***The performance and key insights on the data***

***The problem:***

*The Property Finance bank deals in all Property loans. They have presence across all urban, semi urban and rural areas. Customer first apply for Property loan after that bank validates the customer eligibility for loan based on property and applicant-based metrics*

*The bank wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, previous Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.*

*It’s a classification problem , given information about the application we have to predict whether the they’ll be to pay the loan or not.*

*We’ll start by exploratory data analysis , then preprocessing , and finally we’ll be testing different models such as Logistic regression, decision tree, RF and XGBoost*

*The data consists of the following rows:*

***Loan\_ID : Unique Loan ID***

***Gender : Male/ Female***

***Married : Applicant married (Y/N)***

***Dependents : Number of dependents***

***Education : Applicant Education (Graduate/ Under Graduate)***

***Self\_Employed : Self employed (Y/N)***

***ApplicantIncome : Applicant income***

***CoapplicantIncome : Coapplicant income***

***LoanAmount : Loan amount in thousands of dollars***

***Loan\_Amount\_Term : Term of loan in months***

***Credit\_History : credit history meets guidelines yes or no***

***Property\_Area : Urban/ Semi Urban/ Rural***

***Loan\_Status : Loan approved (Y/N) this is the target variable***

***Exploratory data analysis:***

*We’ll be using seaborn for visualisation and pandas for data manipulation.*

*Mount the google drive to colab to link the data set.*

*from google.colab import drive*

*drive.mount('/content/drive')*

*We’ll import the necessary libraries and load the data :*

*import pandas as pd*

*import numpy as np*

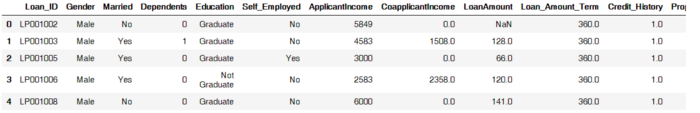
*import matplotlib.pyplot as plt*

*import seaborn as sns*

*import plotly.express as px*

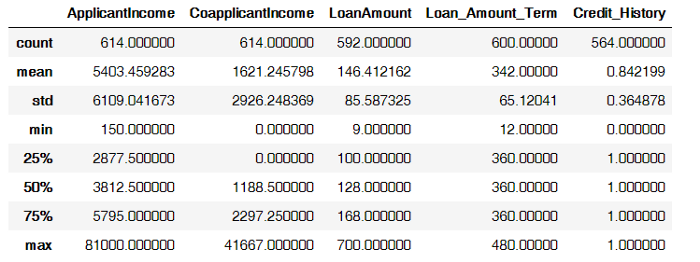
***df=pd.read\_csv("/content/drive/MyDrive/Loan Status Prediction/train.csv")***

*We can look at few top rows using the head function*

*df.head()*

*We can see that there’s some missing data , we can further explore this using the pandas describe function:*

