PERSONAL LOAN

PROJECT REPORT ON

LOAN APPROVAL PREDICTION USING MACHINE LEARNING

A project report submitted in partial submission of the requirements

For the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. Neeraj kumar Achari	206F1A0543
M. Chandu	206F1A0523
K.Gowri Venkata Sai Ketan	206F1A0542
K.Bhusan Kumar	206F1A0520
R. Gunasekhar	206F1A0423

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAI GANAPATHI ENGINEERING COLLEGE

(Approved by AICTE, Recognized of A.P Affiliated to JNTU Kakinada)
Gidijala (V), Anandapuram (M), Visakhapatnam (Dist.) 531173.

Under the Esteemed Guidance of

Mr. A. Srikar, M.TECH

Assistant professor

SAI GANAPATHI ENGINEERING COLLEGE

(Approved by AICTE, Affiliated to JNTU Kakinada)

Gidijala(V), Anandapuram(M), Visakhapatnam(Dist.) – 531173

BONAFIDE CERTIFICATE

This is to certify the project entitled "**LOAN APPROVAL PREDICTION USING MACHINE LEARNING**" that is being submitted by K. Neeraj Kumar Achari(**206F1A0543**), M.Chandu(**206F1A0523**), K.Gowri Venkata Sai Ketan (**206F1A0542**),K.BhusanKumar(**206F1A0520)** and R.Gunasekhar(**206F1A0423**) in partial fulfilment of the requirements for the award of the Degree of **Bachelor of Technology in "Computer Science and Engineering"** in the academic year 2023-2024 to **SAI GANAPATHI ENGINEERING COLLEGE** affiliated to **Jawaharlal Nehru Technological University Kakinada,** is a record of bonafide work carried out by them under my guidance and supervision.

DECLARATION

We declare that the project entitled "**LOAN APPROVAL PREDICTION USING MACHINE LEARNING**" is a bonafide word carried put by us and not been submitted to any other University or college for the award of any degree.

This Results embodied in this project have not been submitted to any other university or institute for the award of any degree or diploma.

K. Neeraj kumar Achari	206F1A0543
M. Chandu	206F1A0523
K.Gowri Venkata Sai Ketan	206F1A0542
K.Bhusan Kumar	206F1A0520
R. Gunasekhar	206F1A0423

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whoseconstant guidance and encouragement crown all the efforts with success.

First and foremost, we would like to thank my project guide **Mr. A. SRIKAR(ASST. PROFESSOR)**, Department of CSE, for giving me an opportunity to workon this challenging topic and providing me guidance. His encouragement, support and suggestions are most valuable for the successful completion of my project and course.

I feel elated to extend my floral gratitude to **Mr.K.Mallikharjun(Head of Department)**, Dept. of Computer Science and Engineering, for his encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of doing the thesis and for providing me all the required facilities.

I would like to take this opportunity to express my profound sense of gratitude to revered **Principal PRADEEP VARMA** for giving me the opportunity of doing thesis and for providing me all the required facilities.

I also take this opportunity to express my heartfelt thanks to the teaching andnon-teaching staff of the department, for their perspective comments and suggestions.

CONTENTS

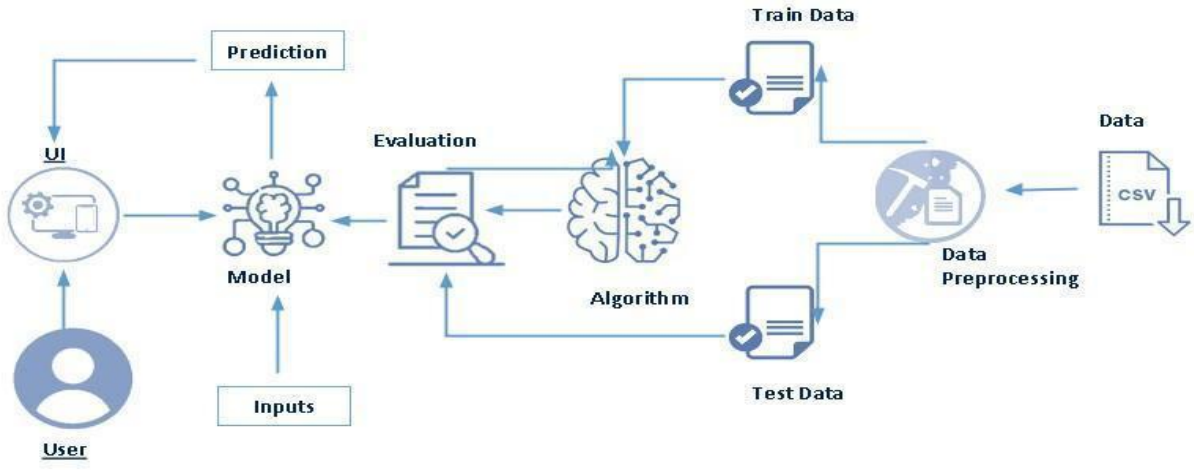
❖ INTRODUCTION

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more.They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's credit worthiness. To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

Technical Architecture:



Project Flow

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.

- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation

- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis

- Model Building
  - Training the model in multiple algorithms
  - Testing the model

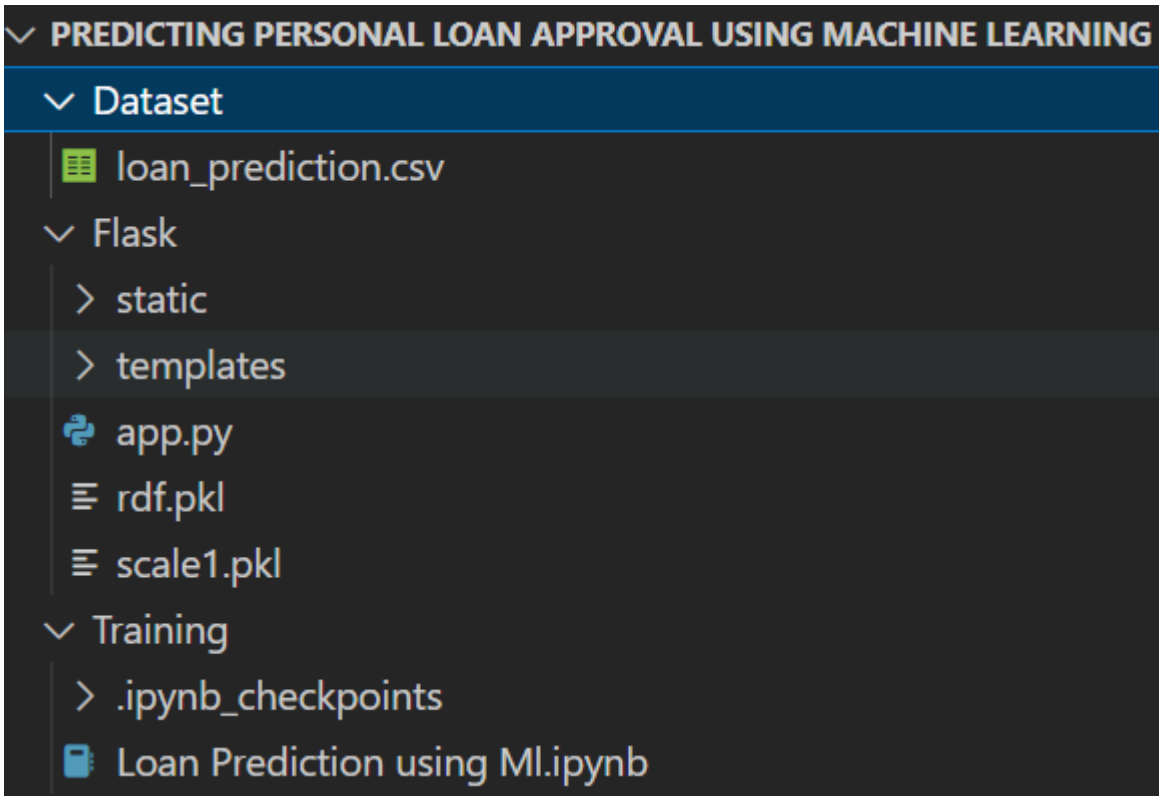
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning

- Model Deployment
  - Save the best model
  - Integrate with Web Framework

- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Project Structure

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- rdf.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file

Define Problem / Problem Understanding

➢ Specify the business problem

A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more. They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

A personal loan is a type of unsecured loan that can be used for a variety of expenses such as home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's credit worthiness. To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

➢ Business Requirements

The business requirements for a machine learning model to predict personal loan approval include the ability to accurately predict loan approval based on applicant information, Minimise the number of false positives (approved loans that default) and false negatives (rejected loans that would have been successful).Provide an explanation for the model's decision, to comply with regulations and improve transparency.

➢ Literature survey

As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Machine Learning (ML), as a typical method for information investigation, has gotten more consideration increasingly. Individuals of various businesses are utilising ML calculations to take care of the issues dependent on their industry information. Banks are facing a significant problem in the approval of the loan. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high.Most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications.There are various algorithms that have been used with varying levels of success. Logistic regression, decision tree, random forest, and neural networks have all been used and have been able to accurately predict loan defaults. Commonly used features in these studies include credit score, income, and employment history, sometimes also other features like age, occupation, and education level.

➢ Social or Business Impact

Social Impact:- Personal loans can stimulate economic growth by providing individuals with the funds they need to make major purchases, start businesses, or invest in their education

Business Model/Impact:- Personal loan providers may charge fees for services such as loan origination, processing, and late payments.Advertising the brand awareness and marketing to reach out to potential borrowers to generate revenue.

❖ DATA COLLECTION AND PREPARATION

➢ Data Collection

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

- Inaccurate data. The collected data could be unrelated to the problem statement.
- Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
- Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
- Data bias. Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove.

Data collection

```
#importing libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
#importing the data set
loan_df=pd.read_csv("trainlap.csv")
```

➢ Data Preparation

Data preparation is the later stage of the ML lifecycle. Firstly, the data is collected from various sources, and later garbage data is cleaned and transformed into real-time machine learning projects to uncover insights or make predictions. Machine learning also helps to find patterns in data to make accurate predictions and construct the data sets and transform the data correctly. In this topic, **"Data Preparation in Machine Learning,"** we will discuss various steps for data preparation in machine learning, data preparation steps, data pre-processing, data splitting, etc. So, let's start with a quick introduction to data preparation in Machine Learning.

Data preparation is defined as a gathering, combining, cleaning, and transforming raw data to make accurate predictions in Machine learning projects.

Data preparation is also known as data "pre-processing," "data wrangling," "data cleaning," "data pre-processing," and "feature engineering." It is the later stage of the machine learning lifecycle, which comes after data collection.

Data preparation is particular to data, the objectives of the projects, and the algorithms that will be used in data modeling techniques.

Prerequisites for Data Preparation

Everyone must explore a few essential tasks when working with data in the data preparation step. These are as follows:

- **Data cleaning:** This task includes the identification of errors and making corrections or improvements to those errors.

- **Feature Selection:** We need to identify the most important or relevant input data variables for the model.

- **Data Transforms:** Data transformation involves converting raw data into a well-suitable format for the model.

- **Feature Engineering:** Feature engineering involves deriving new variables from the available dataset.

- **Dimensionality Reduction:** The dimensionality reduction process involves converting higher dimensions into lower dimension features without changing the information.

Data Preparation is the process of cleaning and transforming raw data to make predictions accurately through using ML algorithms. Although data preparation is considered the most complicated stage in ML, it reduces process complexity later in real-time projects. Various issues have been reported during the data preparation step in machine learning as follows:

- **Missing data:** Missing data or incomplete records is a prevalent issue found in most datasets. Instead of appropriate data, sometimes records contain empty cells, values (e.g., NULL or N/A), or a specific character, such as a question mark, etc.

- **Outliers or Anomalies:** ML algorithms are sensitive to the range and distribution of values when data comes from unknown sources. These values can spoil the entire machine learning training system and the performance of the model. Hence, it is essential to detect these outliers or anomalies through techniques such as visualization technique.

- **Unstructured data format:** Data comes from various sources and needs to be extracted into a different format. Hence, before deploying an ML project, always consult with domain experts or import data from known sources.

- **Limited Features:** Whenever data comes from a single source, it contains limited features, so it is necessary to import data from various sources for feature enrichment or build multiple features in datasets.

- **Understanding feature engineering:** Features engineering helps develop additional content in the ML models, increasing model performance and accuracy in predictions.

Each machine learning project requires a specific data format. To do so, datasets need to be prepared well before applying it to the projects. Sometimes, data in data sets have missing or incomplete information, which leads to less accurate or incorrect predictions. Further, sometimes data sets are clean but not adequately shaped, such as aggregated or pivoted, and some have less business context. Hence, after collecting data from various data sources, data preparation needs to transform raw data. Below are a few significant advantages of data preparation in machine learning as follows:

- It helps to provide reliable prediction outcomes in various analytics operations.
- It helps identify data issues or errors and significantly reduces the chances of errors.
- It increases decision-making capability.
- It reduces overall project cost (data management and analytic cost).
- It helps to remove duplicate content to make it worthwhile for different applications.
- It increases model performance.

Data preparation is one of the critical steps in the machine learning project building process, and it must be done in particular series of steps which includes different tasks. There are some essential steps of the data preparation process in machine learning suggested by different ML experts and professionals as follows:

1. **Understand the problem:** This is one of the essential steps of data preparation for a machine learning model in which we need to understand the actual problem and try to solve it. To build a better model, we must have detailed information on all issues, such as what to do and how to do it. It is also very much effective to retain clients without wasting much effort.

2. **Data collection:** Data collection is probably the most typical step in the data preparation process, where data scientistsneed to collect data from various potential sources. These data sources may be either within enterprise or third parties vendors. Data collection is beneficial to reduce and mitigate biasing in the ML model; hence before collecting data, always analyze it and also ensure that the data set was collected from diverse people, geographical areas, and perspectives.
   There are some common problems that can be addressed using data collection as follows:
   - It is helpful to determine the relevant attributes in the string for the .csv file format.
   - It is used to parse highly nested data structures files such as XML or JSON into tabular form.
   - It is significant in easier scanning and pattern detection in data sets.
   - Data collection is a practical step in machine learning to find relevant data from external repositories.
3. **Profiling and Data Exploration:** After analyzing and collecting data from various data sources, it's time to explore data such as trends, outliers, exceptions, incorrect, inconsistent, missing, or skewed information, etc. Although source data will provide all model findings, it does not contain unseen biases. Data exploration helps to determine problems such as collinearity, which means a situation when the Standardization of data sets and other data transformations are necessary.
4. **Data Cleaning and Validation:** Data cleaning and validation techniques help determine and solve inconsistencies, outliers, anomalies, incomplete data, etc. Clean data helps to find valuable patterns and information in data and ignoresirrelevant data in the datasets. It is very much essential to build high-quality models, and missing or incomplete data is one of the best examples of poor data. Since missing data always reduces prediction accuracy and performance of the model, data must be cleaned and validated through various imputation tools to fill incomplete fields with statistically relevant substitutes.
5. **Data Formatting:** After cleaning and validating data, the following approach is to ensure that the data is correctly formatted or not. If data is formatted incorrectly, it will help build a high-quality model. Since data comes from various sources or is sometimes updated manually,

there are high chances of discrepancies in the data format. For example, if you have collected data from two sources, one source has updated the product's price to USD10.50, and the other has updated the same value to $10.50. Similarly, there may be anomalies in their spelling, abbreviation, etc. This type of data formation leads to incorrect predictions. To reduce these errors, you must format your data inconsistent manner by using some input formatting protocols.

6. **Improve data quality:** Quality is one of the essential parameters in building high-quality models. Quality data helps to reduce errors, missing data, extreme values, and outliers in the datasets. We can understand it with an example such, In one dataset, columns have First Name and Last NAME, and another dataset has Column named as a customer that combines First and Last Name. Then in such cases, intelligent ML algorithms must have the ability to match these columns and join the dataset for a singular view of the customer.
7. **Feature engineering and selection:** Feature engineering is defined as the study of selecting, manipulating, and transforming raw data into valuable features or most relevant variables in supervised machine learning. Feature engineering enables you to build an enhanced predictive model with accurate predictions.
   For example, data can be spitted into various parts to capture more specific information, such as analyzingmarketing performance by the day of the week, not only the month or year. In this situation, segregating the day as a separate categorical value from the data (e.g., "Mon; 07.12.2021") may provide the algorithm with more relevant information. There are various feature engineering techniques used in machine learning as follows:
   - **Imputation:** Feature imputation is the technique to fill incomplete fields in the datasets. It is essential because most machine learning models don't work when there are missing data in the dataset. Although, the missing values problem can be reduced by using techniques such as single value imputation, multiple value imputation, K-Nearest neighbor, deleting the row, etc.
   - **Encoding:** Feature encoding is defined as the method to convert string values into numeric form. This is important as all ML models require all values in numeric format. Feature encoding includes label encoding and One Hot Encoding (also known as get_dummies).

Similarly, feature engineering also includes handling outliers, log transform, scaling, normalization, Standardization, etc.

8. **Splittingdata:**
   After feature engineering and selection, the last step is to split your data into two different sets (training and evaluation sets). Further, always select non-overlapping subsets of your data for the training and evaluation sets to ensure proper testing.

```
In [5]: #it display information about data set
        #it shows default by first and last 5 rows
        print(loan_df)
```

```
      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]
```

Given above is data set

In [6]:
```
#it shows the first 5 rows by default
loan_df.head()
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [7]:
```
#it shows the last 5 rows by default
loan_df.tail()
```

Out[7]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Stat
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	

Head()-it show first 5 rows of data set

Tail()-it show last 5 rows of data set

Describe()-It gives the stastical measures of data

Info()-it gives information about data

In [8]:
```
#it gives stastical measures of a data set
loan_df.describe()
```

Out[8]:

	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

In [9]:
```
#it shows the information of dataset with total rows and columns and shows the data types of each column and memory usage
loan_df.info()
```

```
<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
```

❖ EXPLORATORY DATA ANALYSIS

➢ UNIVARIATE ANALYSIS

Analysing one variable independently in the given data set

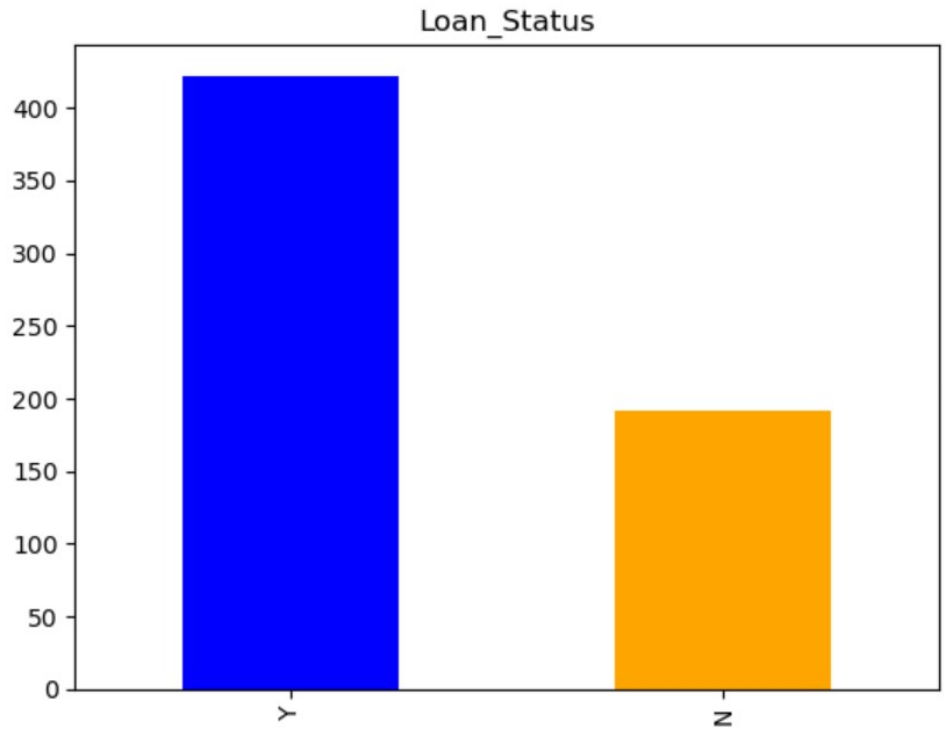
```
[14]
import seaborn as sns
loan_df['Loan_Status'].value_counts()
```

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

