Introduction to Data Science (S1-22_DSECLZG532)-ASSIGNMENT

Group No: 167

Group Member Names:

1. MOITREYA NANDI (BITS ID: 2022DA04610)
2. HITESH NEGI (BITS ID: 2022DA04593)
3. IYER SUMITRA NARAYAN (BITS ID: 2022DA04605)
4.

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--------------

1. **Business Problem:** We are going to create a Machine Learning model which will be competent to predict the capability of
a person to apply for financial loan based on the various feature attributes.

2. **Data Description:** We require a dataset containing the details of loan taker(s) which will actually be needed to train
our technically built model in order to understand the above mentioned business problem in depth so that we can test it
further on some real time scenario to gather the expected result.
The details of the data description are shown as follows:
    * Quantative Attributes (ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term)
    * Categorical Attributes (Gender, Married, Dependents, Education, Self_Employed, Credit_History, Property_Area,
Loan_Status)

3. **Data Source:** From Kaggle website (https://www.kaggle.com/), we have taken the dataset which contains the specific
feature details of user as compare to each of their loan status and all these details will be most suitable to resolve the
overall business problem by utilizing them with respect to our classified model through training and testing approach.

4. **Details of Analytical Tasks Performed:** At the initial part of analysis, we have performed data cleaning, encoding and
several visualization techniques on the existing dataset for better understanding. Also, we have used the anomally detection
algorithm to find out any possible anomally and after that, we have performed classification over our cleansed data to find
out the expected outcome.
```

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 [1]:
```
##---------Type the code below this line-----------------##
!pip install opendatasets

import opendatasets as od
import pandas as pd

od.download("https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset/download?datasetVersionNumber=1"
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/
(https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages (from opendatasets) (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from opendatasets) (4.64.1)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from opendatasets) (8.1.3)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (1.26.14)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2.25.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages (from kaggle->opendatasets) (1.15.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dist-packages (from python-slugify->kaggle->ope
ndatasets) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle->opendatasets)
(2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->kaggle->opendataset
s) (4.0.0)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds (http://bit.ly/kaggle-
creds)
Your Kaggle username: moitreyanandi
Your Kaggle Key: ··········
Downloading loan-prediction-problem-dataset.zip to ./loan-prediction-problem-dataset

100%|██████████| 12.6k/12.6k [00:00<00:00, 5.37MB/s]
```

2.2 Code for converting the above downloaded data into a dataframe

In [2]:
```
##---------Type the code below this line-----------------##
import pandas as pd
df = pd.read_csv("/content/loan-prediction-problem-dataset/train_u6lujuX_CVtuZ9i.csv")
test_df = pd.read_csv("/content/loan-prediction-problem-dataset/test_Y3wMUE5_7gLdaTN.csv")
print("\n**********************Train Data************************")
print(df)
print("\n**********************Test Data************************")
print(test_df)
```

```
**********************Train 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
```

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

In [3]:
```
##---------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 [4]:
```
##---------Type the code below this line-----------------##

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[4]:

	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--------------

---


1. **Size of the dataset:** Please find the overall size of the used dataset as mentioned below-
  * Number of Features/Attributes (column wise) => 13
  * Number of users (row wise) => 614


2. **Type of data attributes:** We can divide the overall data attributes into two parts which are as follows-
    * Quantitative Attributes => Total 4
    * Categorical Attributes => Total 8 (Excluding 'Loan_ID')

3. **Existence of Null Data throughout the dataset:**
    Total null/empty/blank field detected => 148
    (We need to clean this data value in later stages)
```

3. Data Preparation

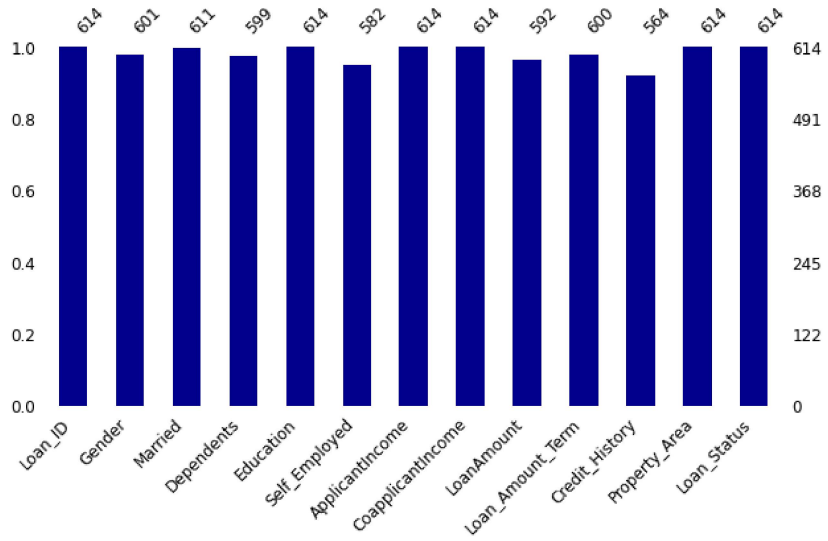
```
If input data is numerical or categorical, do 3.1, 3.2 and 3.4


If input data is text, do 3.3 and 3.4
```

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [5]:
```
##---------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")

print (" ")

# 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 [6]:
```
##---------Type the code below this line-----------------##

# To remove duplicate data
# Since there is no Duplicate data present in our database hence noting to remove. But still we can check
# the dataset if any duplicate value presents or not

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(),')')

#hence proves that there is no duplicate data present in the used dataset

# To impute or remove missing data, we can use the below code but as there are some empty fields present
# w.r.t some features in our used dataset, hence instead of removing them we are going to impute them during
# model creation as a part of one of the ML techniques we used at the later stage
# For specific purpose, the code of missing value removal can be applied as follows:

# 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()

#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
```

```
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[6]:
```
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 [7]:
```
##---------Type the code below this line------------------##

import numpy as np

# Dropping the unwanted attribute from train dataset
df.drop(columns = ['Loan_ID'], inplace = True)

# Justification: "Loan_ID" is not needed in our further analysis at the later stage(s).
# Hence, we are removing this attribute from the dataset

for col in df.columns:
  if df[col].dtypes == str or df[col].dtypes == object:
    print("Column name is {} and unique values are {}".format(col, df[col].unique()))

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 Dataset*****************************
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 Dataset******************************
test_df['Gender'] = test_df['Gender'].apply(lambda X : enc(G, X))
test_df['Married'] = test_df['Married'].apply(lambda X : enc(M, X))
test_df['Dependents'] = test_df['Dependents'].apply(lambda X : enc(D , X))
test_df['Education'] = test_df['Education'].apply(lambda X : enc(Gr , X))
test_df['Self_Employed'] = test_df['Self_Employed'].apply(lambda X : enc(se, X))
test_df['Property_Area'] = test_df['Property_Area'].apply(lambda X : enc(ar,X))
```

```
Column name is Gender and unique values are ['Male' 'Female' nan]
Column name is Married and unique values are ['No' 'Yes' nan]
Column name is Dependents and unique values are ['0' '1' '2' '3+' nan]
Column name is Education and unique values are ['Graduate' 'Not Graduate']
Column name is Self_Employed and unique values are ['No' 'Yes' nan]
Column name is Property_Area and unique values are ['Urban' 'Rural' 'Semiurban']
Column name is Loan_Status and 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 [8]:
```
##---------Type the code below this line------------------##


# Removal of Special characters, Transform into common case by applying either up-casing or down-casing, Tokenization
# and stop words filtration process 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 above four process in this case.
```

In [ ]:
```
##---------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------------------##

#***
# 1) to remove duplicate data, if present

# Answer :
# We have investigated each of the attributes for the count of duplicate data (if any) present in our used dataset
# and have amputated those data fields that doesnt meet the criteria hence are duplicate. The df.duplicated() method
# has been used to identify any duplicate value and we have used df.drop_duplicates()function to remove those duplicate values.



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


# 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 for more analysis purpose

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




#FOR TEXT DATA:
# Removal of Special characters, Transform into common case by applying either up-casing or down-casing, Tokenization
# and stop words filtration process 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 above four process in this case.

#***
```

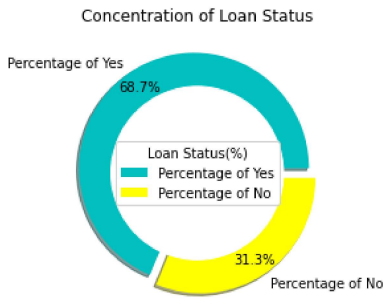
In [ ]:
```
##---------Type the code below this line------------------##
```

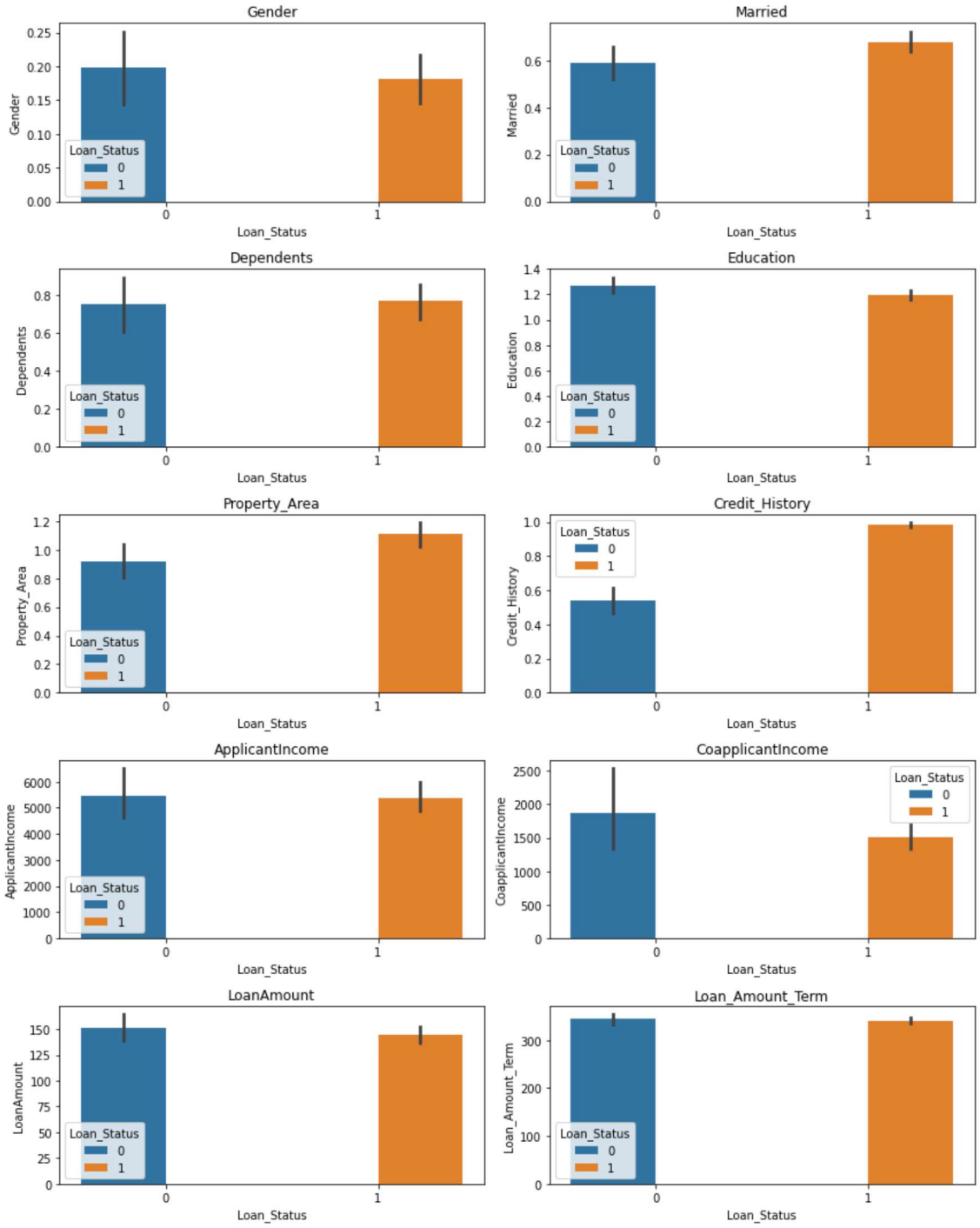
3.5 Identify the target variables.

