Loan Predictor EDA-Classification

Kaggle Link : https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset/download?datasetVersionNumber=1 (https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset/download?datasetVersionNumber=1)

Created by Binayak Mukherjee

1. Business Understanding

Students are expected to identify a classification problem of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

```
# --------------Type the answers below this line--------------


What is the business problem that you are trying to solve?
Answer: We are attempting to create a Classification Model that will be capable to judge wither a person will be eligible
for a Personal Loan or not based on the various features bellow such as income, credit history, loan amount, etc.


What data do you need to answer the above problem?
Answer: We require a Data Set that consists of the Loan Statuses of a set of people so that our model can be trained upon
and later implemented over real time to solve the above mentioned bussiness problem


What are the different sources of data?
Answer: We have take the Data set from Kaggle website that has a containment of users details and data pertaining to their
Loan Statuses which will be well suitable for the creation and training of the bellow model to resolve the Bussiness Problem


What kind of analytics task are you performing?
Answer: We have clean,encoded and visualised the data for better understanding and the later phases we have detected any
possible anomaly and at last performed Classification over the same data for the final resultant
```

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique) from any public data source.

2.1 Download the data directly

In [ ]:
```
##---------Type the code below this line------------------##


import opendatasets as od
import pandas as pd

od.download("https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset/download?datasetVersionNumber=1
#Instructions : Please insert Kaggle Auth ID and Key to download the data in you
#end which will download the data in you mashine from the embeded dataset link
```

2.2 Code for converting the above downloaded data into a dataframe

In [1]:
```
##---------Type the code below this line------------------##
import pandas as pd
df = pd.read_csv("C:/Users/ASUS/Asignment/loan-prediction-problem-dataset/train_u6lujuX_CVtuZ9i.csv")
testdf = pd.read_csv("C:/Users/ASUS/Asignment/loan-prediction-problem-dataset/test_Y3wMUE5_7gLdaTN.csv")
print("_____________________________________Printing the Training Data_____________________________________")
print(df)
print("_____________________________________Printing the Testing Data_____________________________________")
print(testdf)
```

```
_____Printing the Training Data_____
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0         NaN             360.0
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
..               ...                ...         ...               ...
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
_____Printing the Testing Data_____
      Loan_ID Gender Married Dependents     Education Self_Employed  \
0    LP001015   Male     Yes          0      Graduate            No
1    LP001022   Male     Yes          1      Graduate            No
2    LP001031   Male     Yes          2      Graduate            No
3    LP001035   Male     Yes          2      Graduate            No
4    LP001051   Male      No          0  Not Graduate            No
..        ...    ...     ...        ...           ...           ...
362  LP002971   Male     Yes         3+  Not Graduate           Yes
363  LP002975   Male     Yes          0      Graduate            No
364  LP002980   Male      No          0      Graduate            No
365  LP002986   Male     Yes          0      Graduate            No
366  LP002989   Male      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5720                  0       110.0             360.0
1               3076               1500       126.0             360.0
2               5000               1800       208.0             360.0
3               2340               2546       100.0             360.0
4               3276                  0        78.0             360.0
..               ...                ...         ...               ...
362             4009               1777       113.0             360.0
363             4158                709       115.0             360.0
364             3250               1993       126.0             360.0
365             5000               2393       158.0             360.0
366             9200                  0        98.0             180.0

     Credit_History Property_Area
0               1.0         Urban
1               1.0         Urban
2               1.0         Urban
3               NaN         Urban
4               1.0         Urban
..              ...           ...
362             1.0         Urban
363             1.0         Urban
364             NaN     Semiurban
365             1.0         Rural
366             1.0         Rural

[367 rows x 12 columns]
```

2.3 Confirm the data has been correctly by displaying the first 5 and last 5 records.

In [2]:
```
##---------Type the code below this line------------------##

print(df.head(5))
print(df.tail(5))
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
      Loan_ID  Gender Married Dependents Education Self_Employed  \
609  LP002978  Female      No          0  Graduate            No
610  LP002979    Male     Yes         3+  Graduate            No
611  LP002983    Male     Yes          1  Graduate            No
612  LP002984    Male     Yes          2  Graduate            No
613  LP002990  Female      No          0  Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N
```

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [3]:
```
##---------Type the code below this line------------------##
df.columns
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

Out[3]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

```
--------------Type the answers below this line--------------

Size of the dataset
Answer:
    Number of Attributes(Columns):  13
    Number of Rows: 614

What type of data attributes are there?
Answer :
    Categorical: 8
    Numerical: 4
    Excluding Loan ID



Is there any null data that has to be cleaned?
Answer : 148 Total Null/Empty/Blank Fields detected ( which has been cleaned in the later stages )
```

3. Data Preparation

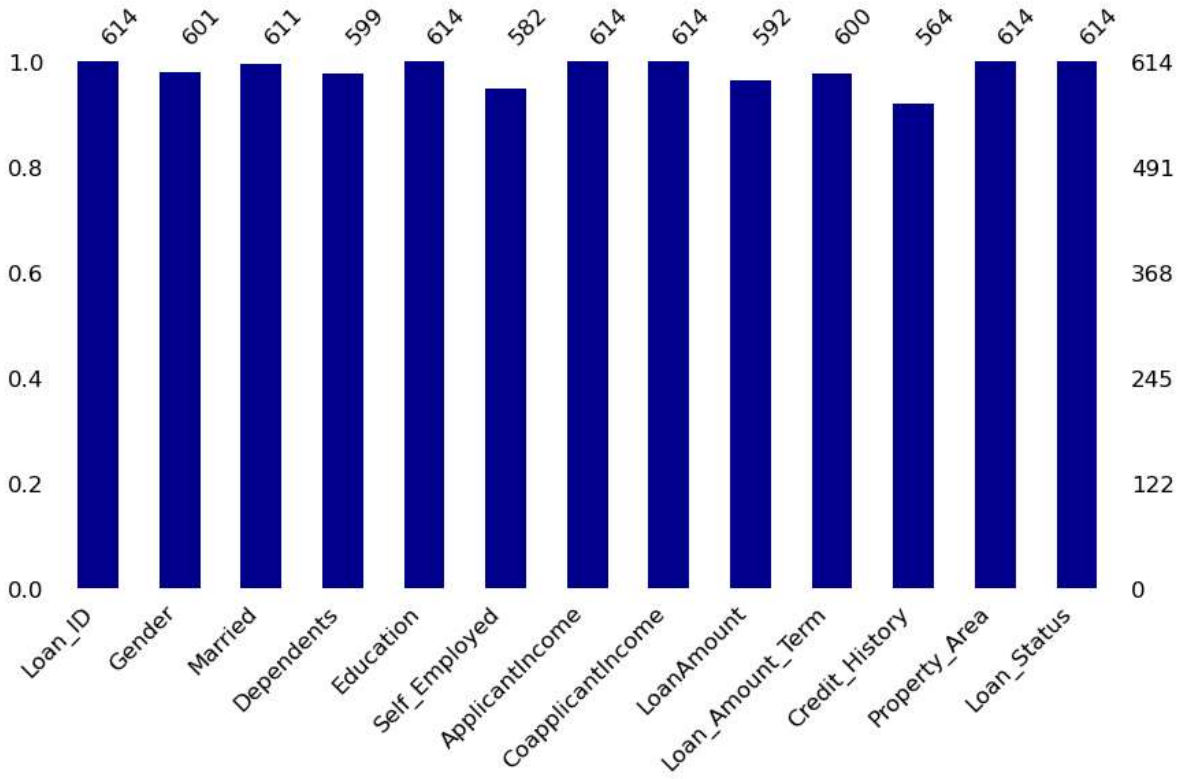
```
If input data is numerical or categorical, do 3.1, 3.2 and 3.4
If input data is text, do 3.3 and 3.4
```

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [4]:
```
##---------Type the code below this line------------------##

duplicate_data = df.duplicated().any()
missing_data = df.isnull().any()
data_inconsistencies = df.isna().sum()