```
[15]
loan_df['Loan_Status'].value_counts()
loan_df['Loan_Status'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Loan_Status')
```

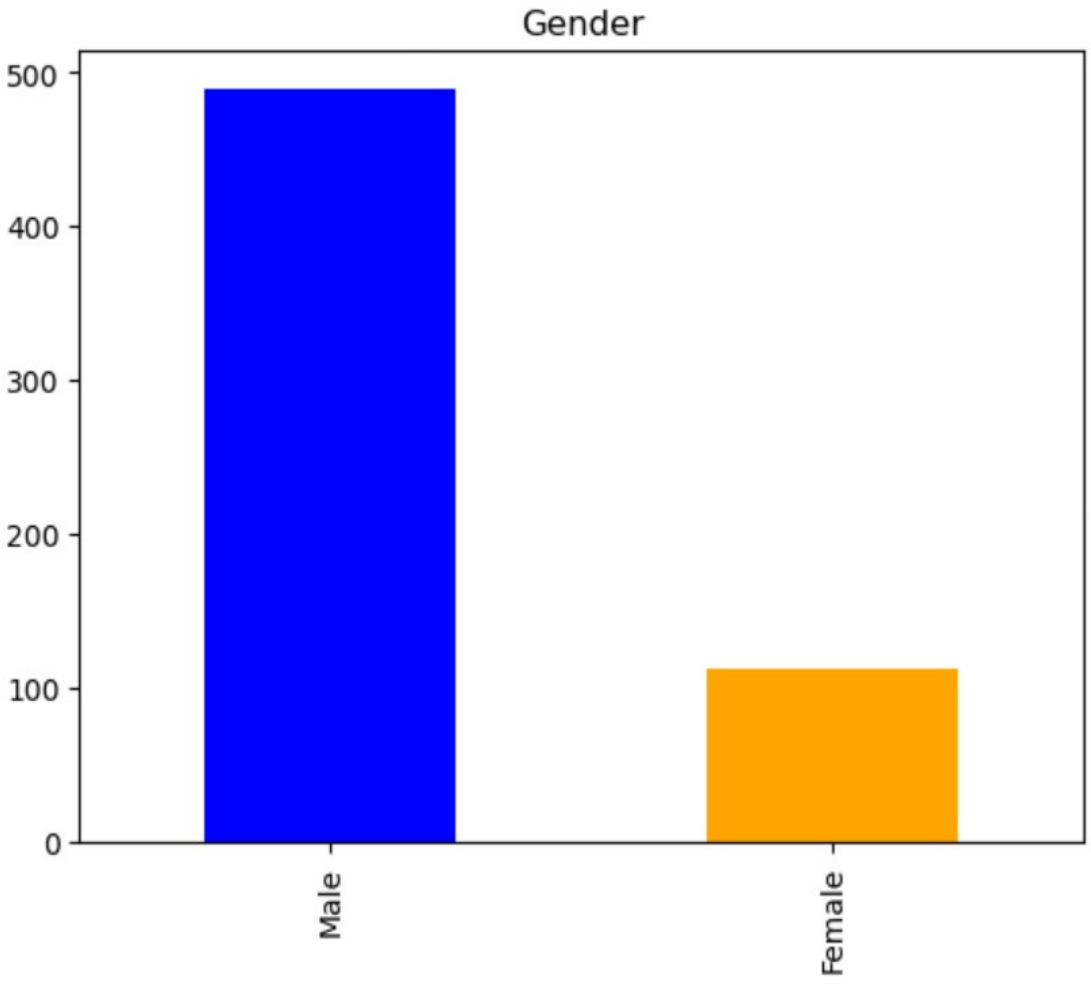
```
Text(0.5, 1.0, 'Loan_Status')
```

Text(0.5, 1.0, 'Loan_Status')



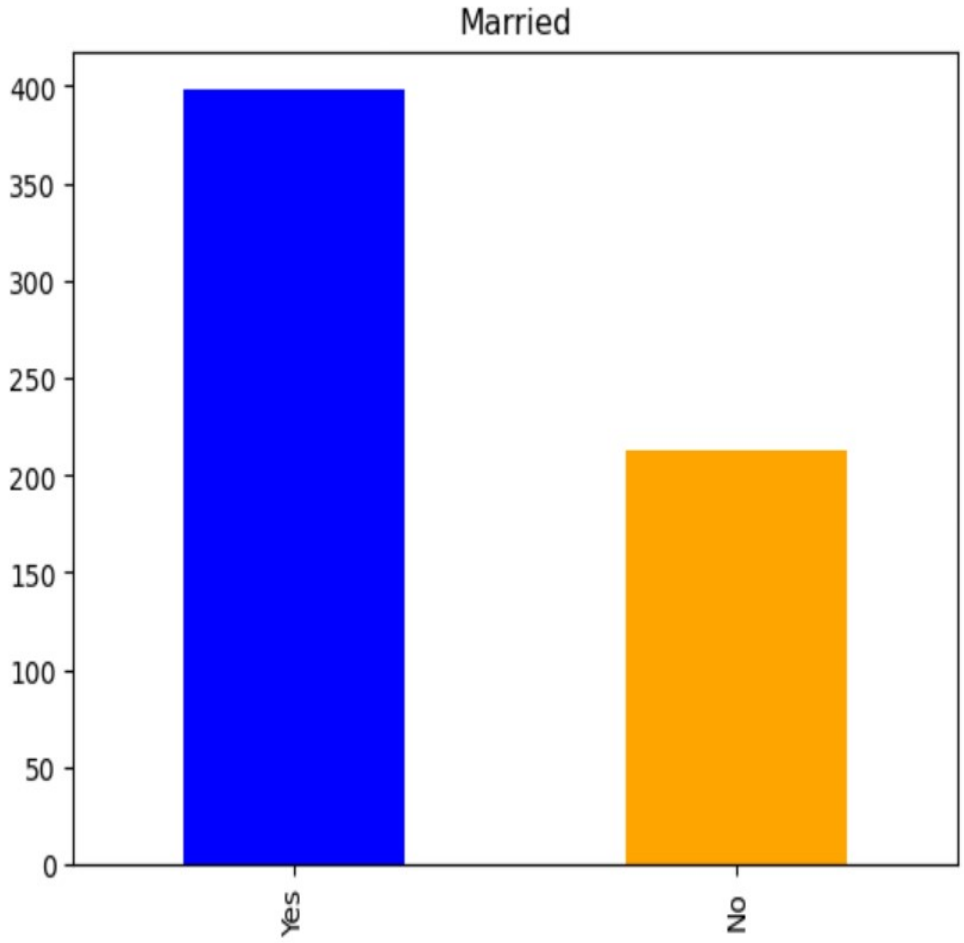
```
loan_df['Gender'].value_counts()
loan_df['Gender'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Gender')
```

```
Text(0.5, 1.0, 'Gender')
```



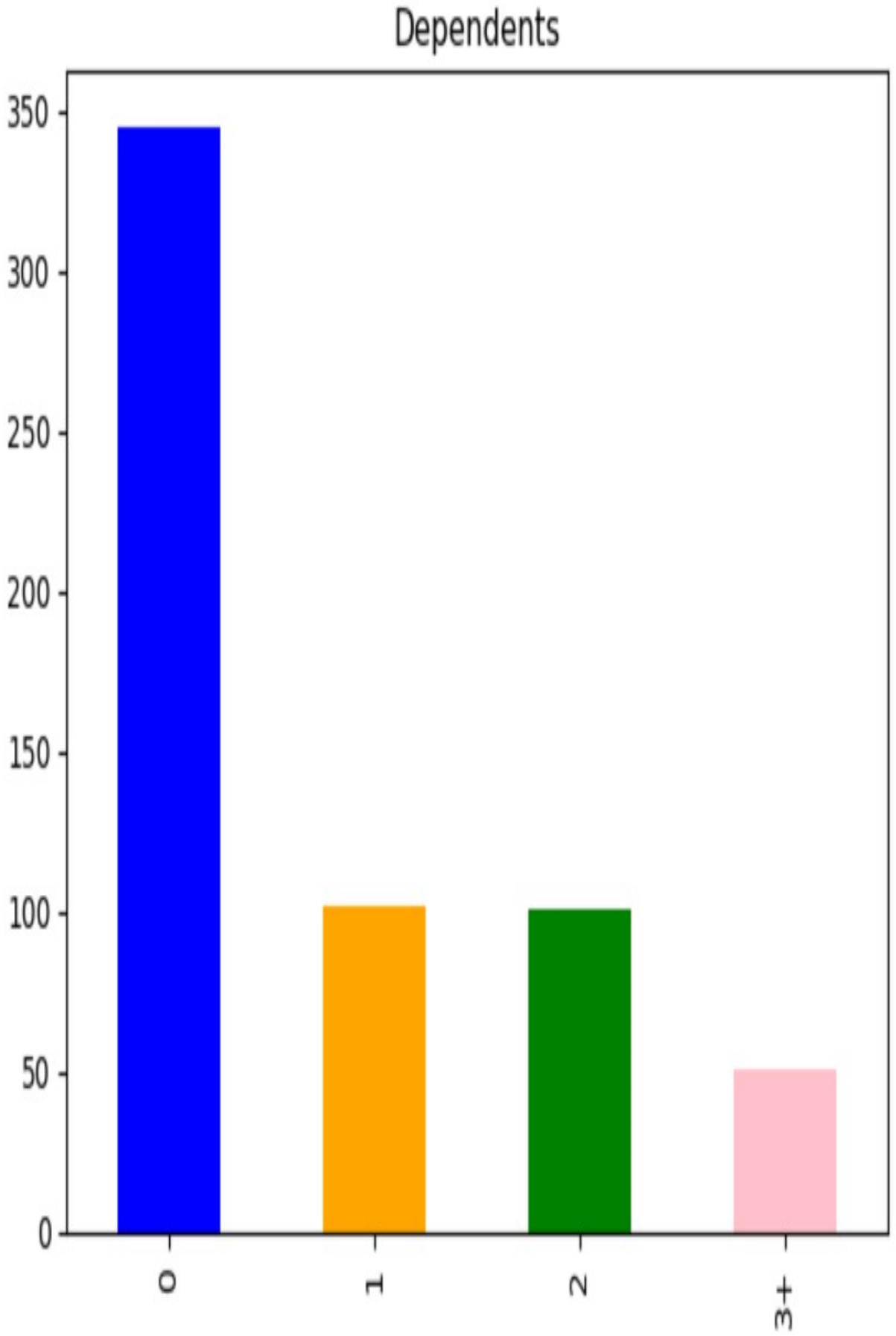
```
loan_df['Married'].value_counts()
loan_df['Married'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Married')
```

```
Text(0.5, 1.0, 'Married')
```



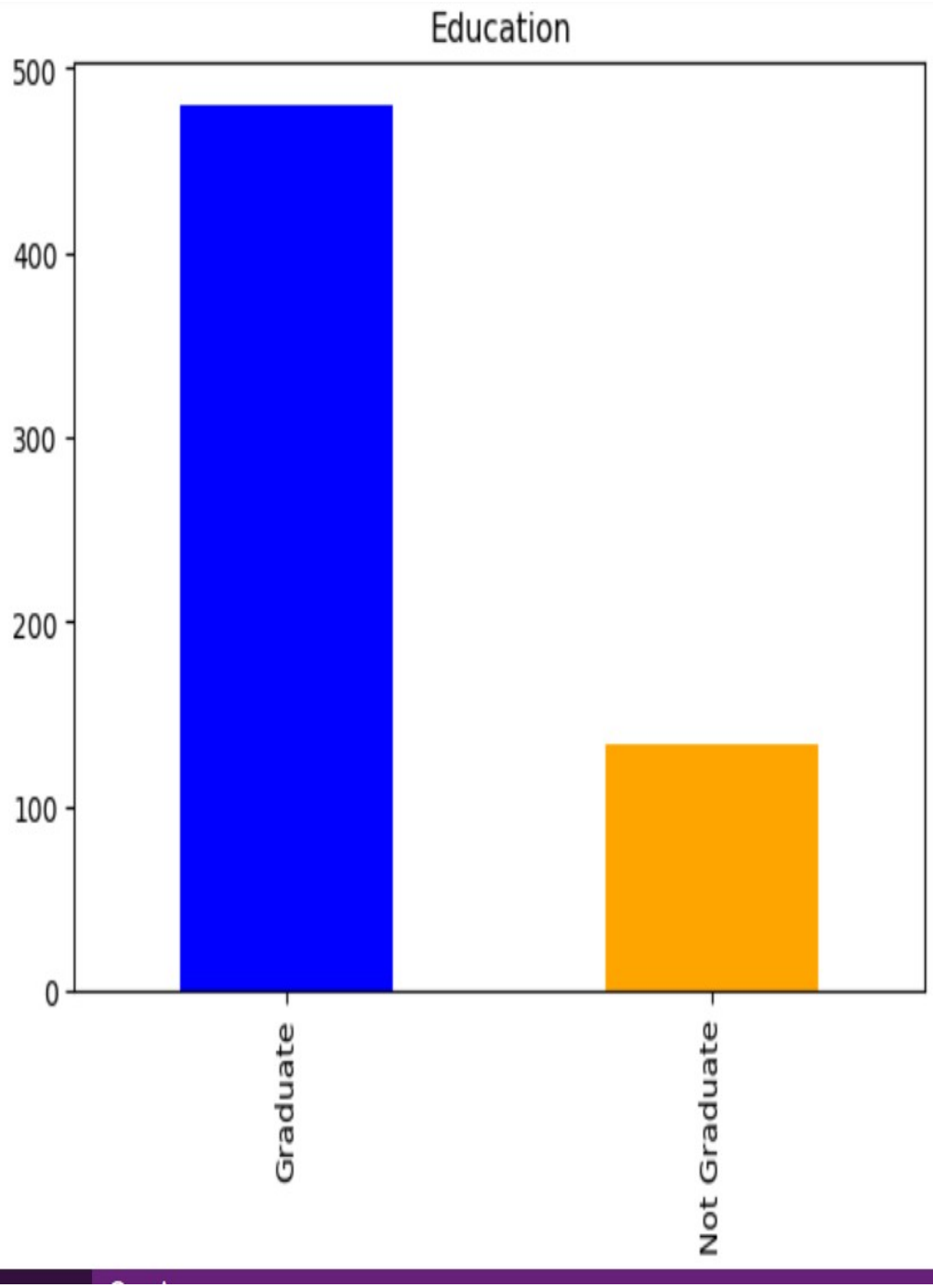
```
loan_df['Dependents'].value_counts()
loan_df['Dependents'].value_counts().plot.bar(color=[ 'blue','orange','green','pink'])
plt.title('Dependents')
```

```
Text(0.5, 1.0, 'Dependents')
```



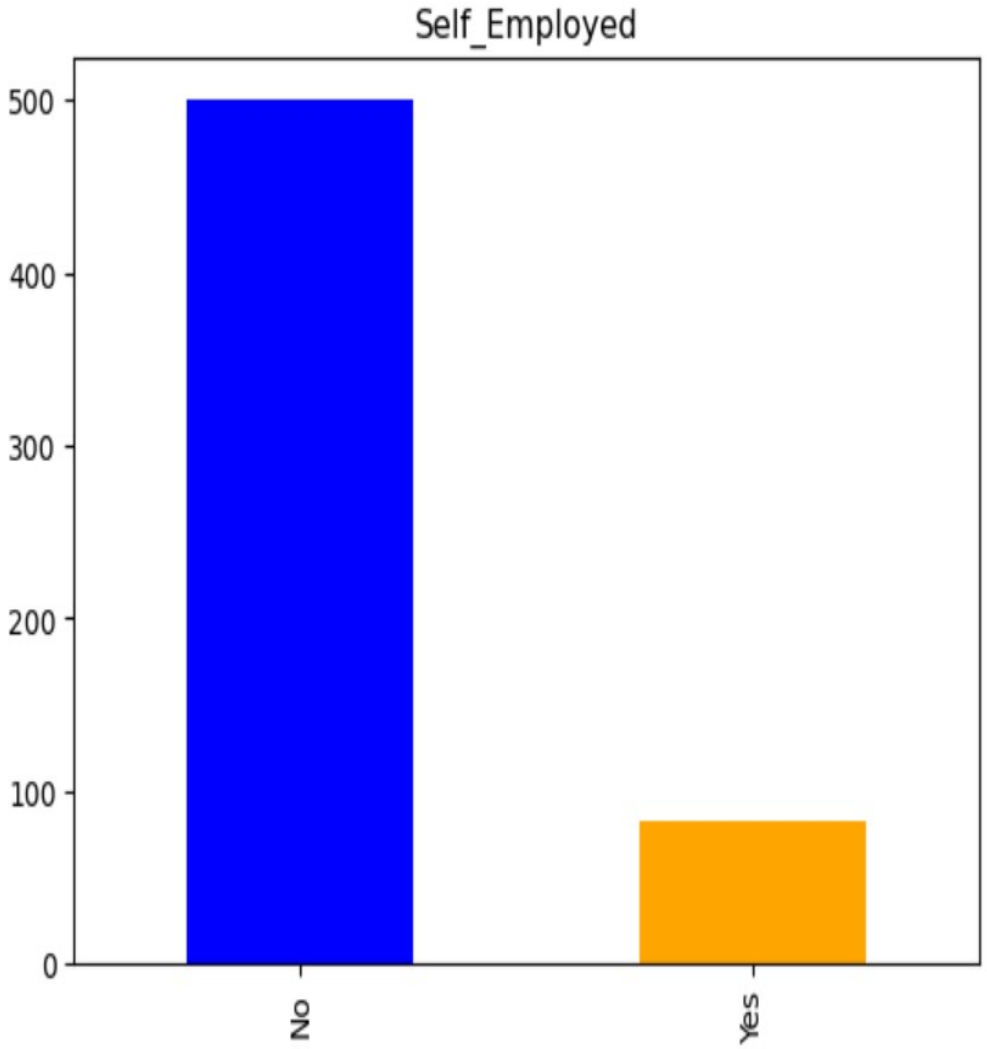
```
loan_df['Education'].value_counts()
loan_df['Education'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Education')
```

```
Text(0.5, 1.0, 'Education')
```



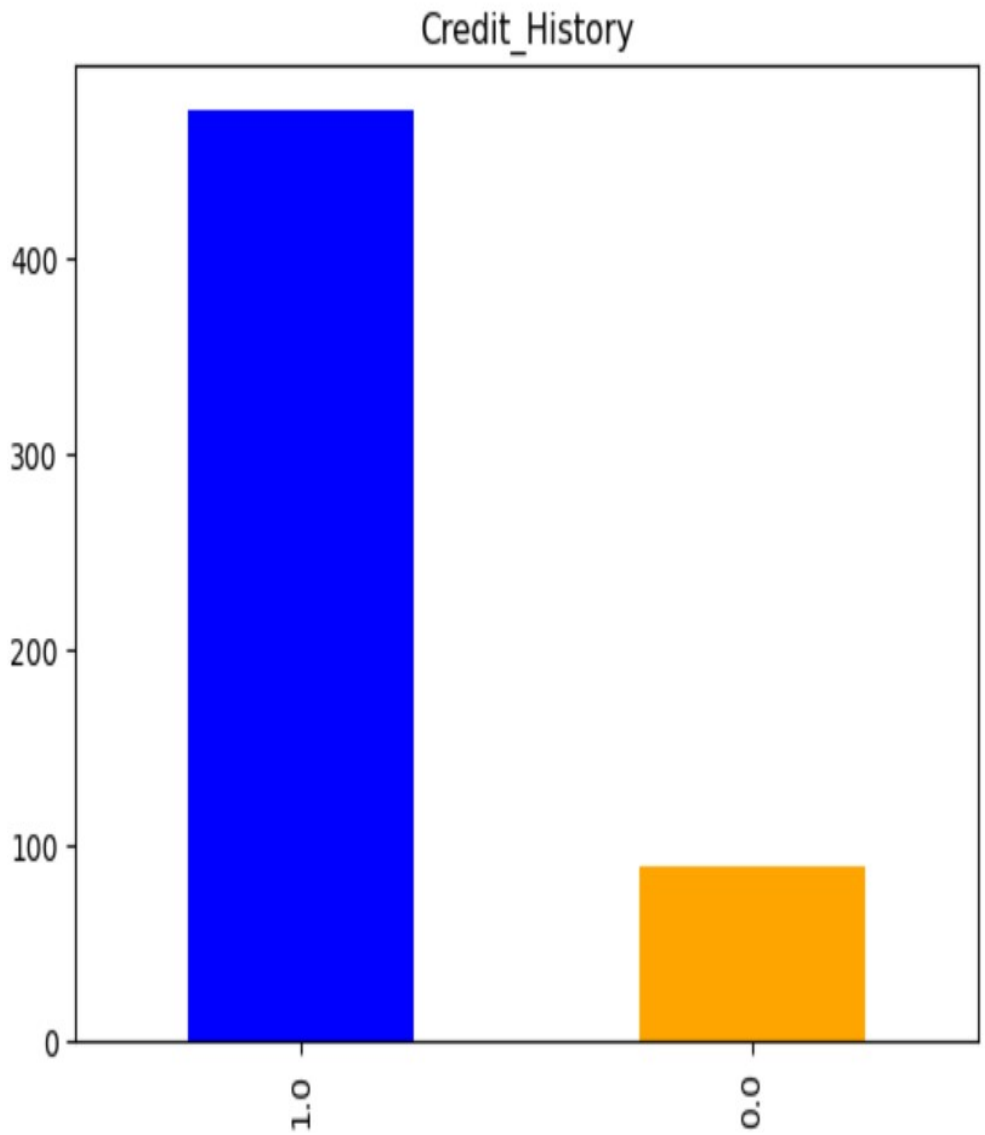
```
loan_df['Self_Employed'].value_counts()
loan_df['Self_Employed'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Self_Employed')
```

```
Text(0.5, 1.0, 'Self_Employed')
```



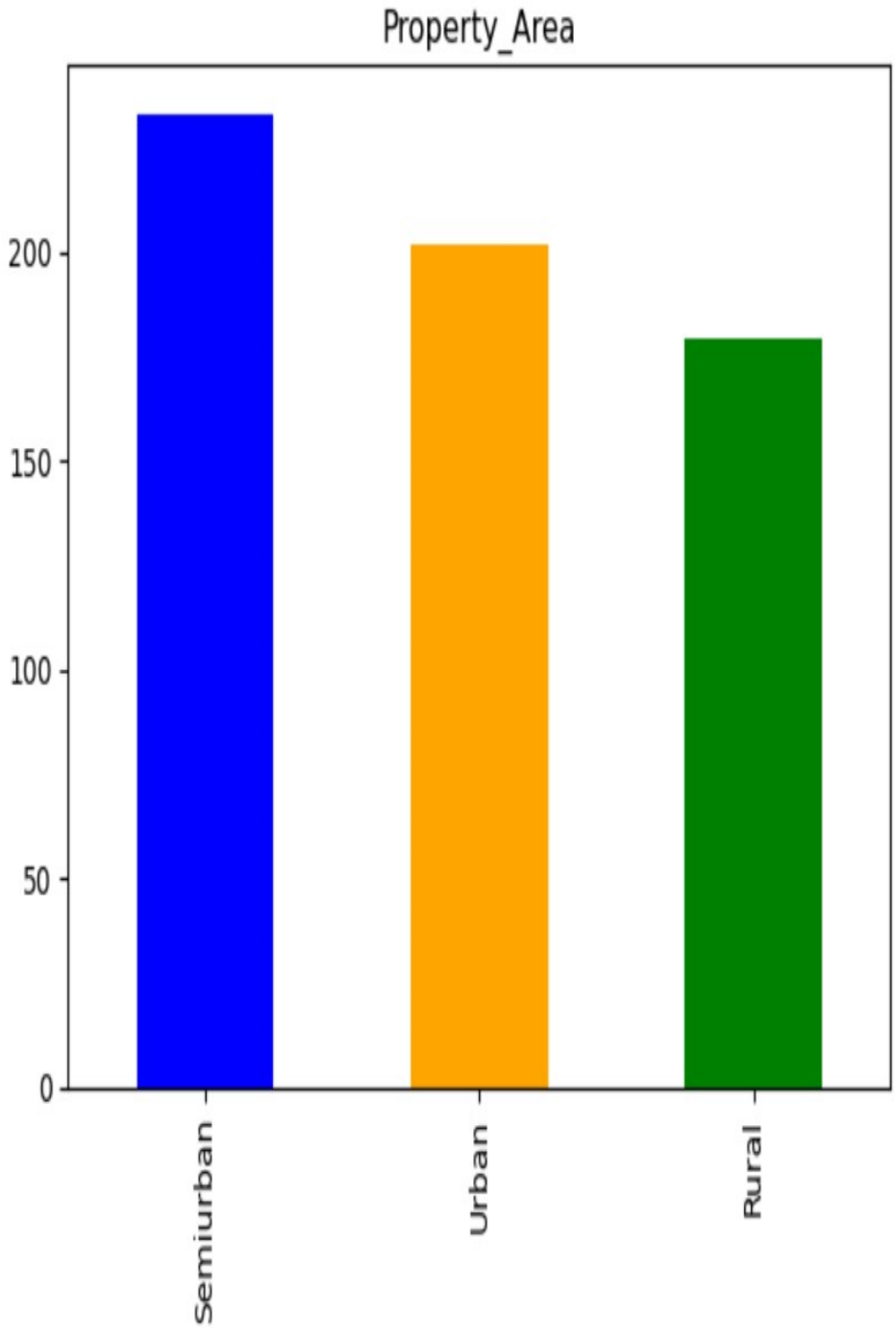
```
loan_df['Credit_History'].value_counts()
loan_df['Credit_History'].value_counts().plot.bar(color=[ 'blue','orange'])
plt.title('Credit_History')
```