- 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

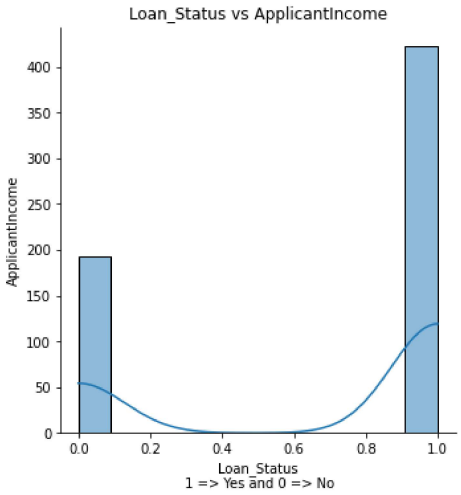
Score: 1 Mark

In [10]:
```
##---------Type the code below this line------------------##
# By observing the nature of the used dataset, identifying the significant aim to determine and predict right personel
# for the Loans we will keep the "Loan_Status" as the Target Variable because in our case, it is the decisive element
# for determining the predictive solution w.r.t our business 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','yellow']
explode = (0.05, 0.05)
plt.pie(slice, labels =activities, colors = cols, shadow = True, autopct ='%1.1f%%', pctdistance=0.85, explode=explode)
my_circle=plt.Circle( (0,0), 0.75, color='white')
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.title('Concentration of Loan Status')
plt.legend(activities, loc="center", title="Loan Status(%)")

# Report the observation of our target variable in form of donut chart representation
plt.show()

print("  ");

#------Now we need to observe the intricate details of the data w.r.t 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"

# Create a 2x5 grid of subplots
fig, axs = plt.subplots(nrows=5, ncols=2, figsize=(12,15))

# 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])
    axs[i].set_title(feature)

plt.figure(figsize=(12,6))
sns.displot(data=df, x ="Loan_Status", kde=True)
plt.title("Loan_Status vs ApplicantIncome")
plt.xlabel("Loan_Status\n 1 => Yes and 0 => No")
plt.ylabel("ApplicantIncome")

# Adjust the spacing between the subplots
fig.tight_layout()

# Report the observations of the interrelationship between features & target variable in form of bar chart representation
plt.show()
```






```
<Figure size 864x432 with 0 Axes>
```

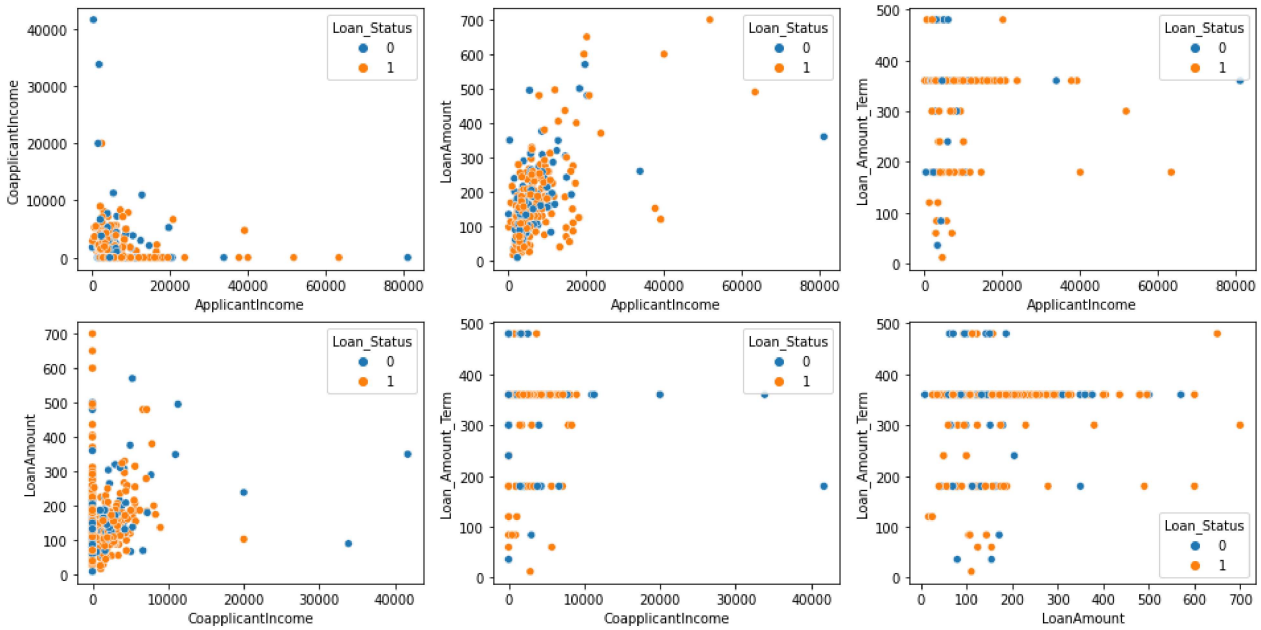


4. Data Exploration using various plots

4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

In [11]:
```
##---------Type the code below this line-------------------##
import itertools
from matplotlib import pyplot as plt
quant_vars=["ApplicantIncome","CoapplicantIncome","LoanAmount","Loan_Amount_Term"]
target="Loan_Status"

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

#Displaying scatterplot
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="Loan_Status", ax=axes[n//3, n%3])
```