## Check for Duplicate value in Dataset
print("Checking for duplicate data--------->")
print(df.duplicated())
if duplicate_data == True:
  print("Count of duplicate data ----------> ", df.duplicated().count())
else:
  print("Count of duplicate data ---------->No duplicate data exists in the given dataset")

print ("                                                                                                                    ")

## Check for Missing Value in Dataset
print("Checking for missing data--------->")
print ("Print the sum of all null values for each possible column in descending order---------->")
print(df.isnull().sum().sort_values(ascending=False))
print("                                        ")

#!pip install missingno     ---> To analyse the missing data

import missingno as msno

msno.bar(df,figsize=(10,5), fontsize=12,color="darkblue")



## Check for Data inconsistencies in Dataset
print("Checking for data inconsistencies (if any)-------->")
print(data_inconsistencies)
```

```
Checking for duplicate data--------->
0      False
1      False
2      False
3      False
4      False
       ...
609    False
610    False
611    False
612    False
613    False
Length: 614, dtype: bool
Count of duplicate data ---------->No duplicate data exists in the given dataset

Checking for missing data--------->
Print the sum of all null values for each possible column in descending order---------->
Credit_History       50
Self_Employed        32
LoanAmount           22
Dependents           15
Loan_Amount_Term     14
Gender               13
Married               3
Loan_ID               0
Education             0
ApplicantIncome       0
CoapplicantIncome     0
Property_Area         0
Loan_Status           0
dtype: int64

Checking for data inconsistencies (if any)-------->
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
```



3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

In [5]:
```
##---------Type the code below this line-----------------##

# To remove duplicate data

duplicate = df.duplicated()
print("Row wise duplicate data status\n", duplicate)
print("                                                  ")
print("Value count of non-duplicate values in the dataset\n(", duplicate.value_counts(),')')
# Droping the duplicated
#df.drop_duplicates()


# To impute or remove missing data
#We can remove the missing values by the code bellow but we have not done that
#for better efficient in the classification model
# import missingno as msno
# df.isnull().sum().sort_values(ascending=False)
# df.dropna(axis=0,how='any', thresh = None, subset = None, inplace=True)


# msno.bar(df,figsize=(10,5), fontsize=12,color="DarkBlue")


# To remove data inconsistencies

df['Loan_ID'].value_counts()
```

```
Row wise duplicate data status
 0      False
1       False
2       False
3       False
4       False
        ...
609     False
610     False
611     False
612     False
613     False
Length: 614, dtype: bool

Value count of non-duplicate values in the dataset
( False    614
dtype: int64 )
```

Out[5]:
```
LP001002    1
LP002328    1
LP002305    1
LP002308    1
LP002314    1
           ..
LP001692    1
LP001693    1
LP001698    1
LP001699    1
LP002990    1
Name: Loan_ID, Length: 614, dtype: int64
```

3.3 Encode categorical data

In [6]:
```
#Droping the Unwanted variable : Just5ification : We have dropped the Loan ID variable as there is no need for
#this attribute in the late stages as there can be no feesible judgement derived from the bellow feature
import numpy as np
df.drop(columns = ['Loan_ID'], inplace = True)
Rep_op=[]


for col in df.columns:
    if df[col].dtypes == str or df[col].dtypes == object:
        print("In the Column name  {} the unique values are {}".format(col, df[col].unique()))
        Rep_op.append(str("In the Column name  {} the unique values are {}".format(col, df[col].unique())))
#-------------------------------- Encoded Logic Specifications ----------------------------------------
G = {'Male' : 0, 'Female' : 1}
M = {'No': 0,  'Yes' : 1}
D = {'0' : 0,  '1' : 1,  '2' : 2,  '3+': 3}
Gr = {'Graduate' : 1,  'Not Graduate' : 2}
se = {'No' : 0, 'Yes' : 1}
ar = {'Urban' : 0,  'Rural' : 1,  'Semiurban' : 2}

def enc(d, el):
    if el in list(d.keys()):
        return d[el]
    else:
        return np.nan
#-------------------------------------------Encoding Train Data--------------------------

df['Gender'] = df['Gender'].apply(lambda X : enc(G, X))
df['Married'] = df['Married'].apply(lambda X : enc(M, X))
df['Dependents'] = df['Dependents'].apply(lambda X : enc(D , X))
df['Education'] = df['Education'].apply(lambda X : enc(Gr , X))
df['Self_Employed'] = df['Self_Employed'].apply(lambda X : enc(se, X))
df['Property_Area'] = df['Property_Area'].apply(lambda X : enc(ar,X))
df['Loan_Status'] = df['Loan_Status'].apply(lambda x: 1 if x == 'Y' else 0)

#-------------------------------------------Encoding Test Data--------------------------
testdf['Gender'] = testdf['Gender'].apply(lambda X : enc(G, X))
testdf['Married'] = testdf['Married'].apply(lambda X : enc(M, X))
testdf['Dependents'] = testdf['Dependents'].apply(lambda X : enc(D , X))
testdf['Education'] = testdf['Education'].apply(lambda X : enc(Gr , X))
testdf['Self_Employed'] = testdf['Self_Employed'].apply(lambda X : enc(se, X))
testdf['Property_Area'] = testdf['Property_Area'].apply(lambda X :enc(ar,X))
```

```
In the Column name  Gender the unique values are ['Male' 'Female' nan]
In the Column name  Married the unique values are ['No' 'Yes' nan]
In the Column name  Dependents the unique values are ['0' '1' '2' '3+' nan]
In the Column name  Education the unique values are ['Graduate' 'Not Graduate']
In the Column name  Self_Employed the unique values are ['No' 'Yes' nan]
In the Column name  Property_Area the unique values are ['Urban' 'Rural' 'Semiurban']
In the Column name  Loan_Status the unique values are ['Y' 'N']
```

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

In [7]:
```
##---------Type the code below this line------------------##

# Removal of Special characters, Transform into common case by either up-casing or down-casing, Tokenization and Filtration of
# can not be used in the selected dataset as no text data is present in this dataset. Hence, it is not possible to apply the a
```

In [8]:
```
##---------Type the code below this line------------------##
```

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

In [9]:
```
##---------Type the code below this line------------------##

#Justification for Section 3.2

# 1) to remove duplicate data, if present

# Answer :
# We have investigated each attribute explicitly for the count of the Duplicated Data present in the Dataset
# and have amputated the data fields that doesnt meet the criteria hence are duplicate. The df.duplicated() methord has
# been used to identify any duplicated values and dropped using the df.drop_duplicates() function



# 2)to impute or remove missing data, if present
# Answer : Null/Empty/NA values have been detected and deleted (dropped) using functions like dropna



# 3)to remove data inconsistencies, if present
# Answer : Since there is a unique identifying feature(Loan_ID) present in the used dataset hence it signifies data
# consistency for all the existing features and no visible redundancy in the data,please be aware that Loan_ID will be
#dropped in the later stages

# Please be aware that in the later stages we have performed anomaly detection process on this dataset as well to
# evaluate minutely if and only if there is any inconsistency present in form of outliers


##---------Type the code below this line------------------##
#1) to remove duplicate data, if present

# Answer :
# We have investigated each attribute explicitly for the count of the Duplicated Data present in the Dataset
# and have amputated the data fields that doesnt meet the criteria hence are duplicate. The df.duplicated() methord has
# been used to identify any duplicated values and dropped using the df.drop_duplicates() function



# 2)to impute or remove missing data, if present
# Answer : Null/Empty/NA values have been detected and deleted (dropped) using functions like dropna



# 3)to remove data inconsistencies, if present
# Answer : Since there is a unique identifying feature(Loan_ID) present in the used dataset hence
# it signifies data consistency for all the existing features and no visible redundancy in the data,
#please be aware that Loan_ID will be dropped in the later stages