```
Text(0.5, 1.0, 'Credit_History')
```



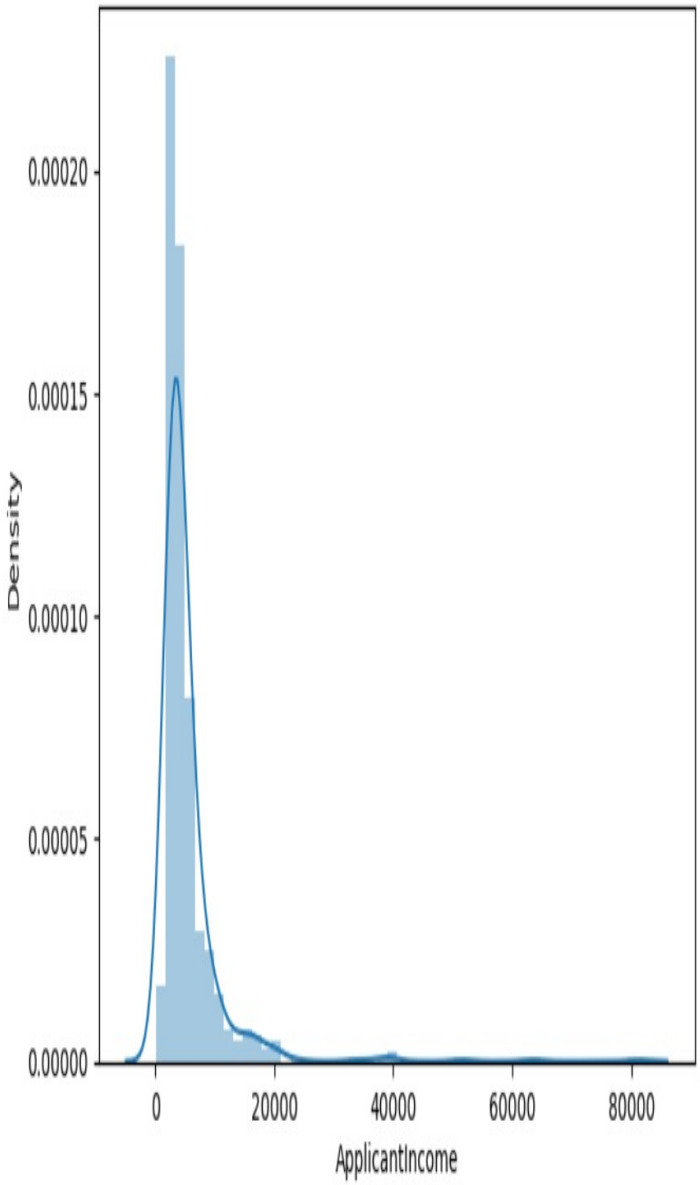
```
loan_df['Property_Area'].value_counts()
loan_df['Property_Area'].value_counts().plot.bar(color=[ 'blue','orange','green'])
plt.title('Property_Area')
```

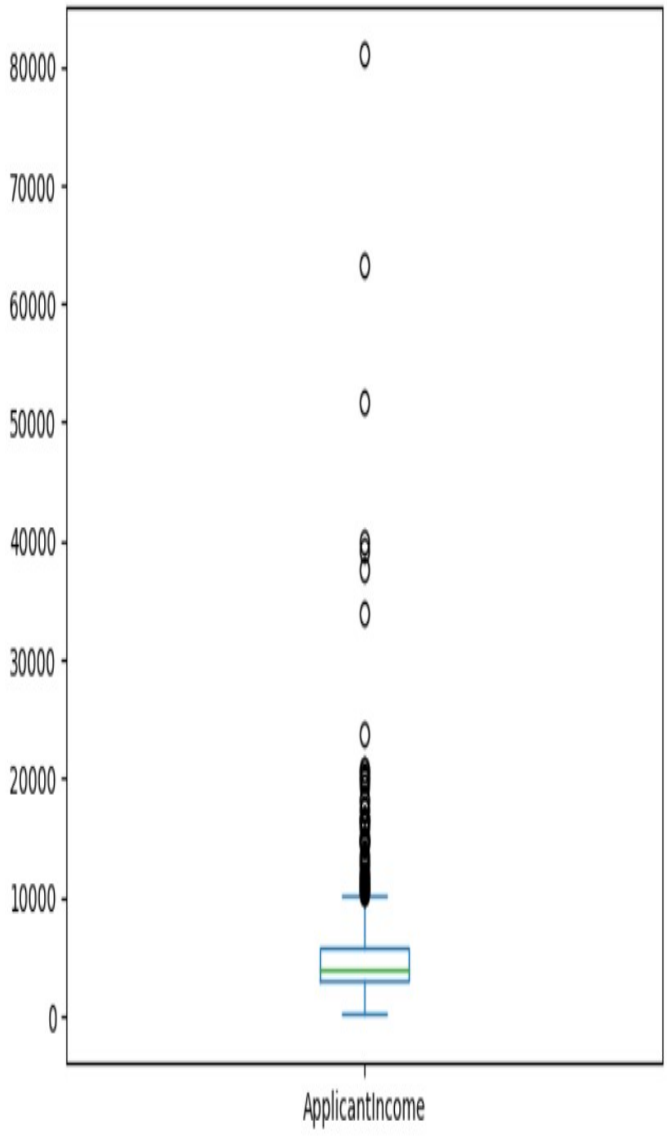
```
Text(0.5, 1.0, 'Property_Area')
```



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

plt.figure(1)
plt.subplot(121)
sns.distplot(loan_df['ApplicantIncome']);

plt.subplot(122)
loan_df['ApplicantIncome'].plot.box(figsize=(16,5))

plt.show()
```

''

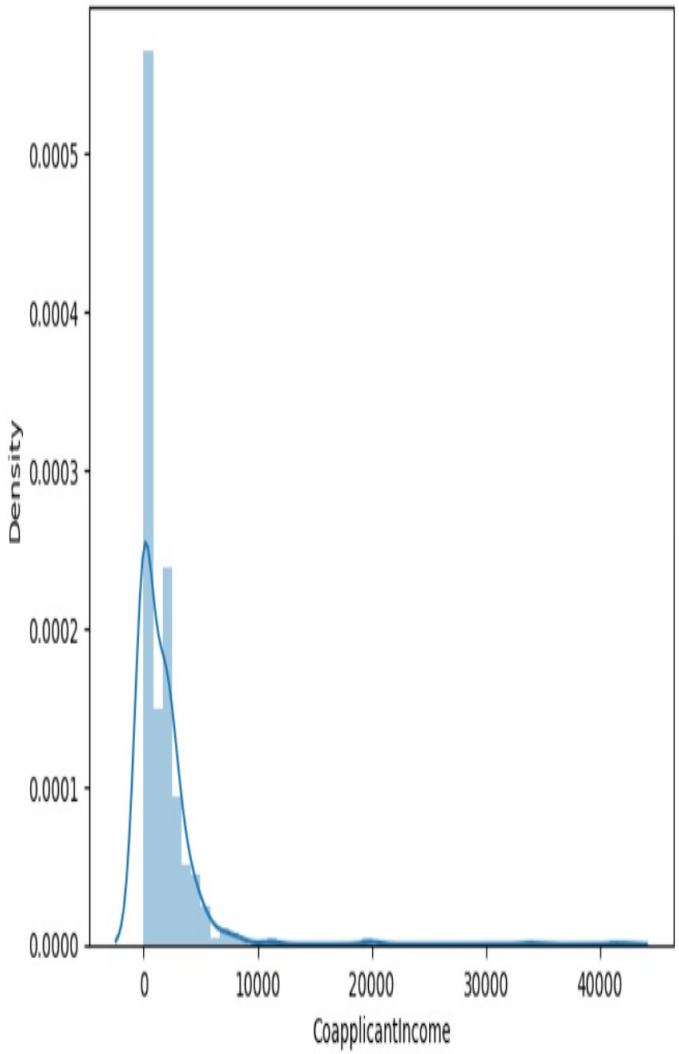


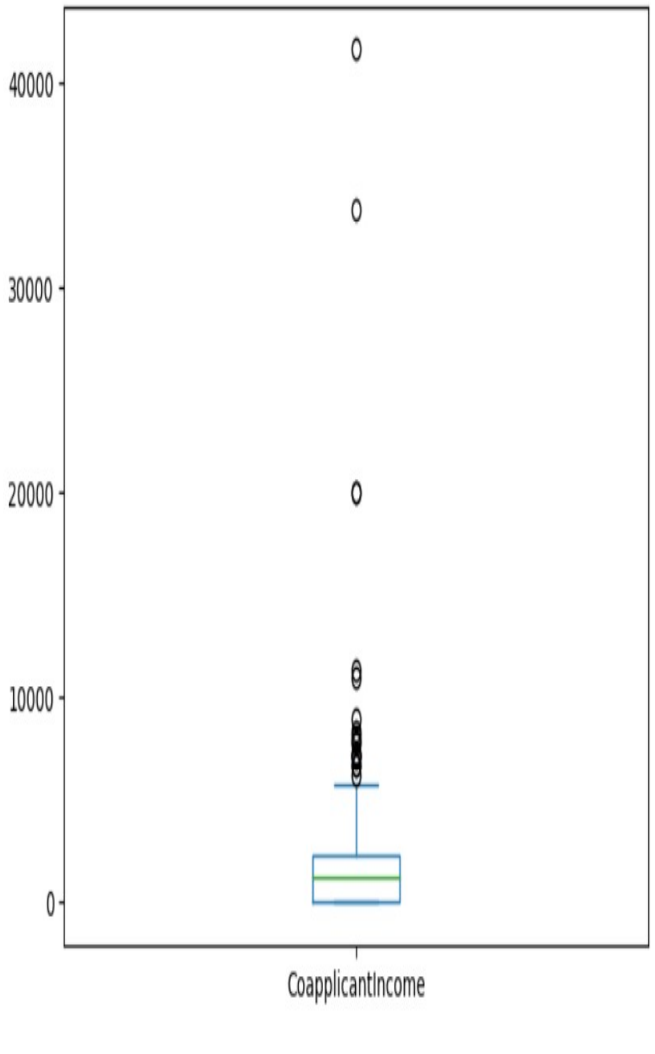


```
import warnings
warnings.filterwarnings('ignore')

plt.figure(1)
plt.subplot(121)
sns.distplot(loan_df['CoapplicantIncome']);

plt.subplot(122)
loan_df['CoapplicantIncome'].plot.box(figsize=(16,5))

plt.show()
```

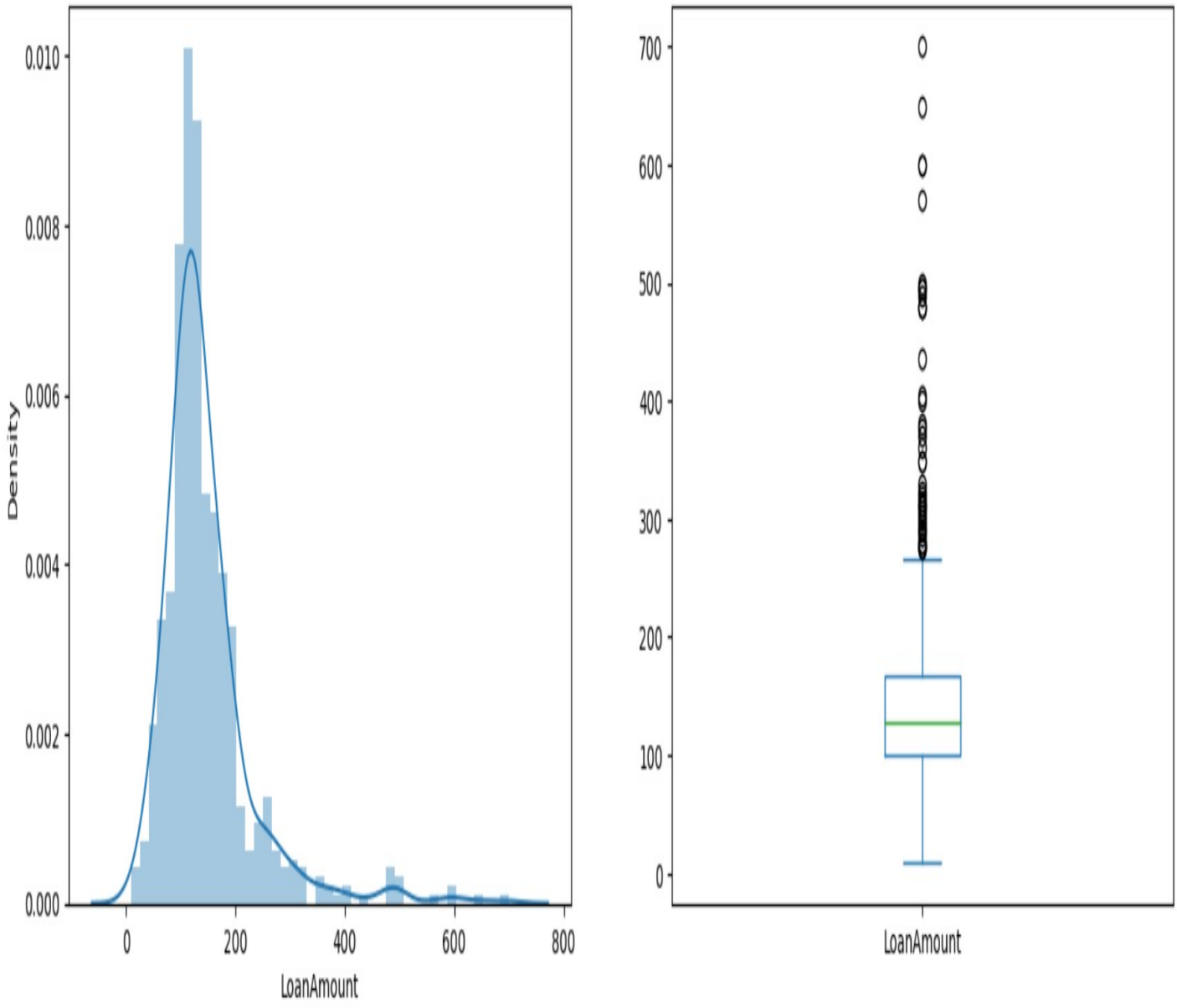
Pythc

'''





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

plt.figure(1)
plt.subplot(121)
sns.distplot(loan_df['LoanAmount']);

plt.subplot(122)
loan_df['LoanAmount'].plot.box(figsize=(16,5))

plt.show()
```

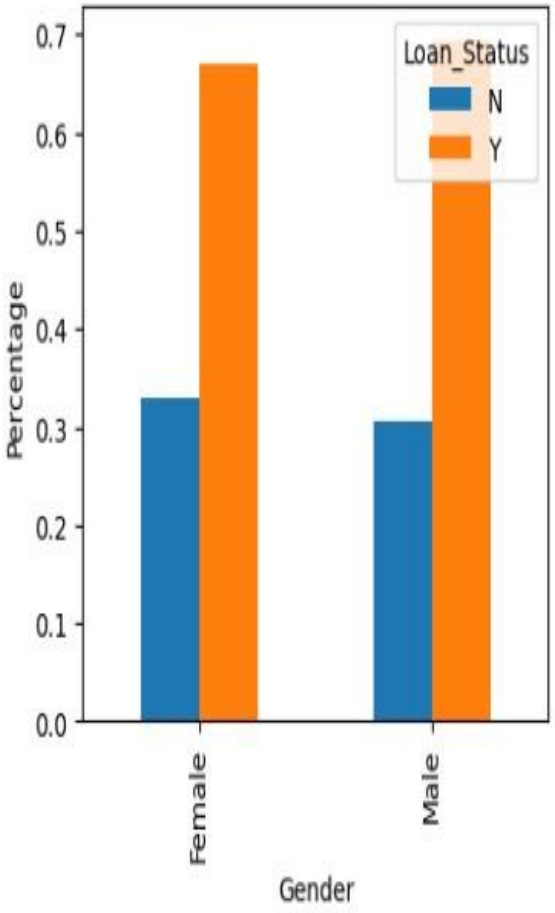


➢ BI VARIATE ANAYSIS

It analysis the variables comparing with target variable

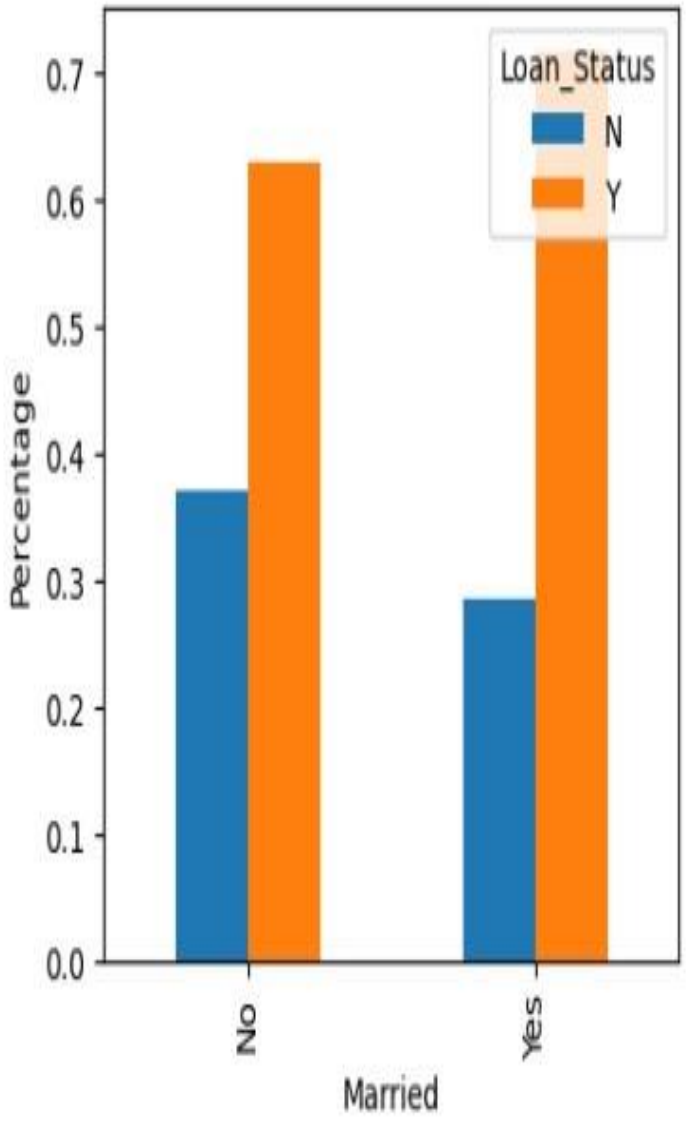
[26]:
```
print(pd.crosstab(loan_df['Gender'],loan_df['Loan_Status']))

Gender=pd.crosstab(loan_df['Gender'],loan_df['Loan_Status'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind="bar", stacked=False, figsize=(4,4))
plt.xlabel('Gender')
p = plt.ylabel('Percentage')
```

```
Loan_Status    N    Y
Gender
Female        37   75
Male         150  339
```



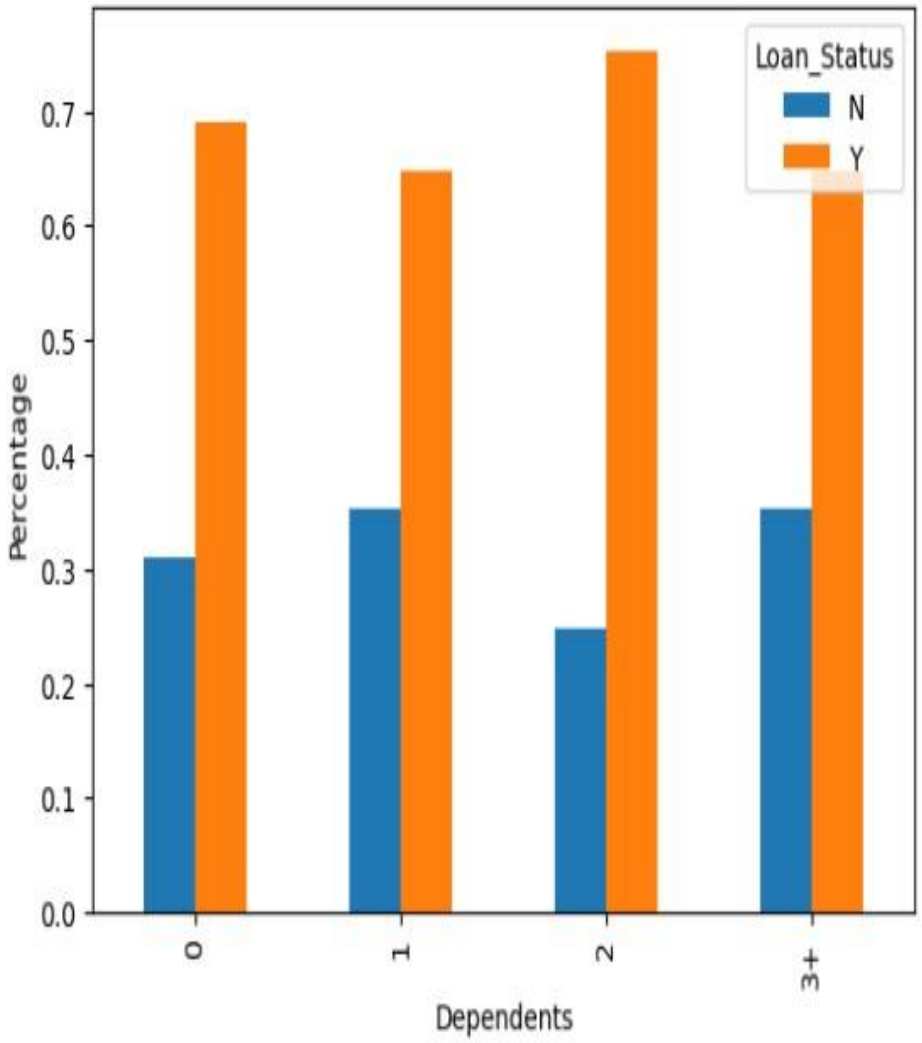
In [27]:
```
print(pd.crosstab(loan_df['Married'],loan_df['Loan_Status']))