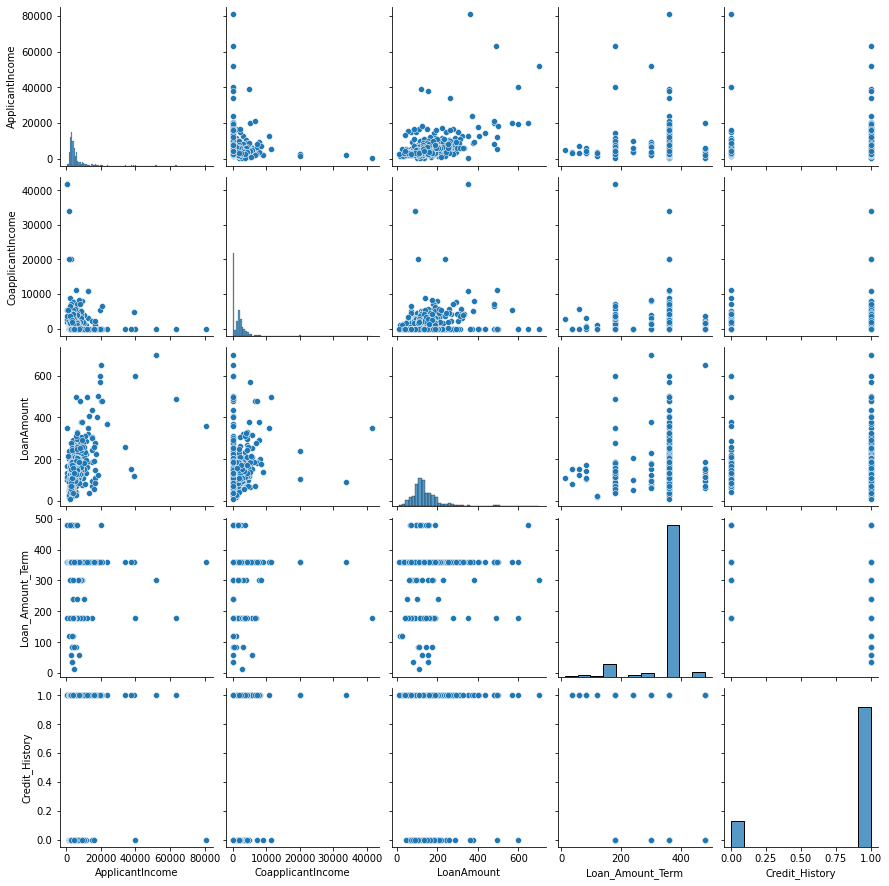
*df.describe()*

**

*Some variables have missing values that we’ll have to deal with , and also there seems to be some outliers for the Applicant Income , Coapplicant income and Loan Amount . We also see that about 84% applicants have a credit\_history. Because the mean of Credit\_History field is 0.84 and it has either (1 for having a credit history or 0 for not)*

*It would be interesting to study the distribution of the numerical variables mainly the Applicant income and the loan amount. To do this we’ll use seaborn for visualization.*

*sns.pairplot(df)*

**

*cat = df.select\_dtypes('object').columns.to\_list()*

*for i in cat[1:-1]:*

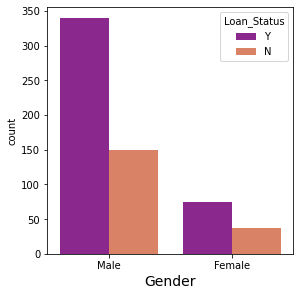
*plt.figure(figsize=(15,10))*

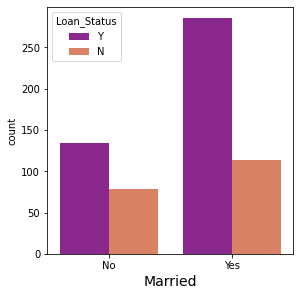
*plt.subplot(2,3,1)*

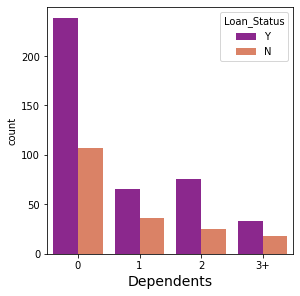
*sns.countplot(x=i ,hue='Loan\_Status', data=df ,palette='plasma')*

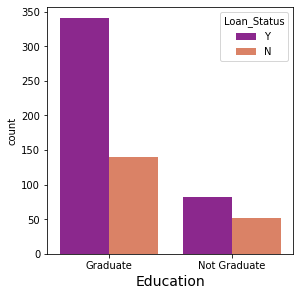
*plt.xlabel(i, fontsize=14)*

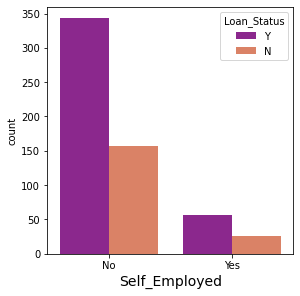
*To plot the Categorical (split by loan status)*

