Splitting Data Into Train And Test
The dataset is already download in .csv format

IMPORTING THE PACKAGE

In [1]:
```python
import numpy as np
import pandas as pd

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

Load the dataset

In [2]:
```python
df=pd.read_csv("C:\loan_prediction.csv")
```

In [3]:
```python
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]:
```python
df.shape
```

Out[4]: (614, 13)

Handle the Missing values

In [5]:
```python
#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]:
```python
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]:
```python
df['Gender'].unique()
```

Out[19]:
```
array(['Male', 'Female'], dtype=object)
```

In [20]:
```python
df['Gender'].replace({'Male':1,'Female':0},inplace=True)
```

In [21]:
```python
df['Married'].unique()
```

Out[21]:
```
array(['No', 'Yes'], dtype=object)
```

In [22]:
```python
df['Married'].replace({'Yes':1,'No':0},inplace=True)
```

In [23]:
```python
df['Dependents'].unique()
```

Out[23]:
```
array(['0', '1', '2', '3'], dtype=object)
```

In [24]:
```python
df['Dependents'].replace({'0':0,'1':1,'2':2,'3':3},inplace=True)
```

In [25]:
```python
df['Self_Employed'].unique()
```

Out[25]:
```
array(['No', 'Yes'], dtype=object)
```

In [26]:
```python
df['Self_Employed'].replace({'Yes':1,'No':0},inplace=True)
```

In [27]:
```python
df['Property_Area'].unique()
```

Out[27]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [28]:
```python
df['Property_Area'].replace({'Urban':2,'Rural':0,'Semiurban':1},inplace=True)
```

In [29]:
```python
df['Loan_Status'].unique()
```

Out[29]:
```
array(['Y', 'N'], dtype=object)
```

In [30]:
```python
df['Loan_Status'].replace({'Y':1,'N':0},inplace=True)
```

In [31]:
```python
df['Education'].unique()
```

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

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

In [33]:
```python
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]:
```python
# 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]:
```python
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]:
```python
from imblearn.combine import SMOTETomek
```

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

In [38]:
```python
#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]:
```python
#creating the new x and y for balance data
x_bal,y_bal = smote.fit_resample(x,y)
```

In [40]:
```python
#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    359
0    316
Name: Loan_Status, dtype: int64
```
Scaling The Data

In [41]: **from** sklearn.preprocessing **import** StandardScaler

In [42]:
```python
sc = StandardScaler()
x_bal = sc.fit_transform(x_bal)
```

In [43]:
```python
x_bal = pd.DataFrame(x_bal)
```

In [44]:
```python
x_bal.head()
```

Out[...

	0	1	2	3	4	5	6	7	8	9
0	-1.696389	0.529939	-1.159202	-0.718409	0.607592	-0.329203	0.099167	-0.468799	-0.277759	0.304575	0.5
1	-1.690621	0.529939	0.862662	0.324443	0.607592	-0.329203	-0.120245	-0.025950	-0.170708	0.304575	0.5
2	-1.684854	0.529939	0.862662	-0.718409	0.607592	3.037643	-0.394597	-0.468799	-1.000358	0.304575	0.5
3	-1.679087	0.529939	0.862662	-0.718409	-1.645841	-0.329203	-0.466868	0.223666	-0.277759	0.304575	0.5
4	-1.673319	0.529939	-1.159202	-0.718409	0.607592	-0.329203	0.125337	-0.468799	0.003251	0.304575	0.5

We will perform scaling only on the input values

Splitting Data Into Train And Test

In [45]:
```python
# splitting the data into training and testing set

from sklearn.model_selection import train_test_split
```

In [...
```python
x_train, x_test, y_train, y_test = train_test_split(x_bal, y_bal, test_size = 0.33, rando
```

In [47]:
```python
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(452, 12)
(452,)
(223, 12)
(223,)
```

In [48]:
```python
x_train
```

Out[...

	0	1	2	3	4	5	6	7	8	9
221	-0.191109	0.529939	-1.159202	-0.718409	0.607592	-0.329203	-0.557509	0.179910	-0.090419	0.304575
516	0.933526	-1.887011	-1.159202	-0.718409	-1.645841	-0.329203	-0.324579	-0.468799	-0.491863	0.304575
660	-0.837053	-1.887011	-1.159202	-0.718409	-1.645841	-0.329203	0.358960	-0.468799	-0.143945	0.304575
527	-1.090816	0.529939	0.862662	-0.718409	-1.645841	-0.329203	0.435564	-0.468799	-0.358048	0.304575
84	-1.148490	0.529939	-1.159202	-0.718409	0.607592	-0.329203	0.260866	-0.468799	-0.277759	0.304575
...	...	...	...	...	...	...	...	...	...	...

	0	1	2	3	4	5	6	7	8	9
71	-1.229233	0.529939	0.862662	0.324443	0.607592	3.037643	-0.499450	-0.468799	-0.277759	0.304575
106	-0.998539	-1.887011	-1.159202	-0.718409	0.607592	-0.329203	-0.192516	-0.468799	-1.294750	0.304575
270	0.137630	-1.887011	-1.159202	0.324443	0.607592	-0.329203	-0.300835	-0.468799	-0.384811	0.304575
435	1.273800	0.529939	0.862662	2.410147	0.607592	-0.329203	-0.313834	0.537889	0.123684	0.304575
102	-1.033143	0.529939	0.862662	0.324443	0.607592	-0.329203	1.612867	-0.468799	0.592035	-2.393236

452 rows × 12 columns

In [49]:
```
x_test
```

Out[...]

	0	1	2	3	4	5	6	7	8	9
396	1.008501	0.529939	0.862662	-0.718409	0.607592	-0.329203	-0.163573	0.250977	-0.411574	0.304575
302	0.356790	0.529939	0.862662	1.367295	0.607592	-0.329203	0.529672	0.461242	0.324406	0.304575
673	-0.444872	0.529939	-1.159202	-0.718409	-1.645841	-0.329203	-0.605343	0.835666	-0.050275	0.304575
541	1.152685	0.529939	-1.159202	0.324443	0.607592	-0.329203	0.224124	-0.468799	-0.384811	0.304575
377	0.881619	0.529939	0.862662	-0.718409	0.607592	-0.329203	-0.301182	-0.064714	-1.147554	0.304575
...	...	...	...	...	...	...	...	...	...	...
482	1.654445	0.529939	-1.159202	-0.718409	0.607592	-0.329203	-0.596331	9.467980	-0.679203	0.304575
380	0.904689	0.529939	0.862662	1.367295	-1.645841	-0.329203	-0.575187	-0.041221	-1.080647	-0.594695
9	-1.644482	0.529939	0.862662	0.324443	0.607592	-0.329203	1.310959	2.752131	2.786594	0.304575
22	-1.552205	0.529939	0.862662	-0.718409	0.607592	3.037643	0.742325	-0.468799	0.672324	0.304575
290	0.270280	-1.887011	-1.159202	-0.718409	0.607592	-0.329203	-0.502396	-0.468799	-1.267987	0.304575

223 rows × 12 columns

In [50]:
```
y_train
```

Out[50]:
```
221    1
516    0
660    0
527    0
84     0
      ..
71     1
106    1
270    1
435    1
102    1
Name: Loan_Status, Length: 452, dtype: int64
```

In [51]:
```
y_test
```

```
Out[51]:396     0
         302     1
         673     0
         541     0
         377     0
                 ..
         482     0
         380     1
         9       0
         22      1
         290     0
         Name: Loan_Status, Length: 223, dtype: int64
```

In [ ]: