Data Pre-Processing

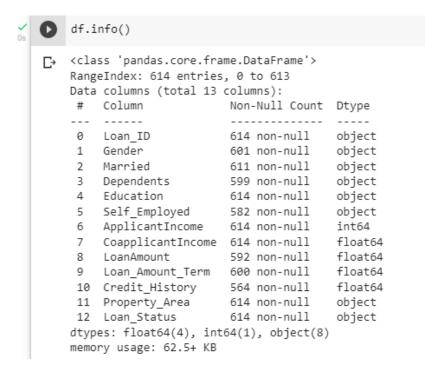
Date	01 November 2022
Team ID	PNT2022TMID12860
Project Name	Smart Lender - Applicant Credibility Prediction for
	Loan Approval

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

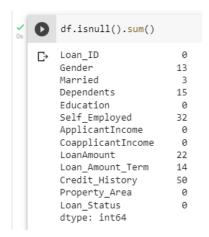
- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

CHECKING FOR NULL VALUES

• To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.



• For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip the handling of the missing values step.



- From the above code of analysis, we can infer that columns such as gender, married, dependents, self-employed, loan amount, loan amount term, and credit history are having the missing values, we need to treat them in a required way.
- We will fill in the missing values in numeric data type using the mean value of that particular column and categorical data type using the most repeated value.

```
[13] df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
     df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
     #replacing + with space for filling the nan values
     df['Dependents']=df['Dependents'].replace('3+',3)
     df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
     df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
     df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount']. mode()[0])
     df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0])
     df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mode()[0])
 df.isnull().sum()
 Loan_ID
     Gender
     Married
     Dependents
     Education
     Self Employed
     ApplicantIncome
     CoapplicantIncome
     LoanAmount
     Loan_Amount_Term
     Credit_History
                          0
     Property_Area
                          0
     Loan_Status
                          0
     dtype: int64
```

HANDLING CATEGORICAL VALUES

- To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project, we are using manual encoding with the help of list comprehension.
- In our project, Gender, married, dependents, self-employed, co-applicants income, loan amount, loan amount term, credit history With list comprehension encoding is done.

```
(16) from sklearn.preprocessing import LabelEncoder
        le=LabelEncoder()
       df.Gender=le.fit transform(df.Gender)
        df.Loan Status=le.fit transform(df.Loan Status)
        df.Married=le.fit_transform(df.Married)
        df.Education=le.fit_transform(df.Education)
        df.Self Employed=le.fit transform(df.Self Employed)
        df.Property_Area=le.fit_transform(df.Property_Area)
   #changing the datype of each float column to int
        df['Gender']=df['Gender'].astype('int64')
        df['Married']=df['Married'].astype('int64')
        df['Dependents']=df['Dependents'].astype('int64')
        df['Self Employed']=df['Self Employed'].astype('int64')
       df['CoapplicantIncome']=df['CoapplicantIncome'].astype('int64')
       df['LoanAmount']=df['LoanAmount'].astype('int64')
        df['Loan_Amount_Term']=df['Loan_Amount_Term'].astype('int64')
        df['Credit History']=df['Credit History'].astype('int64')
```

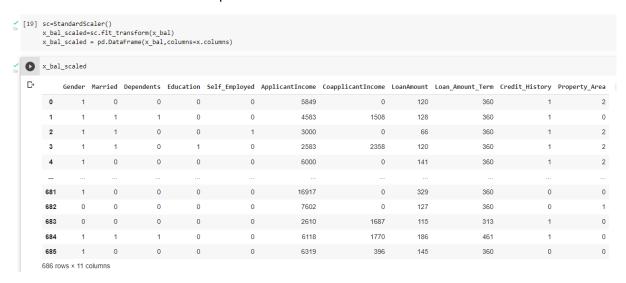
BALANCING THE DATASET

- Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset ,we will get biased results, which means our model is able to predict only one class element
- For Balancing the data we are using the SMOTE Method.

```
#Balancing the dfset by using smote
 from imblearn.combine import SMOTETomek
smote = SMOTETomek (0.95)
 y = df['Loan_Status']
 x = df.drop(columns=["Loan_ID", 'Loan_Status'], axis=1)
 x bal,y bal =smote.fit resample(x,y)
 print(y.value_counts())
 print(y_bal.value_counts())
1
    422
    192
Name: Loan_Status, dtype: int64
1 354
Name: Loan_Status, dtype: int64
 /usr/local/lib/python3.7/dist-packages/imblearn/utils/_validation.py:591:
   FutureWarning,
```

SCALING THE DATA

- Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction
- Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.



SPLITTING DATA INTO TRAIN AND TEST DATA

Here x and y variables are created. On the x variable, df is passed by dropping the target variable.
 And on y target variable is passed. For splitting training and testing data, we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, and random_state.

Code is uploaded in the following drive link:

https://colab.research.google.com/drive/10YDz5VLr60QmNikdFTSWBxKUMaqxPi2w