      result["Loan_ID"]=test_data["Loan_ID"]
      result["Loan_Status"].replace(1,'Y',inplace=True)
      result["Loan_Status"].replace(0,'N',inplace=True)
      result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7159090909090909

2 of kfold 7
accuracy score is  0.7386363636363636

3 of kfold 7
accuracy score is  0.7386363636363636

4 of kfold 7
accuracy score is  0.6931818181818182

5 of kfold 7
accuracy score is  0.6818181818181818

6 of kfold 7
accuracy score is  0.735632183908046

7 of kfold 7
accuracy score is  0.6896551724137931

Mean Validation Accuracy  0.7133527392148081
```

Out[58]:

	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

In [62]:

```
from sklearn.ensemble import RandomForestClassifier
i=1
mean=0
kf=StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
for train_index,test_index in kf.split(x,y):
  print('\n{} of kfold {} '.format(i,kf.n_splits))
  xtr,xvl=x.loc[train_index],x.loc[test_index]
  ytr,yvl=y.loc[train_index],y.loc[test_index]
  RF=RandomForestClassifier(random_state=1,max_depth=7)
```

```
        RF.fit(xtr,ytr)
        pred_test=RF.predict(xvl)
        score=accuracy_score(yvl,pred_test)
        mean=mean+score
        print('accuracy score is ',score)
        i=i+1
        if i>kf.n_splits:
          pred_test=LRM.predict(test_copy1)
      print('\nMean Validation Accuracy ',mean/(i-1),'\n')
      result=test_copy.copy()
      result["Loan_Status"]=pred_test
      result["Loan_ID"]=test_data["Loan_ID"]
      result["Loan_Status"].replace(1,'Y',inplace=True)
      result["Loan_Status"].replace(0,'N',inplace=True)
      result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7840909090909091

2 of kfold 7
accuracy score is  0.8522727272727273

3 of kfold 7
accuracy score is  0.8295454545454546

4 of kfold 7
accuracy score is  0.7954545454545454

5 of kfold 7
accuracy score is  0.7613636363636364

6 of kfold 7
accuracy score is  0.8275862068965517

7 of kfold 7
accuracy score is  0.7816091954022989

Mean Validation Accuracy  0.8045603821465891
```

Out[62]:

	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

In [72]:

```
from sklearn.model_selection import GridSearchCV,train_test_split
l=[1,2,3,4,5,6,7,8,9]
l1=[10,20,30,40,50,60,70,80,90]
paramgrid = {'max_depth': l, 'n_estimators': l1}
grid_search=GridSearchCV(RandomForestClassifier(random_state=1),paramgrid)
xt,xv,yt,yv=train_test_split(x,y,test_size=0.2,random_state=1)
grid_search.fit(xt,yt)
```

```
GridSearchCV(estimator=RandomForestClassifier(random_state=1),
```

Out[72]:
```
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                         'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90]})
```

In [73]:
```
grid_search.best_estimator_
```

Out[73]:
```
RandomForestClassifier(max_depth=4, n_estimators=70, random_state=1)
```

In [75]:
```
from sklearn.ensemble import RandomForestClassifier
i=1
mean=0
kf=StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
for train_index,test_index in kf.split(x,y):
  print('\n{} of kfold {} '.format(i,kf.n_splits))
  xtr,xvl=x.loc[train_index],x.loc[test_index]
  ytr,yvl=y.loc[train_index],y.loc[test_index]
  RF=RandomForestClassifier(random_state=1,max_depth=4,n_estimators=70)
  RF.fit(xtr,ytr)
  pred_test=RF.predict(xvl)
  score=accuracy_score(yvl,pred_test)
  mean=mean+score
  print('accuracy score is ',score)
  i=i+1
  if i>kf.n_splits:
    pred_test=LRM.predict(test_copy1)
print('\nMean Validation Accuracy ',mean/(i-1),'\n')
result=test_copy.copy()
result["Loan_Status"]=pred_test
result["Loan_ID"]=test_data["Loan_ID"]
result["Loan_Status"].replace(1,'Y',inplace=True)
result["Loan_Status"].replace(0,'N',inplace=True)
result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7954545454545454

