**Loan Application Status Prediction**



**Problem Statement**

In this every need money to survive. Money isn’t everything but everything needs money. To earn money investment is must. Here comes the loan. People apply for in banks and loan distribution is core part of every banks business. Nobody can deny this fact that the main asset of bank is loan because from loan they earn a lot, however risk is major part of the loan. So, every banking company or farm always wants that their money should go in safe hands. That is why in today’s world every company have their own process of verification of that person and validation, still there is not 100% guarantee that the customer will repay the loan amount or not. Here machine learning comes into picture, ML is much capable to identify that whether the loan can be given to applicant or not. In machine learning, machine learn each and every aspect of the data and then tells whether the loan can be given or not but in other hand in banks sometimes some situation comes when bank have to give loan to applicant of single strong factor, which is not possible because the frequency of these cases will be one in thousands or more applications.

Machine always helps the peoples it reduces the risk as well. Prediction of loan approval is very helpful for bank employees. The purpose of this blog is do provide good, fast and effective way to choose deserving candidate who applied for the loan. This model will save time and efforts of bank employee. Here first model will lean the data and then when test data will be provided to the model, it’ll predict and tell whether loan should be given to applicant or not.

The dataset includes details of applicants who have applied for loan. The dataset includes details like credit history, loan amount, their income, dependents etc.

*Independent Variables:*

- Loan\_ID

- Gender

- Married

- Dependents

- Education

- Self\_Employed

- ApplicantIncome

- CoapplicantIncome

- Loan\_Amount

- Loan\_Amount\_Term

- Credit History

- Property\_Area

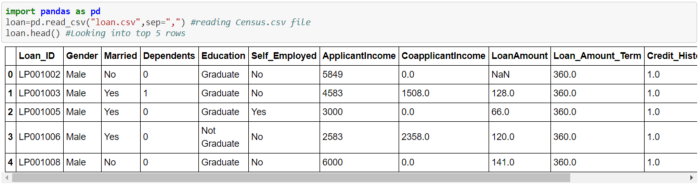
*Dependent Variable (Target Variable):*

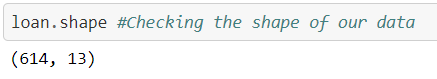
-Loan\_Status

The purpose of this blog is to build a model that can predict whether the loan of the applicant will be approved or not on the basis of the details provided in the dataset.

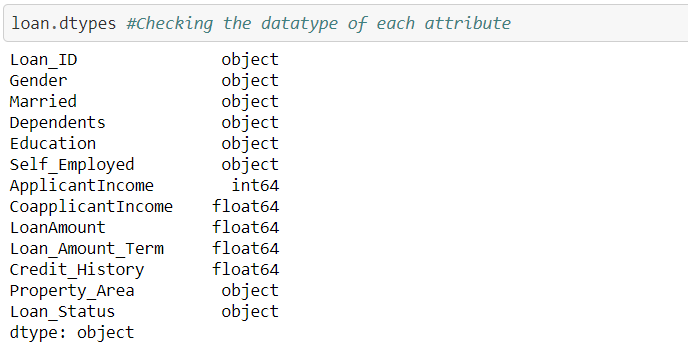
**Data Analysis**

Data Analysis is a procedure for gathering raw data than converting it into useful and informative data that will help for making decisions clear by the user. Data will be collect, analyzed to answer the questions.



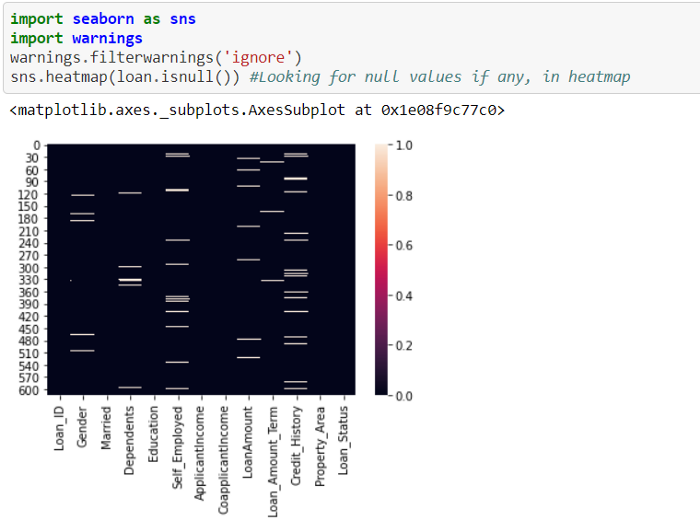


614 rows and 13 columns are available in this dataset to predict the Loan status.



