Balancing The Dataset

The dataset is already download in .csv format

IMPORTING THE PACKAGE

In [1]:

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

In [2]:

```
df=pd.read_csv("C:\loan_prediction.csv")
```

In [3]:

```
df
```

O...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Lo
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	
...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	

614 rows × 13 columns

In [4]:

```
df.shape
```

Out[4]:

```
(614, 13)
```

Handle the Missing values

In [5]:

```
#checking the null values
df.isnull().sum()
```

Out[5]:

```
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
dtype: int64
```

Treating the Null Value

We will fill the missing values in numeric data type using the mean value of that particular column and categorical data type using the most repeated value

In [6]:
```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

In [7]:
```
numerical_features
```

Out[7]:
```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History'],
      dtype='object')
```

In [8]:
```
categorical_features
```

Out[8]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [9]:
```
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
```

In [10]:
```
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
```

In [11]:
```
#replace + with non value
df['Dependents'] = df['Dependents'].str.replace('+','')
```

In [12]:
```
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
```

In [13]:
```
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
```

In [14]:
```
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mode()[0])
```

In [1...
```
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0])
```

In [16]:
```
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mode()[0])
```

In [17]:
```
#checking the null values now
df.isnull().sum()
```

Out[17]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
```

```
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

Now the null value is retreated

Handling Categorical Values

In [18]:

```
df.select_dtypes(include='object').columns
```

Out[18]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [19]:

```
df['Gender'].unique()
```

Out[19]:

```
array(['Male', 'Female'], dtype=object)
```

In [20]:

```
df['Gender'].replace({'Male':1,'Female':0},inplace=True)
```

In [21]:

```
df['Married'].unique()
```

Out[21]:

```
array(['No', 'Yes'], dtype=object)
```

In [22]:

```
df['Married'].replace({'Yes':1,'No':0},inplace=True)
```

In [23]:

```
df['Dependents'].unique()
```

Out[23]:

```
array(['0', '1', '2', '3'], dtype=object)
```

In [24]:

```
df['Dependents'].replace({'0':0,'1':1,'2':2,'3':3},inplace=True)
```

In [25]:

```
df['Self_Employed'].unique()
```

Out[25]:

```
array(['No', 'Yes'], dtype=object)
```

In [26]:

```
df['Self_Employed'].replace({'Yes':1,'No':0},inplace=True)
```

In [27]:

```
df['Property_Area'].unique()
```

Out[27]:

```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [28]:

```
df['Property_Area'].replace({'Urban':2,'Rural':0,'Semiurban':1},inplace=True)
```

In [29]:

```
df['Loan_Status'].unique()
```

Out[29]:

```
array(['Y', 'N'], dtype=object)
```

In [30]:

```
df['Loan_Status'].replace({'Y':1,'N':0},inplace=True)
```

In [31]:

```
df['Education'].unique()
```

Out[31]:
```
array(['Graduate', 'Not Graduate'], dtype=object)
```

In [32]:
```
df['Education'].replace({'Graduate':1,'Not Graduate':0},inplace=True)
```

In [33]:
```
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")
```

In [34]:
```
# dummy columns are created for the categories in Loan_ID
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Loan_ID'] = le.fit_transform(df.Loan_ID)
```

In [35]:
```
df.head()
```

Ou...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loan
0	0	1	0	0	1	0	5849	0	
1	1	1	1	1	1	0	4583	1508	
2	2	1	1	0	1	1	3000	0	
3	3	1	1	0	0	0	2583	2358	
4	4	1	0	0	1	0	6000	0	

Balancing The Dataset

In [36]:
```
from imblearn.combine import SMOTETomek
```

In [37]:
```
smote = SMOTETomek(0.90)
```

In [38]:
```
#dividing the dataset into dependent and independent y and x respectively

y = df['Loan_Status']
x = df.drop(columns=['Loan_Status'],axis=1)
```

In [39]:
```
#creating the new x and y for balance data
x_bal,y_bal = smote.fit_resample(x,y)
```

In [40]:
```
#printing the value before and after balancing
print(y.value_counts())
print(y_bal.value_counts())
```

```
1    422
0    192
Name: Loan_Status, dtype: int64
1    371
0    328
Name: Loan_Status, dtype: int64
```

In [ ]: