**Python Code**

#Initializing import statements  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import accuracy\_score  
from sklearn.model\_selection import StratifiedKFold  
from sklearn import metrics  
import matplotlib  
matplotlib.use('TkAgg')  
import matplotlib.pyplot as plt  
import seaborn as sns  
import warnings  
warnings.filterwarnings("ignore")  
  
#Reading in data  
train=pd.read\_csv("train\_u6lujuX\_CVtuZ9i.csv")  
test=pd.read\_csv("test\_Y3wMUE5\_7gLdaTN.csv")  
  
#Making copies of the files  
train\_original=train.copy()  
test\_original=test.copy()  
  
#Understanding the data  
print("\nTraining file columns:")  
print(train.columns)  
print("\nTesting file columns:")  
print(test.columns)  
  
#Understanding the data types  
print("\nTraining file data types:")  
print(train.dtypes)  
print("\nTesting file data types:")  
print(test.dtypes)  
  
#Understanding shape(dimensions) of the data  
print("\nTraining file dimensions:")  
print(train.shape)  
print("\nTesting file dimensions:")  
print(test.shape)  
  
#Target Variable (Loan Status) proportions  
print("\nLoan Status Frequency:")  
print(train['Loan\_Status'].value\_counts(normalize=False))  
  
#Target Variable (Loan Status) proportions  
print("\nLoan Status Proportion:")  
print(train['Loan\_Status'].value\_counts(normalize=True))  
  
#Plotting Loan Status Frequency and Proportion  
plt.figure(1)  
plt.subplot(121)  
train['Loan\_Status'].value\_counts().plot.bar(title = 'Loan Status Frequency')  
plt.subplot(122)  
train['Loan\_Status'].value\_counts(normalize=True).plot.bar(title = 'Loan Status Proportion')  
  
  
#Plotting Gender, Marital Status, Employment Status and Credit History Proportions  
plt.figure(1)  
plt.subplot(221)  
train['Gender'].value\_counts(normalize=True).plot.bar(title= 'Gender')  
plt.subplot(222)  
train['Married'].value\_counts(normalize=True).plot.bar(title= 'Married')  
plt.subplot(223)  
train['Self\_Employed'].value\_counts(normalize=True).plot.bar(title= 'Self Employed')  
plt.subplot(224)  
train['Credit\_History'].value\_counts(normalize=True).plot.bar(title= 'Credit History')  
  
#Plotting Dependents, Education and Property Area  
plt.figure(1)  
plt.subplot(131)  
train['Dependents'].value\_counts(normalize=True).plot.bar(figsize=(24,24), title= 'Dependents')  
plt.subplot(132)  
train['Education'].value\_counts(normalize=True).plot.bar(title= 'Education', fontsize = 6)  
plt.subplot(133)  
train['Property\_Area'].value\_counts(normalize=True).plot.bar(title= 'Property Area', fontsize = 9)  
  
#Plotting Applicant Income  
plt.figure(1)  
plt.subplot(121)  
sns.distplot(train['ApplicantIncome']);  
plt.subplot(122)  
train['ApplicantIncome'].plot.box()  
  
#Plotting Applicant Income by Education Level  
#train.boxplot(column = 'ApplicantIncome', by = 'Education')  
  
  
#Plotting Coapplicant Income  
plt.figure(1)  
plt.subplot(121)  
sns.distplot(train['CoapplicantIncome']);  
plt.subplot(122)  
train['CoapplicantIncome'].plot.box(figsize=(16,5))  
  
  
#Plotting Loan Amount  
plt.figure(1)  
plt.subplot(121)  
df = train.dropna() #Removes midding values  
sns.distplot(df['LoanAmount']);  
plt.subplot(122)  
train['LoanAmount'].plot.box(figsize=(16,5))  
  
#Plotting Gender vs Loan Status  
Gender=pd.crosstab(train['Gender'],train['Loan\_Status'])  
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 8)  
  
  
#Plotting Marital Status vs Loan Status  
Married=pd.crosstab(train['Married'],train['Loan\_Status'])  
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(8,8))  
  