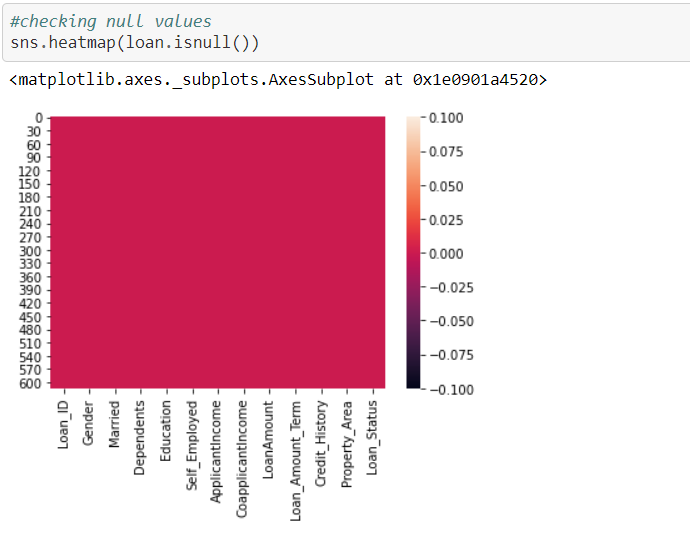
Target variable i.e. Loan\_Status is object so Classification will be used to learn model.

**Exploratory Data Analysis**



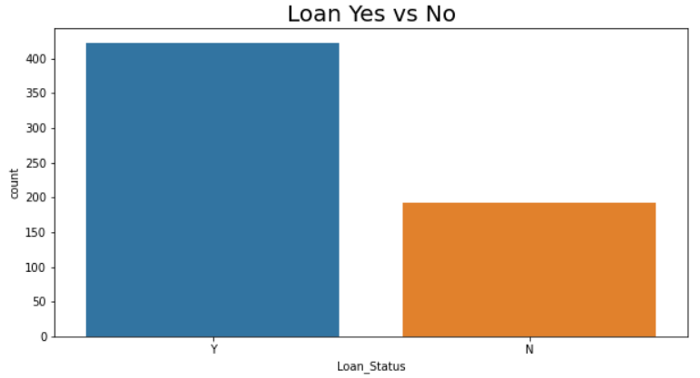
It’s clearly visible that there are too many null values present in the dataset.

*#Replacing null values of object column with mode of it.*  
**import** **numpy** **as** **np**  
collist=loan.columns.values  
**for** i **in** range(0,len(collist)):  
 **if** loan[collist[i]].dtype == "object":  
 loan[collist[i]].fillna(loan[collist[i]].mode()[0], inplace=**True**)*#Replacing non object values i.e. int64 and float64 null values with mean of it.*  
**import** **numpy** **as** **np**  
collist=loan.columns.values  
**for** i **in** range(0,len(collist)):  
 **if** loan[collist[i]].dtype != "object":  
 loan[collist[i]].fillna(loan[collist[i]].mean(), inplace=**True**)



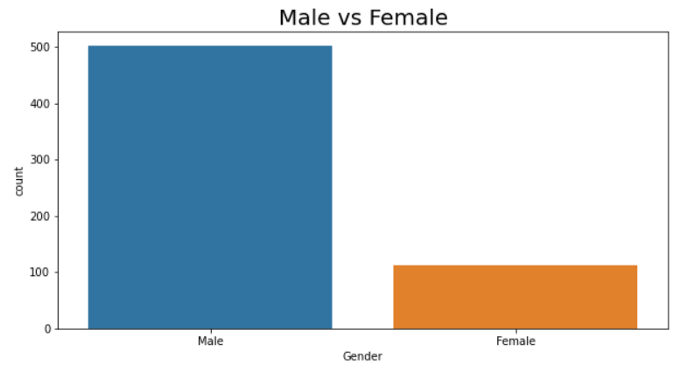
All the null values are removed from the datasets.