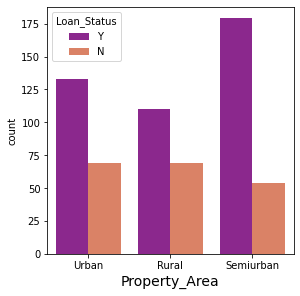
**

**

**

**

**

**

*To check the corelation and plot the heat map*

*df.corr()*

|  | ***ApplicantIncome*** | ***CoapplicantIncome*** | ***LoanAmount*** | ***Loan\_Amount\_Term*** | ***Credit\_History*** |
| --- | --- | --- | --- | --- | --- |
| ***ApplicantIncome*** | *1.000000* | *-0.116605* | *0.570909* | *-0.045306* | *-0.014715* |
| ***CoapplicantIncome*** | *-0.116605* | *1.000000* | *0.188619* | *-0.059878* | *-0.002056* |
| ***LoanAmount*** | *0.570909* | *0.188619* | *1.000000* | *0.039447* | *-0.008433* |
| ***Loan\_Amount\_Term*** | *-0.045306* | *-0.059878* | *0.039447* | *1.000000* | *0.001470* |
| ***Credit\_History*** | *-0.014715* | *-0.002056* | *-0.008433* | *0.001470* | *1.000000* |

**

*Since Loan Amount has missing values , we can’t plot it directly. One solution is to fill the missing values rows then plot it, we can do this using the below functions. For gender and married comes under categorical data so we used mode to fill missing values*

*To check the null values*

***df.isnull().sum()***

*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*

*For gender and married comes under categorical data so we used mode to fill missing values*

*To fil l the missing values in Gender columns.*

*df["Gender"].mode()*

*df["Gender"]=df["Gender"].fillna("Male")*

*To fil l the missing values in Married columns*

*df["Married"].mode()*

*df["Married"]=df["Married"].fillna("Yes")*

*To fil l the missing values in Dependent columns*

***df["Dependents"].mode()***

*df["Dependents"]=df["Dependents"].fillna("0")*

*To fil l the missing values in Dependent columns*

*df["Self\_Employed"].mode()*

*df["Self\_Employed"]=df["Self\_Employed"].fillna("No")*

*To fill the LoanAmount it comes under median,*

*the Loan amount term and Credit History comes under mean*

*df["LoanAmount"].describe(),df["LoanAmount"].median()*

*(count 592.000000*

*mean 146.412162*

*std 85.587325*

*min 9.000000*

*25% 100.000000*

*50% 128.000000*

*75% 168.000000*

*max 700.000000*

*Name: LoanAmount, dtype: float64, 128.0)*

*df["LoanAmount"]=df["LoanAmount"].fillna(df["LoanAmount"].median())*

*df["Loan\_Amount\_Term"]=df["Loan\_Amount\_Term"].fillna(df["Loan\_Amount\_Term"].mean())*

*df["Credit\_History"]=df["Credit\_History"].fillna(df["Credit\_History"].mean())*

*Now we drop the loan\_ID and Loan\_status*

*X=df.drop(columns=["Loan\_ID","Loan\_Status"],axis=1)*

*y=df["Loan\_Status"]*

## *Encoding data to numeric*

*converting categorical values to numbers*

*X["Dependents"]=X["Dependents"].map({"0":1,"1":1,"2":2,"3+":3})*

*X["Property\_Area"]=X["Property\_Area"].map({"Urban": 3, "Semiurban": 2,"Rural": 1})*

*y=y.map({"Y":1,"N":0})*

*Creating dummy coloumn for train and test*

*df\_dum=pd.get\_dummies(X,columns=["Gender","Married","Education","Self\_Employed","Dependents","Property\_Area"])*

*df\_dum.head()*

*X\_new=df\_dum*

*Now we do the test train the model*

*from sklearn.model\_selection import train\_test\_split*

*sum(y)/len(y)*

*0.6872964169381107*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_new, y, test\_size = 0.3, random\_state = 0)*

*sum(y\_train)/len(y\_train) , sum(y\_test)/len(y\_test)*

*(0.6713286713286714, 0.7243243243243244)*

# *Machine learning models*

*First of all we will divide our dataset into two variables X as the features we defined earlier and y as the Loan\_Status the target value we want to predict.*