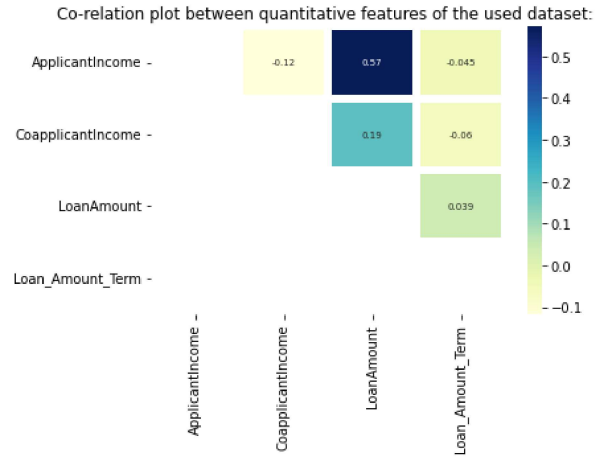


4.2 EDA using visuals

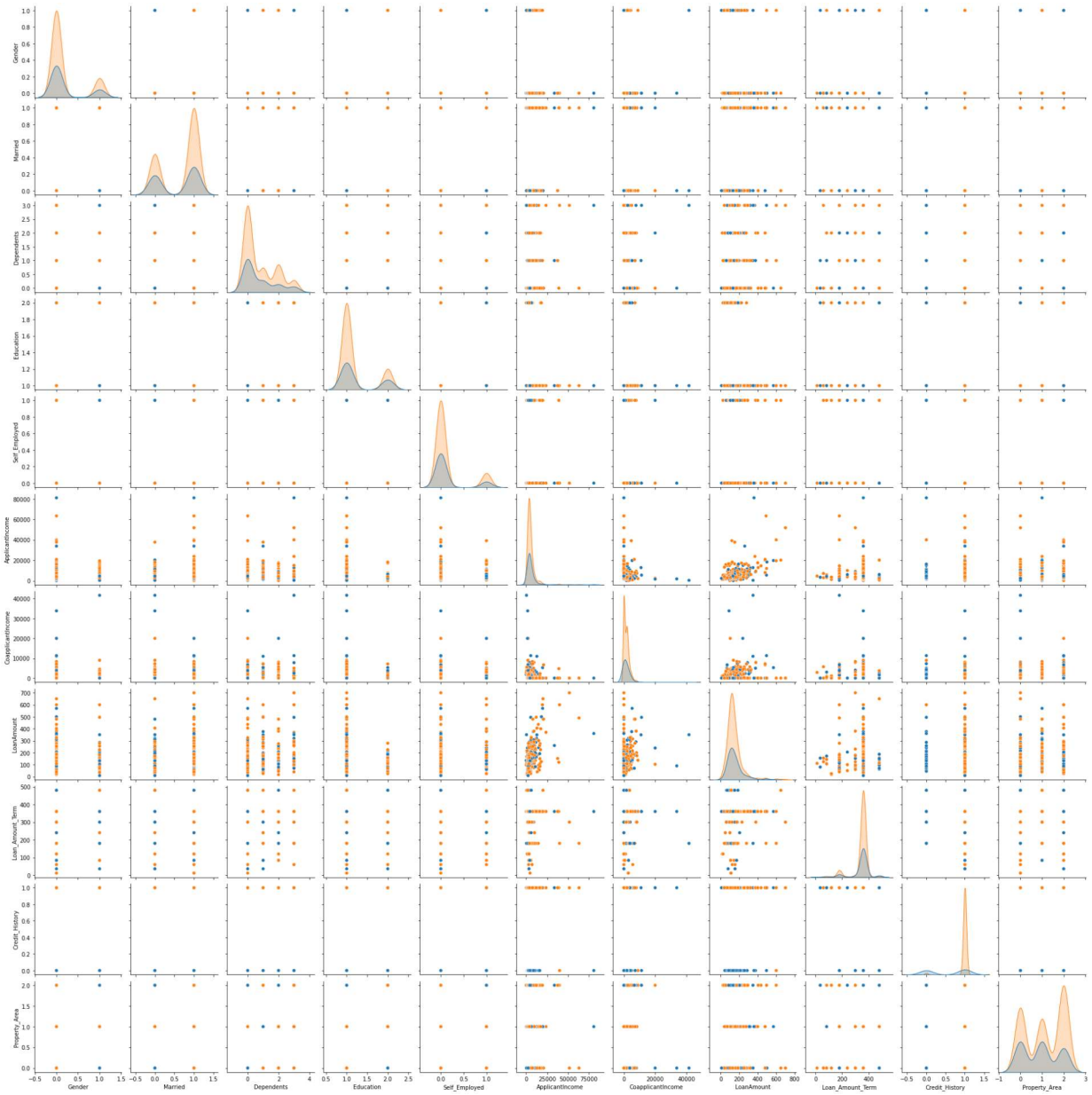
- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice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

Score: 2 Marks

In [12]:
```
##---------Type the code below this line-----------------##
import numpy as np
import seaborn as sns

#Usage of 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 plot between quantitative features of the used dataset: ")
dataplot = sns.heatmap(df[quant_vars].corr(), cmap="YlGnBu", annot=True,linewidths=6,annot_kws={'size': 7},
                       mask=np.tril(df[quant_vars].corr()))
plt.figure(figsize = (16,5))
plt.show()


# ***
# Justification of using the plot (Correlation Heatmap):
#   A correlation heatmap is a heatmap that shows a 2D correlation matrix between two discrete dimensions,
#   using colored cells to represent data from usually a monochromatic scale. The values of the first dimension
#   appear as the rows of the table while of the second dimension as a column. The color of the cell is proportional
#   to the number of measurements that match the dimensional value. This makes correlation heatmaps ideal for data
#   analysis since it makes patterns easily readable and highlights the differences and variation in the same data.
#   A correlation heatmap, like a regular heatmap, is assisted by a colorbar making data easily readable and comprehensible.
#   Hence, we are choosing this plot.
# ***




# Usage of Pairplot -- To represent the interrelationship between all the existing features of the dataset,
# taking target variable as signifier

sns.pairplot(df, hue="Loan_Status")
plt.show()


# ***
# Justification of using the plot (Pairplot):
#   Pairplots are used to visualize the pairwise relationships between variables in a dataset.
#   Basically, pairplot is needed to plot multiple pairwise bivariate distributions present in a dataset
#   as a part of exploratory data analysis. The diagonal plots are the univariate plots, and this displays the
#   relationship for the (n, 2) combination of variables in a DataFrame as a matrix of plots.


#   Pairplots are useful for many reasons:
#   a) Identifying Co-relations between the available features
#   b) Multivariate Analysis to explore the data in depth
#   c) Outlier Detection to find out the existing uncommon trend (if any) present within the data
#   d) Data Preparation (helps us to be aware of missing or unevenly distributed data throughout the dataset)
# ***
```




```
<Figure size 1152x360 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**)

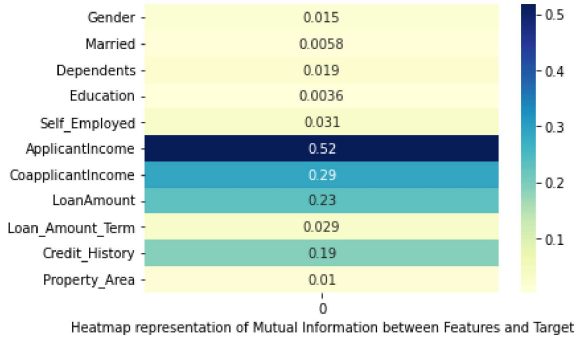
For Text data

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**)

Score: 3 Marks

In [13]:
```
##---------Type the code below this line-----------------##

#--------------1. Use of Mutual Information (Information Gain)--------------------#
#!pip install scipy

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()

cont_feats = [col for col in df.columns if col!= 'Loan_Status']

imputer = IterativeImputer(random_state=42)
imputed = imputer.fit_transform(df[cont_feats])
imputed_df= pd.DataFrame(imputed, columns=cont_feats)


print("*********1. Data Wrangling using Mutual Information (Information Gain)*********")
for i in imputed_df.columns:
    print ("\nMutual Information between ",i,"and Loan_Status")
    print("Mutual Information Scores",
          float(mutual_info_classif(imputed_df[i].to_frame(), df["Loan_Status"], discrete_features=[True])))
    mi.append(float(mutual_info_classif(imputed_df[i].to_frame(), df["Loan_Status"], discrete_features=[True])))

#Plotting the M.I.
index= cont_feats
values=mi
hmdf=pd.DataFrame(values,index=index)
ax=sns.heatmap(hmdf,annot=True,cmap="YlGnBu")
ax.set(xlabel="Heatmap representation of Mutual Information between Features and Target")
plt.show()


#---------------4. Use of Chi-Squared Test-------------------#

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



category_df = df[['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Credit_History',
                  'Property_Area', 'Loan_Status']]

col_names = category_df.columns
chisqmatrix=pd.DataFrame(category_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("\n\n *********2. Data Wrangling using Chi-Squared Test************")
print("\n ===Contigency Table for the dataset we used=== \n")
print(crosstab)
print("\n Degree of Freedom: ", dof)
print("\n Calculated p-value: ", p)

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

```
*********1. Data Wrangling using Mutual Information (Information Gain)*********

Mutual Information between  Gender and Loan_Status
Mutual Information Scores 0.01454935307984008

Mutual Information between  Married and Loan_Status
Mutual Information Scores 0.0057549771333510556

Mutual Information between  Dependents and Loan_Status
Mutual Information Scores 0.019480900101761636

Mutual Information between  Education and Loan_Status
Mutual Information Scores 0.0035908300621147193

Mutual Information between  Self_Employed and Loan_Status
Mutual Information Scores 0.03109583702015946

Mutual Information between  ApplicantIncome and Loan_Status
Mutual Information Scores 0.5172157504577801

Mutual Information between  CoapplicantIncome and Loan_Status
Mutual Information Scores 0.2896438136085336

Mutual Information between  LoanAmount and Loan_Status
Mutual Information Scores 0.23010763653371055

Mutual Information between  Loan_Amount_Term and Loan_Status
Mutual Information Scores 0.028836442394426842

Mutual Information between  Credit_History and Loan_Status
Mutual Information Scores 0.1897258685428141

Mutual Information between  Property_Area and Loan_Status
Mutual Information Scores 0.010202944321684893
```



Heatmap representation of Mutual Information between Features and Target

```
 *********2. Data Wrangling using Chi-Squared Test************

 ===Contigency Table for the dataset we used===

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

 Degree of Freedom:  4

 Calculated 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

In [14]:
```
##---------Type the code below this line-----------------##

# ***
# Method 1: Mutual Information Gain (Usage with justification & Observation)
# ---------------------------------------------------------------------------

# Mutual Information Gain is a measure used in several machine learning algorithms to identify the importance
# of a feature or attribute within the overall datasetduring data classification phase. The mutual information
# technique is basically used to measure the non-linear relations between two random variables in order to
# understand how much information can be obtained from a random variable by observing & comparing with another
# random variable. Therefore, it is closely linked with concept of entropy or the amount of uncertainty in
# classifying the overall data after splitting it based on a particular feature. That's why, here we have used
# Information Gain on this dataset to identify the most important features from the available ones so that
# we can use them further during model training phase.