Married=pd.crosstab(loan_df['Married'],loan_df['Loan_Status'])
Married.div(Married.sum(1).astype(float), axis=0).plot(kind="bar", figsize=(4,4))
plt.xlabel('Married')
p = plt.ylabel('Percentage')
```

Loan_Status	N	Y
Married		
No	79	134
Yes	113	285



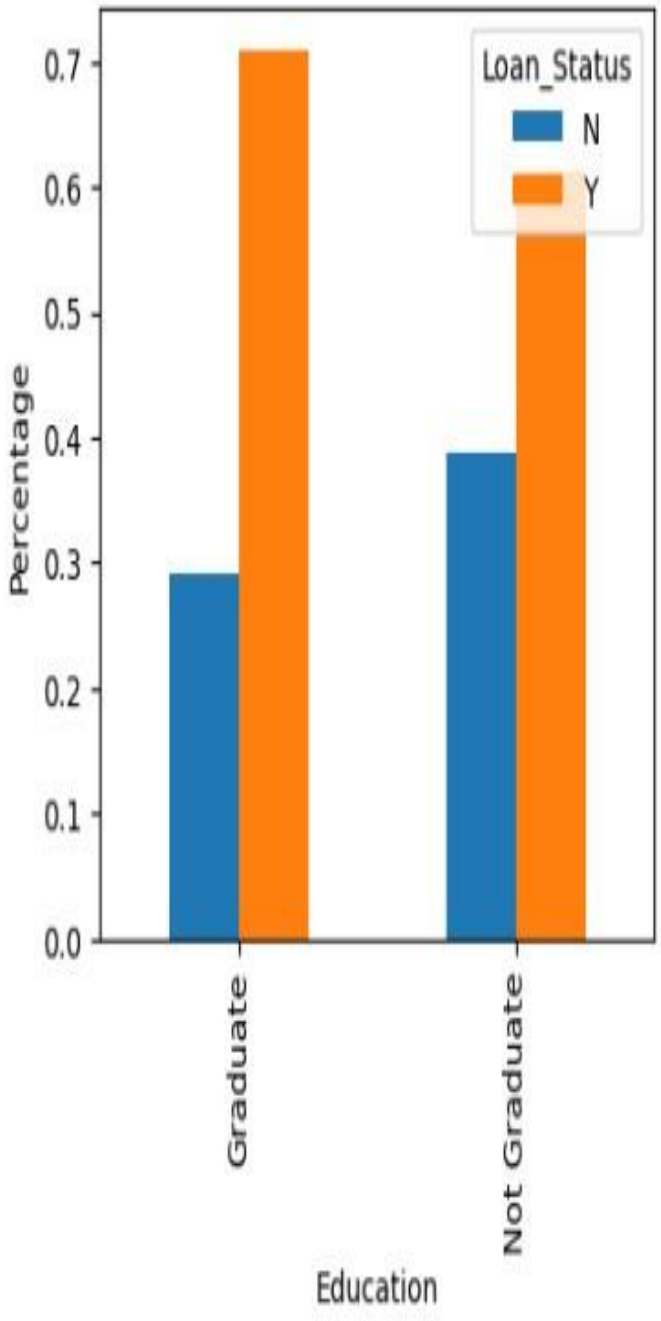
In [28]:
```
print(pd.crosstab(loan_df['Dependents'],loan_df['Loan_Status']))

Dependents=pd.crosstab(loan_df['Dependents'],loan_df['Loan_Status'])
Dependents.div(Dependents.sum(1).astype(float), axis=0).plot(kind="bar")
plt.xlabel('Dependents')
p = plt.ylabel('Percentage')
```

Loan_Status	N	Y
Dependents		
0	107	238
1	36	66
2	25	76
3+	18	33



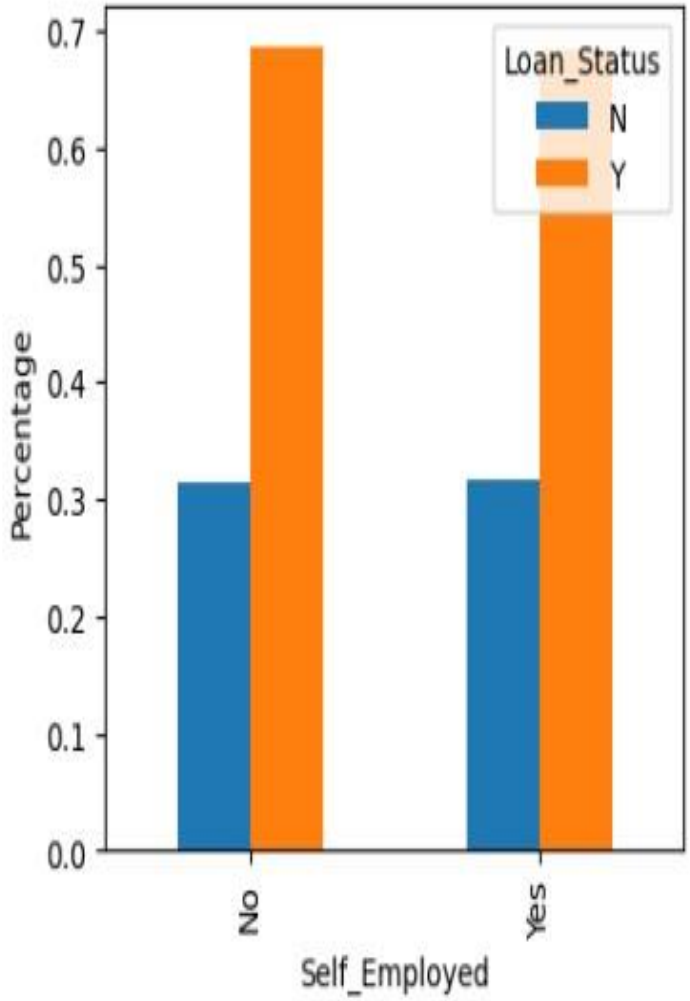
```
print(pd.crosstab(loan_df['Education'],loan_df['Loan_Status']))

Education=pd.crosstab(loan_df['Education'],loan_df['Loan_Status'])
Education.div(Education.sum(1).astype(float), axis=0).plot(kind="bar", figsize=(4,4))
plt.xlabel('Education')
p = plt.ylabel('Percentage')
```

Loan_Status	N	Y
Education		
Graduate	140	340
Not Graduate	52	82



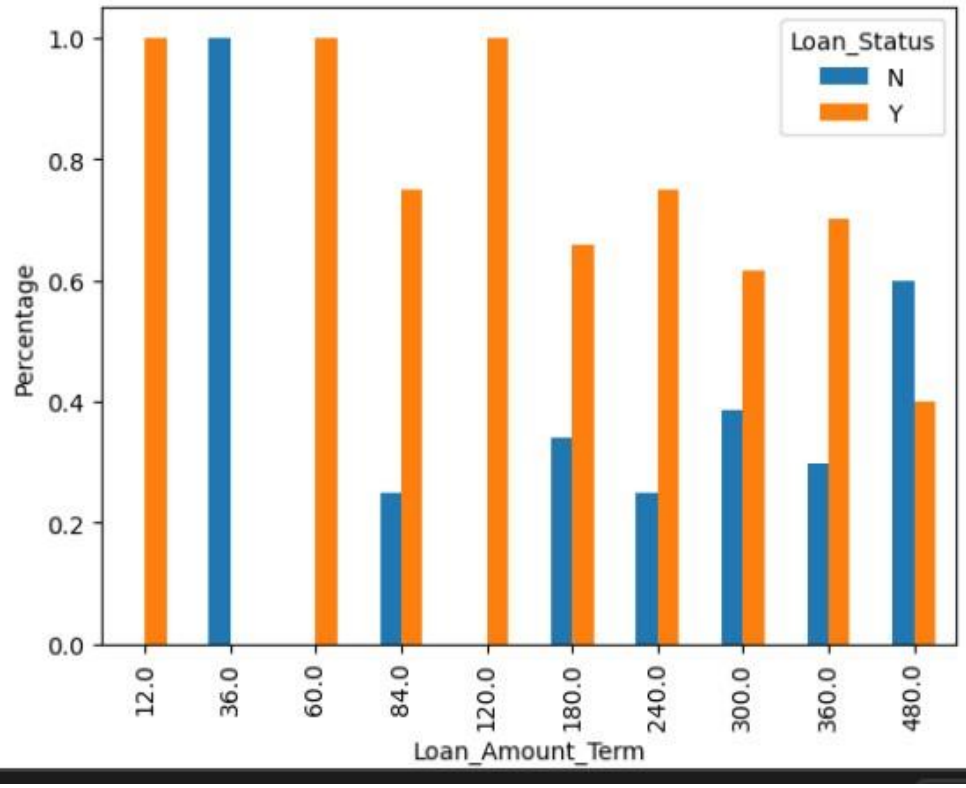
[30]:
```
print(pd.crosstab(loan_df['Self_Employed'],loan_df['Loan_Status']))

Self_Employed=pd.crosstab(loan_df['Self_Employed'],loan_df['Loan_Status'])
Self_Employed.div(Self_Employed.sum(1).astype(float), axis=0).plot(kind="bar", figsize=(4,4))
plt.xlabel('Self_Employed')
p = plt.ylabel('Percentage')
```

```
Loan_Status      N    Y
Self_Employed
No             157  343
Yes             26   56
```



In [33]:
```
print(pd.crosstab(loan_df['Loan_Amount_Term'],loan_df['Loan_Status']))

Property_Area=pd.crosstab(loan_df['Loan_Amount_Term'],loan_df['Loan_Status'])
Property_Area.div(Property_Area.sum(1).astype(float), axis=0).plot(kind="bar")
plt.xlabel('Loan_Amount_Term')
P = plt.ylabel('Percentage')
```

```
Loan_Status         N    Y
Loan_Amount_Term
12.0                0    1
36.0                2    0
60.0                0    2
84.0                1    3
120.0               0    3
180.0              15   29
240.0               1    3
300.0               5    8
360.0             153  359
480.0               9    6
```



In [34]:
```
loan_df.head()
```

Out[34]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [35]:
```
maxValue = loan_df[['ApplicantIncome']].max()
```

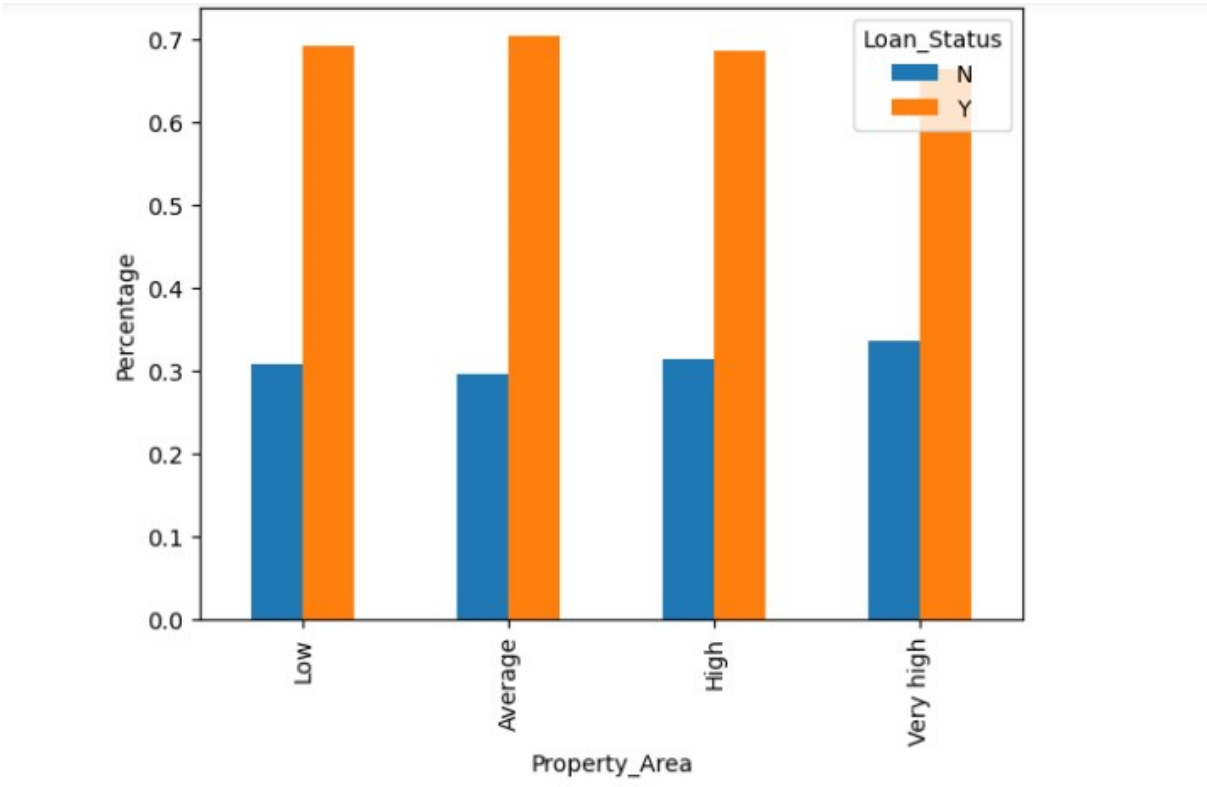
In [36]:
```
print(maxValue)
```

```
ApplicantIncome    81000
dtype: int64
```

In [37]:
```
minValue = loan_df[['ApplicantIncome']].min()
print(minValue)
```

```
ApplicantIncome    150
dtype: int64
```

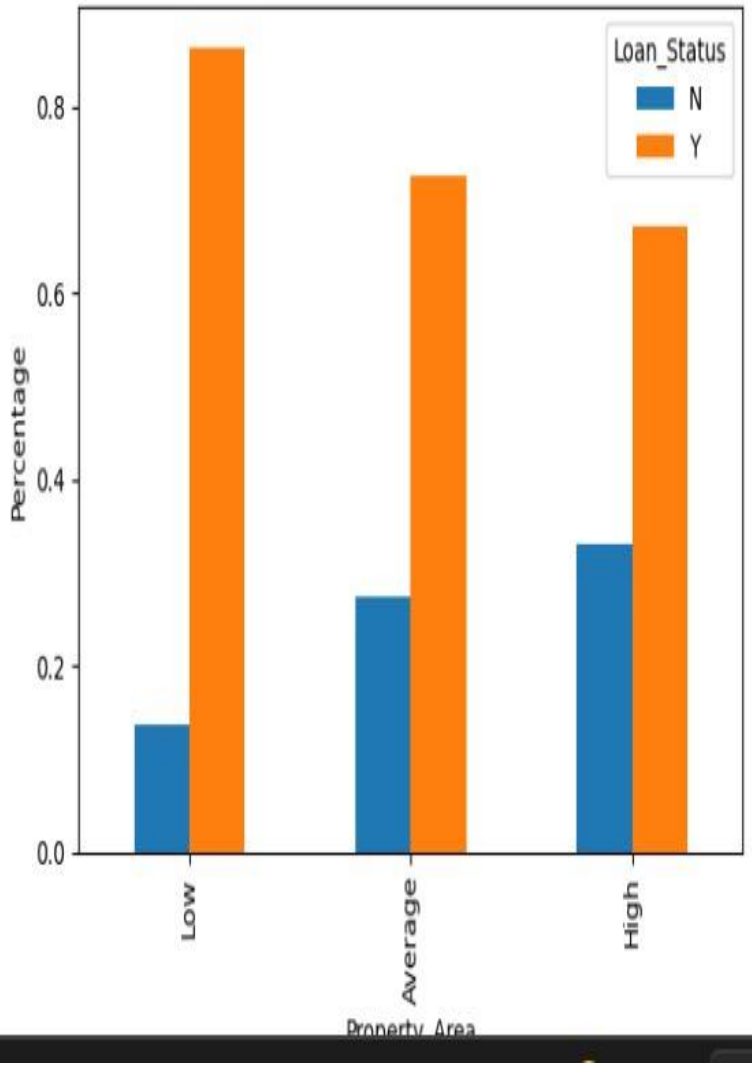
In [38]:
```
# Making bins for Applicant income variable
bins=[150,2500,4000,6000,81000]
group=['Low','Average','High', 'Very high']
loan_df['Income_bin']=pd.cut(loan_df['ApplicantIncome'],bins,labels=group)
Income_bin=pd.crosstab(loan_df['Income_bin'],loan_df['Loan_Status'])
Income_bin.div(Income_bin.sum(1).astype(float), axis=0).plot(kind="bar")
plt.xlabel('Property_Area')
P = plt.ylabel('Percentage')
```



In [40]:
```
maxValue = loan_df[['CoapplicantIncome']].max()
print(maxValue)
minValue = loan_df[['CoapplicantIncome']].min()
print(minValue)
```

```
CoapplicantIncome    41667.0
dtype: float64
CoapplicantIncome    0.0
dtype: float64
```

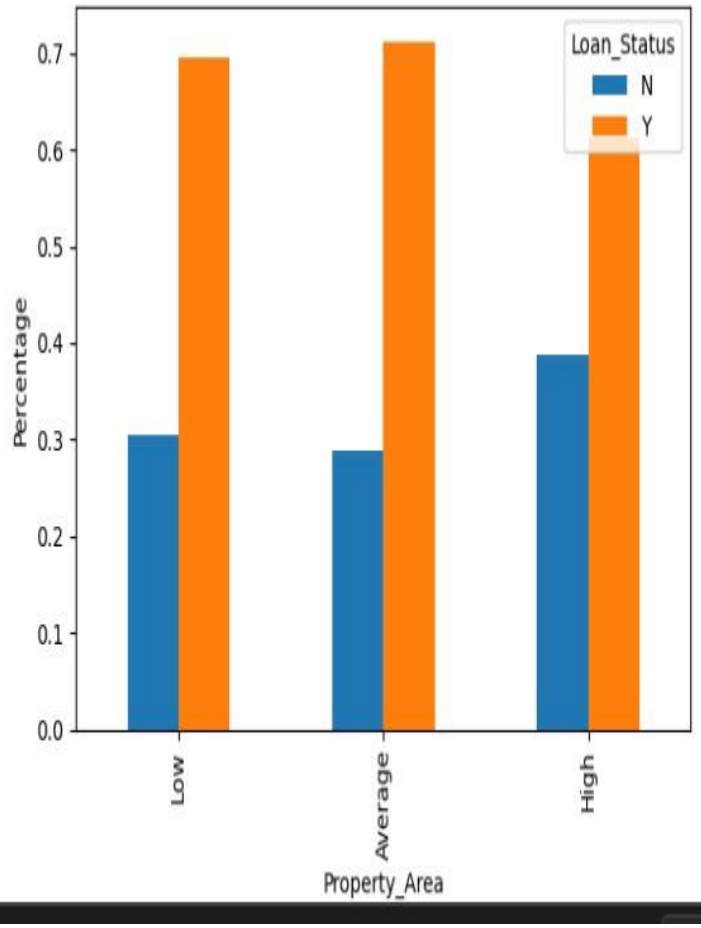
In [41]:
```
# Making bins for Coapplicant income variable
bins=[0,1000,3000,42000]
group=['Low','Average','High']
loan_df['Coapplicant_Income_bin']=pd.cut(loan_df['CoapplicantIncome'],bins,labels=group)
Coapplicant_Income_bin=pd.crosstab(loan_df['Coapplicant_Income_bin'],loan_df['Loan_Status'])
Coapplicant_Income_bin.div(Coapplicant_Income_bin.sum(1).astype(float), axis=0).plot(kind="bar")
plt.xlabel('Property_Area')
P = plt.ylabel('Percentage')
```



In [42]:
```
maxValue = loan_df[['LoanAmount']].max()
print(maxValue)
minValue = loan_df[['LoanAmount']].min()
print(minValue)
```

```
LoanAmount    700.0
dtype: float64
LoanAmount    9.0
dtype: float64
```

In [43]:
```
# Making bins for LoanAmount variable
bins=[0,100,200,700]
group=['Low','Average','High']
loan_df['LoanAmount_bin']=pd.cut(loan_df['LoanAmount'],bins,labels=group)
LoanAmount_bin=pd.crosstab(loan_df['LoanAmount_bin'],loan_df['Loan_Status'])
LoanAmount_bin.div(LoanAmount_bin.sum(1).astype(float), axis=0).plot(kind="bar")
plt.xlabel('Property_Area')
P = plt.ylabel('Percentage')
```



- We will drop the bins because it will add to the data set

```
In [44]: train=loan_df.drop(['Income_bin', 'Coapplicant_Income_bin', 'LoanAmount_bin'], axis=1,inplace=True)

In [45]: loan_df.isnull().sum()

Out[45]: 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
```

- Multi variate analysis

It will find the co relation between the variables

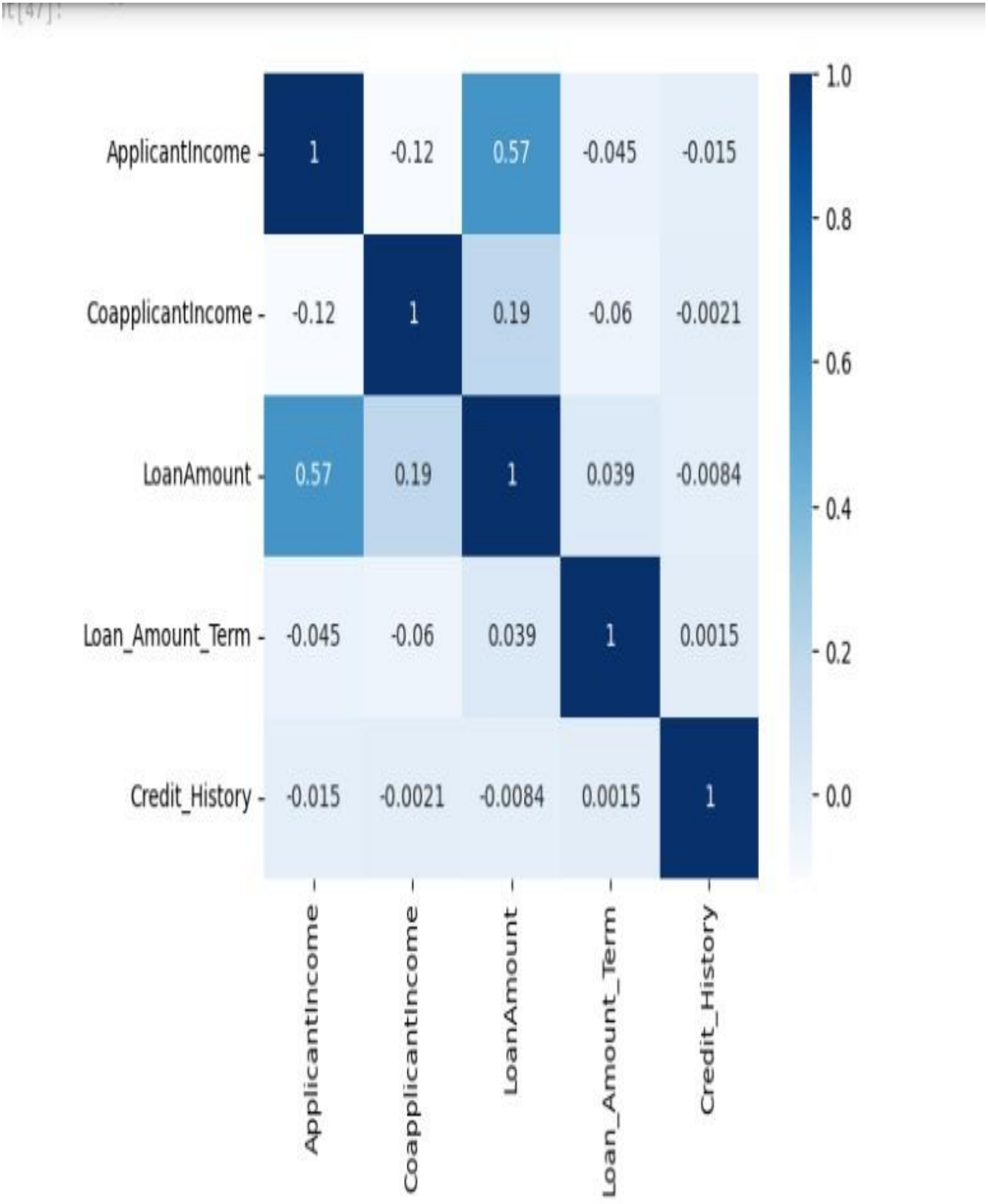
Multivariate analysis

In [46]:
```
# Print correlation matrix (heat map)

import warnings
warnings.filterwarnings('ignore')
correlation=loan_df.corr()
print(correlation)
```

```
                   ApplicantIncome  CoapplicantIncome  LoanAmount  \
ApplicantIncome           1.000000          -0.116605    0.570909
CoapplicantIncome        -0.116605           1.000000    0.188619
LoanAmount                0.570909           0.188619    1.000000
Loan_Amount_Term         -0.045306          -0.059878    0.039447
Credit_History           -0.014715          -0.002056   -0.008433

                   Loan_Amount_Term  Credit_History
ApplicantIncome           -0.045306       -0.014715
CoapplicantIncome         -0.059878       -0.002056
LoanAmount                 0.039447       -0.008433
Loan_Amount_Term           1.000000        0.001470
Credit_History             0.001470        1.000000
```

In [47]:
```
sns.heatmap(correlation,cmap="Blues",annot=True)
```

Out[47]:
```
<Axes: >
```


ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
1
-0.12
0.57
-0.045
-0.015
-0.12
1
0.19
-0.06
-0.0021
0.57
0.19
1
0.039
-0.0084
-0.045
-0.06
0.039
1
0.0015
-0.015
-0.0021
-0.0084
0.0015
1
1.0
0.8
0.6
0.4
0.2
0.0

DATA PREPROCESING

➢ Handling Missing Values

In [51]:
```
loan_df.head()
```

Out[51]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [52]:
```
#we have null values in data set
loan_df.isnull().sum()
```

Out[52]:
```
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
```

In [53]:
```
#we will see data types of data set
loan_df.dtypes
```

Out[53]:
```
Gender              object
Married             object
Dependents          object
Education           object
Self_Employed       object
ApplicantIncome      int64
CoapplicantIncome  float64
LoanAmount         float64
Loan_Amount_Term   float64
Credit_History     float64
Property_Area       object
Loan_Status         object
dtype: object
```

In [54]:
```
#printing numerical and categorical values present in the data set
import warnings
warnings.filterwarnings('ignore')

numerical_values =loan_df.select_dtypes(include = [np.number]).columns
categorical_values =loan_df.select_dtypes(include = [np.object]).columns
print(numerical_values)
print(categorical_values)
```

```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History'],
      dtype='object')
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [55]:
```
#filling null values
loan_df['Gender'] =loan_df['Gender'].fillna(loan_df['Gender'].mode()[0])
loan_df['Married'] = loan_df['Married'].fillna(loan_df['Married'].mode()[0])
loan_df['Dependents'] = loan_df['Dependents'].str.replace('+','')
loan_df['Dependents'] = loan_df['Dependents'].fillna(loan_df['Dependents'].mode()[0])
loan_df['LoanAmount'] = loan_df['LoanAmount'].fillna(loan_df['LoanAmount'].mode()[0])
loan_df['Self_Employed'] = loan_df['Self_Employed'].fillna(loan_df['Self_Employed'].mode()[0])
loan_df['Loan_Amount_Term'] = loan_df['Loan_Amount_Term'].fillna(loan_df['Loan_Amount_Term'].mode()[0])
loan_df['Credit_History'] = loan_df['Credit_History'].fillna(loan_df['Credit_History'].mode()[0])
```

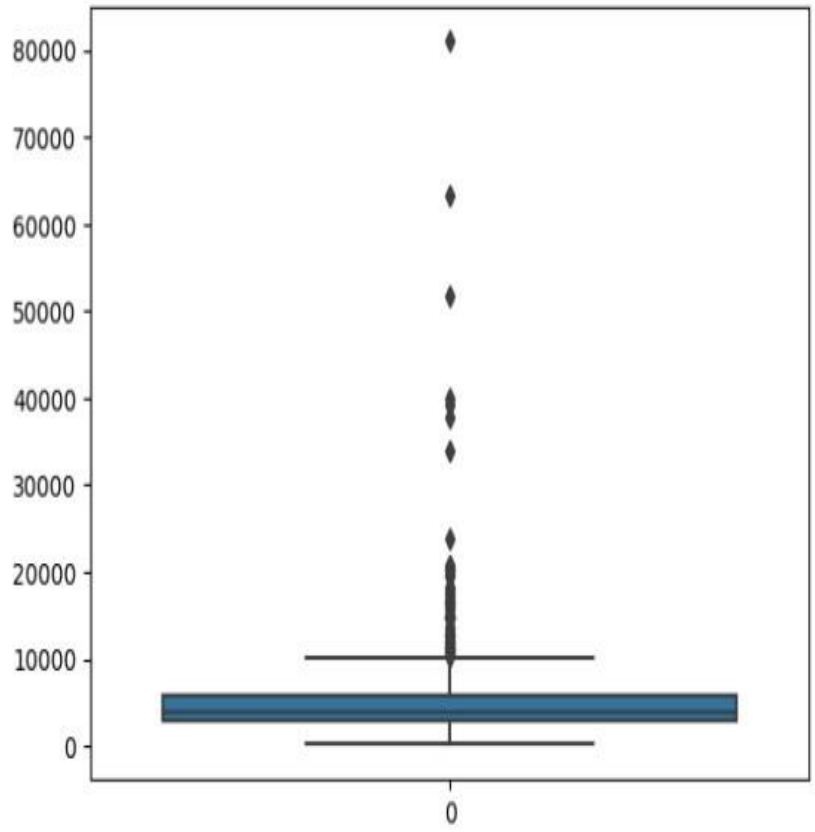
In [56]:
```
#we filled all the null values by using stastics
loan_df.isnull().sum()
```

Out[56]:
```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

➢Outliers

In [58]:
```
sns.boxplot(loan_df["ApplicantIncome"])
```

Out[58]:
```
<Axes: >
```



In [59]:
```
q1=loan_df['ApplicantIncome'].quantile(0.25)
q3=loan_df['ApplicantIncome'].quantile(0.75)
iqr=q3-q1
q1,q3,iqr
```

Out[59]:
```
(2877.5, 5795.0, 2917.5)
```

In [60]:
```
upper_limit=q3+(1.5*iqr)
lower_limit=q3+(1.5*iqr)
lower_limit,upper_limit
```

Out[60]:
```
(10171.25, 10171.25)
```

In [59]:
```
q1=loan_df['ApplicantIncome'].quantile(0.25)
q3=loan_df['ApplicantIncome'].quantile(0.75)
iqr=q3-q1
q1,q3,iqr
```

Out[59]: (2877.5, 5795.0, 2917.5)

In [60]:
```
upper_limit=q3+(1.5*iqr)
lower_limit=q3+(1.5*iqr)
lower_limit,upper_limit
```

Out[60]: (10171.25, 10171.25)

In [61]:
```
#finding the outliers
loan_df.loc[(loan_df['ApplicantIncome']>upper_limit)|(loan_df['ApplicantIncome']<lower_limit)]
```

Out[61]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	120.0	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 12 columns

In [63]:
```
q1=loan_df['CoapplicantIncome'].quantile(0.25)
q3=loan_df['CoapplicantIncome'].quantile(0.75)
iqr=q3-q1
q1,q3,iqr
```

Out[63]: (0.0, 2297.25, 2297.25)

In [64]:
```
upper_limit=q3+(1.5*iqr)
lower_limit=q3+(1.5*iqr)
lower_limit,upper_limit
```

Out[64]: (5743.125, 5743.125)

In [65]:
```
#finding the outliers
loan_df.loc[(loan_df['CoapplicantIncome']>upper_limit)|(loan_df['CoapplicantIncome']<lower_limit)]
```

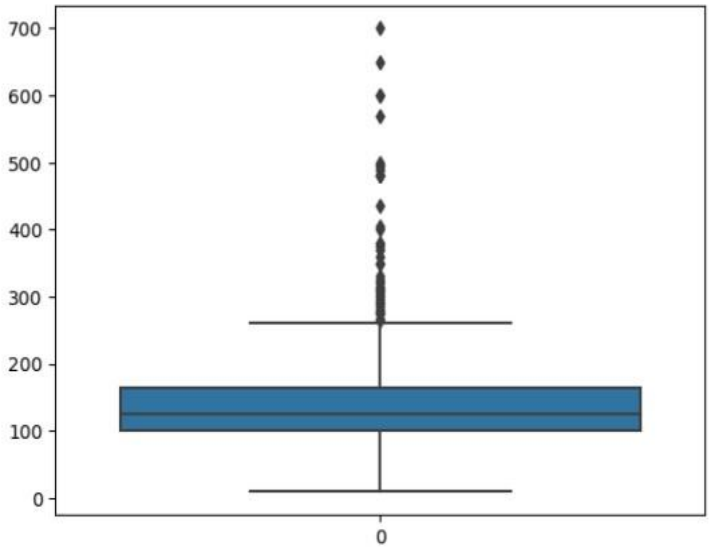
Out[65]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	120.0	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 12 columns

In [66]:
```
sns.boxplot(loan_df["LoanAmount"])
```

Out[66]:
```
<Axes: >
```



In [67]:
```
q1=loan_df['LoanAmount'].quantile(0.25)
q3=loan_df['CoapplicantIncome'].quantile(0.75)
iqr=q3-q1
q1,q3,iqr
```

Out[67]:
```
(100.25, 2297.25, 2197.0)
```

In [68]:
```
upper_limit=q3+(1.5*iqr)
lower_limit=q3+(1.5*iqr)
lower_limit,upper_limit
```

Out[68]:
```
(5592.75, 5592.75)
```

In [69]:
```
#finding the outliers
loan_df.loc[(loan_df['CoapplicantIncome']>upper_limit)|(loan_df['CoapplicantIncome']<lower_limit)]
```

Out[69]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	120.0	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 12 columns

➢ Handling categorical values

We need to categorical values into numerical values because model can understand only numerical values for building the model using machine learning algorithms.

In [74]:
```
loan_df.isnull().sum()
```

Out[74]:
```
Gender              0
Married             0
Dependents          0
Education           0
Self_Employed       0
ApplicantIncome     0
CoapplicantIncome   0
LoanAmount          0
Loan_Amount_Term    0
Credit_History      0
Property_Area       0
Loan_Status         0
dtype: int64
```

In [75]:
```
from sklearn.preprocessing import LabelEncoder

loan_df['Gender'].replace({'Male':1,'Female':0},inplace=True)
loan_df['Dependents'].replace({'0':0,'1':1,'2':2,'3':3},inplace=True)
loan_df['Married'].replace({'Yes':1,'No':0},inplace=True)
loan_df['Self_Employed'].replace({'Yes':1,'No':0},inplace=True)
loan_df['Property_Area'].replace({'Urban':2,'Rural':0,'Semiurban':1},inplace=True)
loan_df['Education'].replace({'Graduate':1,'Not Graduate':0},inplace=True)
loan_df['Loan_Status'].replace({'Y':1,'N':0},inplace=True)
```

In [76]:
```
loan_df['CoapplicantIncome']=loan_df['CoapplicantIncome'].astype("int64")
loan_df['LoanAmount']=loan_df['LoanAmount'].astype("int64")
loan_df['Loan_Amount_Term']=loan_df['Loan_Amount_Term'].astype("int64")
loan_df['Credit_History']=loan_df['Credit_History'].astype("int64")
```

In [77]:
```
loan_df.head()
```

Out[77]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1	0	0	1	0	5849	0	120	360	1	2	1
1	1	1	1	1	0	4583	1508	128	360	1	0	0
2	1	1	0	1	1	3000	0	66	360	1	2	1
3	1	1	0	0	0	2583	2358	120	360	1	2	1
4	1	0	0	1	0	6000	0	141	360	1	2	1

➢ Handling Imbalance data

[78]:
```
import numpy as np
import pandas as pd
from imblearn.combine import SMOTETomek
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score
import warnings
warnings.filterwarnings('ignore')
```

[79]:
```
#for handling the imbalancing data i use the smote technique
#smote stands for the Synthetic Minority Over-sampling TEchnique
from imblearn.combine import SMOTETomek
```

[n [ ]:
```
```

[80]:
```
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTETomek

from imblearn.combine import SMOTETomek
smote = SMOTETomek(random_state=42)

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

[81]:
```
x_bal,y_bal = smote.fit_resample(x,y)
print(y.value_counts())
print(y_bal.value_counts())
```

```
1    422
0    192
Name: Loan_Status, dtype: int64
1    356
0    356
Name: Loan_Status, dtype: int64
```

- Scaling the data

Scaling the data

In [82]:
```
scaler = StandardScaler()
x_bal = scaler.fit_transform(x_bal)
x_bal = pd.DataFrame(x_bal)
x_bal.head()
```

Out[82]:

	0	1	2	3	4	5	6	7	8	9	10
0	0.551389	-1.155174	-0.691351	0.638055	-0.335410	0.120158	-0.589954	-0.280508	0.307468	0.638055	1.36871
1	0.551389	0.865671	-0.691351	0.638055	2.981424	-0.385427	-0.589954	-0.983753	0.307468	0.638055	1.36871
2	0.551389	0.865671	-0.691351	-1.567262	-0.335410	-0.459428	0.290829	-0.280508	0.307468	0.638055	1.36871
3	0.551389	-1.155174	-0.691351	0.638055	-0.335410	0.146954	-0.589954	-0.007024	0.307468	0.638055	1.36871
4	0.551389	0.865671	1.434963	0.638055	2.981424	0.043495	0.977376	1.633882	0.307468	0.638055	1.36871

MODEL BUILDING

Importing all libraries for model building

Model Building

In [83]:
```
import numpy as np
import pandas as pd
from imblearn.combine import SMOTETomek
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import  GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB




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

➢ Splitting the data

We need to split the data for training and testing the model and evalution the model

Splitting the data set

```
4]: x_train, x_test, y_train, y_test = train_test_split(x_bal, y_bal, test_size = 0.33, random_state = 42)
```

Now we will train and test the model after splitting the data

We will use different algorithms for training the model and we will see performance of the model by using the model evalution metrics

K-Nearest Neighbor (KNN) Algorithm for Machine Learning

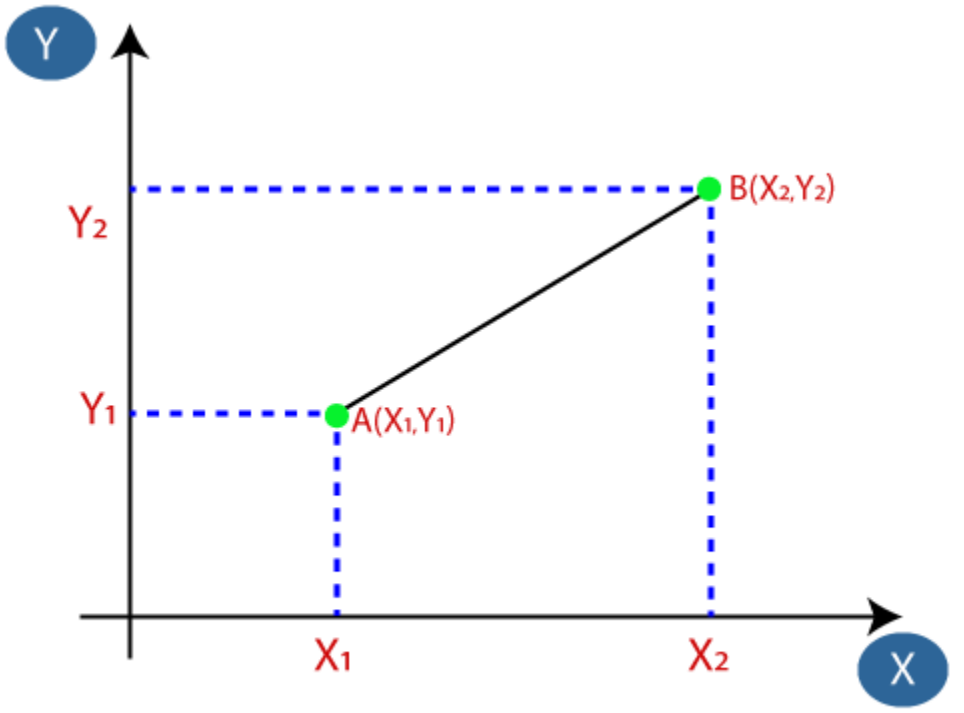
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

The K-NN working can be explained on the basis of the below algorithm:

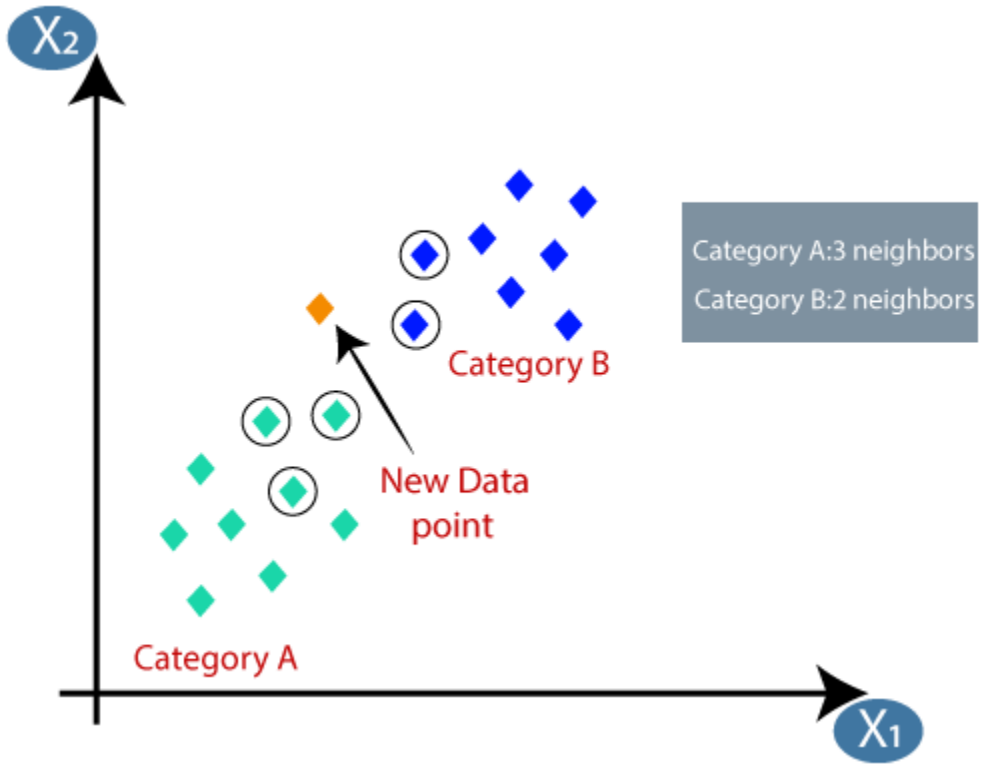
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

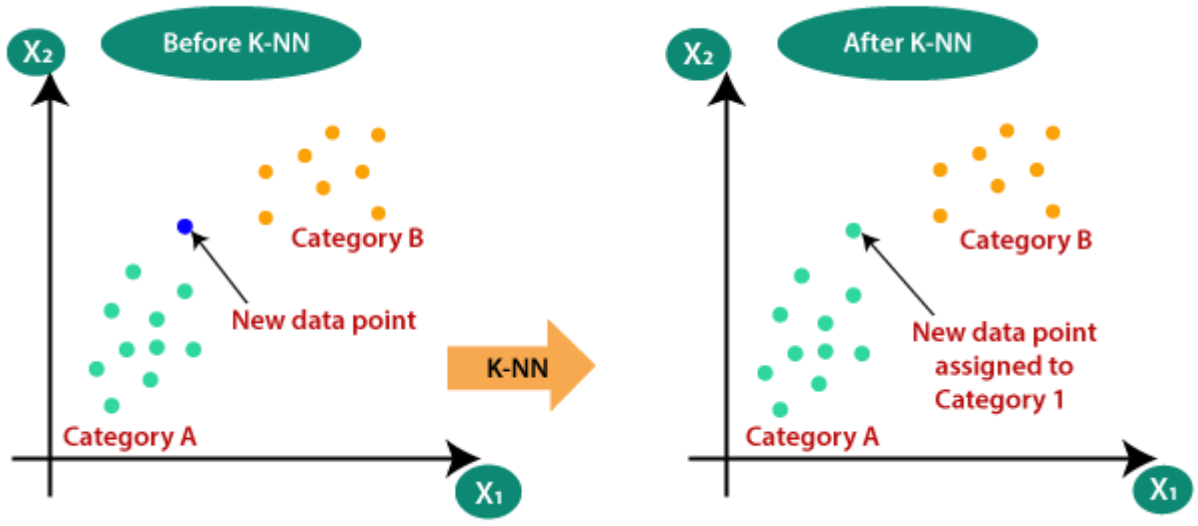
Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

KNeighboursclassifier

In [85]:
```
from sklearn.neighbors import KNeighborsClassifier
def kneighborsClassifier(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    yPred = knn.predict(x_test)
    print("****KNN****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=knn.predict(x_test)
    y_pred1=knn.predict(x_train)
    knn_test_accuracy = accuracy_score(y_test,y_pred)
    knn_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ',knn_test_accuracy)
    print('Training accuracy: ', knn_train_accuracy)

kneighborsClassifier(x_train, x_test, y_train, y_test)
```

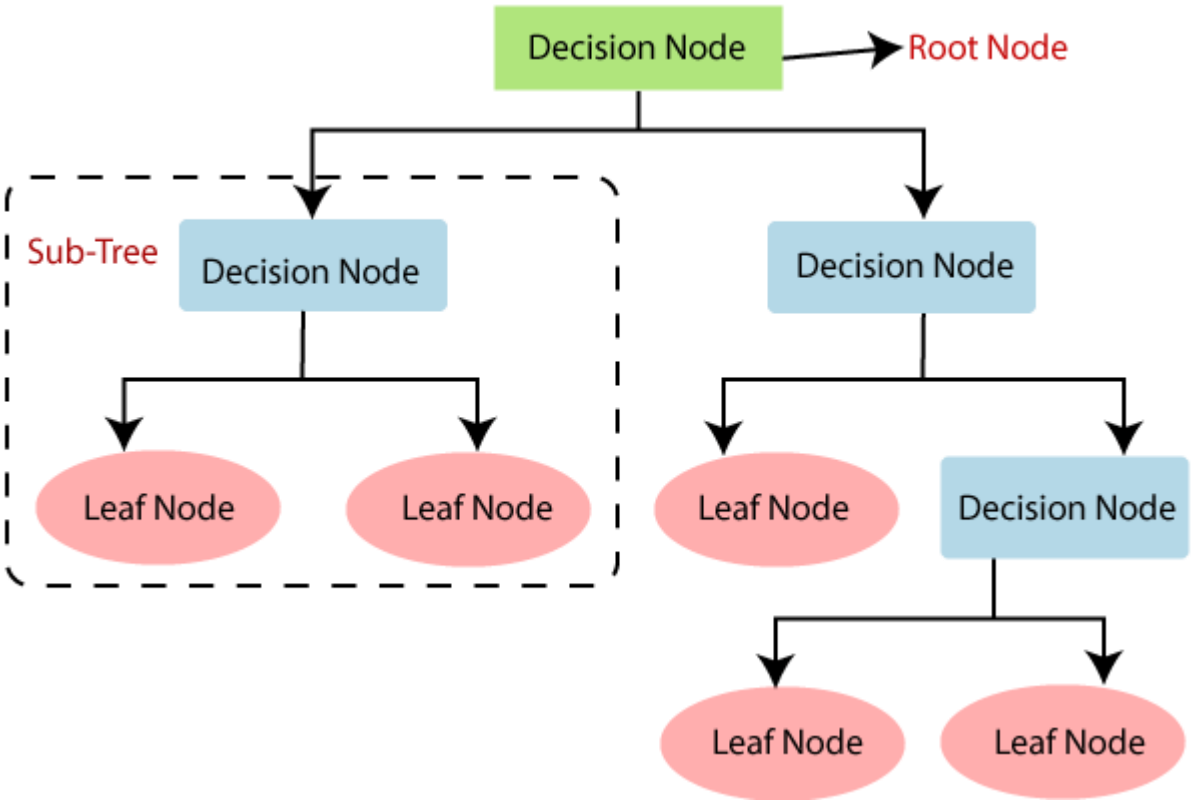
```
****KNN****
Confusion matrix
[[80 43]
 [13 99]]
Classification report
              precision    recall  f1-score   support

           0       0.86      0.65      0.74       123
           1       0.70      0.88      0.78       112

    accuracy                           0.76       235
   macro avg       0.78      0.77      0.76       235
weighted avg       0.78      0.76      0.76       235

Testing accuracy:  0.7617021276595745
Training accuracy:  0.8490566037735849
```



Decision Tree Classification Algorithm

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

> Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

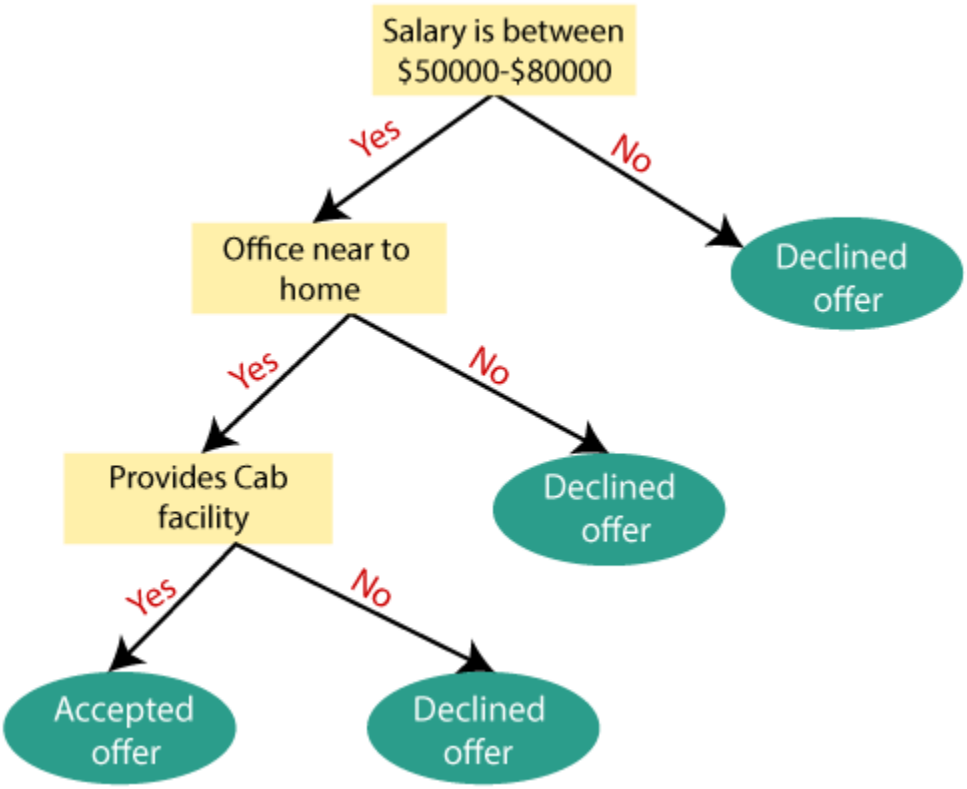
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$Entropy(s) = -P(yes)\log_2 P(yes) - P(no)\log_2 P(no)$$

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$Gini\ Index = 1 - \sum_j P_j^2$$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.

- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Decision tree classifier

In [86]:
```
def decisionTreeClassifier(x_train, x_test, y_train, y_test):
    dt = DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print("****DecisionTreeClassifier****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=dt.predict(x_test)
    y_pred1=dt.predict(x_train)
    decision_tree_test_accuracy = accuracy_score(y_test,y_pred)
    decision_tree_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ', decision_tree_test_accuracy)
    print('Training accuracy: ', decision_tree_train_accuracy)

decisionTreeClassifier(x_train, x_test, y_train, y_test)
```

```
****DecisionTreeClassifier****
Confusion matrix
[[95 28]
 [16 96]]
Classification report
              precision    recall  f1-score   support

           0       0.86      0.77      0.81       123
           1       0.77      0.86      0.81       112

    accuracy                           0.81       235
   macro avg       0.82      0.81      0.81       235
weighted avg       0.82      0.81      0.81       235

Testing accuracy:  0.8127659574468085
Training accuracy:  1.0
```

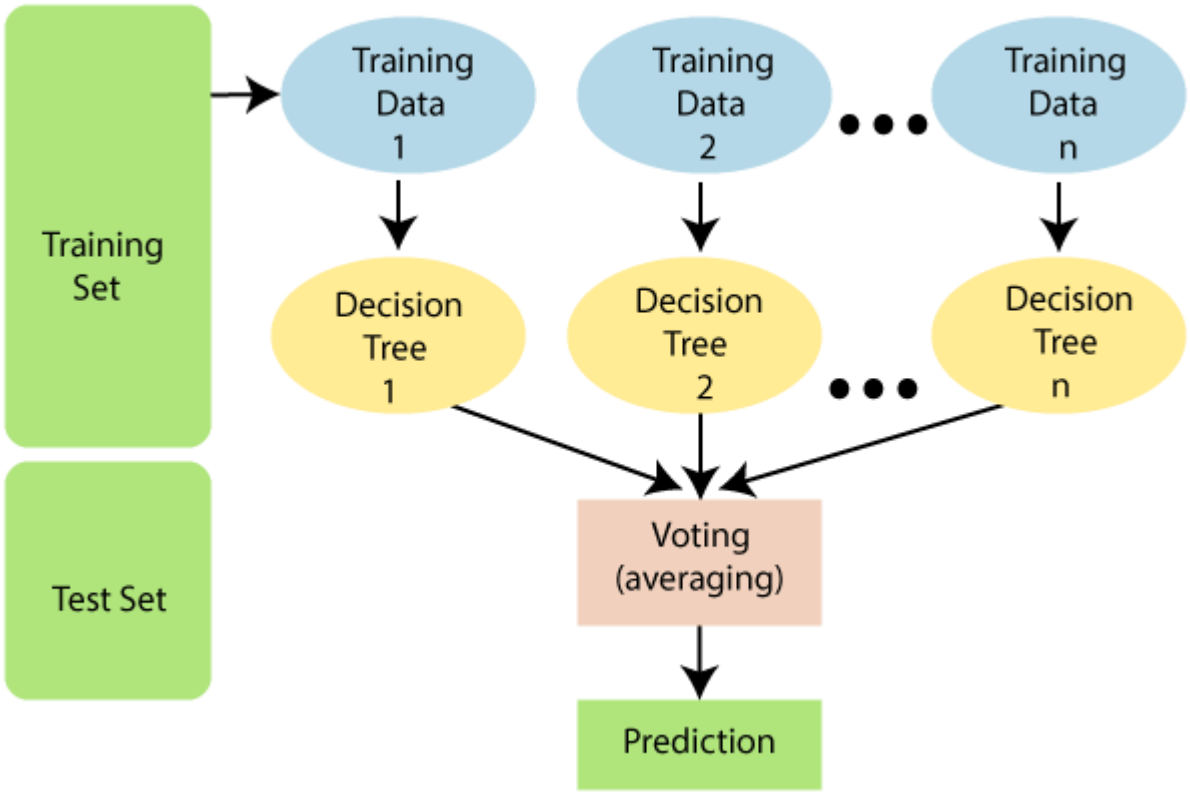
Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, **"*Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.*"** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



> Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<="" li="">

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

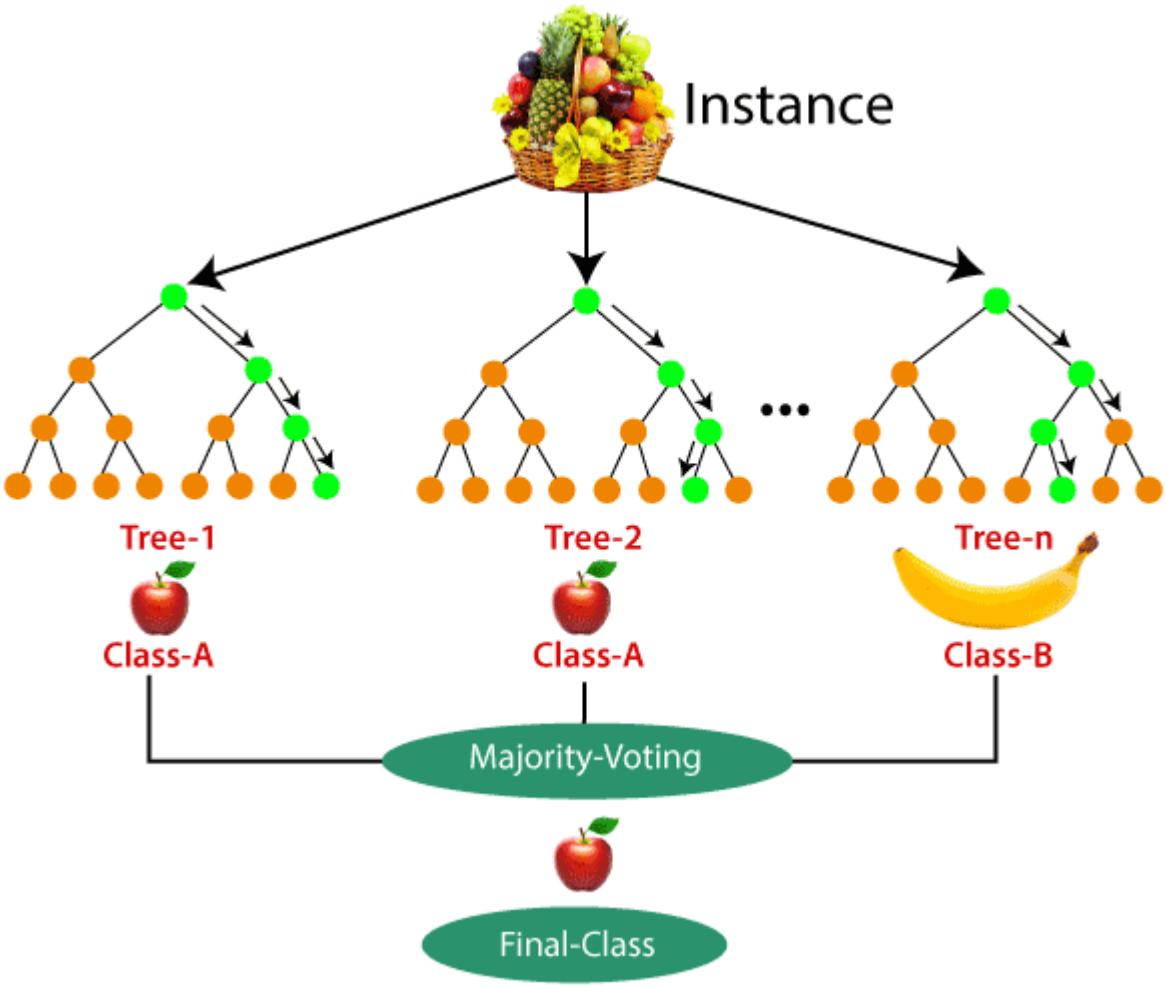
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Random forest classifier

In [87]:
```
def randomForestClassifier(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print("****RandomForestClassifier****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=rf.predict(x_test)
    y_pred1=rf.predict(x_train)
    random_forest_test_accuracy = accuracy_score(y_test,y_pred)
    random_forest_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ', random_forest_test_accuracy)
    print('Training accuracy: ',random_forest_train_accuracy)

randomForestClassifier(x_train, x_test, y_train, y_test)
```

```
****RandomForestClassifier****
Confusion matrix
[[90 33]
 [13 99]]
Classification report
              precision    recall  f1-score   support

           0       0.87      0.73      0.80       123
           1       0.75      0.88      0.81       112

    accuracy                           0.80       235
   macro avg       0.81      0.81      0.80       235
weighted avg       0.81      0.80      0.80       235

Testing accuracy:  0.8042553191489362
Training accuracy:  1.0
```

XGBoost Algorithm

XGBoost is a robust machine-learning algorithm that can help you understand your data and make better decisions.

XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models.

XGBoost is designed for speed, ease of use, and performance on large datasets. It does not require optimization of the parameters or tuning, which means that it can be used immediately after installation without any further configuration.

XGBoost Features

XGBoost is a widespread implementation of gradient boosting. Let’s discuss some features of XGBoost that make it so attractive.

- XGBoost offers regularization, which allows you to control overfitting by introducing L1/L2 penalties on the weights and biases of each tree. This feature is not available in many other implementations of gradient boosting.
- Another feature of XGBoost is its ability to handle sparse data sets using the weighted quantile sketch algorithm. This algorithm allows us to deal with non-zero entries in the feature matrix while retaining the same computational complexity as other algorithms like stochastic gradient descent.
- XGBoost also has a block structure for parallel learning. It makes it easy to scale up on multicore machines or clusters. It also uses cache awareness, which helps reduce memory usage when training models with large datasets.
- Finally, XGBoost offers out-of-core computing capabilities using disk-based data structures instead of in-memory ones during the computation phase.

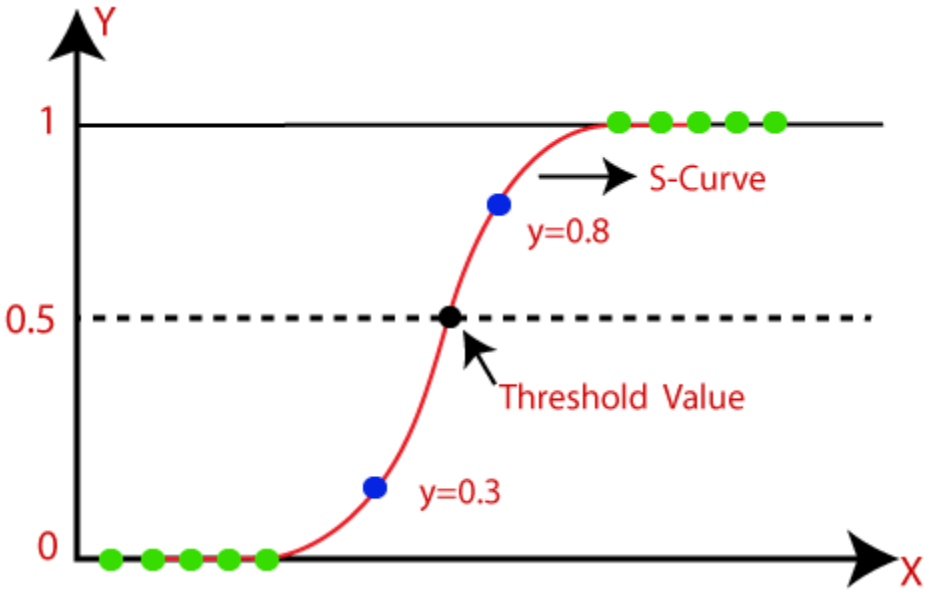
xgboost classifier

[88]:
```
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print("****Gradient BoostingClassifier****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=xg.predict(x_test)
    y_pred1=xg.predict(x_train)
    xgboost_test_accuracy = accuracy_score(y_test,y_pred)
    xgboost_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ', xgboost_test_accuracy)
    print('Training accuracy: ', xgboost_train_accuracy)

xgboost(x_train, x_test, y_train, y_test)
```

```
****Gradient BoostingClassifier****
Confusion matrix
[[ 86  37]
 [ 10 102]]
Classification report
              precision    recall  f1-score   support

           0       0.90      0.70      0.79       123
           1       0.73      0.91      0.81       112

    accuracy                           0.80       235
   macro avg       0.81      0.80      0.80       235
weighted avg       0.82      0.80      0.80       235