  
#Plotting Dependents vs Loan Status  
Dependents=pd.crosstab(train['Dependents'],train['Loan\_Status'])  
Dependents.div(Dependents.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True)  
  
  
#Plotting Education vs Loan Status  
Education=pd.crosstab(train['Education'],train['Loan\_Status'])  
Education.div(Education.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(8,8), fontsize = 8)  
  
  
#Plotting Self Employment vs Loan Status  
Self\_Employed=pd.crosstab(train['Self\_Employed'],train['Loan\_Status'])  
Self\_Employed.div(Self\_Employed.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(8,8))  
  
#Plotting Credit History vs Loan Status  
Credit\_History=pd.crosstab(train['Credit\_History'],train['Loan\_Status'])  
Credit\_History.div(Credit\_History.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(8,8))  
  
  
#Plotting Property Area vs Loan Status  
Property\_Area=pd.crosstab(train['Property\_Area'],train['Loan\_Status'])  
Property\_Area.div(Property\_Area.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 8)  
  
#Plotting Applicant Income vs Loan Status  
bins=[0,2500,4000,6000,81000]  
group=['Low','Average','High', 'Very high']  
train['Income\_bin']=pd.cut(df['ApplicantIncome'],bins,labels=group)  
Income\_bin=pd.crosstab(train['Income\_bin'],train['Loan\_Status'])  
Income\_bin.div(Income\_bin.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 6, figsize = (8,8))  
plt.xlabel('ApplicantIncome')  
plt.ylabel('Percentage')  
  
#Plotting Co-applicant Income vs Loan Status  
bins=[0,1000,3000,42000]  
group=['Low','Average','High']  
train['Coapplicant\_Income\_bin']=pd.cut(df['CoapplicantIncome'],bins,labels=group)  
Coapplicant\_Income\_bin=pd.crosstab(train['Coapplicant\_Income\_bin'],train['Loan\_Status'])  
Coapplicant\_Income\_bin.div(Coapplicant\_Income\_bin.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 6, figsize = (8,8))  
plt.xlabel('CoapplicantIncome')  
plt.ylabel('Percentage')  
  
#Plotting Total Income vs Loan Status  
train['Total\_Income']=train['ApplicantIncome']+train['CoapplicantIncome']  
bins=[0,2500,4000,6000,81000]  
group=['Low','Average','High', 'Very high']  
train['Total\_Income\_bin']=pd.cut(train['Total\_Income'],bins,labels=group)  
Total\_Income\_bin=pd.crosstab(train['Total\_Income\_bin'],train['Loan\_Status'])  
Total\_Income\_bin.div(Total\_Income\_bin.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 6, figsize = (8,8))  
plt.xlabel('Total\_Income')  
plt.ylabel('Percentage')  
  
#Plotting Loan Amount vs Loan Status  
bins=[0,100,200,700]  
group=['Low','Average','High']  
train['LoanAmount\_bin']=pd.cut(df['LoanAmount'],bins,labels=group)  
LoanAmount\_bin=pd.crosstab(train['LoanAmount\_bin'],train['Loan\_Status'])  
LoanAmount\_bin.div(LoanAmount\_bin.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, fontsize = 6, figsize = (8,8))  
plt.xlabel('LoanAmount')  
plt.ylabel('Percentage')  
  
  
#Plotting corelation between data fields and Loan Status  
train=train.drop(['Income\_bin', 'Coapplicant\_Income\_bin', 'LoanAmount\_bin', 'Total\_Income\_bin', 'Total\_Income'], axis=1)  
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)  
matrix = train.corr()  
f, ax = plt.subplots(figsize=(9, 12))  
sns.set(font\_scale=0.5)  
sns.heatmap(matrix, vmax=.8, square=True, cmap="BuPu")  
  
#Printing missing values from training dataset  
print('\n')  
print(train.isnull().sum())  
  