# As per our used methodology, our observations from the resultant output (chart representation) are as follows:

#     a) Most significant features based on the information gain score between features & target variable are
#                "ApplicantIncome" (Mutual Information Score with target variable is 0.52),
#                "CoapplicantIncome" (Mutual Information Score with target variable is 0.29),
#                "LoanAmount" (Mutual Information Score with target variable is 0.23)
#                "Credit_History" (Mutual Information Score with target variable is 0.19)
#                "Self_Employed"(Mutual Information Score with target variable is 0.031)

#        By observing the above mentioned score values, we must include the most significant features into further
#         considerations in order to use them during data classification stage through training and testing

#     b) For rest of the features (i.e, Gender, Married, Dependents, Education, Self_Employed, Loan_Amount_Term,
#        Property_Area), the mutual information score is very less comparatively.
#        Hence, we don't need to put them for further consideration.


# Method 2: Chi-Squared test  (Usage with justification & Observation)
# ---------------------------------------------------------------------------

# Initially, we have prepared the Contingency table based on the existing values present in our target variable
# (Loan_Status).And after calculation, we can see that the P-Value is significantly small  1.4459216888673977e-131
# (i.e, 1.4459216888673977 × 10^-131) which indicates the strong evidence against the Null Hypothesis  and there is
# no association between variables.
# Since the p-value is very small, we can reject the null hypothesis and conclude that there is a significant
# association between the two categorical variables.

# ***
```

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-----------------##

import numpy as np
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


train_df = imputed_df.copy()

cat_cols = ['Gender', 'Married', 'Dependents', 'Education', "Self_Employed", 'Property_Area', 'Credit_History']

for col in train_df.columns:
  if col in cat_cols:
    train_df[col] = train_df[col].astype(str)
    test_df[col] = test_df[col].astype(str)

final_features = ['Credit_History', 'CoapplicantIncome', 'LoanAmount', 'ApplicantIncome']

features = train_df[final_features]

target = df.iloc[:,-1]


x_train, x_test, y_train, y_test  = train_test_split(features, target, test_size=0.31, random_state=42)

model_pipeline = Pipeline(steps=[('imputer', IterativeImputer()),
                                 ('standardization', StandardScaler()),
                                 ('model', DecisionTreeClassifier(random_state = 42))
                                ])

model_pipeline.fit(x_train, y_train)

Y_Tr_Pred = model_pipeline.predict(x_train)

y_predict = model_pipeline.predict(x_test)

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


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



#***********
# Justification of implementing Decision Tree Classification approach as ML technique:
# ------------------------------------------------------------------------------------

# *GENERAL EXPLANATION:
# =====================
# In Machine learning, Classification signifies two step process - Learning step and Prediction step.
# In Learning step, the model is developed based on the given Training data and in Prediction step, that
# developed model is used to predict the response for given data.

# Decision Tree which is considered to be one of the supervised learning algorithms can be used for solving
# regression and classification problems. The goal of using a Decision Tree is to create a training model that
# can use to predict the class or value of the target variable by learning simple decision rules inferred from
# prior data(training data).

# *AFFINITY TO REPRESENT REAL-LIFE SCENARIO:
# ==========================================
# As per the used dataset, here we are using the concept of categorical variable decision tree (Decision Tree which
# has a categorical target variable: "Loan_Status"in our case). As defined in the problem statement, we have to predict
# from the available data whether a customer is eligible to apply for any financial loan(Yes/No). Here, we know that the
# "ApplicantIncome" of customers is is a significant variable but the financial service company (bank) does not have the
# income details for all customers. Now, as we know this is an important variable, hence we can build a decision tree to
# predict customer's loan taking ability based on several features like "ApplicantIncome", "LoanAmount", "Credit_History"
# and various other variables.

# *QUALITATIVE IMPORTANCE AS ML TECHNIQUE:
# ========================================
# 1. Interpretation evaluation => The truthfulness of the prediction depends on the predictive performance of the decision
# tree. The explanations for short trees are very simple and general because for each split the instance falls into either
# one or the other leaf. In such case, Binary decisions are easy to understand.

# 2. Feature importance => Using decision tree, it is possible to get insights by observing relative importance of different
```

```
# features in predicting the target variable. The overall feature importance depends on how much each feature split reduces
# the variance (for Regression) and also it focuses on purity and impurity in a node (for Information Gain)

# 3. Designed as Non-Linear model => Decision trees are a prime example of non-linear models. It works by dividing the overall
# data into regions based on “if-then” type of questions which is easy to interpret the most important features as well.

#***********
```

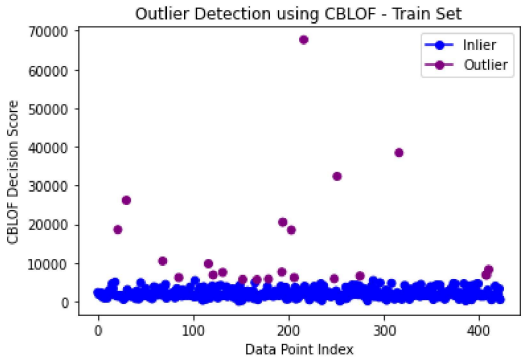
```
==============Performance Analysis Score of ML Technique: Decision Tree Classification===============

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 0.75
Test recall_score is 0.7983870967741935
Test accuracy_score is 0.6963350785340314
Test f1_score is 0.7734375
```