**import** **matplotlib.pyplot** **as** **plt**  
plt.figure(figsize = (10,5))  
sns.countplot(x="Loan\_Status", data=loan)  
plt.title("Loan Yes vs No", fontsize = 20)  
plt.show()



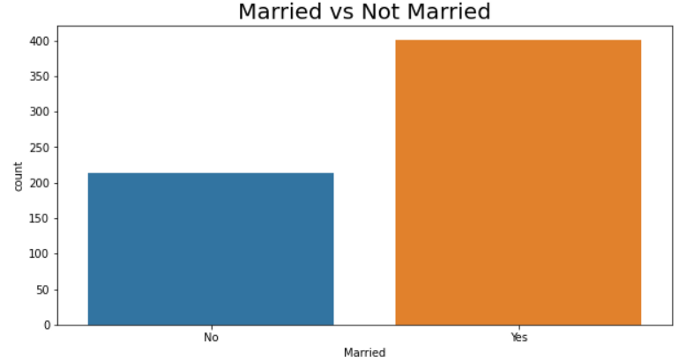
There are 422 Yes(loan approved) and 192 No (Not approved) values present.

**import** **matplotlib.pyplot** **as** **plt**   
plt.figure(figsize = (10,5))   
sns.countplot(x="Gender", data=loan)   
plt.title("Male vs Female", fontsize = 20)   
plt.show()



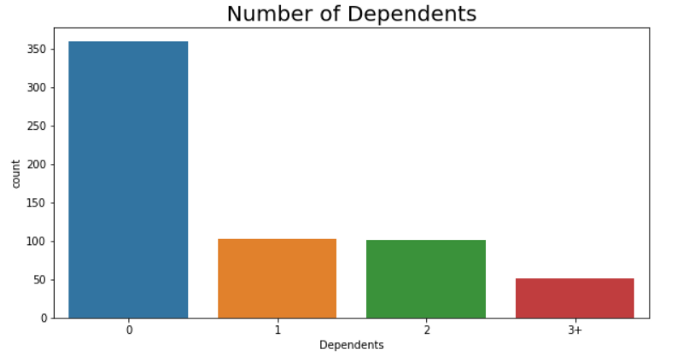
Here Males applicants are much more than Female

**import** **matplotlib.pyplot** **as** **plt**   
plt.figure(figsize = (10,5))   
sns.countplot(x=”Married”, data=loan)   
plt.title(“Married vs Not Married”, fontsize = 20)   
plt.show()



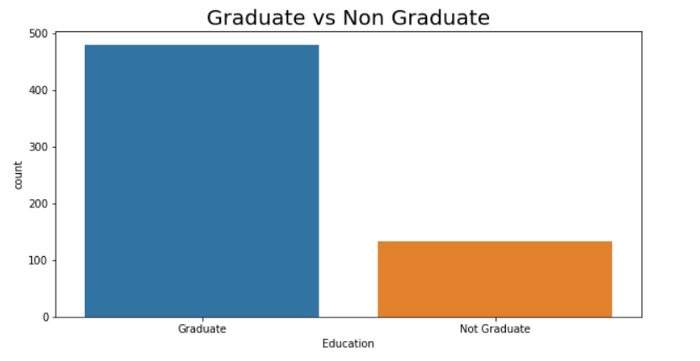
It’s clearly visible that Married people applied for loan most.

**import** **matplotlib.pyplot** **as** **plt**  
plt.figure(figsize = (10,5))  
sns.countplot(x="Dependents", data=loan)  
plt.title("Number of Dependents", fontsize = 20)  
plt.show()



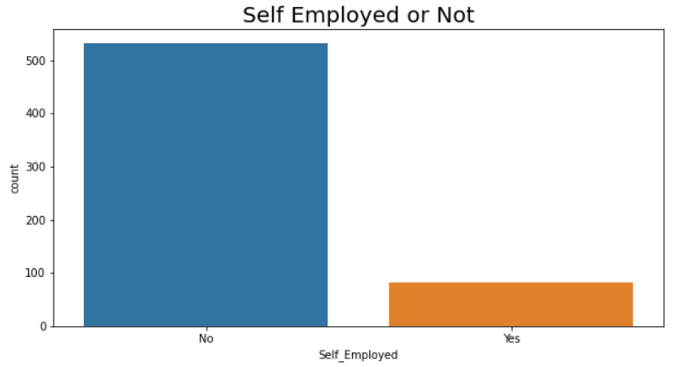
The person who does not have any Dependent, those person applied for loan most.

