Data Science (Predicting Loan Approval)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

submitted by
PARASA VENKATA VAIBHAV(221810302043)

under the guidance of
Mrs.V. Rekha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT(GITAM)

(Declared as Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS September-2021

DECLARATION

I submit this summer internship entitled "**Predicting of Loan Approval**" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology" in "Computer Science Engineering". I declare that it was carried out independently by me under the guidance of Mrs. V. Rekha, GITAM (Deemed To Be University), Hyderabad, India. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD **PARASA VENKATA VAIBHAV**

Date:02-11-2021 221810302043

GITAM CERTIFICATE

Department of Computer Science and Engineering

GITAM(Deemed to be University)
Rudraram Mandal, Sangareddy district, Patancheruvu, Hyderabad, Telangana 502329

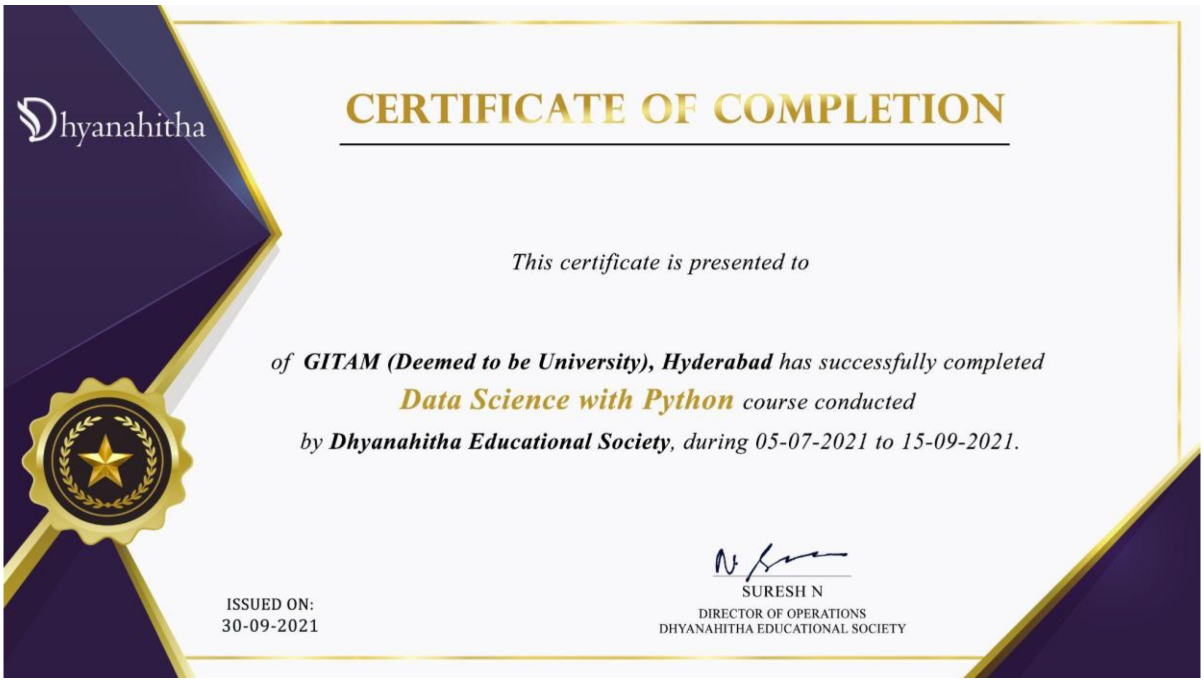
CERTIFICATE

This is to certify that the Summer Internship Report entitled **“Prediction of Loan Approval”** is being submitted by P.V.Vaibhav (221810302043) in partial fulfillment of the requirement for the award of **Bachelor of Technology in “COMPUTER SCIENCE ENGINEERING”** at GITAM University, Hyderabad.

It is faithful record work carried out by him at the **Computer Science Engineering Department**, GITAM School of Technology, GITAM Deemed to be University ,Hyderabad Campus under my guidance and supervision.

Mrs. V. Rekha	**Prof. S. Phani Kumar**
Assistant Professor	Professor and HOD
Department of CSE	Department of CSE

INTERNSHIP CERTIFICATE



Dhyanahitha

CERTIFICATE OF COMPLETION

This certificate is presented to

of ***GITAM (Deemed to be University), Hyderabad*** *has successfully completed* ***Data Science with Python*** *course conducted by* ***Dhyanahitha Educational Society****, during 05-07-2021 to 15-09-2021.*

ISSUED ON:
30-09-2021

[illegible]
SURESH N
DIRECTOR OF OPERATIONS
DHYANAHITHA EDUCATIONAL SOCIETY

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. we would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honorable Pro-Vice Chancellor, Prof. N. Siva Prasad for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to Prof. N. Seetha Ramaiah, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved Prof. S. Phani Kumar, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to Mrs. V. Rekha, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by him for the success of the Internship Project.