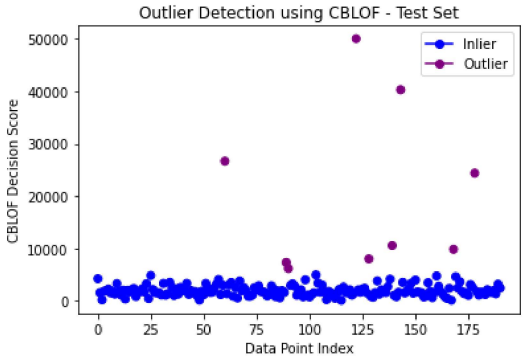
6.2 ML technique 2 + Justification

In [16]:
```
##---------Type the code below this line-----------------##
!pip install pyod

#CBLOF implementation
from pyod.models.cblof import CBLOF
import matplotlib

cblof = CBLOF(n_clusters=5, contamination = 0.05)
cblof.fit(x_train)

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

# Test data
y_test_scores = cblof.decision_function(x_test)
y_test_pred = cblof.predict(x_test) # 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("\nThe training data:", count_stat(y_train_pred))
print("\nThe testing data:", count_stat(y_test_pred))
# Threshold for the defined comtanimation rate
print("\nThe threshold for the defined comtanimation rate:" , cblof.threshold_)


# Define label names and corresponding colors
label_names = ["Inlier", "Outlier"]
colors = ["blue", "purple"]

# 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("Outlier Detection using CBLOF - Train Set")
plt.xlabel("Data Point Index")
plt.ylabel("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("Outlier Detection using CBLOF - Test Set")
plt.xlabel("Data Point Index")
plt.ylabel("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 implementing CBLOF (Cluster Based Local Outlier Factor) approach as ML technique:
# ---------------------------------------------------------------------------------------------------
#
# *GENERAL EXPLANATION:
# =====================
# In Machine Learning, anomaly detection is the process of identifying unexpected items or events in data sets,
# which differ from the norm. Anomaly detection is often applied on unlabeled data which is known as unsupervised
# anomaly detection. The two basic assumptions related to anomaly detection are as follows:

#    a)Anomalies has its own impacts as well as importance (No straightforward uniform rule to identify anomalies,
#      it can varries from case to case)
#    b)Their features differ from the normal instances significantly.

# In general, we can say that a measure for identifying the physical significance of an outlier in form of designed
# cluster is called Cluster-Based Local Outlier Factor (CBLOF). It is a meaningful process which provides importance
# to the behavioral trend(s)/pattern(s) of local data.


# *AFFINITY TO REPRESENT REAL-LIFE SCENARIO:
# ==========================================
# In digital world, a pattern can either be seen physically or it can be observed mathematically by applying algorithms
# and it can be represented using vectorfeature values. Pattern recognition can be defined as the classification of data
# based on knowledge already gained or on statistical information extracted from patterns and/or their representation.
# So, we can describe that pattern recognition involves the classification and cluster of patterns in following ways:

#   a) In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated
#   using a set of training patterns or domain knowledge. It is used in supervised learning.

#   b) Clustering generated a partition of the data which helps decision making, the specific decision-making activity
#   of interest to us. IT is used in unsupervised learning.

# Therefore, it can be stated as cluster based local outlier factor technique represents the distinct behavioral patterns
```

```
# in form of inliers and outliers  by grouping them within individual clusters from the trained dataset which is used in
# data classification stage previously. In short, it is a fast algorithm for mining outliers(if any), whose effectiveness
# is verified by the experimental results.


# *QUALITATIVE IMPORTANCE AS ML TECHNIQUE:
# ========================================
# 1. Unsupervised learning approach => CBLOF is often applied on unlabeled data, taking only the internal structure of the
# dataset into account. This challenge is known as unsupervised anomaly detection. It is very much useful and is addressed
# in many practical applications like in network intrusion detection, fraud detection as well as in the life science and
# medical domain which deals with constantly change data evolving with time.

# 2. Clustering => CBLOF represents the overall data points into several smaller groups (popularly known as clusters) and
# each of these clusters has its ownimportance based on their individual existence. In case, CBLOF can be useful to represent
# the relationship between data points in form of clustered set(s) if the data has some complex structure or complicated in
# terms of interpretability.

# 3. Identification of Local Density => Density based clustering refers the unsupervised methods that identify the distinctive
#  clusters in the data. By observing, local density of data points can be helpful to identify the outliers present in the
#  data, where it is not that easy to rectify them using some simple statistical approach or common visualization.
#
#***********
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/
(https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting pyod
  Downloading pyod-1.0.7.tar.gz (147 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 147.7/147.7 KB 3.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from pyod) (1.2.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from pyod) (3.5.3)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.8/dist-packages (from pyod) (1.22.4)
Requirement already satisfied: numba>=0.51 in /usr/local/lib/python3.8/dist-packages (from pyod) (0.56.4)
Requirement already satisfied: scipy>=1.5.1 in /usr/local/lib/python3.8/dist-packages (from pyod) (1.10.1)
Requirement already satisfied: scikit_learn>=0.20.0 in /usr/local/lib/python3.8/dist-packages (from pyod) (1.2.1)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from pyod) (1.15.0)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.8/dist-packages (from pyod) (0.13.5)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.8/dist-packages (from numba>=0.51->pyod)
(0.39.1)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.8/dist-packages (from numba>=0.51->pyod) (6.0.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from numba>=0.51->pyod) (57.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit_learn>=0.20.0->pyo
d) (3.1.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (23.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (4.38.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (1.4.4)
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib->pyod) (8.4.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.8/dist-packages (from statsmodels->pyod) (0.5.3)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.8/dist-packages (from statsmodels->pyod) (1.3.5)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=0.25->statsmodels->pyod)
(2022.7.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from importlib-metadata->numba>=0.51->pyo
d) (3.15.0)
Building wheels for collected packages: pyod
  Building wheel for pyod (setup.py) ... done
  Created wheel for pyod: filename=pyod-1.0.7-py3-none-any.whl size=181101 sha256=b4dd576d4a794eefcb2ddf9b1f66ca74c15e5029ddd3
f0e3cd28a1fd5ba18e0b
  Stored in directory: /root/.cache/pip/wheels/f7/e2/c1/1c7fd8b261e72411f6509afb429c84532e40ddcd96074473f4
Successfully built pyod
Installing collected packages: pyod
Successfully installed pyod-1.0.7

The training data: {0: 401, 1: 22}

The testing data: {0: 182, 1: 9}

The threshold for the defined comtanimation rate: 5633.212969372864
```





7. Conclusion

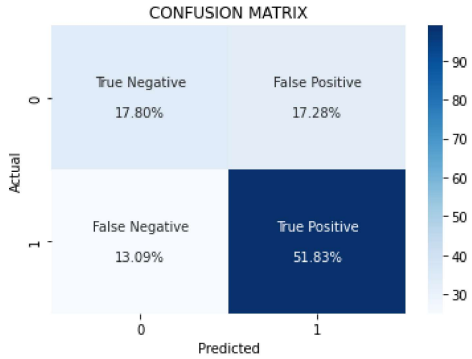
Compare the performance of the ML techniques used.