**import** **matplotlib.pyplot** **as** **plt**  
plt.figure(figsize = (10,5))  
sns.countplot(x="Education", data=loan)  
plt.title("Graduate vs Non Graduate", fontsize = 20)  
plt.show()



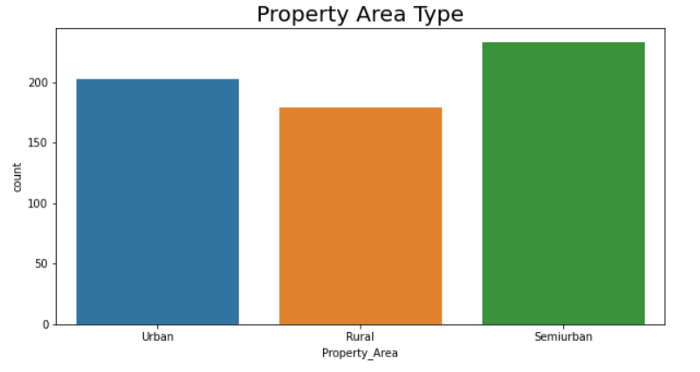
Graduate People applied for loan most.

**import** **matplotlib.pyplot** **as** **plt**  
plt.figure(figsize = (10,5))  
sns.countplot(x="Self\_Employed", data=loan)  
plt.title("Self Employed or Not", fontsize = 20)  
plt.show()



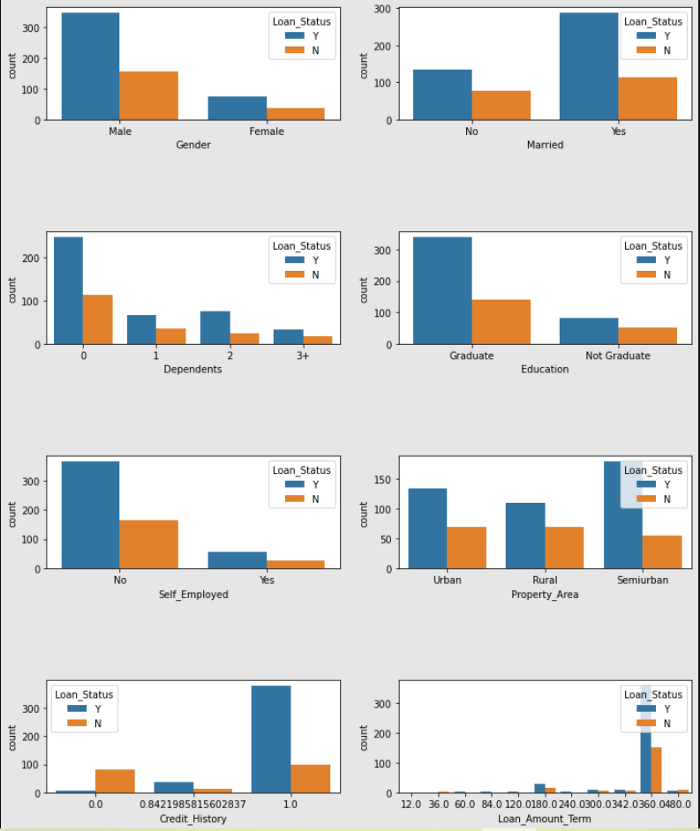
Those persons who are not self employed, apply for loan most

**import** **matplotlib.pyplot** **as** **plt**  
plt.figure(figsize = (10,5))  
sns.countplot(x="Property\_Area", data=loan)  
plt.title("Property Area Type", fontsize = 20)  
plt.show()



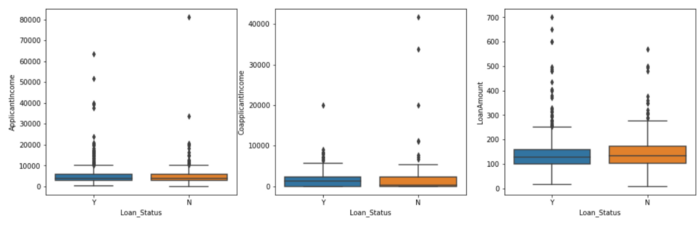
Semi Urban people applied for Loan most

label\_list=['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area','Credit\_History','Loan\_Amount\_Term']  
fig,axes = plt.subplots(4,2,figsize=(12,15))  
**for** i,cat **in** enumerate(label\_list):  
 row,col = i//2,i%2 *#getting size of plots in row and cols*   
 sns.countplot(x=cat,data=loan,hue='Loan\_Status',ax=axes[row,col]) *#Plotting count plot with hue Loan Status*  
plt.subplots\_adjust(hspace=1) *# Plotting the graphs*