#Filling in missing values of categorical variables with mode (training dataset)  
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)  
train['Loan\_Amount\_Term'].fillna(train['Loan\_Amount\_Term'].mode()[0], inplace=True)  
  
#Filling in numerical variable with median  
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)  
  
#Filling in missing values of all variables  
test['Gender'].fillna(train['Gender'].mode()[0], inplace=True)  
test['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)  
test['Self\_Employed'].fillna(train['Self\_Employed'].mode()[0], inplace=True)  
test['Credit\_History'].fillna(train['Credit\_History'].mode()[0], inplace=True)  
test['Loan\_Amount\_Term'].fillna(train['Loan\_Amount\_Term'].mode()[0], inplace=True)  
test['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)  
  
#Treating outliers/making the distribution more normal  
train['LoanAmount\_log'] = np.log(train['LoanAmount'])  
train['LoanAmount\_log'].hist(bins=20)  
test['LoanAmount\_log'] = np.log(test['LoanAmount'])  
  
#Model Making  
train=train.drop('Loan\_ID',axis=1)  
test=test.drop('Loan\_ID',axis=1)  
X = train.drop('Loan\_Status',1)  
y = train.Loan\_Status  
X=pd.get\_dummies(X)  
train=pd.get\_dummies(train)  
test=pd.get\_dummies(test)  
x\_train, x\_cv, y\_train, y\_cv = train\_test\_split(X,y, test\_size =0.3)  
model = LogisticRegression()  
model.fit(x\_train, y\_train)  
pred\_cv = model.predict(x\_cv)  
print("Accuracy Score for training data: ")  
print(accuracy\_score(y\_cv, pred\_cv))  
pred\_test = model.predict(test)  
submission=pd.read\_csv("Sample\_Submission\_ZAuTl8O\_FK3zQHh.csv")  
submission['Loan\_Status']=pred\_test  
submission['Loan\_ID']=test\_original['Loan\_ID']  
submission['Loan\_Status'].replace(0, 'N',inplace=True)  
submission['Loan\_Status'].replace(1, 'Y',inplace=True)  
pd.DataFrame(submission, columns=['Loan\_ID','Loan\_Status']).to\_csv('to\_submit.csv')  
  
#Using Stratified K-Fold Cross Validation to Improve Model  
i = 1  
kf = StratifiedKFold(n\_splits=5, 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]  
  
 model = LogisticRegression(random\_state=1)  
 model.fit(xtr, ytr)  
 pred\_test = model.predict(xvl)  
 score = accuracy\_score(yvl, pred\_test)  
 print('accuracy\_score', score)  
 i += 1  
pred\_test = model.predict(test)  
pred = model.predict\_proba(xvl)[:, 1]  
  
#Visualizing ROC curve  
fpr, tpr, \_ = metrics.roc\_curve(yvl, pred)  
auc = metrics.roc\_auc\_score(yvl, pred)  
plt.figure(figsize=(12,8))  
plt.plot(fpr,tpr,label="validation, auc="+str(auc))  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.legend(loc=4)  
  
#Adding varibles I think might affect Loan Status  
train['Total\_Income']=train['ApplicantIncome']+train['CoapplicantIncome']  
test['Total\_Income']=test['ApplicantIncome']+test['CoapplicantIncome']  
train['Total\_Income\_log'] = np.log(train['Total\_Income'])  
test['Total\_Income\_log'] = np.log(test['Total\_Income'])  
train['EMI']=train['LoanAmount']/train['Loan\_Amount\_Term']  
test['EMI']=test['LoanAmount']/test['Loan\_Amount\_Term']  
train['Balance Income']=train['Total\_Income']-(train['EMI']\*1000)  
test['Balance Income']=test['Total\_Income']-(test['EMI']\*1000)  
sns.distplot(train['Total\_Income\_log']);  
sns.distplot(train['EMI']);  
sns.distplot(train['Balance Income']);  
  
#Removing Irrelevant Variables(used to make the new ones)  
train=train.drop(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan\_Amount\_Term'], axis=1)  
test=test.drop(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan\_Amount\_Term'], axis=1)