# Please be aware that in the later stages we have performed anomaly detection process on this dataset as well to evaluate
# minutely if and only if there is any inconsistency present in form of outliers
```

In [10]:
```
##---------Type the code below this line------------------##
```

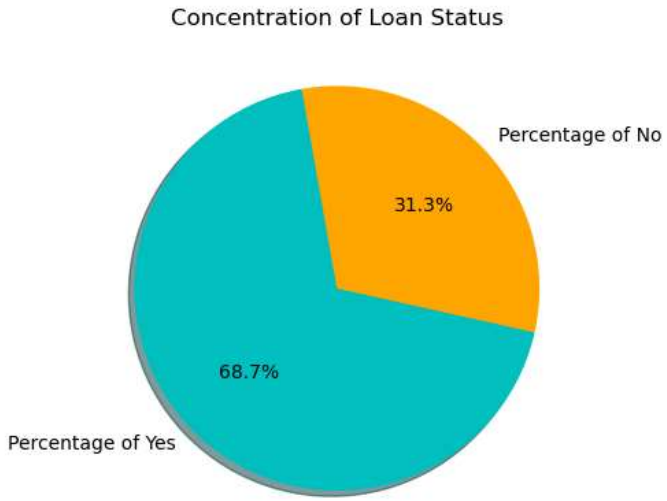
3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

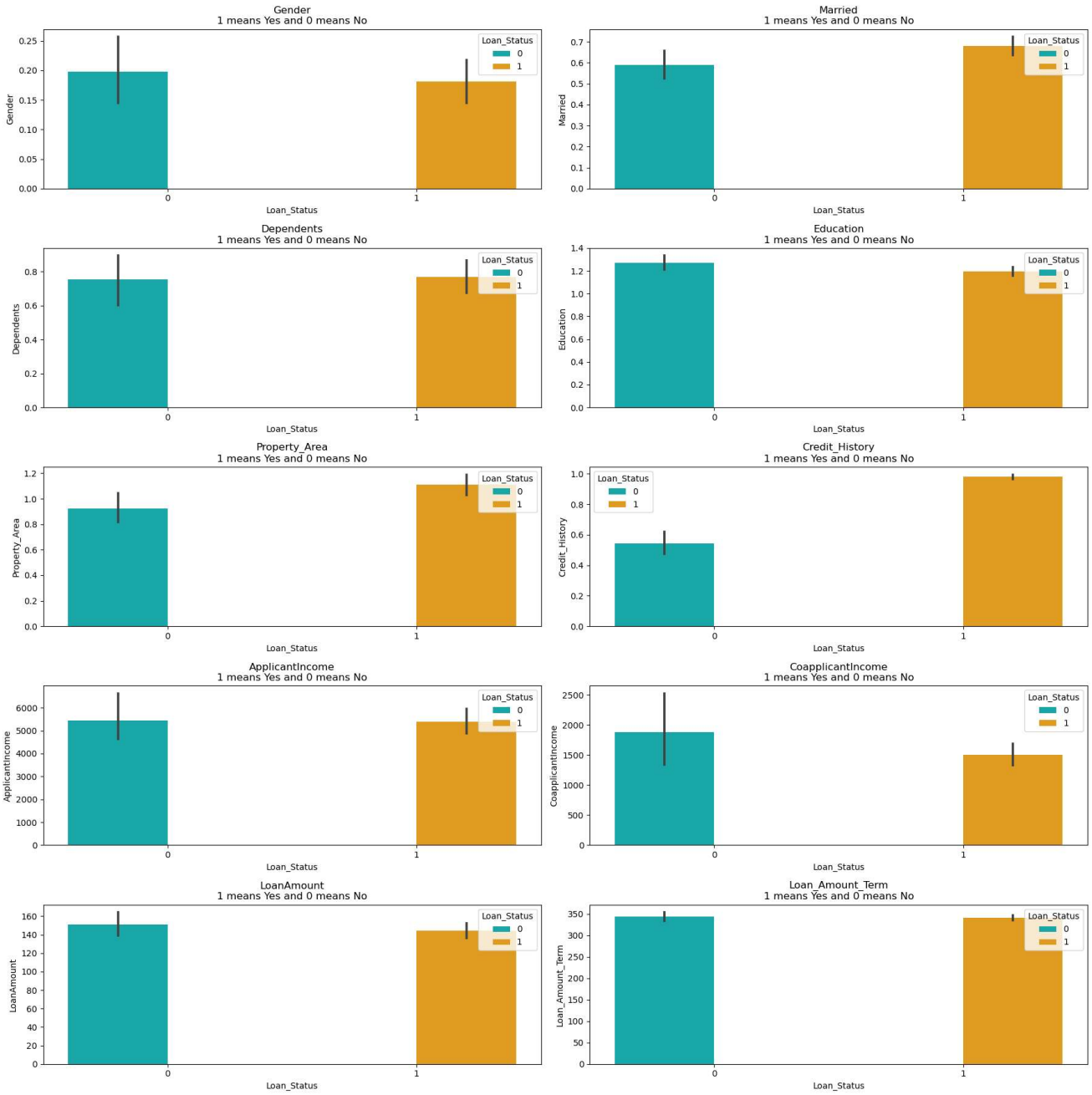
Score: 1 Mark

In [11]:

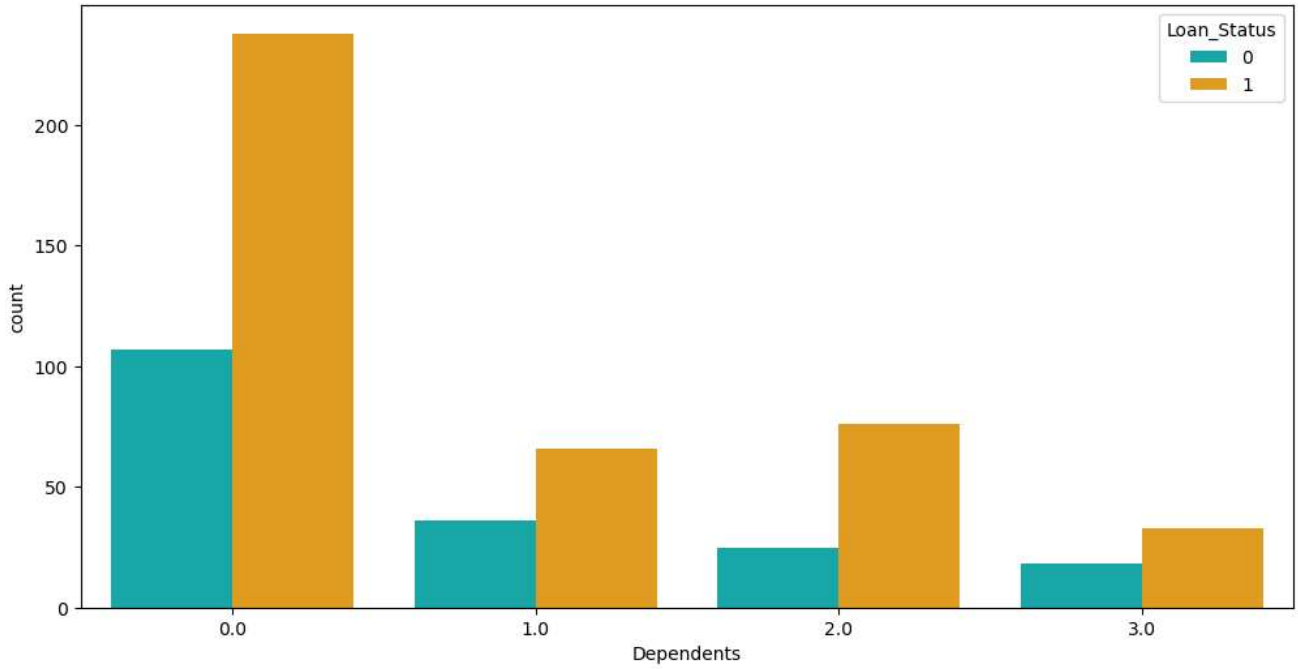
```
##---------Type the code below this line------------------##
#From the Data Set and the collective aim to determine and predict right personel
#for the Loans we will keep the Loan Status as the Target Variable
#as it is the decisive element for determining the solution of our problem


import matplotlib.pyplot as plt
import seaborn as sns


totaldatacount=df["Loan_Status"].count()
yesnocount=df['Loan_Status'].value_counts()


yescount=round((yesnocount[1]/totaldatacount)*100,2)
nocount=round((yesnocount[0]/totaldatacount)*100,2)
slice = [yescount,nocount]
activities = ["Percentage of Yes","Percentage of No"]
cols = ['c','orange']
plt.pie(slice, labels =activities, colors = cols, startangle = 100, shadow = True, autopct ='%1.1f%%')
plt.title('Concentration of Loan Status')
# Print the chart
plt.show()
print("Percentages of Yes = ",yescount,"% and Percentage of No Count",nocount,"%")
#------------ Now we need to observer the intricate details of the data with respect to our Target Variable - Loan_Status
# Define the features and target variable
feature_vars=["Gender", "Married", "Dependents", "Education", "Property_Area", "Credit_History","ApplicantIncome",
              "CoapplicantIncome","LoanAmount","Loan_Amount_Term"]
target="Loan_Status"
colors = ['c','orange']
# Create a 2x5 grid of subplots
fig, axs = plt.subplots(nrows=5, ncols=2, figsize=(18,18))
# Flatten the subplots into a 1D array
axs = axs.ravel()
# Iterate over each feature and plot a bar plot on a separate subplot
for i, feature in enumerate(feature_vars):
    sns.barplot(data=df, x=target, y=feature, hue="Loan_Status", ax=axs[i],palette=colors)
    axs[i].set_title(feature+"\n 1 means Yes and 0 means No ")
# Adjust the spacing between the subplots
fig.tight_layout()
# Display the plot
plt.show()

#Dependents
plt.figure(figsize=(12,6))
sns.countplot(x='Dependents', hue='Loan_Status', data=df,palette=colors,);
sns.displot(data=df, x ="Loan_Status",kde=True)
plt.title("Loan_Status vs ApplicantIncome")
plt.xlabel("Loan_Status\n 1 means Yes and 0 means No")
plt.ylabel("ApplicantIncome")



###Observations of the above steps :
#If we observe and draw inferance from the bellow projections we will see that the consistency
# of Yes and No with respect to our Target Variable(Loan Status)
# is Yes =  68.73 % and No 31.27 %
# with denotes to a significant and adequte trend in terms of the
# polarity of the data.

# Now lets observe the inter-feature comparisons which shows that there is ample
# and just representation of both types/polarity
# of data with respect to the Loan Status present amongst our features
# which will be fundamental in the determination of the
# classes and other data based judgements that will be derived from
# further investigation and application of techniques to
# effectively draw and infer from the dataset we have chosen
```

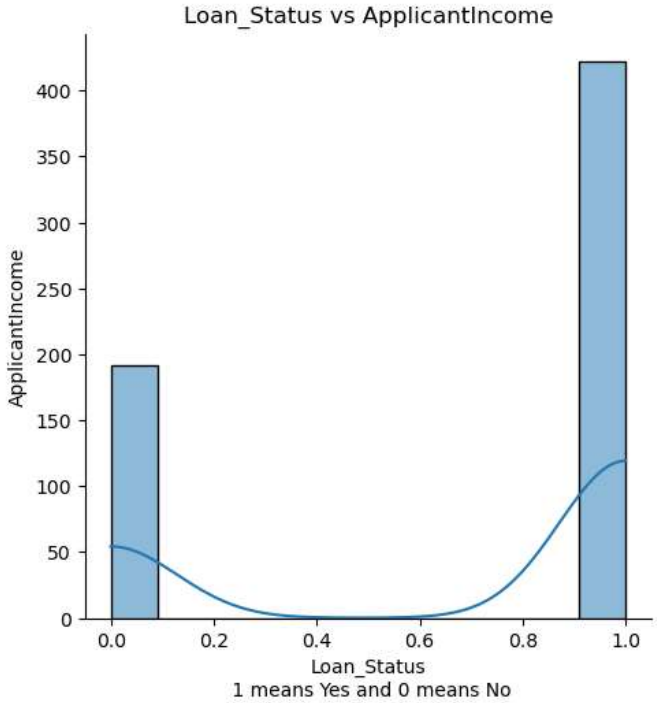


Percentages of Yes = 68.73 % and Percentage of No Count 31.27 %



Out[11]: Text(4.944444444444445, 0.5, 'ApplicantIncome')



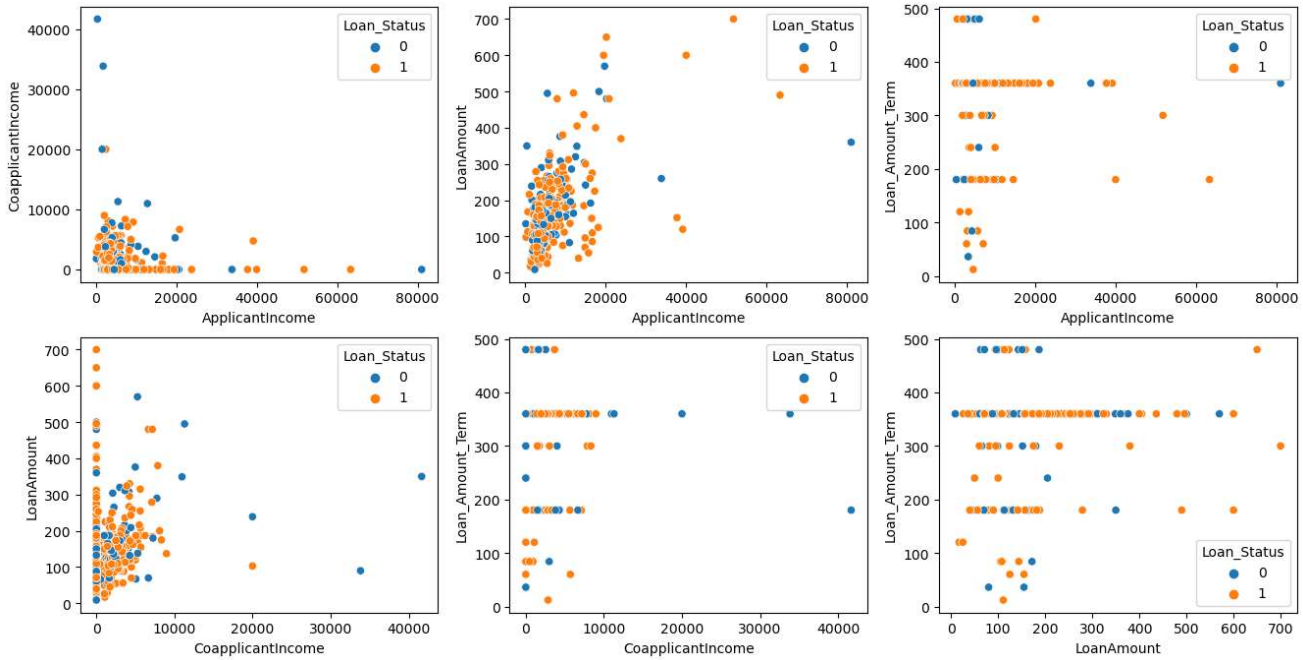


4. Data Exploration using various plots

4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

In [12]:
```
##---------Type the code below this line-----------------##

import itertools
import seaborn as sns
from matplotlib import pyplot as plt
quantitativevars=["ApplicantIncome","CoapplicantIncome","LoanAmount","Loan_Amount_Term"]
targetvar="Loan_Status"

com = list(itertools.combinations(quantitativevars, 2))