Loan Approval Status: Near 70% of Loan applications got accepted. Sex: Men applied for loan way more than Female and got approved more than female.   
Martial Status: 70% of the people are Married, Their loan loans also got approved more.   
Dependents: Many number of people have no dependent and their loan also got approved.   
Education: More than 80% of the people are Graduate and graduates have higher proportion of loan approval.   
Employment: 80% of population is not self employed.   
Property Area: Semi-urban have applied for loan more and their loan applications got approved respectively.   
Credit History: The person who have credit credit cards those peoples loan is approved more.   
Loan Amount Term: The person who took for 360 months i.e. 30 years, those loan approved most.

*label\_list = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']  
fig,axes = plt.subplots(1,3,figsize=(17,5))*#size of plot***for****i,cat\_col****in****enumerate(label\_list):  
sns.boxplot(y=cat\_col,data=loan,x='Loan\_Status',ax=axes[i])  
plt.subplots\_adjust(hspace=1)*#plotting the graph



In Numerical columns, there is no significant relation with Loan approval status.

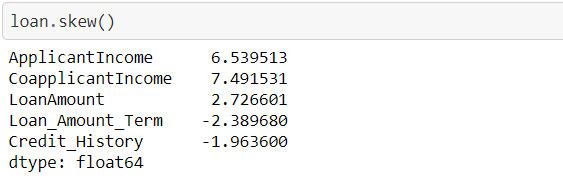
Applicant Income, Co applicant Income and Loan Amount for all 3 things amount doesn’t matter. For 3 things Loan status was approved and rejected almost same.

*#Dropping Loan\_ID*   
loan.drop('Loan\_ID',axis=1,inplace=**True**)

*Dropping Loan\_ID because it’s unique in each row, If it’ll drop, it’ll not reflect to prediction.*

**Pre-Processing Pipeline**

**Removing Skewness**



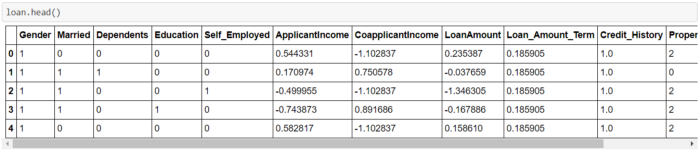
Skewness of these columns are too high, let’s reduce the skewness using PowerTransformer

**from** **sklearn.preprocessing** **import** PowerTransformer  
PT=PowerTransformer()  
**for** i **in** loan.columns:  
 **if** loan[i].dtype != "object":  
 **if** ((len(pd.unique(loan[i]))) > 3):  
 **if** abs(loan.loc[:,i].skew())>0.55:  
 loan.loc[:,i]=PT.fit\_transform(loan.loc[:,i].values.reshape(-1,1))

**Label Encoding**

Let’s perform label encoding to convert object type columns into numeric type

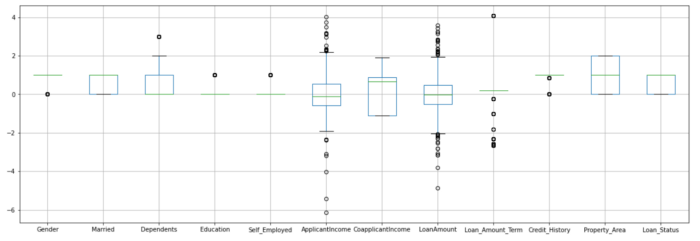
label\_list=list(loan.select\_dtypes(['object']).columns)  
**from** **sklearn.preprocessing** **import** LabelEncoder  
le=LabelEncoder()  
**for** i **in** label\_list:  
 loan[i] = le.fit\_transform(loan[i])



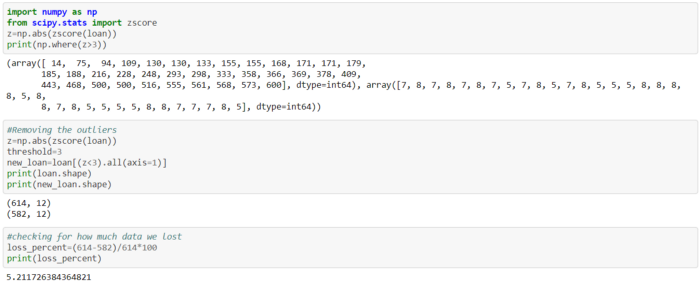
Now all type of columns are numeric.