Testing accuracy:  0.8
Training accuracy:  0.9559748427672956
```

Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic

> regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

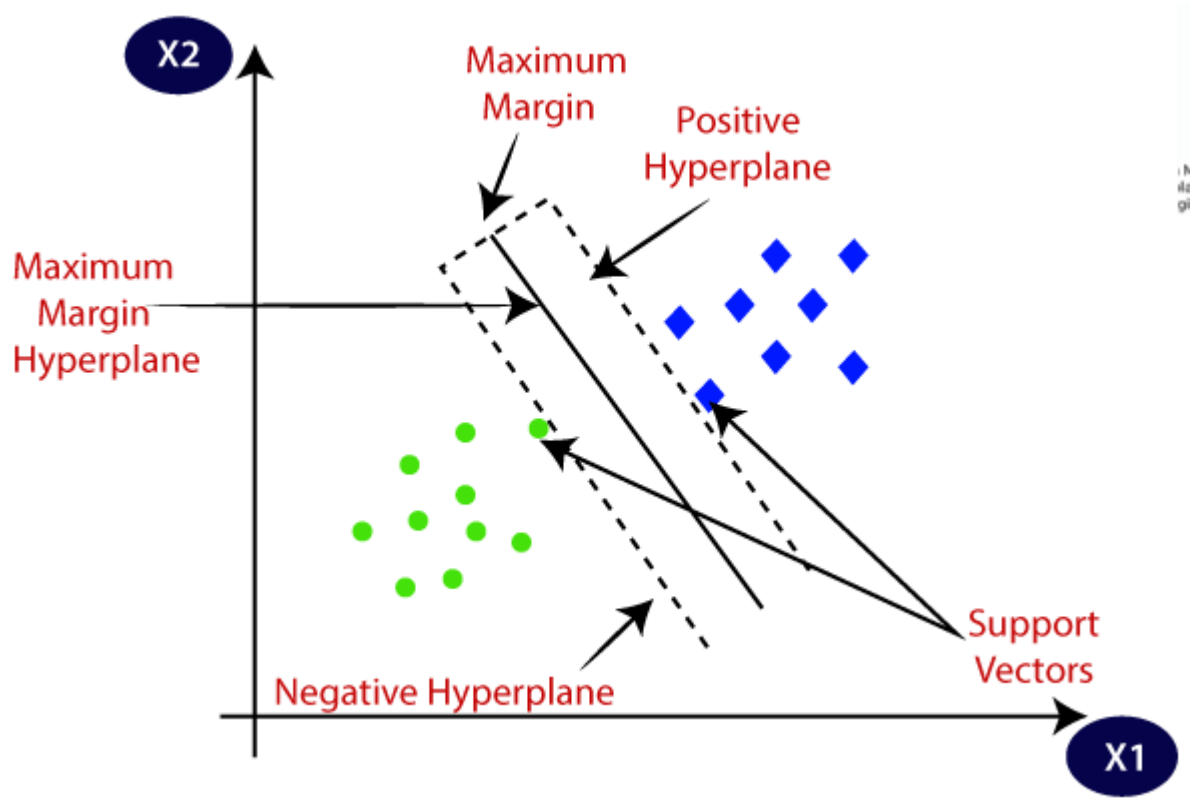
- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High"

Support Vector Machine Algorithm

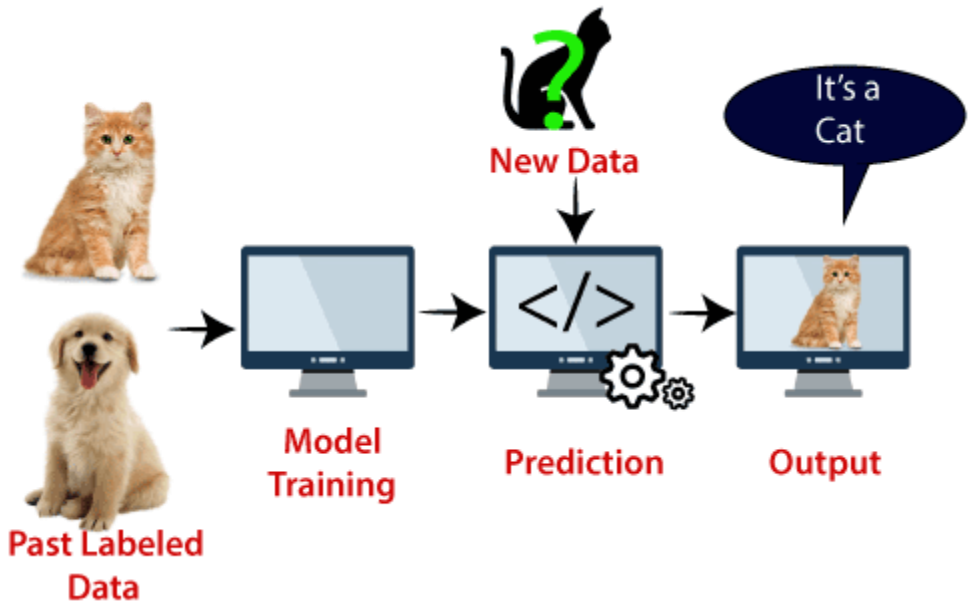
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

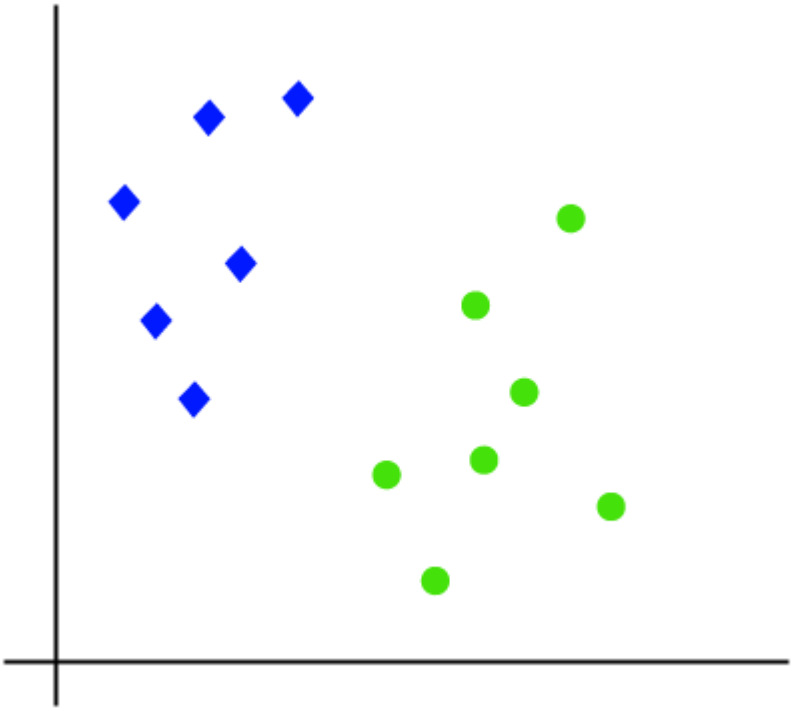
Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

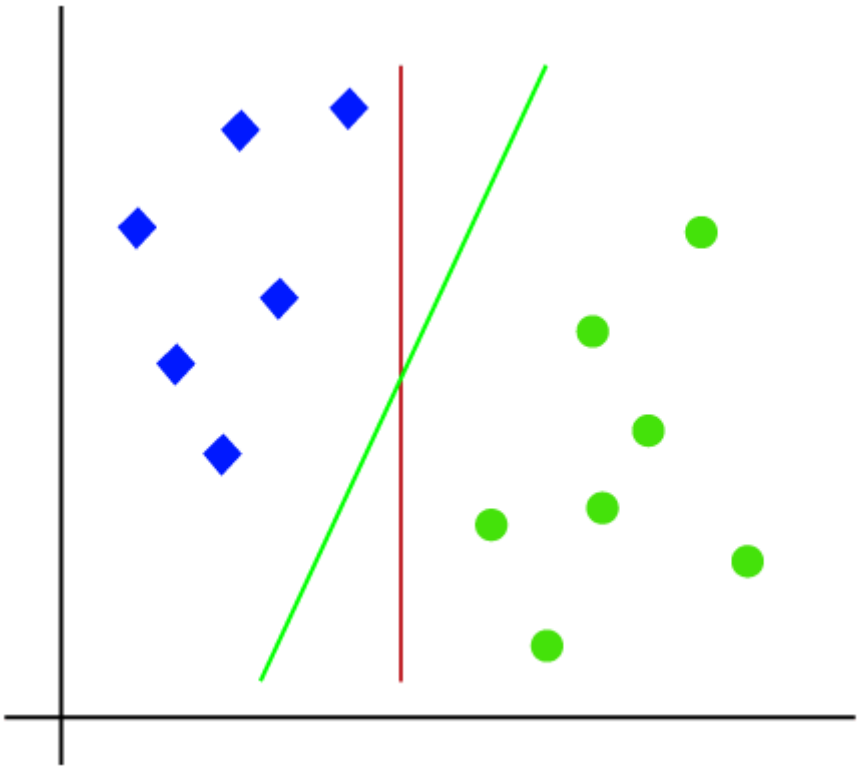
How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

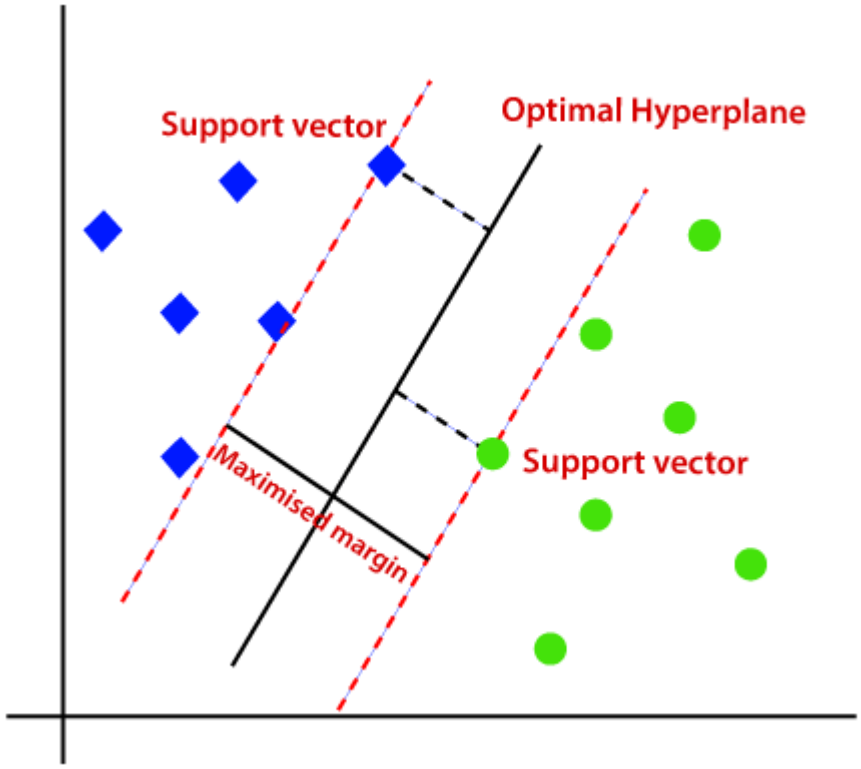


So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



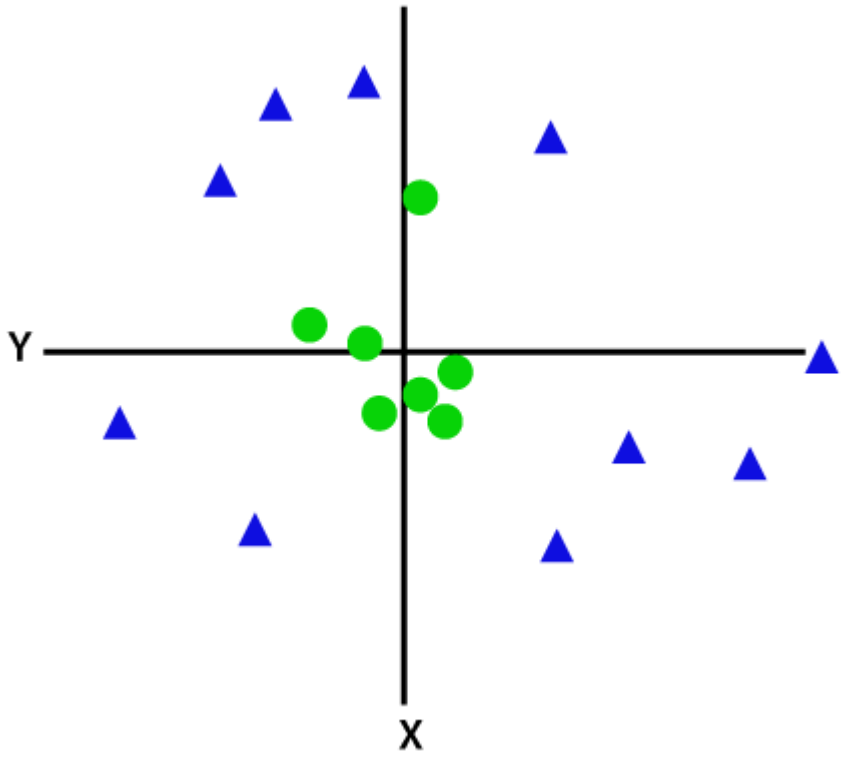
Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point

of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

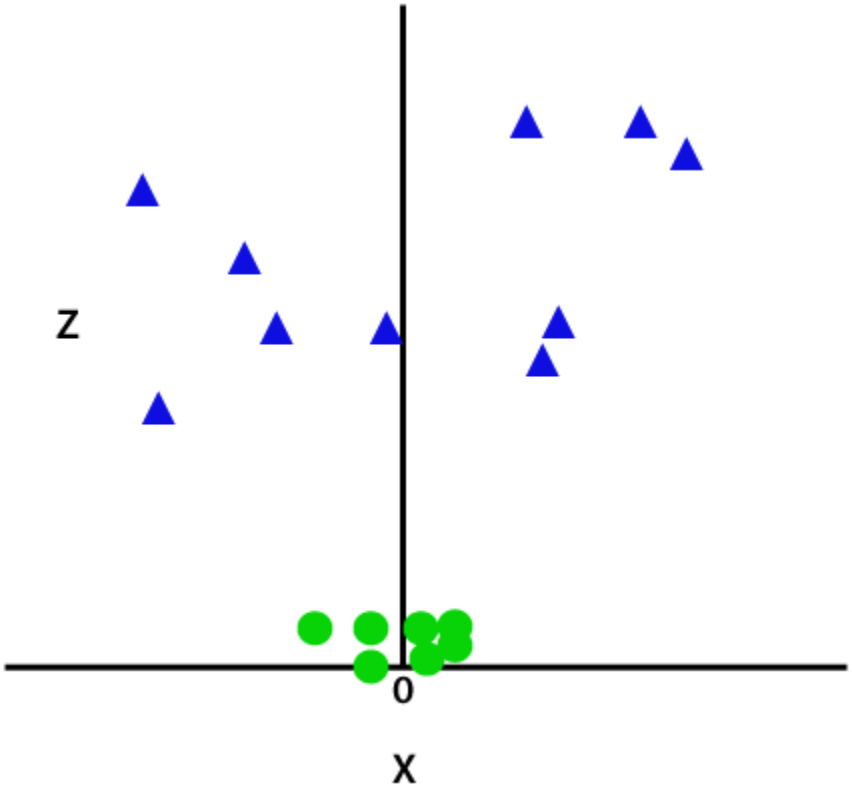
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



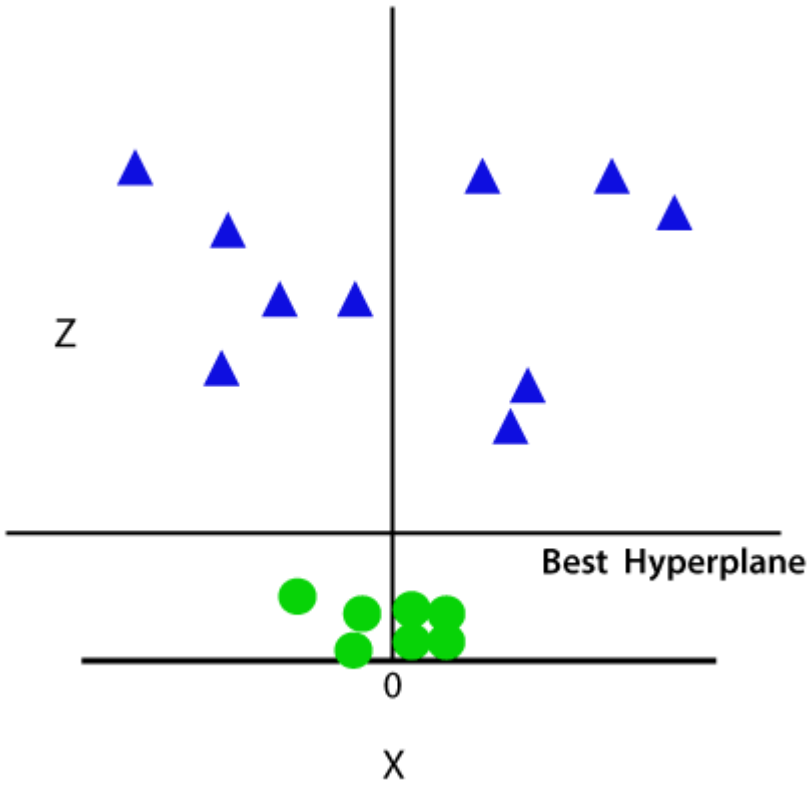
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2 +y^2$$

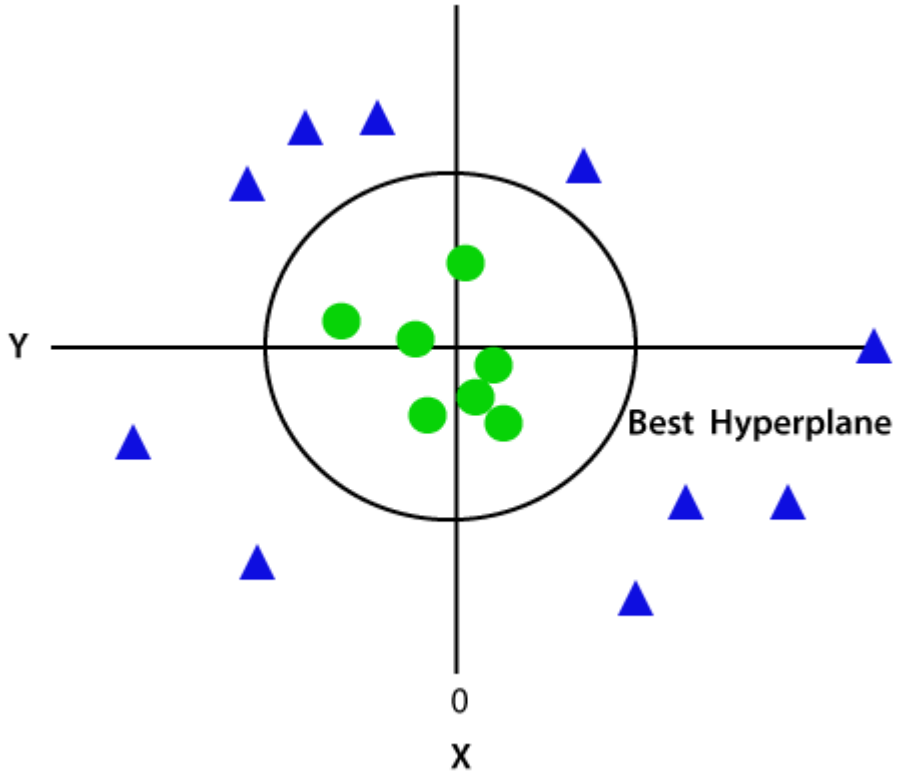
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

Python Implementation of Support Vector Machine

Now we will implement the SVM algorithm using Python. Here we will use the same dataset **user_data**, which we have used in Logistic regression and KNN classification.

Gaussian Naive Bayes (GNB)

Gaussian Naive Bayes (GNB) is a classification technique used in Machine Learning (ML) based on the probabilistic approach and Gaussian distribution. Gaussian Naive Bayes assumes that each parameter (also called features or predictors) has an independent capacity of predicting the output variable. The combination of the prediction for all parameters is the final prediction, that returns a probability of the dependent variable to be classified in each group.

The final classification is assigned to the group with the higher probability.

```
[128…
models={
    "Logistic Regression":LogisticRegression(),
    "svm":SVC(),
    "GaussianNB":GaussianNB()
}
for i in range(len(list(models))):
    model=list(models.values())[i]
    model.fit(x_train, y_train) #train model

    #make predictions
    y_train_pred =model.predict(x_train)
    y_test_pred =model.predict(x_test)

    #Training set performance
    model_train_accuracy=accuracy_score(y_train, y_train_pred)


    model_train_f1=f1_score(y_train, y_train_pred,average='weighted')



    #Testing set performance
    model_test_accuracy=accuracy_score(y_test, y_test_pred)

    model_test_f1=f1_score(y_test, y_test_pred,average='weighted')

    print(list(models.keys())[i])


    print('Model performance for Training set')
    print("- Accuracy: {:.4f}".format(model_train_accuracy))
    print("- F1 score: {:.4f}".format(model_train_f1))



    print('----------------------------------------------------------------------')
    print('Model performance for Testing set')
    print("- Accuracy: {:.4f}".format(model_test_accuracy))
    print("- F1 score: {:.4f}".format(model_test_f1))



    print('='*35)
    print('\n')
```

```
Logistic Regression
Model performance for Training set
- Accuracy: 0.7925
- F1 score: 0.7897
----------------------------------------------------------------------
Model performance for Testing set
- Accuracy: 0.7702
- F1 score: 0.7689
==================================


svm
Model performance for Training set
- Accuracy: 0.8344
- F1 score: 0.8302
----------------------------------------------------------------------
Model performance for Testing set
- Accuracy: 0.7660
- F1 score: 0.7593
==================================


GaussianNB
Model performance for Training set
- Accuracy: 0.7799
- F1 score: 0.7783
----------------------------------------------------------------------
Model performance for Testing set
- Accuracy: 0.7489
- F1 score: 0.7480
==================================
```

➢ PERFORMANCE TESTING

In [93]:
```
from sklearn.model_selection import cross_val_score
forest_reg = RandomForestClassifier(n_estimators = 10, criterion = "entropy", random_state = 42)
forest_reg.fit(x_train,y_train)
```

Out[93]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=42)
```

In [94]:
```
yPred = forest_reg.predict(x_test)
y_pred=forest_reg.predict(x_test)
y_pred1=forest_reg.predict(x_train)
random_forest_test_accuracy = accuracy_score(y_test,y_pred)
random_forest_train_accuracy = accuracy_score(y_train,y_pred1)
print('Testing accuracy: ', random_forest_test_accuracy)
print('Training accuracy: ',random_forest_train_accuracy)
f1_score(yPred,y_test, average='weighted')
cv = cross_val_score(forest_reg,x,y,cv=5)
np.mean(cv)
```

```
Testing accuracy:  0.7872340425531915
Training accuracy:  0.9853249475890985
```

Out[94]:
```
0.7638278022124483
```

➢ HYPER PARAMETER TUNING (SELECTING THE BEST MODEL)

When you're training machine learning models, each dataset and model needs a different set of hyperparameters, which are a kind of variable. The only way to determine these is through multiple experiments, where you pick a set of hyperparameters and run them through your model. This is called *hyperparameter tuning*. In essence, you're training your model sequentially with different sets of hyperparameters. This process can be manual, or you can pick one of several automated hyperparameter tuning methods.

Whichever method you use, you need to track the results of your experiments. You'll have to apply some form of statistical analysis, such as the loss function, to determine which set of hyperparameters gives the best result. Hyperparameter tuning is an important and computationally intensive process.

Hyperparameters are external configuration variables that data scientists use to manage machine learning model training. Sometimes called *model hyperparameters*, the hyperparameters are manually set before training a model. They're different from parameters, which are internal parameters automatically derived during the learning process and not set by data scientists.

Examples of hyperparameters include the number of nodes and layers in a neural network and the number of branches in a decision tree. Hyperparameters determine key features such as model architecture, learning rate, and model complexity.

Selecting the right set of hyperparameters is important in terms of model performance and accuracy. Unfortunately, there are no set rules on which hyperparameters work best nor their optimal or default values. You need to experiment to find the optimum hyperparameter set. This activity is known as *hyperparameter tuning* or *hyperparameter optimization*.

Hyperparameters directly control model structure, function, and performance. Hyperparameter tuning allows data scientists to tweak model performance for optimal results. This process is an essential part of machine learning, and choosing appropriate hyperparameter values is crucial for success.

For example, assume you're using the learning rate of the model as a hyperparameter. If the value is too high, the model may converge too quickly with suboptimal results. Whereas if the rate is too low, training takes too long and results may not converge. A good and balanced choice of hyperparameters results in accurate models and excellent model performance.