fig, axes = plt.subplots(2,3,figsize=(16,8))
for n,elem in enumerate(com):
  sns.scatterplot(data=df, x=elem[0],y=elem[1], hue=targetvar, ax=axes[n//3, n%3])
```



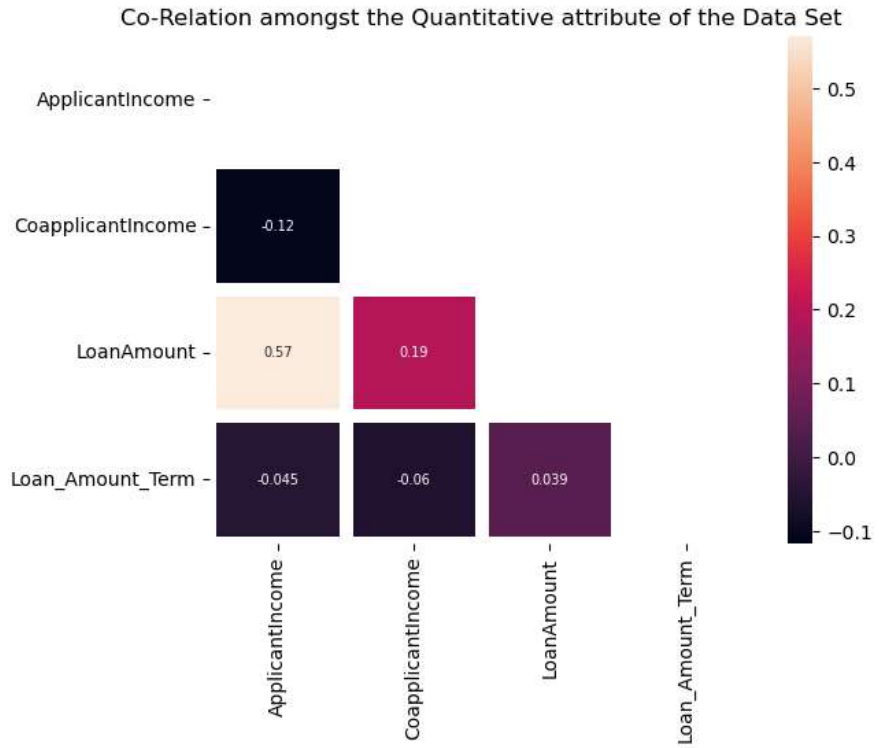
4.2 EDA using visuals

- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

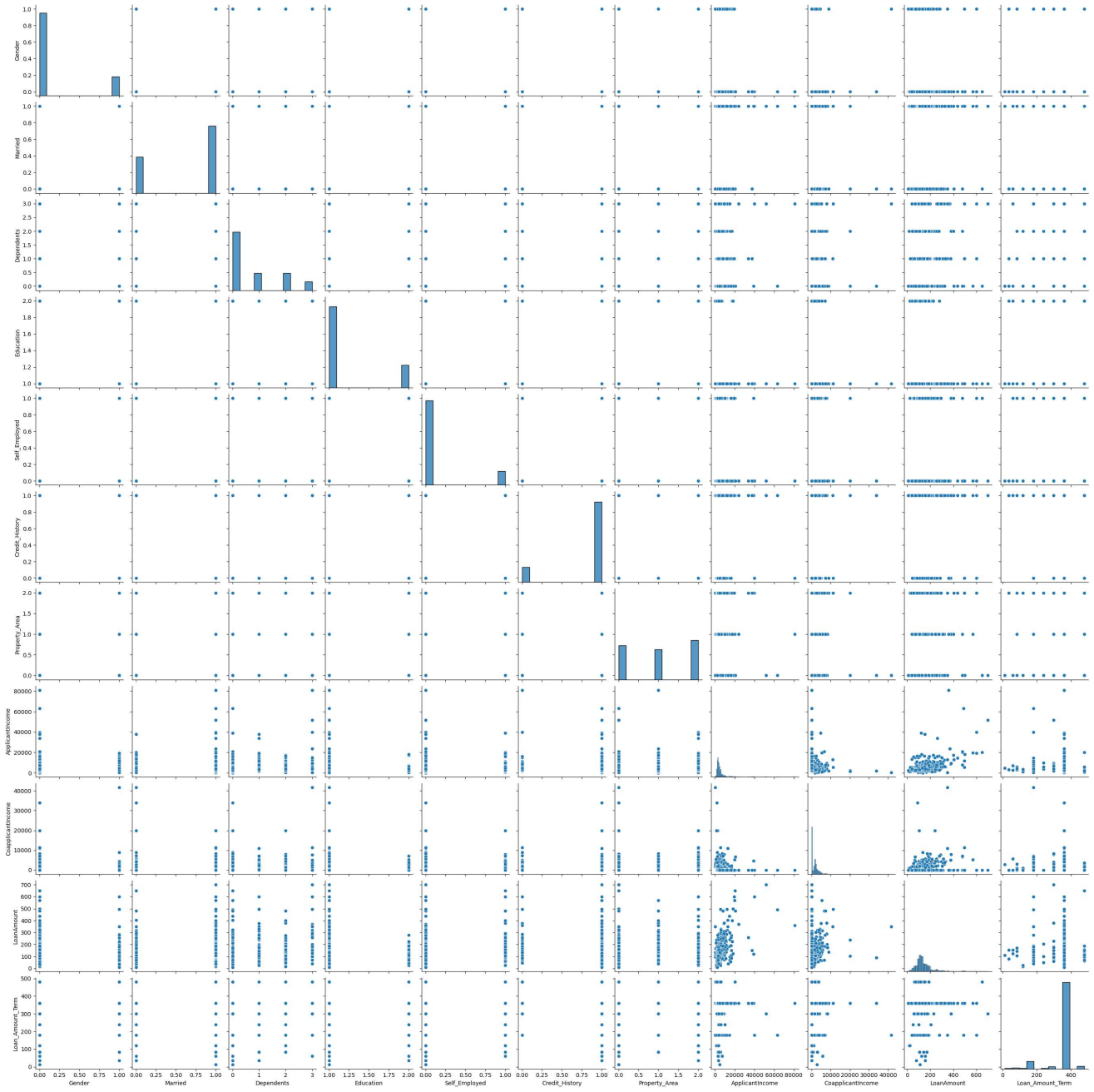
Score: 2 Marks

In [13]:

```
##---------Type the code below this line------------------##
import numpy as np
import itertools
#Co-Relation Plot -- Only works for the quantitative attributes, not for categorical attributes
quant_vars=["ApplicantIncome","CoapplicantIncome","LoanAmount","Loan_Amount_Term"]
plt.title("Co-Relation amongst the Quantitative attribute of the Data Set")
dataplot = sns.heatmap(df[quant_vars].corr(), annot=True,linewidths=6,annot_kws={'size': 7},
                       mask=np.triu(df[quant_vars].corr()))
plt.figure(figsize = (16,5))
plt.show()

# Justification of using the plot - Heatmap :
#Heatmap plots are commonly used to visualise a dataset as they generate a Easy-to-Interpret graphical format of the
#data and the unique display of the attributes and their relation amongst the attributes which is represented by
#different colours are visually differentiating and at a glance we can understand the nature of the data effectively
# (ie. patterns,relationships etc)
# Conclusion :
# In our dataset, the heatmap can be used to visualize the correlation between various features and the target
# variable (Loan_Status). It can also help to identify any strong correlations between the features themselves,
# which can be useful in feature selection and dimensionality reduction if required.


ppquant_vars=['Gender','Married','Dependents','Education','Self_Employed','Credit_History','Property_Area',"ApplicantIncome",
              "CoapplicantIncome","LoanAmount","Loan_Amount_Term"]
import seaborn as sns
sns.pairplot(data=df[ppquant_vars])

# Display the plot
plt.show()


# Justification of using the plot - Pairplot :
#Pairplots are used to visualize the pairwise relationships between variables in a dataset.
#A pairplot is a grid of scatterplots,bar plots and an extensive array of varied projections, where each scatterplot
#shows the relationship between two variables.
#The diagonal of the grid shows a histogram or a density plot for each variable. Pairplots are often used in
#exploratory data analysis to quickly identify patterns and relationships in a dataset.