**Removing Outliers**

**import** **matplotlib.pyplot** **as** **plt**  
**import** **seaborn** **as** **sns**  
loan.boxplot(figsize=[20,8])  
plt.subplots\_adjust(bottom=0.25)  
plt.show()



Too many outliers present in the dataset. Let’s remove it.



5 percent of data is lost, which is not bad, So let’s continue

**Building Machine Learning Models**

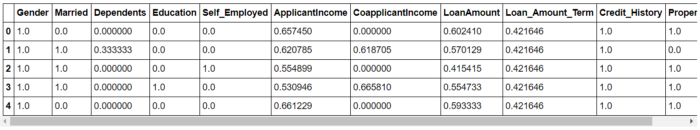
**Separating Input and Output Variables**

x = loan.drop("Loan\_Status", axis=1)  
y = loan["Loan\_Status"]

**Scaling**

Scaling is required because there is too much difference in minimum and maximum value of columns.

**from** **sklearn.preprocessing** **import** StandardScaler,MinMaxScaler  
scaler=MinMaxScaler() *#Initializting MinMaxScaler*  
scale\_x=scaler.fit\_transform(x) *#fitting our data into MinMaxScaller*  
scaled\_x = pd.DataFrame(scale\_x, index=x.index, columns=x.columns)  
x=scaled\_x  
x.head() *#Priting top 5 rows of our data*



**Finding Best Random State**

***from******sklearn.model\_selection******import****train\_test\_split****from******sklearn.linear\_model******import****LogisticRegression****from******sklearn.metrics******import****accuracy\_score****from******sklearn.metrics******import****confusion\_matrix,classification\_report,auc****import******warnings*** *warnings.filterwarnings('ignore')  
maxAccu=0  
maxRS=0****for****i****in****range(1400,1450):  
x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.30,random\_state=i)  
LR = LogisticRegression()  
LR.fit(x\_train,y\_train)  
predrf = LR.predict(x\_test)  
acc = accuracy\_score(y\_test, predrf)****if****acc>maxAccu:  
maxAccu=acc  
maxRS=i  
print("Best accuracy is",maxAccu," on Random\_state ",maxRS)*

#Output  
Best accuracy is 0.8810810810810811 on Random\_state 1424

**Train Test Split**

Splitting train and test data, 70% data will be train and 30% data will be test

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.30,random\_state=maxRS)

**Finding Best Algorithm**

*#importing all the required libraries to find best Algorithm*  
**from** **sklearn.naive\_bayes** **import** BernoulliNB  
**from** **sklearn.svm** **import** SVC  
**from** **sklearn.tree** **import** DecisionTreeClassifier  
**from** **sklearn.neighbors** **import** KNeighborsClassifier  
**from** **sklearn.ensemble** **import** RandomForestClassifier  
**from** **sklearn.model\_selection** **import** cross\_val\_score  
model=[LogisticRegression(),KNeighborsClassifier(),BernoulliNB(),SVC(),DecisionTreeClassifier(),RandomForestClassifier()]  
**for** m **in** model:  
 m.fit(x\_train,y\_train)  
 pred=m.predict(x\_test)  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 print("|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||")  
 print('accuracy score of ->', m)  
 print(accuracy\_score(y\_test,pred))  
 print(confusion\_matrix(y\_test,pred))  
 print(classification\_report(y\_test,pred))  
 score=cross\_val\_score(m,x,y,cv=5)  
 print(score)  
 print(score.mean())  
 print("Difference between Accuracy score and cross validation score is - ",accuracy\_score(y\_test,pred)-score.mean())  
 print("|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||")  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

OUTPUT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> LogisticRegression()  
0.8810810810810811  
[[ 28 22]  
 [ 0 135]]  
 precision recall f1-score support  
  