## *Models we will use:*

* *****Decision Tree*****
* *****Random Forest*****
* *****XGBoost*****
* *****Logistic Regression*****

## *The Process of Modeling the Data:*

*Importing the model*

*Fitting the model*

*Predicting Loan Status*

*Classification report by Loan Status*

*Overall accuracy*

## *Logistic Regression[¶](https://www.kaggle.com/code/yonatanrabinovich/loan-prediction-dataset-ml-project" \l "Logistic-Regression" \t "https://www.kaggleusercontent.com/kf/46856503/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..aUc4dNK5Xv6Wu5dmcSjJ2A.7Esty61JDV_Apg7h3x5MmQvL8kspLWnYJRCpldDm946Lw1L8i2l4wnvXcovQ5stpcL8VUt2pq52aZXJvdZhNihJ5bo4PIxkis9Xw_NmiC3v0CkYhMh_Qjd9iA0ekJ-FmXXBjehqGnTFUC1AJT4JDTs0fRu6CDKzFNaTd-IS3-LTJx8G_4LszfWc2TpI66MLBBFcJggv45Y0o7ukSTuroJ2MQMFfbOpTKnQcvLoMdz9YQP-gczEjN7DwERYJPP_lXRImJftTMrJujvKUvCfKDSSjPesT20k39Pw_GQiimbud3Ib_2lN81WVnrV38jn3qITr-jL48qlA8yPElGXgiQBjCdujTs1BUF_wPOO0xx7tlqiasSQX55q_IycW68Z4MKJy7ymUwCoL3eFA44zxYDtDbtCmeiwY31KZAzE6AtBXyuDluXGeDxizUZtzdx4xbHOICTF84SeiREQ6-ZH6A10BuduN1INLdrfA1hxcYbJEosl9yCDLn8T3dLqHsZynkcB4HpJlpDkbIUncWUG7jxUl6HYZimcsL5xF9aWUjQkB_WsrDkyjQUCUtBHQ5SkOzth62xTS4CA52pFG8lhCBnTzUC8rMpTezN4Nd7XyasxyrPj4WKq2dm22fNTtZXmczZxDuGXMVI6B_7IrwtKstgFlu9nusEMwzAW_y8Wuk_Cqwt6ihXOUGqAt9ApMwlZLQv.e3S5LCbPoNH4z-ZkElzomg/_self)*

*Now, I will explore the Logistic Regression model.*

|  |
| --- |
| *IMG_256* |

*Now import necessary libraries*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.metrics import classification\_report, accuracy\_score,plot\_confusion\_matrix*

*LR = LogisticRegression()*

*LR.fit(X\_train, y\_train)*

*y\_predict = LR.predict(X\_test)*

*#  prediction Summary by species*

*print(classification\_report(y\_test, y\_predict))*

*# Accuracy score*

*LR\_SC = accuracy\_score(y\_predict,y\_test)*

*print('accuracy is',accuracy\_score(y\_predict,y\_test))*

*precision recall f1-score support*

*0 0.92 0.43 0.59 51*

*1 0.82 0.99 0.89 134*

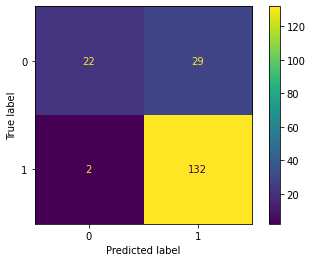
*Accuracy 0.83 185*

*macro avg 0.87 0.71 0.74 185*

*weighted avg 0.85 0.83 0.81 185*

***Accuracy is 0.8324324324324325***

*Plot the confusion matrix*

*)*

***LR.score(X\_test,y\_test)***

***0.8324324324324325***

***Conclusion:***

1. *Credit\_History is a very important variable because of its high correlation with Loan\_Status\_prediction therefor showind high Dependancy for the latter.*
2. *The Logistic Regression algorithm is the most accurate:******approximately 83%******.*