2 of kfold 7
accuracy score is  0.8522727272727273

3 of kfold 7
accuracy score is  0.8295454545454546

4 of kfold 7
accuracy score is  0.7954545454545454

5 of kfold 7
accuracy score is  0.7840909090909091

6 of kfold 7
accuracy score is  0.8390804597701149

7 of kfold 7
accuracy score is  0.7816091954022989

Mean Validation Accuracy  0.8110725481415136
```

Out[75]:

	Loan_ID	Loan_Status
0	LP001015	Y

	Loan_ID	Loan_Status
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

In [78]:
```
from xgboost import XGBClassifier
i=1
mean=0
kf=StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
for train_index,test_index in kf.split(x,y):
  print('\n{} of kfold {} '.format(i,kf.n_splits))
  xtr,xvl=x.loc[train_index],x.loc[test_index]
  ytr,yvl=y.loc[train_index],y.loc[test_index]
  XGB=XGBClassifier(random_state=1,max_depth=4,n_estimators=70)
  XGB.fit(xtr,ytr)
  pred_test=XGB.predict(xvl)
  score=accuracy_score(yvl,pred_test)
  mean=mean+score
  print('accuracy score is ',score)
  i=i+1
  if i>kf.n_splits:
    pred_test=LRM.predict(test_copy1)
print('\nMean Validation Accuracy ',mean/(i-1),'\n')
result=test_copy.copy()
result["Loan_Status"]=pred_test
result["Loan_ID"]=test_data["Loan_ID"]
result["Loan_Status"].replace(1,'Y',inplace=True)
result["Loan_Status"].replace(0,'N',inplace=True)
result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7954545454545454

2 of kfold 7
accuracy score is  0.8295454545454546

3 of kfold 7
accuracy score is  0.7954545454545454

4 of kfold 7
accuracy score is  0.7954545454545454

5 of kfold 7
accuracy score is  0.75

6 of kfold 7
accuracy score is  0.8390804597701149

7 of kfold 7
accuracy score is  0.7701149425287356

Mean Validation Accuracy  0.7964434990297059
```

Out[78]:

	Loan_ID	Loan_Status

	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y

In [79]:
```
l=[1,2,3,4,5,6,7,8,9]
l1=[10,20,30,40,50,60,70,80,90]
paramgrid = {'max_depth': l, 'n_estimators': l1}
grid_search=GridSearchCV(XGBClassifier(random_state=1),paramgrid)
xt,xv,yt,yv=train_test_split(x,y,test_size=0.2,random_state=1)
grid_search.fit(xt,yt)
```

Out[79]:
```
GridSearchCV(estimator=XGBClassifier(random_state=1),
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9],
                         'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90]})
```

In [80]:
```
grid_search.best_estimator_
```

Out[80]:
```
XGBClassifier(n_estimators=20, random_state=1)
```

In [81]:
```
from xgboost import XGBClassifier
i=1
mean=0
kf=StratifiedKFold(n_splits=7,random_state=1,shuffle=True)
for train_index,test_index in kf.split(x,y):
  print('\n{} of kfold {} '.format(i,kf.n_splits))
  xtr,xvl=x.loc[train_index],x.loc[test_index]
  ytr,yvl=y.loc[train_index],y.loc[test_index]
  XGB=XGBClassifier(random_state=1,n_estimators=20)
  XGB.fit(xtr,ytr)
  pred_test=XGB.predict(xvl)
  score=accuracy_score(yvl,pred_test)
  mean=mean+score
  print('accuracy score is ',score)
  i=i+1
  if i>kf.n_splits:
    pred_test=LRM.predict(test_copy1)
print('\nMean Validation Accuracy ',mean/(i-1),'\n')
result=test_copy.copy()
result["Loan_Status"]=pred_test
result["Loan_ID"]=test_data["Loan_ID"]
result["Loan_Status"].replace(1,'Y',inplace=True)
result["Loan_Status"].replace(0,'N',inplace=True)
result[["Loan_ID","Loan_Status"]].head()
```

```
1 of kfold 7
accuracy score is  0.7954545454545454

2 of kfold 7
accuracy score is  0.8409090909090909

3 of kfold 7
```

```
accuracy score is  0.8181818181818182

4 of kfold 7
accuracy score is  0.8068181818181818

5 of kfold 7
accuracy score is  0.7840909090909091

6 of kfold 7
accuracy score is  0.8275862068965517

7 of kfold 7
accuracy score is  0.7816091954022989

Mean Validation Accuracy  0.8078071353933423
```

Out[81]:

	Loan_ID	Loan_Status
0	LP001015	Y
1	LP001022	Y
2	LP001031	Y
3	LP001035	Y
4	LP001051	Y