 0 1.00 0.56 0.72 50  
 1 0.86 1.00 0.92 135  
  
 accuracy 0.88 185  
 macro avg 0.93 0.78 0.82 185  
weighted avg 0.90 0.88 0.87 185  
  
[0.81300813 0.77235772 0.7804878 0.85365854 0.81967213]  
0.8078368652538984  
Difference between Accuracy score and cross validation score is - 0.07324421582718266  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> KNeighborsClassifier()  
0.8486486486486486  
[[ 29 21]  
 [ 7 128]]  
 precision recall f1-score support  
  
 0 0.81 0.58 0.67 50  
 1 0.86 0.95 0.90 135  
  
 accuracy 0.85 185  
 macro avg 0.83 0.76 0.79 185  
weighted avg 0.84 0.85 0.84 185  
  
[0.80487805 0.75609756 0.7804878 0.81300813 0.75409836]  
0.7817139810742371  
Difference between Accuracy score and cross validation score is - 0.06693466757441158  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> BernoulliNB()  
0.8810810810810811  
[[ 28 22]  
 [ 0 135]]  
 precision recall f1-score support  
  
 0 1.00 0.56 0.72 50  
 1 0.86 1.00 0.92 135  
  
 accuracy 0.88 185  
 macro avg 0.93 0.78 0.82 185  
weighted avg 0.90 0.88 0.87 185  
  
[0.81300813 0.7804878 0.7804878 0.85365854 0.81967213]  
0.809462881514061  
Difference between Accuracy score and cross validation score is - 0.07161819956702009  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> SVC()  
0.8810810810810811  
[[ 28 22]  
 [ 0 135]]  
 precision recall f1-score support  
  
 0 1.00 0.56 0.72 50  
 1 0.86 1.00 0.92 135  
  
 accuracy 0.88 185  
 macro avg 0.93 0.78 0.82 185  
weighted avg 0.90 0.88 0.87 185  
  
[0.81300813 0.7804878 0.7804878 0.85365854 0.81967213]  
0.809462881514061  
Difference between Accuracy score and cross validation score is - 0.07161819956702009  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> DecisionTreeClassifier()  
0.7081081081081081  
[[35 15]  
 [39 96]]  
 precision recall f1-score support  
  
 0 0.47 0.70 0.56 50  
 1 0.86 0.71 0.78 135  
  
 accuracy 0.71 185  
 macro avg 0.67 0.71 0.67 185  
weighted avg 0.76 0.71 0.72 185  
  
[0.67479675 0.67479675 0.76422764 0.73170732 0.74590164]  
0.718286018925763  
Difference between Accuracy score and cross validation score is - -0.010177910817654956  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
accuracy score of -> RandomForestClassifier()  
0.827027027027027  
[[ 30 20]  
 [ 12 123]]  
 precision recall f1-score support  
  
 0 0.71 0.60 0.65 50  
 1 0.86 0.91 0.88 135  
  
 accuracy 0.83 185  
 macro avg 0.79 0.76 0.77 185  
weighted avg 0.82 0.83 0.82 185  
  
[0.79674797 0.74796748 0.76422764 0.81300813 0.80327869]  
0.7850459816073571  
Difference between Accuracy score and cross validation score is - 0.04198104541966996  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

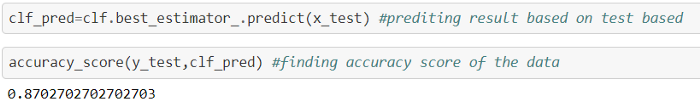
Random Forest Classifier have highest Accuracy more than 80 % and the difference between Cross Validation Score and Accuracy score it less. So Random Forest Classifier will be used here to learn model.

