Importing Libraries and Dataset

Firstly, we must import libraries:

Pandas – To load the Data frame.

Matplotlib – To visualize the data features i.e., barplot.

Seaborn – To see the correlation between features using heatmap.

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**df=pd.read\_csv("ML Project 1 Dataset.csv")**

⦁Once we imported the dataset, let’s view it using the below command.

**df.head()**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

APP\_ID CIBIL\_SCORE\_VALUE NEW\_CUST CUS\_CATGCODE EMPLOYMENT\_TYPE AGE SEX NO\_OF\_DEPENDENTS MARITAL EDU\_QUA P\_RESTYPE P\_CATEGORY EMPLOYEE\_TYPE MON\_IN\_OCC INCOM\_EXP\_GMI LTV TENURE STATUS

0 12345 0 YES 1 0 31 F 3 0 0 1 4 2 36 0 0.767104 12 0

1 12347 0 NO 1 1 40 F 2 1 1 0 1 1 12 2 0.619077 24 0

2 12349 0 YES 1 0 27 F 3 0 0 1 2 2 72 0 0.848949 36 0

3 12351 2 NO 1 1 33 M 2 0 1 0 2 1 120 1 0.515646 12 0

4 12353 2 NO 1 1 29 F 1 0 1 1 2 1 24 2 0.614123 24 1

Data Pre-processing and Visualization

Get the number of columns of object datatype.

**obj=(df.dtypes =='object')**

**print("categorical variable:", len(list(obj[obj].index)))**

**Output**

**categorical variable: 2**

As Loan\_ID is completely unique and not correlated with any of the other column, So we will drop it using .drop() function.

**df.drop(['APP\_ID'],axis=1,inplace=True)**

Visualize all the unique values in columns using barplot. This will simply show which value is dominating as per our dataset.

**obj=(df.dtypes =='object')**

**object\_cols =list(obj[obj].index)**

**plt.figure(figsize=(25,50))**

**index=1**

**for col in object\_cols:**

**y=df[col].value\_counts()**

**plt.subplot(11,4, index)**

**plt.xticks(rotation=90)**

**sns.barplot(x=list(y.index),y=y)**

**index +=1**



As all the categorical values are binary so we can use Label Encoder for all such columns and the values will change into int datatype.

**from sklearn import preprocessing**

**label\_encoder=preprocessing.LabelEncoder()**

**obj=(df.dtypes=='object')**

**for col in list(obj[obj].index):**

**df[col]=label\_encoder.fit\_transform(df[col])**

Again check the object datatype columns. Let’s find out if there is still any left.

**obj = (df.dtypes == 'object')**

**print("categorical variables:",len(list(obj[obj].index)))**

**Output :**

**Categorical variables: 0**

**plt.figure(figsize=(12,6))**

**sns.heatmap(df.corr(),cmap='BrBG',fmt='.2f',linewidths=2,annot=True)**



The above heatmap is showing the correlation between Loan Amount and Applicant Income. It also shows that Credit History has a high impact on Loan Status.

Now we will find out if there is any missing values in the dataset using below code.

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

**CIBIL\_SCORE\_VALUE 0**  
**NEW\_CUST 0**  
**CUS\_CATGCODE 0**  
 **EMPLOYMENT\_TYPE 0**  
**AGE 0**  
 **SEX 0**  
 **NO\_OF\_DEPENDENTS 0**  
 **MARITAL 0**  
 **EDU\_QUA 0**  
 **P\_RESTYPE 0**  
 **P\_CATEGORY 0**  
 **EMPLOYEE\_TYPE 0**  
 **MON\_IN\_OCC 0**  
 **INCOM\_EXP\_GMI 0**  
**LTV 0**  
 **TENURE 0**  
**STATUS 0**  
**dtype: int64**

As there is no missing value then we must proceed to model training.

Splitting Dataset

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

**x = df.drop(['STATUS'],axis=1)**

**y = df['STATUS']**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.4,random\_state=1)**

**x\_train.shape,x\_test.shape,y\_train.shape,y\_test.shape**

**Output:**

**((7979, 16), (5320, 16), (7979,), (5320,))**

Model Training and Evaluation

As this is a classification problem so we will be using these models:

KNeighborsClassifiers

RandomForestClassifiers

Support Vector Classifiers (SVC)

Logistics Regression

To predict the accuracy we will use the accuracy score function from scikit-learn library.

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.svm import SVC**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn import metrics**

**knn = KNeighborsClassifier(n\_neighbors=3)**

**rfc = RandomForestClassifier(n\_estimators = 7,criterion = 'entropy', random\_state = 7)**

**svc = SVC()**

**lc = LogisticRegression()**

**for clf in(rfc,knn,svc,lc):**

**clf.fit(x\_train,y\_train)**

**y\_pred = clf.predict(x\_train)**

**print("Accuracy score of ",clf.\_\_class\_\_.\_\_name\_\_,"=",100\*metrics.accuracy\_score(y\_train,y\_pred))**

**Accuracy score of RandomForestClassifier = 96.36545933074319**  
**Accuracy score of KNeighborsClassifier = 77.1650582779797**  
**Accuracy score of SVC = 62.10051384885325**  
**Accuracy score of LogisticRegression = 63.980448677779165**

**for clf in(rfc,knn,svc,lc):**

**clf.fit(x\_train,y\_train)**

**y\_pred = clf.predict(x\_test)**

**print("Accuracy score of ",clf.\_\_class\_\_.\_\_name\_\_,"=",100\*metrics.accuracy\_score(y\_test,y\_pred)**)

**Accuracy score of RandomForestClassifier = 62.16165413533835**  
**Accuracy score of KNeighborsClassifier = 54.81203007518797**  
**Accuracy score of SVC = 62.55639097744361**  
**Accuracy score of LogisticRegression = 63.1203007518797**

Conclusion:

Random Forest Classifier is giving the best accuracy with an accuracy score of 82% for the testing dataset. And to get much better results ensemble learning techniques like Bagging and Boosting can also be used.