As previously stated, hyperparameter tuning can be manual or automated. While manual tuning is slow and tedious, a benefit is that you better understand how hyperparameter weightings affect the model. But in most instances, you would normally use one of the well-known hyperparameter learning algorithms.

The process of hyperparameter tuning is iterative, and you try out different combinations of parameters and values. You generally start by defining a target variable such as accuracy as the primary metric, and you intend to maximize or minimize this variable. It’s a good idea to use cross-validation techniques, so your model isn't centered on a single portion of your data.

Numerous hyperparameter tuning algorithms exist, although the most commonly used types are Bayesian optimization, grid search and randomized search.

Bayesian optimization

Bayesian optimization is a technique based on Bayes’ theorem, which describes the probability of an event occurring related to current knowledge. When this is applied to hyperparameter optimization, the algorithm builds a probabilistic model from a set of hyperparameters that optimizes a specific metric. It uses regression analysis to iteratively choose the best set of hyperparameters.

Grid search

With grid search, you specify a list of hyperparameters and a performance metric, and the algorithm works through all possible combinations to determine the best fit. Grid search works well, but it’s relatively tedious and computationally intensive, especially with large numbers of hyperparameters.

Random search

Although based on similar principles as grid search, random search selects groups of hyperparameters randomly on each iteration. It works

well when a relatively small number of the hyperparameters primarily determine the model outcome.

In [98]:
```
#Manual hyper parameter tuning
```

In [99]:
```
model=RandomForestClassifier(n_estimators=300,criterion='entropy',
                             max_features='sqrt',min_samples_leaf=10,random_state=100).fit(x_train,y_train)
predictions=model.predict(x_test)
print(confusion_matrix(y_test,predictions))
print(accuracy_score(y_test,predictions))
print(classification_report(y_test,predictions))
```

```
[[ 81  42]
 [  7 105]]
0.7914893617021277
              precision    recall  f1-score   support

           0       0.92      0.66      0.77       123
           1       0.71      0.94      0.81       112

    accuracy                           0.79       235
   macro avg       0.82      0.80      0.79       235
weighted avg       0.82      0.79      0.79       235
```

100…
```
#No hyper parameter tuning
def randomForestClassifier(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print("*****RandomForestClassifier*****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=rf.predict(x_test)
    y_pred1=rf.predict(x_train)
    random_forest_test_accuracy = accuracy_score(y_test,y_pred)
    random_forest_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ', random_forest_test_accuracy)
    print('Training accuracy: ',random_forest_train_accuracy)

randomForestClassifier(x_train, x_test, y_train, y_test)
```

```
****RandomForestClassifier****
Confusion matrix
[[ 90  33]
 [ 11 101]]
Classification report
              precision    recall  f1-score   support

           0       0.89      0.73      0.80       123
           1       0.75      0.90      0.82       112

    accuracy                           0.81       235
   macro avg       0.82      0.82      0.81       235
weighted avg       0.83      0.81      0.81       235

Testing accuracy:  0.8127659574468085
Training accuracy:  1.0
```

```
101  #Manual hyper parameter tuning
     def randomForestClassifier(x_train, x_test, y_train, y_test):
         rf = RandomForestClassifier(n_estimators=300,criterion='entropy',
                                     max_features='sqrt',min_samples_leaf=10,random_state=100)
         rf.fit(x_train,y_train)
         yPred = rf.predict(x_test)
         print("****RandomForestClassifier****")
         print("Confusion matrix")
         print(confusion_matrix(y_test ,yPred) )
         print("Classification report")
         print(classification_report (y_test, yPred))
         y_pred=rf.predict(x_test)
         y_pred1=rf.predict(x_train)
         random_forest_test_accuracy = accuracy_score(y_test,y_pred)
         random_forest_train_accuracy = accuracy_score(y_train,y_pred1)
         print('Testing accuracy: ', random_forest_test_accuracy)
         print('Training accuracy: ',random_forest_train_accuracy)

     randomForestClassifier(x_train, x_test, y_train, y_test)
```

```
****RandomForestClassifier****
Confusion matrix
[[ 81  42]
 [  7 105]]
Classification report
              precision    recall  f1-score   support

           0       0.92      0.66      0.77       123
           1       0.71      0.94      0.81       112

    accuracy                           0.79       235
   macro avg       0.82      0.80      0.79       235
weighted avg       0.82      0.79      0.79       235

Testing accuracy:  0.7914893617021277
Training accuracy:  0.8511530398322851
```

Randomized search cv

In [102…]
```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [2,4]
# Minimum number of samples required to split a node
min_samples_split = [2, 5]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
# Method of selecting samples for training each tree
bootstrap = [True, False]
```

In [103…]
```
# Create the param grid
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
print(param_grid)
```

```
{'n_estimators': [10, 17, 25, 33, 41, 48, 56, 64, 72, 80], 'max_features': ['auto', 'sqrt'], 'max_depth': [2, 4], 'min_samples_split': [2, 5], 'min
_samples_leaf': [1, 2], 'bootstrap': [True, False]}
```

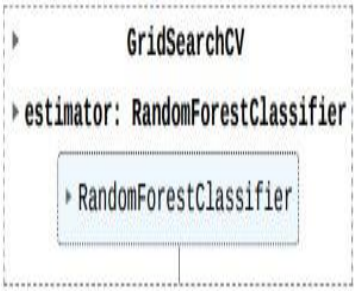
In [104…]
```
rf_Model = RandomForestClassifier()
```

In [105…]
```
from sklearn.model_selection import GridSearchCV
rf_Grid = GridSearchCV(estimator = rf_Model, param_grid = param_grid, cv = 10, verbose=2, n_jobs = 4)
```

In [106…]
```
rf_Grid.fit(x_train, y_train)
```

```
Fitting 10 folds for each of 320 candidates, totalling 3200 fits
```

Out[106]:



In [111…
```
rf_Grid.best_params_
```

Out[111]:
```
{'bootstrap': True,
 'max_depth': 4,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 5,
 'n_estimators': 33}
```

In [112…
```
rf_RandomGrid.best_params_
```

Out[112]:
```
{'n_estimators': 41,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 4,
 'bootstrap': False}
```

[113…
```
print (f'Train Accuracy - : {rf_Grid.score(x_train,y_train):.3f}')
print (f'Test Accuracy - : {rf_Grid.score(x_test,y_test):.3f}')
```

```
Train Accuracy - : 0.832
Test Accuracy - : 0.787
```

[114…
```
print (f'Train Accuracy - : {rf_RandomGrid.score(x_train,y_train):.3f}')
print (f'Test Accuracy - : {rf_RandomGrid.score(x_test,y_test):.3f}')
```

```
Train Accuracy - : 0.847
Test Accuracy - : 0.796
```

finally by doing hyper parameter tuning we see the results =>Based on that random forest is selected for model building

[122…
```
def randomForestClassifier(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print("****RandomForestClassifier****")
    print("Confusion matrix")
    print(confusion_matrix(y_test ,yPred) )
    print("Classification report")
    print(classification_report (y_test, yPred))
    y_pred=rf.predict(x_test)
    y_pred1=rf.predict(x_train)
    random_forest_test_accuracy = accuracy_score(y_test,y_pred)
    random_forest_train_accuracy = accuracy_score(y_train,y_pred1)
    print('Testing accuracy: ', random_forest_test_accuracy)
    print('Training accuracy: ',random_forest_train_accuracy)

randomForestClassifier(x_train, x_test, y_train, y_test)
```

```
****RandomForestClassifier****
Confusion matrix
[[ 92  31]
 [ 10 102]]
Classification report
              precision    recall  f1-score   support

           0       0.90      0.75      0.82       123
           1       0.77      0.91      0.83       112

    accuracy                           0.83       235
   macro avg       0.83      0.83      0.83       235
weighted avg       0.84      0.83      0.82       235

Testing accuracy:  0.825531914893617
Training accuracy:  1.0
```

MODEL DEPLOYMENT

For model deployment we need:

1.model.py
2.app.py
3.html and css
4.model.pkl
5.editor for deployment model using flask like visual studio code,spyder,pycharm

➢ App.py

Code-

```
from flask import Flask, escape, request, render_template
import pickle
import numpy as np

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template("index.html")


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method ==  'POST':
        gender = request.form['gender']
        married = request.form['married']
        dependents = request.form['dependents']
        education = request.form['education']
        employed = request.form['employed']
        credit = float(request.form['credit'])
        area = request.form['area']
        ApplicantIncome = float(request.form['ApplicantIncome'])
        CoapplicantIncome = float(request.form['CoapplicantIncome'])
        LoanAmount = float(request.form['LoanAmount'])
        Loan_Amount_Term = float(request.form['Loan_Amount_Term'])

        # gender
        if (gender == "Male"):
            male=1
        else:
            male=0

        # married
        if(married=="Yes"):
            married_yes = 1
        else:
            married_yes=0
```

```
# dependents
if(dependents=='1'):
    dependents_1 = 1
    dependents_2 = 0
    dependents_3 = 0
elif(dependents == '2'):
    dependents_1 = 0
    dependents_2 = 1
    dependents_3 = 0
elif(dependents=="3+"):
    dependents_1 = 0
    dependents_2 = 0
    dependents_3 = 1
else:
    dependents_1 = 0
    dependents_2 = 0
    dependents_3 = 0

# education
if (education=="Not Graduate"):
    not_graduate=1
else:
    not_graduate=0

# employed
if (employed == "Yes"):
    employed_yes=1
else:
    employed_yes=0

# property area

if(area=="Semiurban"):
    semiurban=1
    urban=0
elif(area=="Urban"):
    semiurban=0
    urban=1
else:
    semiurban=0
    urban=0
```

```
        ApplicantIncomelog = np.log(ApplicantIncome)
        totalincomelog = np.log(ApplicantIncome+CoapplicantIncome)
        LoanAmountlog = np.log(LoanAmount)
        Loan_Amount_Termlog = np.log(Loan_Amount_Term)

        prediction = model.predict([[credit, ApplicantIncomelog,LoanAmountlog,
Loan_Amount_Termlog, totalincomelog, male, married_yes, dependents_1,
dependents_2, dependents_3, not_graduate, employed_yes,semiurban, urban ]])

        # print(prediction)

        if(prediction=="N"):
            prediction="No"
        else:
            prediction="Yes"


        return render_template("prediction.html", prediction_text="loan status is
{}".format(prediction))




    else:
        return render_template("prediction.html")



if __name__ == "__main__":
    app.run(debug=True)
```

Model.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("train.csv")
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] =
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] =
df['Credit_History'].fillna(df['Credit_History'].mean())
df['Gender'].mode()[0]
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] =
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
df.isnull().sum()
df['Total_income'] = df['ApplicantIncome']+df['CoapplicantIncome']
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
df['LoanAmountLog'] = np.log(df['LoanAmount'])
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
df['Total_Income_Log'] = np.log(df['Total_income'])
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount",
"Loan_Amount_Term", "Total_income", 'Loan_ID', 'CoapplicantIncomeLog']
df = df.drop(columns=cols, axis=1)
d1 = pd.get_dummies(df['Gender'], drop_first= True)
d2 = pd.get_dummies(df['Married'], drop_first= True)
d3 = pd.get_dummies(df['Dependents'], drop_first= True)
d4 = pd.get_dummies(df['Education'], drop_first= True)
d5 = pd.get_dummies(df['Self_Employed'], drop_first= True)
d6 = pd.get_dummies(df['Property_Area'], drop_first= True)



df1 = pd.concat([df, d1, d2, d3, d4, d5, d6], axis = 1)
df=df1

cols = ['Gender', 'Married', "Dependents", "Education", "Self_Employed",
'Property_Area']
df = df.drop(columns=cols, axis=1)
test = pd.read_csv("test.csv")
# filling numerical missing data
test['LoanAmount']=test['LoanAmount'].fillna(test['LoanAmount'].mean())
test['Loan_Amount_Term']=test['Loan_Amount_Term'].fillna(test['Loan_Amount_
Term'].mean())
test['Credit_History']=test['Credit_History'].fillna(test['Credit_History']
.mean())

# filling categorical missing data
test['Gender']=test['Gender'].fillna(test['Gender'].mode()[0])
test['Married']=test['Married'].fillna(test['Married'].mode()[0])
test['Dependents']=test['Dependents'].fillna(test['Dependents'].mode()[0])
test['Self_Employed']=test['Self_Employed'].fillna(test['Self_Employed'].mo
de()[0])

test['Total_income'] = test['ApplicantIncome']+test['CoapplicantIncome']

# apply log transformation to the attribute
test['ApplicantIncomeLog'] = np.log(test['ApplicantIncome'])

test['CoapplicantIncomeLog'] = np.log(test['CoapplicantIncome'])
```

```
test['LoanAmountLog'] = np.log(test['LoanAmount'])

test['Loan_Amount_Term_Log'] = np.log(test['Loan_Amount_Term'])

test['Total_Income_Log'] = np.log(test['Total_income'])

cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount",
"Loan_Amount_Term", "Total_income", 'Loan_ID', 'CoapplicantIncomeLog']
test = test.drop(columns=cols, axis=1)

t1 = pd.get_dummies(test['Gender'], drop_first= True)
t2 = pd.get_dummies(test['Married'], drop_first= True)
t3 = pd.get_dummies(test['Dependents'], drop_first= True)
t4 = pd.get_dummies(test['Education'], drop_first= True)
t5 = pd.get_dummies(test['Self_Employed'], drop_first= True)
t6 = pd.get_dummies(test['Property_Area'], drop_first= True)

df1 = pd.concat([test, t1, t2, t3, t4, t5, t6], axis = 1)
test=df1

cols = ['Gender', 'Married', "Dependents", "Education", "Self_Employed",
'Property_Area']
test = test.drop(columns=cols, axis=1)
x = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()

model.fit(x_train, y_train)
print("Accuracy is", model.score(x_test, y_test)*100)
from sklearn.tree import DecisionTreeClassifier
model2 = DecisionTreeClassifier()
model2.fit(x_train, y_train)
print("Accuracy is", model2.score(x_test, y_test)*100)
from sklearn.linear_model import LogisticRegression
model3 = LogisticRegression()
model3.fit(x_train, y_train)
print("Accuracy is", model3.score(x_test, y_test)*100)
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
import pickle
file=open("model.pkl", 'wb')
pickle.dump(model, file)
```

➢ Html and css code

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpb
zKgwra6" crossorigin="anonymous">
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
rel="stylesheet">

    <title>loan Prediction</title>
  </head>
  <body>

<!-- This example requires Tailwind CSS v2.0+ -->
<div class="relative bg-white overflow-hidden">
  <div class="max-w-7xl mx-auto">
    <div class="relative z-10 pb-8 bg-white sm:pb-16 md:pb-20 lg:max-w-
2xl lg:w-full lg:pb-28 xl:pb-32">
      <svg class="hidden lg:block absolute right-0 inset-y-0 h-full w-48
text-white transform translate-x-1/2" fill="currentColor" viewBox="0 0
100 100" preserveAspectRatio="none" aria-hidden="true">
        <polygon points="50,0 100,0 50,100 0,100" />
      </svg>

      <div class="relative pt-6 px-4 sm:px-6 lg:px-8">
        <nav class="relative flex items-center justify-between sm:h-10
lg:justify-start" aria-label="Global">
          <div class="flex items-center flex-grow flex-shrink-0 lg:flex-grow-
0">
            <div class="flex items-center justify-between w-full md:w-auto">
              <a href="#">
                <span class="sr-only">Workflow</span>
                <img class="h-8 w-auto sm:h-10"
src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg">
```

```
            </a>
            <div class="-mr-2 flex items-center md:hidden">
              <button type="button" class="bg-white rounded-md p-2 inline-flex items-center justify-center text-gray-400 hover:text-gray-500 hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-inset focus:ring-indigo-500" aria-expanded="false">
                <span class="sr-only">Open main menu</span>
                <!-- Heroicon name: outline/menu -->
                <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-hidden="true">
                  <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
                </svg>
              </button>
            </div>
          </div>
        </div>
        <div class="hidden md:block md:ml-10 md:pr-4 md:space-x-8">
          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">Home</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">Prediction</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">About us</a>

          <a href="#" class="font-medium text-gray-500 hover:text-gray-900">contact</a>

        </div>
      </nav>
    </div>

    <!--
      Mobile menu, show/hide based on menu open state.

      Entering: "duration-150 ease-out"
        From: "opacity-0 scale-95"
        To: "opacity-100 scale-100"
      Leaving: "duration-100 ease-in"
        From: "opacity-100 scale-100"
```

```
        To: "opacity-0 scale-95"
      -->
       <div class="absolute top-0 inset-x-0 p-2 transition transform origin-top-right md:hidden">
         <div class="rounded-lg shadow-md bg-white ring-1 ring-black ring-opacity-5 overflow-hidden">
           <div class="px-5 pt-4 flex items-center justify-between">
             <div>
               <img class="h-8 w-auto" src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg" alt="">
             </div>
             <div class="-mr-2">
               <button type="button" class="bg-white rounded-md p-2 inline-flex items-center justify-center text-gray-400 hover:text-gray-500 hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-inset focus:ring-indigo-500">
                 <span class="sr-only">Close main menu</span>
                 <!-- Heroicon name: outline/x -->
                 <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-hidden="true">
                   <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12" />
                 </svg>
               </button>
             </div>
           </div>
         <div class="px-2 pt-2 pb-3 space-y-1">
           <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-50">Home</a>

           <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-50">prediction</a>

           <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-50">about us</a>

           <a href="#" class="block px-3 py-2 rounded-md text-base font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-
```

```
50">contact</a>
        </div>

       </div>
     </div>

     <main class="mt-10 mx-auto max-w-7xl px-4 sm:mt-12 sm:px-6
md:mt-16 lg:mt-20 lg:px-8 xl:mt-28">
       <div class="sm:text-center lg:text-left">
         <h1 class="text-4xl tracking-tight font-extrabold text-gray-900
sm:text-5xl md:text-6xl">
           <span class="block xl:inline">Loan Prediction</span>
           <span class="block text-indigo-600 xl:inline">Machine Learnig
</span>
         </h1>
         <p class="mt-3 text-base text-gray-500 sm:mt-5 sm:text-lg
sm:max-w-xl sm:mx-auto md:mt-5 md:text-xl lg:mx-0">
           Lorem ipsum dolor sit amet consectetur adipisicing elit. Excepturi
ad perspiciatis dolores, deleniti culpa odit dolorem harum dolore ex amet.
         </p>
         <div class="mt-5 sm:mt-8 sm:flex sm:justify-center lg:justify-
start">
           <div class="rounded-md shadow">
             <a href="./predict" class="w-full flex items-center justify-center
px-8 py-3 border border-transparent text-base font-medium rounded-md
text-white bg-indigo-600 hover:bg-indigo-700 md:py-4 md:text-lg
md:px-10">
               Prediction
             </a>
           </div>

         </div>
       </div>
     </main>
   </div>
  </div>
  <div class="lg:absolute lg:inset-y-0 lg:right-0 lg:w-1/2">
    <img class="h-56 w-full object-cover sm:h-72 md:h-96 lg:w-full lg:h-
full" src="https://images.unsplash.com/photo-1551434678-
e076c223a692?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=285
0&q=80" alt="">
  </div>
```

```
</div>



    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2v
I5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

  </body>
</html>
```

- Prediction:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpb
zKgwra6" crossorigin="anonymous">
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
rel="stylesheet">

    <title>prediction</title>
  </head>
  <body>
  <section class="text-gray-600 body-font">
  <div class="container px-5 py-24 mx-auto">
    <div class="flex flex-col text-center w-full mb-20">

      <h1 class="sm:text-3xl text-2xl font-medium title-font mb-4 text-
gray-900">Loan prediction project</h1>
      <p class="lg:w-2/3 mx-auto leading-relaxed text-base">fill the form
for prediction</p>
      <br><br><h2><b>{{prediction_text}}</b></h2>
```

```
    </div>
    <div>

    </div>
    <a class="btn btn-primary" href="./" role="button">Back</a>
<form action='/predict' method='POST'>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">
gender</label>
  <select class="form-select" id="gender" name="gender" aria-
label="Default select example">
  <option selected>-- select gender --</option>
  <option value="Male">Male</option>
  <option value="Female">Female</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label"> married
status</label>
  <select class="form-select" id="married" name="married" aria-
label="Default select example">
  <option selected>-- select married status --</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-
label">Dependents</label>
  <select class="form-select" id="dependents" name="dependents" aria-
label="Default select example">
  <option selected>-- select dependents --</option>
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3+">3+</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-
label">Education</label>
  <select class="form-select" id="education" name="education" aria-
label="Default select example">
```

```
  <option selected>-- select education --</option>
  <option value="Graduate">Graduate</option>
  <option value="Not Graduate">Not Graduate</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-
label">Self_Employed</label>
  <select class="form-select" id="employed" name="employed" aria-
label="Default select example">
  <option selected>-- select Self_Employed --</option>
  <option value="Yes">Yes</option>
  <option value="No">No</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-
label">Credit_History</label>
  <select class="form-select" id="credit" name="credit" aria-
label="Default select example">
  <option selected>-- select Credit_History --</option>
  <option value="1.000000">1.000000</option>
  <option value="0.000000">0.000000</option>
  <option value="0.842199">0.842199</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-
label">Property_Area</label>
  <select class="form-select" id="area" name="area" aria-label="Default
select example">
  <option selected>-- select Property_Area --</option>
  <option value="Semiurban">Semiurban</option>
  <option value="Urban">Urban</option>
  <option value="Rural">Rural</option>
</select>
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Enter
ApplicantIncome</label>
  <input type="text" class="form-control" id="ApplicantIncome"
name="ApplicantIncome" placeholder="ApplicantIncome">
</div>
```

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Enter
CoapplicantIncome</label>
  <input type="text" class="form-control" id="CoapplicantIncome"
name="CoapplicantIncome" placeholder="CoapplicantIncome">
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Enter
LoanAmount</label>
  <input type="text" class="form-control" id="LoanAmount"
name="LoanAmount" placeholder="LoanAmount">
</div>
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Enter
Loan_Amount_Term</label>
  <input type="text" class="form-control" id="Loan_Amount_Term"
name="Loan_Amount_Term" placeholder="Loan_Amount_Term">
</div>




    <button type="submit" class="btn btn-primary">Predict</button>
    </form>


  </div>
</section>


    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2v
I5DkLtS3qm9Ekf" crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper
.min.js" integrity="sha384-
```

```
SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1
RDvt3wT4gWFG" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/js/bootstrap.min.js" integrity="sha384-
j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQ
qNXNE6DZiVc" crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

----------PROJECT END---------