# Pairplots are effective to detect the bellow mentioned properties in a given data :
# 1) Identifying Co-relations
# 2) Outlier Detection
# 3) Multivariate Analysis
# 4) Data Preparation ( Informs us regarding missing or un evenly distributed data in the Dataset)
# Conclusion : In our dataset , the pairplot can be used to explore the relationship between
# different pairs of features and the target variable . This can help in identifying any non linear
# or complex relationships that may exist in the data, which may not be visible through
# other type of visualizations

#######Overall Conclusion :
# In our dataset, the heatmap can be used to visualize the correlation between various features and the target variable
# (Loan_Status).It can also help to identify any strong correlations between the features themselves,
# which can be useful in feature selection and dimensionality reduction if required.
```



<Figure size 1600x500 with 0 Axes>



5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

In [14]:
```
##---------Type the code below this line------------------##.
'''
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History

'''
#-----Mutual Information (Information Gain)
from sklearn.metrics import mutual_info_score
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

mi=[]
dfcol = df.copy()
contigency_Features = [col for col in df.columns if col!= 'Loan_Status']
imputer_vals = IterativeImputer(random_state=42)
imputed = imputer_vals.fit_transform(df[contigency_Features])
imputedf= pd.DataFrame(imputed, columns=contigency_Features)
print("========Filter 1(Answer - 1) : Mutual Information using Information Gain========")
for i in imputedf.columns:
    print ("\nComputed Mutual Information  ",i,"and Loan_Status")
    print("The Mutual Information Scores",float(mutual_info_classif(imputedf[i].to_frame(), df["Loan_Status"],
                                                                    discrete_features=[True])))
    mi.append(float(mutual_info_classif(imputedf[i].to_frame(), df["Loan_Status"], discrete_features=[True])))
#Plotting the M.I.
index= contigency_Features
values=mi
hmdf=pd.DataFrame(values,index=index)
ax=sns.heatmap(hmdf,annot=True)
ax.set(xlabel="Mutual Information --> Features VS Target")
plt.show()


#-----Chi-Squared test

from scipy.stats import chi2_contingency
from scipy.stats import chi2
from sklearn.preprocessing import LabelEncoder



col_names = df.columns
chisqmatrix=pd.DataFrame(df,columns=col_names,index=col_names)

significance_level = 0.05

for icol in col_names:
    for jcol in col_names:
# Converting to cross tab as for CHi-square test we have
# to first convert variables into contigency table
        crosstab = pd.crosstab(df[icol],df[jcol], margins=True)
#Getting p-value and other usefull information

stat,p,dof,expected=chi2_contingency(crosstab)
print("========Filter 2(Answer - 2) Chi-Squared test========")
print("\n========Output of The Contigency Table========\n")
print(crosstab)
print("\n Degree of Freedom --> ", dof)
print("\n p-value--> ", p)

if p <= significance_level:
    print('\nREJECT NULL HYPOTHESIS')
else:
    print('\nACCEPT NULL HYPOTHESIS')
```

========Filter 1(Answer - 1) : Mutual Information using Information Gain========

Computed Mutual Information   Gender and Loan_Status
The Mutual Information Scores 0.01454935307984008

Computed Mutual Information   Married and Loan_Status
The Mutual Information Scores 0.0057549771333510556

Computed Mutual Information   Dependents and Loan_Status
The Mutual Information Scores 0.019480900101761636

Computed Mutual Information   Education and Loan_Status
The Mutual Information Scores 0.0035908300621147193

Computed Mutual Information   Self_Employed and Loan_Status
The Mutual Information Scores 0.03109583702015946

Computed Mutual Information   ApplicantIncome and Loan_Status
The Mutual Information Scores 0.5172157504577801

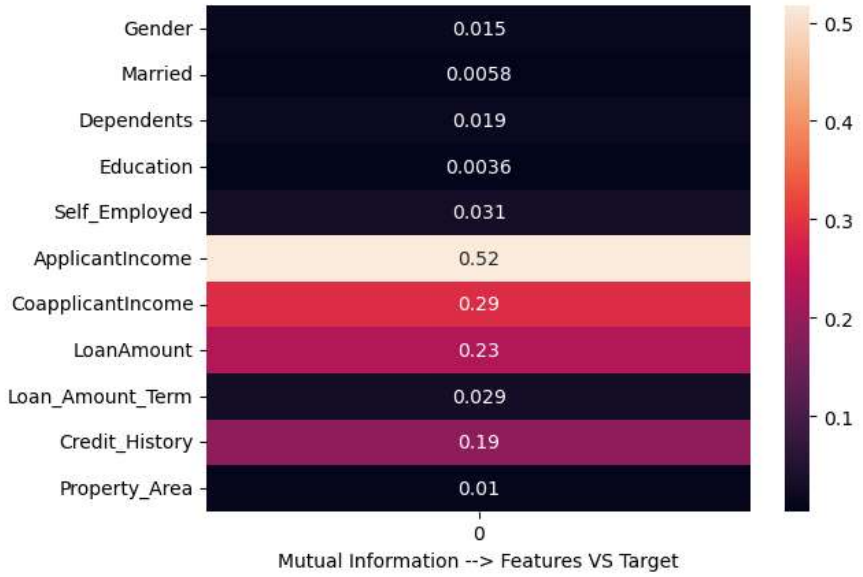
Computed Mutual Information   CoapplicantIncome and Loan_Status
The Mutual Information Scores 0.2896438136085336

Computed Mutual Information   LoanAmount and Loan_Status
The Mutual Information Scores 0.23010763653371055

Computed Mutual Information   Loan_Amount_Term and Loan_Status
The Mutual Information Scores 0.028836442394426842

Computed Mutual Information   Credit_History and Loan_Status
The Mutual Information Scores 0.1897258685428141

Computed Mutual Information   Property_Area and Loan_Status
The Mutual Information Scores 0.010202944321684893



========Filter 2(Answer - 2) Chi-Squared test========

========Output of The Contigency Table========

Loan_Status	0	1	All
Loan_Status			
0	192	0	192
1	0	422	422
All	192	422	614

 Degree of Freedom --> 4

 p-value--> 1.4459216888673977e-131

REJECT NULL HYPOTHESIS

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

```
Method 1 : Information Gain

Information gain is a measure used in decision tree learning and other machine learning algorithms to determine the
importance of a feature or attribute in classifying data. It measures the reduction in entropy or the amount of uncertainty
in the classification of data after splitting it based on a particular feature.

From the above Mutual Information results, we can make the following observations:

1) ApplicantIncome has the highest Mutual Information score of 0.5172, which indicates that it is the most important
numerical feature in predicting whether a loan application will be approved or not.

2) CoapplicantIncome also has a relatively high Mutual Information score of 0.2896, suggesting that it is also an important
numerical feature in determining loan approval status.

3) LoanAmount has a lower Mutual Information score of 0.2301 compared to ApplicantIncome and CoapplicantIncome, indicating
that it may be less important in predicting loan approval status.

4) Credit_History has a Mutual Information score of 0.1897, suggesting that it is an important attribute in determining loan
approval status, but less important than ApplicantIncome and CoapplicantIncome.

5) Self_Employed has a relatively high Mutual Information score of 0.0311 compared to other categorical features, indicating
that it may be an important attribute in predicting loan approval status.

The Mutual Information scores for Gender, Married, Dependents, Education, Loan_Amount_Term, and Property_Area are relatively
low, indicating that these attributes may not be as informative in predicting loan approval status as the numerical features
and Credit_History.<\b>

Final Conclusion : Overall, the Mutual Information scores suggest that ApplicantIncome, CoapplicantIncome, Credit_History,
and Self_Employed are important features to consider when predicting loan approval status in this dataset.




Method 2 : Chi-Squared test


Based on the Contingency table we calculated the Chi-Squared Value we see that the P-Value is significantly small
1.4169071302770516e-102 (ie 1.4169071302770516 × 10^-102)which indicated the strong evidence against the Null Hypothesis
and that there is no association between variables.