We are also thankful to all the staff members of the Computer Science and Engineering department who have cooperated in making our Internship Project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our internship Project.

Sincerely

PARASA VENKATA VAIBHAV

ABSTRACT

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. The concepts of data science extend but not limited to machine learning, deep learning and artificial intelligence. Through data science the unwanted data can be eliminated easily, which means clean data can obtained very swiftly. The relation among parameters can also found out easily with not much effort being applied. In the project of car classification evaluation, the concepts of data science play an important role as the main aim is to predict the "decision" parameter.

In the project every parameter is mapped with "decision" parameter in order to find the correlation among them. Also, the proportions of all parameters are distributed to get a clear understanding of density of values. This helps us approach a method to make predictions more accurate and suitable.

Index

1. Information About Data Science

1.1 What is Data Science

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining, deep learning and big data.

1.2 Need of Data Science

Companies need to use data to run and grow their everyday business. The fundamental goal of data science is to help companies make quicker and better decisions, which can take them to the top of their market, or at least – especially in the toughest red oceans – be a matter of long-term survival

Industries require data scientists to assist them in making a smarter decision. To predict the information everyone requires data scientists. Big Data and Data Science hold the key to the future. Data Science is important for better marketing.

1.3 Uses of Data Science

1. Internet Search
2. Digital Advertisements(Targeted Advertising and re-targeting)
3. Website Recommender Systems
4. Advanced Image Recognition
5. Speech Recognition
6. Gaming
7. Price Comparison Websites
8. Airline Route Planning
9. Fraud and Risk Detection

10. Delivery Logistics

Apart from the applications mentioned above, data science is also used in Marketing, Finance, Human Resources, Health Care, Government, Augmented Reality, Robots, Self-Driving Cars.

2. Information About Machine Learning

2.1 Introduction:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

2.2 Importance of Machine Learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, cybersex fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works.

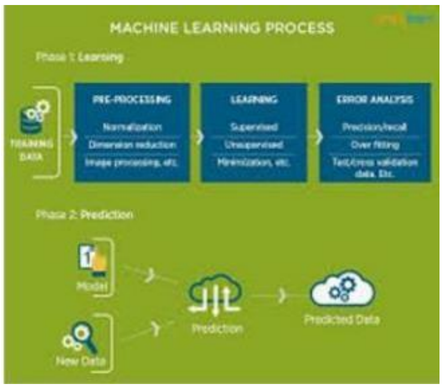


Fig 2.2.1 : The Process Flow

2.3 Uses of Machine Learning

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data. By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

2.4 Types of Machine Learning Algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

2.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labeled training data set – that is, a data set that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multi class classification.