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-----------------##

print("\n***************Comparison Based Performance Analysis using Decision Tree Classification***************\n")
print("Train precision_score is {}".format(precision_score(y_train, Y_Tr_Pred)))
print("Train recall_score is {}".format(recall_score(y_train, Y_Tr_Pred)))
print("Train accuracy_score is {}".format(accuracy_score(y_train, Y_Tr_Pred)))
print("Train f1_score is {}".format(f1_score(y_train, Y_Tr_Pred)))


print("\n\nTest precision_score is {}".format(precision_score(y_test, y_predict)))
print("Test recall_score is {}".format(recall_score(y_test, y_predict)))
print("Test accuracy_score is {}".format(accuracy_score(y_test, y_predict)))
print("Test f1_score is {}\n".format(f1_score(y_test, y_predict)))
print ("Details of Classification Report from the consequent outcomes: \n\n", classification_report(y_test, y_predict))


conf_mat = confusion_matrix(y_test, y_predict)

# Plot confusion matrix as heatmap
group_names = ['True Negative','False Positive','False Negative','True Positive']
group_percentages = ["{0:.2%}".format(value) for value in conf_mat.flatten()/np.sum(conf_mat)]
labels = [f"{v1}\n\n{v2}" for v1, v2 in zip(group_names,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(conf_mat, annot=labels, fmt='', cmap='Blues')

plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('CONFUSION MATRIX')
plt.show()


# ******************OBSERVATION FROM DECISITION TREE CLASSIFIER******************
#
# a) The decision tree classifier model shows a relatively good performance on the test data. The precision score is 0.75,
# indicating that out of all the predicted positive cases, 75% are actually positive.

# b) The recall score is 0.7983870967741935, which indicates that 79.84% are correctly identified as positive by the model
# out of all the actual positive cases.

# c) The accuracy score is 0.6963350785340314, means that the model can predict the correct label for 69.64% of the cases
# in the test set.

# d) Finally, the f1 score is 0.7734375, which is the harmonic mean of precision and recall, providing a balanced measure
# of performance by the model.

# 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 throughout the data classification phase.



# ******************OBSERVATION FROM Cluster-Based Local Outlier Factor (CBLOF)******************
#
# a) The CBLOF model was trained on two datasets. The Training dataset contains 401 normal data points(labeled as 0) and 22
# anomalous data points(labeled as 1), while the Testing dataset contains 182 normal data points(labeled as 0) and 9 anomalous
# data points(labeled as 1).

# b) Besides, 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 5633.212969372864

# Final Conclusion: From the overall analysis, we can propose that the data is segregated after using the CBLOF model and the
# possible outliers are identified successfully as well as effectively.
```

```
***************Comparison Based Performance Analysis using 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 0.75
Test recall_score is 0.7983870967741935
Test accuracy_score is 0.6963350785340314
Test f1_score is 0.7734375

Details of Classification Report from the consequent outcomes:

              precision    recall  f1-score   support

           0       0.58      0.51      0.54        67
           1       0.75      0.80      0.77       124

    accuracy                           0.70       191
   macro avg       0.66      0.65      0.66       191
weighted avg       0.69      0.70      0.69       191
```



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----------

The overall analysis report with respect to the above mentioned business problem are described in following recognizable way:

The business problem is described as whether a loan application of customer will be approved or not based on several incidental features. To solve this problem, a machine learning model is proposed and that would be trained on a dataset containing detailed information about the several personal & economical traits of different users and their individual approval status. After that, the model would be used to predict the validity of loan application depending on the given approval status.

Several steps which are associated with the overall analysis process are given below :-

- **Data Cleaning and Preprocessing:** The loan prediction(training) dataset is preprocessed by handling missing values, encoding categorical features, and scaling numerical features. Only, one attribute (Loan_ID) of users is dropped from the dataset as it would be of no use at the later stage.
- **Exploratory Data Analysis:** Different visualization techniques such as graphical chart representations, heatmaps and pair plots are used to explore the relationships between the individual features and also with the target variable to understand the potential importance and also to identify any existing outliers or anomalies present in the data.
- **Feature Selection:** The most significant features are identified by determining the individual mutual information score with respect to the target variable. These valuable features are used at the further analysis stages.
- **Model Selection:** By observing the hidden pattern of data as well as the nature of the dataset, we have selected the Decision tree classifier algorithm and the CBLOF algorithm to find out the effective solution with respect to our Business Problem
- **Model Evaluation:** The selected model is evaluated using different metrics such as accuracy, precision, recall, and F1-score. The classification report and the corresponding confusion matrix are also represented as per the calculated outcomes.

During analysis, we face some of the challenges and specific observations which is as follows:

- **Missing values:** As per our observation, the dataset has a significant number of missing values present, particularly in the features related to income and credit history. Hence, several techniques are used to impute these missing values to make the data more receptive as well as interpretable for the models used at data classification and anomaly detection phases respectively.

Overall, the loan prediction problem has provided a good opportunity to explore various techniques in terms of data cleansing, feature selection, and usage of different machine learning model. This project highlighted the visual representation of valuable data from a source of raw information and also describes the importance of model evaluation using multiple metrics and estimating their qualitative performance on unseen data effectively.

##NOTE All Late Submissions will incur a penalty of -2 marks. Do ensure on time submission to avoid penalty.

Good Luck!!!