Identified top 5 significant features by evaluating each feature independently with respect to the target variable by
exploring are mentioned bellow :


Based on the computed Mutual Information scores for each feature with respect to the target variable (Loan_Status), we can
identify the top 5 significant features as follows:

1)ApplicantIncome: This feature has the highest Mutual Information score of 0.5172, indicating that it
is highly informative for predicting the Loan_Status.
2)CoapplicantIncome: This feature has the second-highest Mutual Information score of 0.2896,
suggesting that it is also highly informative for predicting the Loan_Status.
3)LoanAmount: This feature has a Mutual Information score of 0.2301, indicating
that it is moderately informative for predicting the Loan_Status.
4)Credit_History: This feature has a Mutual Information score of 0.1897,
suggesting that it is moderately informative for predicting the Loan_Status.
5)Self_Employed: This feature has a Mutual Information score of 0.0311,
indicating that it is slightly informative for predicting the Loan_Status.
```

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis
4. Anomaly detection
5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

In [15]:
```
##---------Type the code below this line------------------##
## Technique 1 :    DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split as train_test_split
from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier

trainData = imputedf.copy()
CategoryCols = ['Gender', 'Married','Education','Self_Employed', 'Property_Area', 'Credit_History', 'Dependents']
for col in trainData.columns:
    if col in CategoryCols:
        trainData[col] = trainData[col].astype(str)
        testdf[col] = testdf[col].astype(str)
Final_Attrs = ['LoanAmount', 'ApplicantIncome','Credit_History', 'CoapplicantIncome']
features = trainData[Final_Attrs]
target = df.iloc[:,-1]
XTrain, XTest, YTrain, YTest  = train_test_split(features, target, test_size=0.21, random_state=42)
PipelineData = Pipeline(steps=[('imputer', IterativeImputer()),
                               ('standardization', StandardScaler()),
                               ('model', DecisionTreeClassifier(random_state = 42))
                              ])
PipelineData.fit(XTrain, YTrain)
TrainYPred = PipelineData.predict(XTrain)



### Visualition of the Model - Descision Tree Classifier


ypredict = PipelineData.predict(XTest)


print("\n==============Performance Analysis Score of ML Technique: Decision Tree Classification==============\n")
print("Train precision_score is {}".format(precision_score(YTrain, TrainYPred)))
print("Train recall_score is {}".format(recall_score(YTrain, TrainYPred)))
print("Train accuracy_score is {}".format(accuracy_score(YTrain, TrainYPred)))
print("Train f1_score is {}".format(f1_score(YTrain, TrainYPred)))




print("\n\nTest precision_score is {}".format(precision_score(YTest, ypredict)))
print("Test recall_score is {}".format(recall_score(YTest, ypredict)))
print("Test accuracy_score is {}".format(accuracy_score(YTest, ypredict)))
print("Test f1_score is {}\n".format(f1_score(YTest, ypredict)))



## Justification of using Decision Tree Classifier:

#1)Interpretability: Decision trees are a very interpretable machine learning algorithm,
#meaning that the decisions made by the algorithm can be easily visualized and
# understood by humans. This can be useful in situations where it is
# important to understand the reasoning
#behind the algorithm's decisions, such as in loan approval decisions.

#2)Non-linear relationships: Decision trees can model non-linear relationships between the
#features and the target variable, which may be present in the Loan Prediction dataset.
#This can help to identify complex patterns and
#interactions between the features that may not be
#captured by simple linear models.

#3)Feature importance: Decision trees can also provide insight into the relative
#importance of different features in predicting the target variable.
#This can help to identify the most important predictors for loan approval,
#which can be useful for improving the accuracy and interpretability of the model.
```

==========Performance Analysis Score of ML Technique: Decision Tree Classification==========

Train precision_score is 1.0
Train recall_score is 1.0
Train accuracy_score is 1.0
Train f1_score is 1.0

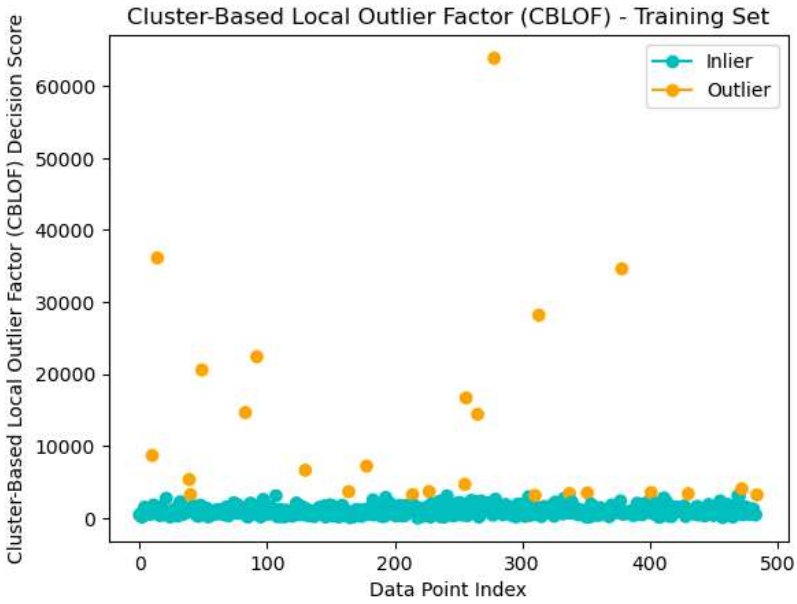
Test precision_score is 0.7560975609756098
Test recall_score is 0.7469879518072289
Test accuracy_score is 0.6821705426356589
Test f1_score is 0.7515151515151515

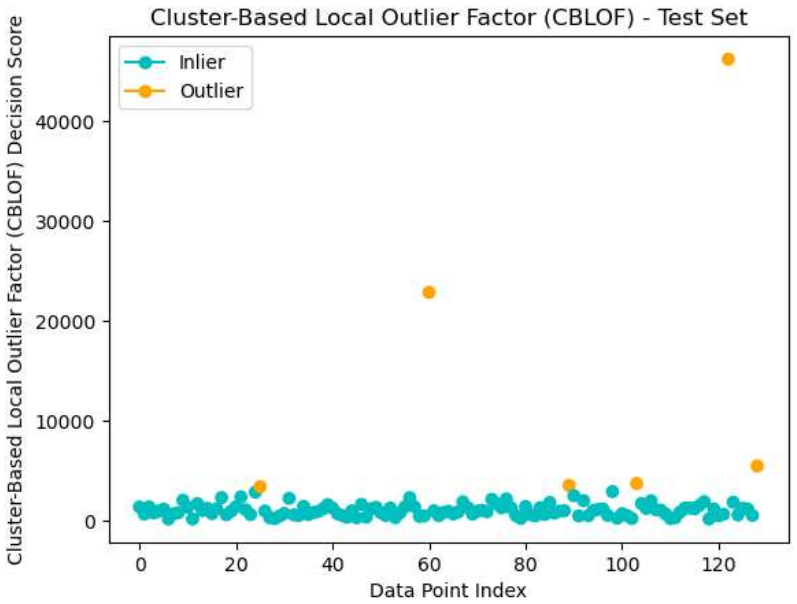
6.2 ML technique 2 + Justification

In [16]:
```
## Technique 2 :   Cluster-Based Local Outlier Factor (CBLOF) implementation
from pyod.models.cblof import CBLOF
import matplotlib
cblof = CBLOF(n_clusters=10, contamination = 0.05)
cblof.fit(XTrain)

# Training data
y_train_scores = cblof.decision_function(XTrain)
y_train_pred = cblof.predict(XTrain)

# Test data
y_test_scores = cblof.decision_function(XTest)
y_test_pred = cblof.predict(XTest) # outlier labels (0 or 1)

def count_stat(vector):
    # Because it is '0' and '1', we can run a count statistic.
    unique, counts = np.unique(vector, return_counts=True)
    return dict(zip(unique, counts))

print("The training data:", count_stat(y_train_pred))
print("The training data:", count_stat(y_test_pred))
# Threshold for the defined comtanimation rate
print("The threshold for the defined comtanimation rate:" , cblof.threshold_)


# Define label names and corresponding colors
label_names = ["Inlier", "Outlier"]
colors = ["c", "orange"]

# Plot the decision scores for the training data
plt.scatter(range(len(y_train_scores)), y_train_scores, c=y_train_pred,cmap=matplotlib.colors.ListedColormap(colors))
plt.title("Cluster-Based Local Outlier Factor (CBLOF) - Training Set")
plt.xlabel("Data Point Index")
plt.ylabel("Cluster-Based Local Outlier Factor (CBLOF) Decision Score")
plt.legend(handles=[plt.Line2D([], [], marker='o', color=colors[0], label=label_names[0]),
                    plt.Line2D([], [], marker='o', color=colors[1], label=label_names[1])])
plt.show()

# Plot the decision scores for the test data
plt.scatter(range(len(y_test_scores)), y_test_scores, c=y_test_pred,cmap=matplotlib.colors.ListedColormap(colors))
plt.title("Cluster-Based Local Outlier Factor (CBLOF) - Test Set")
plt.xlabel("Data Point Index")
plt.ylabel("Cluster-Based Local Outlier Factor (CBLOF) Decision Score")
plt.legend(handles=[plt.Line2D([], [], marker='o', color=colors[0], label=label_names[0]),
                    plt.Line2D([], [], marker='o', color=colors[1], label=label_names[1])])
plt.show()

#Justification of using CBLOF-Cluster-Based Local Outlier Factor
#The CBLOF algorithm can be a good choice for outlier detection in certain
#situations, depending on the characteristics of the dataset and the specific
#goals of the analysis. Here are some potential justifications for using CBLOF:

#1)Local density: The CBLOF algorithm takes into account the local density of
#data points, which can help it identify anomalies in   complex,high-dimensional datasets.
#This can be useful in situations where the anomalies are not easily
#identifiable by simple statistical measures or visual inspection.