**Hyper Parameter Tuning**

Performing hyper parameter tuning to get good and more accurate result from the model

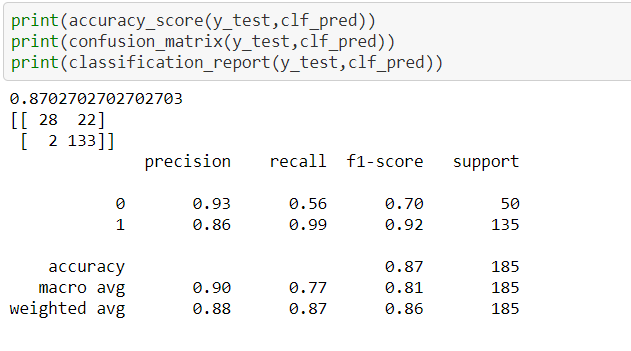
***from******sklearn.model\_selection******import****RandomizedSearchCV  
  
parameters = {"max\_depth":[1,2,3,4,5,6,7,8,9,10,15,20],  
"max\_features": [3,5,7,9],  
"min\_samples\_leaf":[2,3,4,5,6]}  
  
clf = RandomizedSearchCV(RandomForestClassifier(), parameters)  
clf.fit(x\_train,y\_train)*#fitting train and test data *clf.best\_params\_*#Best parameters

#Output  
{'min\_samples\_leaf': 5, 'max\_features': 9, 'max\_depth': 3}



Model learnt almost 87%

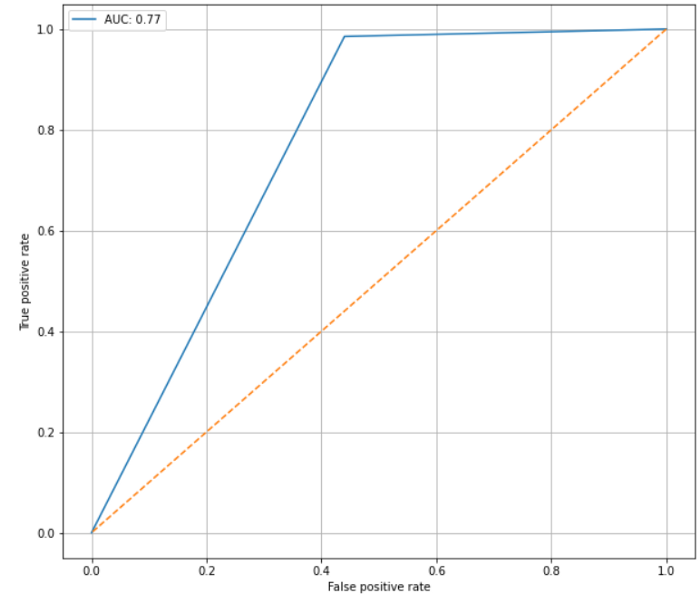
Let’s look into classification matrix of model



**AUC ROC curve**

Plotting AUC ROC curve to see the false positive rate and true positive rate

***from******sklearn.metrics******import****roc\_curve,auc****import******matplotlib.pyplot******as******plt*** *fpr,tpr,thresholds=roc\_curve(y\_test,clf\_pred)*# calculating fpr, tpr *rf\_auc = auc(fpr, tpr)*#Model Accuracy *plt.figure(figsize=(10,9))*#plotting the figure, size of 10\*9 *plt.plot(fpr, tpr, label = 'AUC:****%0.2f****' % rf\_auc)  
plt.plot([1,0],[1,0], linestyle = '--')  
plt.legend(loc=0)*#adding accuracy score at bottom right *plt.xlabel('False positive rate')  
plt.ylabel('True positive rate')  
plt.grid()*#adding the grid

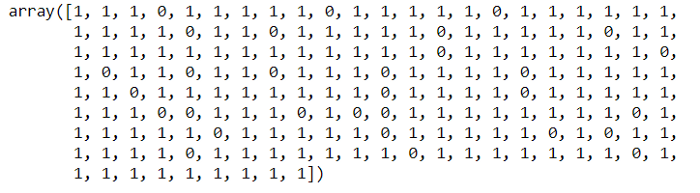


Loan status Predicted vs Actual

**Saving the model**

Model is saved for future use

**import** **joblib**  
joblib.dump(clf.best\_estimator\_,"Loan.obj")  
SVR\_from\_joblib=joblib.load("Loan.obj")  
Predicted = SVR\_from\_joblib.predict(x\_test)  
Predicted



Predicted output

**Concluding Remarks**

In this case study, a Machine Learning model is developed to predict, whether loan should be approved or not. Here several features were mined from the dataset and combined together with the help of Machine Leaning, to do the loan application status prediction. With the help of the above techniques, proposed model is able to predict the status of the load with an accuracy score of 87%. However, there is still ways to do improvement in this model.

In the future, our model can be predict the status of the loan more accurately.