#2)Unsupervised learning: CBLOF is an unsupervised machine learning algorithm,
#which means it can be used to detect outliers without requiring labelled data.
#This can be useful in situations where it is difficult or impractical to label data,
#or when the dataset is constantly changing or evolving.

#3)Clustering: CBLOF first clusters the data points into smaller groups,
#which can help it identify local anomalies within each cluster.
#This can be useful in situations where the data has
#a complex structure or is composed of multiple sub-groups or clusters.

#4)Scalability: CBLOF is relatively scalable, meaning it can
#be applied to large datasets without requiring significant computational resources.
#This can be useful in situations where the dataset is very large or complex.


#How does this algorithm help us in resolution of our problem
#The Loan Prediction dataset contains many features that may be related to each other in complex ways,
#making it difficult to identify important predictors using simple statistical or visual methods.
#CBLOF can help to identify local anomalies within clusters of similar loan applications,
#which may provide additional insight into the complex
#relationships between the features.



#Conclusion with respect to the selection of both the ML Techniques namely
#(Decision Tree Classifier,CBLOF-(Cluster-Based Local Outlier Factor)
#are mentioned bellow :
```

```
#In conclusion, using both CBLOF and Decision Tree Classifier can
#provide complementary insights into the Loan Prediction
#dataset.CBLOF can help to identify outliers and anomalies within
#clusters of similar loan applications, while Decision Tree
#Classifier can provide interpretable models that capture complex
#relationships and feature importance and hence the reason
#of our selection of the above mentioned methodologies to fetch us the
#best result for the given bussiness problem
```

```
The training data: {0: 460, 1: 25}
The training data: {0: 123, 1: 6}
The threshold for the defined comtanimation rate: 3155.3036389502304
```





7. Conclusion

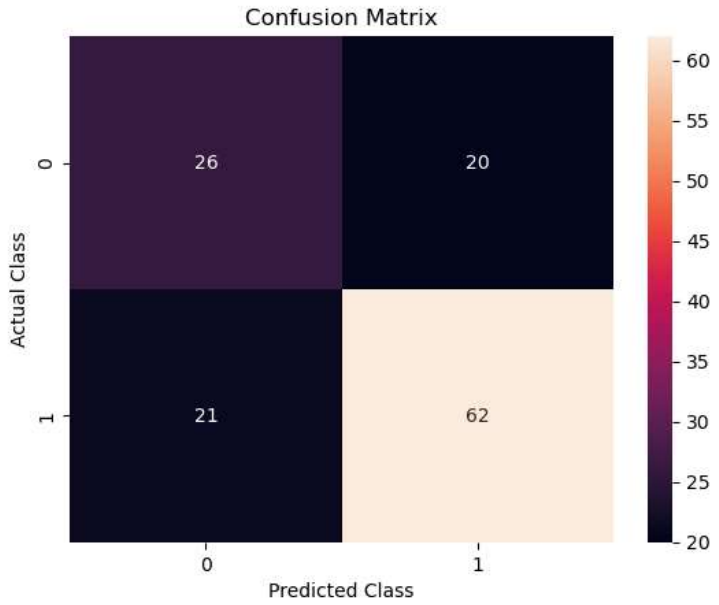
Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

In [17]:
```
##---------Type the code below this line-----------------##

# Please Refer the above section the F1 Score and the Contigency Matrix
# Please find the confusion Matrix bellow
# F1 score and Precision are mentioned there

print("\n**********************Performance of ML Technique-1: Decision Tree Classification**********************\n")
print("\n\nTest precision_score is {}".format(precision_score(YTest, ypredict)))
print("Test recall_score is {}".format(recall_score(YTest, ypredict)))
print("Test accuracy_score is {}".format(accuracy_score(YTest, ypredict)))
print("Test f1_score is {}\n".format(f1_score(YTest, ypredict)))


print("\n********************** Confusion Matrix **********************\n")
conf_mat = confusion_matrix(YTest, ypredict)
print("Confusion Matrix:\n", conf_mat)


# Plot confusion matrix as heatmap
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('Actual Class')
plt.title('Confusion Matrix')
plt.show()




#### Observation from Decision Tree Classifier
#The decision tree classifier model shows a relatively good performance on the test data. The precision score is 0.756,
#indicating that out of all the predicted positive cases, 75.6% are actually positive. The recall score is 0.747,
#indicating that out of all the actual positive cases, 74.7% were correctly identified as positive by the model.
#The accuracy score is 0.682, indicating that the model predicts the correct label for 68.2% of the cases in the test set.
#Finally, the f1 score is 0.752, which is the harmonic mean of precision and recall, providing a balanced measure of the
#model's performance.



#Final Conclusion :Overall, the decision tree classifier model shows a good balance between precision and recall indicating
#that it can effectively identify both positive and negative cases.


####Observation from Cluster-Based Local Outlier Factor (CBLOF)
#The CBLOF model was trained on two datasets. The first dataset(Training) contains 460 normal data points (labeled as 0) and
#25 anomalous data points (labeled as 1), while the second dataset(Testing) contains 123 normal data points and 6 anomalous
#datapoints.Furthermore, the model has defined a contamination rate, which is the expected percentage of anomalies in the
#dataset and the model has defined a threshold for the contamination rate, which is 3155.3036389502304.

#Final Conclusion : In conclusion,  with the application of the CBLOF model we were able to fish out the Outliers amongst
#our data effectively
```

```
**********************Performance of ML Technique-1: Decision Tree Classification**********************



Test precision_score is 0.7560975609756098
Test recall_score is 0.7469879518072289
Test accuracy_score is 0.6821705426356589
Test f1_score is 0.7515151515151515


********************** Confusion Matrix **********************

Confusion Matrix:
 [[26 20]
 [21 62]]
```



8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

----------Type the answers below this line----------

```
##---------Type the answer below this line------------------##

Business Problem and Solution :

The business problem of predicting whether a loan application will be approved or not is discussed. Depending on the
features that are available in the dataset. The model will be used to predict the approval status of loan applications.

Several steps are taken to build and evaluate the machine learning model:

Data cleaning and preprocessing: The loan prediction dataset is preprocessed by handling missing values, encoding
categorical features, and scaling numerical features.

Exploratory data analysis: Visualizations such as heatmaps and pair plots are used to explore the relationships between the
features and the target variable, and identify any potential outliers or anomalies.

Feature selection: The most significant features are identified by computing their mutual information with respect to the
target variable.

Model selection: Taking the data and the nature of the dataset into consideration we have selected the Decision tree
classifier algorithm and the CBLOF algorithm for effective resolution of our Business Problem


Model evaluation: The selected model is evaluated using metrics such as accuracy, precision, recall, and F1-score.


While working through solving the problem, some of the challenges and observations made are:

Missing values: The dataset has a significant number of missing values, particularly in the features related to income and
credit history. Several strategies were used to impute these missing values to make the data more receptive and palatable
for the Models mentioned prior


Feature Selection :  Some of the features in the dataset required transformation or combination to make them more
informative,relevant and practical for it to be in a consortium with our selected model in other words to be  relevant and
worthy to derive a effective judgment for our Business Problem


Overall, the loan prediction problem provides a good opportunity to explore various techniques in data cleaning, feature
selection, and machine learning model building. The project also highlights the importance of evaluating models using
multiple metrics and assessing their performance on unseen data.
```