2.4.2 Unsupervised Learning :

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

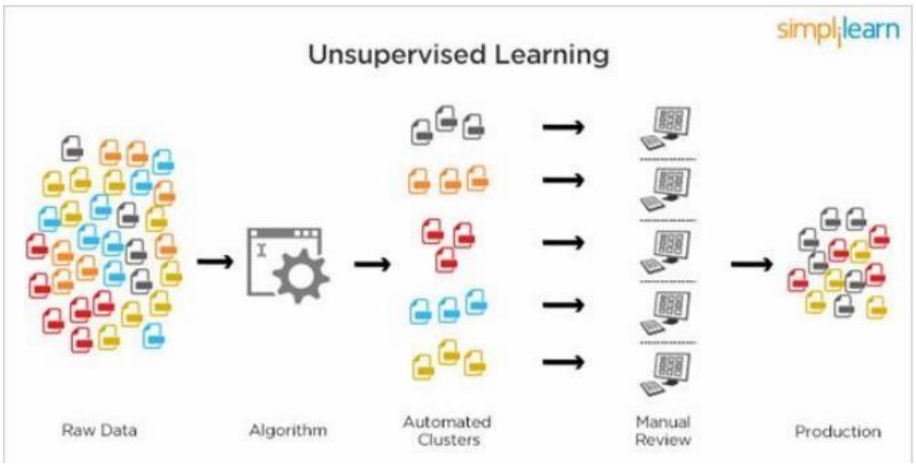


Fig 2.4.2.1: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

2.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

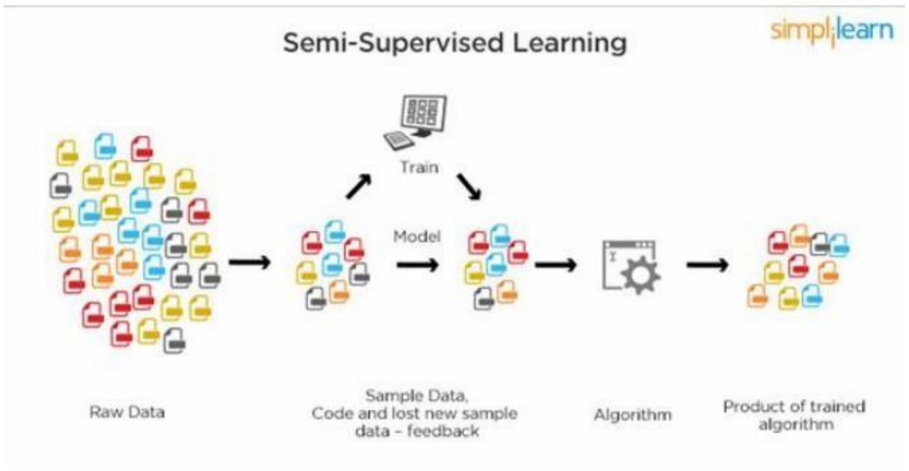

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Fig2.4.3.1: Semi Supervised Learning

2.5 Relation between Data Mining, Machine Learning and Deep Learning

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

3. INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 Introduction

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles.
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

3.2 History of Python

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

3.3 Features of Python

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

● Databases: Python provides interfaces to all major commercial databases.

● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 Python Setup

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

3.4.1 Installation(using python IDLE):

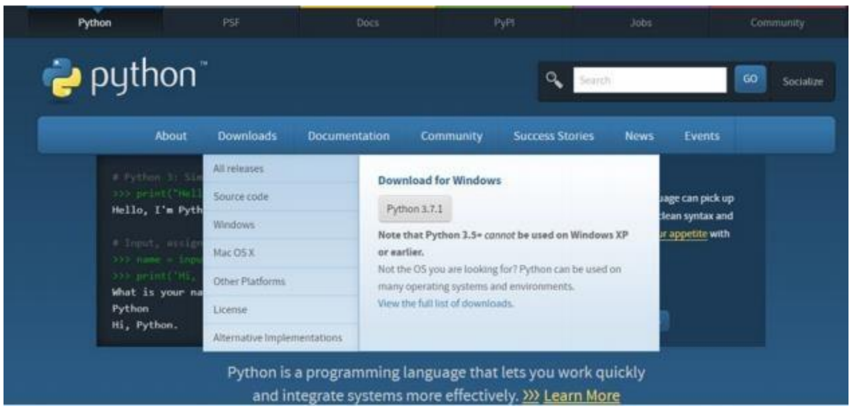
● Installing python is generally easy, and nowadays many Linux and Mac OS

distributions include a recent python.

● Download python from www.python.org

● When the download is completed, double click the file and follow the instructions to install it.

● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



```
Hello, I'm Pyth
What is your na
Python
Hi, Python.
```

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Fig 3.4.1.1: Python download

3.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- In windows
  - Step 1: Open Anaconda.com/downloads in web browser.
  - Step 2: Download python 3.4 version for

(32-bitgraphic installer/64 -bit graphic installer)

  - Step 3: select installation type(all users)
  - Step 4: Select path

(i.e. add anaconda to path & register anaconda as default python 3.4)

next click install and next click finish.

  - Step 5: Open jupyter notebook (it opens in default browser)

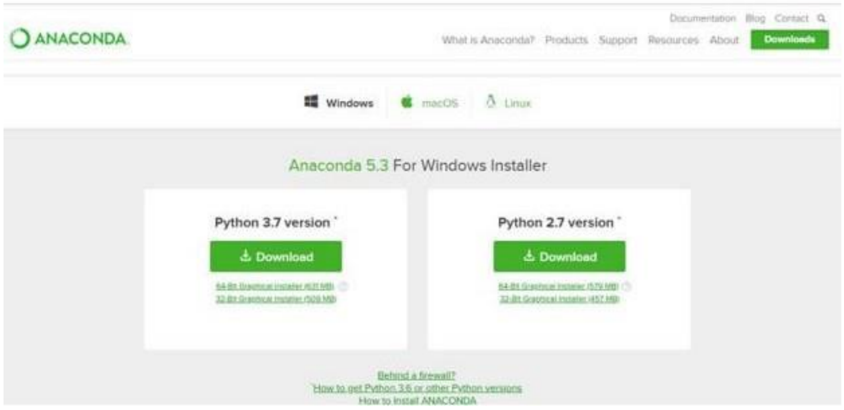


Fig 3.4.2.1: Anaconda download

3.5 Python Variable Types

- Variables are nothing but reserved memory locations to store values.This means that when you create a variable you reserve some space in memory.

- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
    - Numbers
    - Strings
    - Lists
    - Tuples
    - Dictionary

3.5.1 Python Numbers :

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3.5.3 Python Lists:

● Lists are the most versatile of Python's compound data types.

● A list contains items separated by commas and enclosed within square brackets ([]).

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

3.5.5 Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 Python Functions

3.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()). Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.7 Oops Concepts

3.7.1 Class :

● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

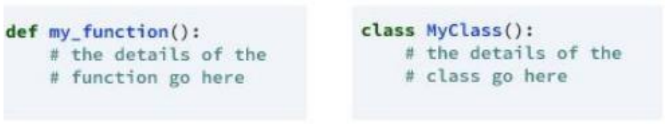
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

● Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

● Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name (i.e.():) when we define a class. Similarly, the body of our class is indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Fig 3.6.2.1: Defining a Class

3.7.2 __init__method in Class:

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores : __init_().

4. Project Name : Loan Approval Prediction

4.1 Project Requirements

4.1.1 Packages used.

- ✓ Pandas
- ✓ Matplotlib
- ✓ Numpy
- ✓ Seaborn
- ✓ Scikit-learn

4.1.2 Versions of the packages.

- ✓ Pandas : 1.3.3
- ✓ Matplotlib : 3.4.3
- ✓ Numpy : 1.21.2
- ✓ Seaborn:0.10.1
- ✓ Scikit-learn:0.23.1

4.1.3 Algorithms used.

- ✓ Logistic Regression Algorithm
- ✓ Ensemble Algorithm: Random forest

4.2 Problem Statement

To accurately predict which customer's loan and should approve and which to reject, in order to minimize the risk of loan default.

4.3 Data set Description

- “Loan_ID” : Unique Loan ID
- “Gender” : Male/Female
- “Married” : Applicant married(Y/N)
- “Dependents” : Number of dependents (0, 1, 2, 3+)
- “Education” : Applicant Education (Graduate / Under Graduate)
- “Self_Employed” : Self-employed (Y/N)
- “ApplicantIncome” : Applicant income

- “CoapplicantIncome” : Coapplicant income
- “LoanAmount” : Loan amount in thousands
- “Loan_Amount_Term” : Term of loan in months
- “Credit_History” : credit history meets guidelines (0, 1)
- “Property_Area” : Urban / Semi Urban / Rural
- “Loan_Status” : Loan approved (Y/N)

4.4 Objective of the Case Study

"Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers."

Loan prediction is a very common real-life problem that every retail bank faces in their lending operations. If the loan approval process is automated, it can save a lot of man hours and improve the speed of service to the customers. The increase in customer satisfaction and savings in operational costs are significant. However, the benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default.

5. Data Preprocessing/Feature Engineering and EDA

5.1 Preprocessing of The Data

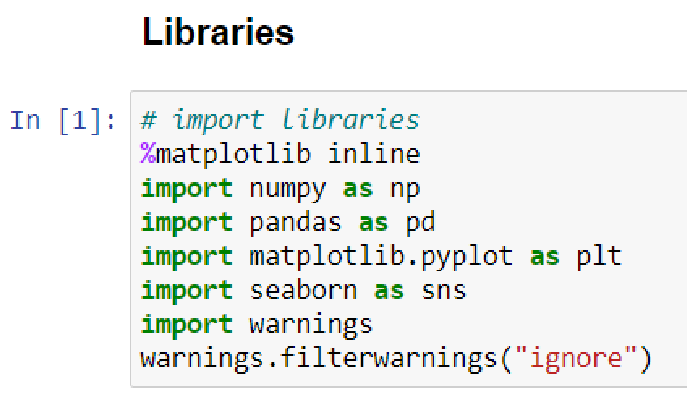
Prepossessing of the data actually involves the following steps:

5.1.1 Getting the Dataset

We can get the data set from the database or we can get the data from client.

5.1.2 Importing the Libraries

we have to import the libraries as per the requirement of the algorithm.



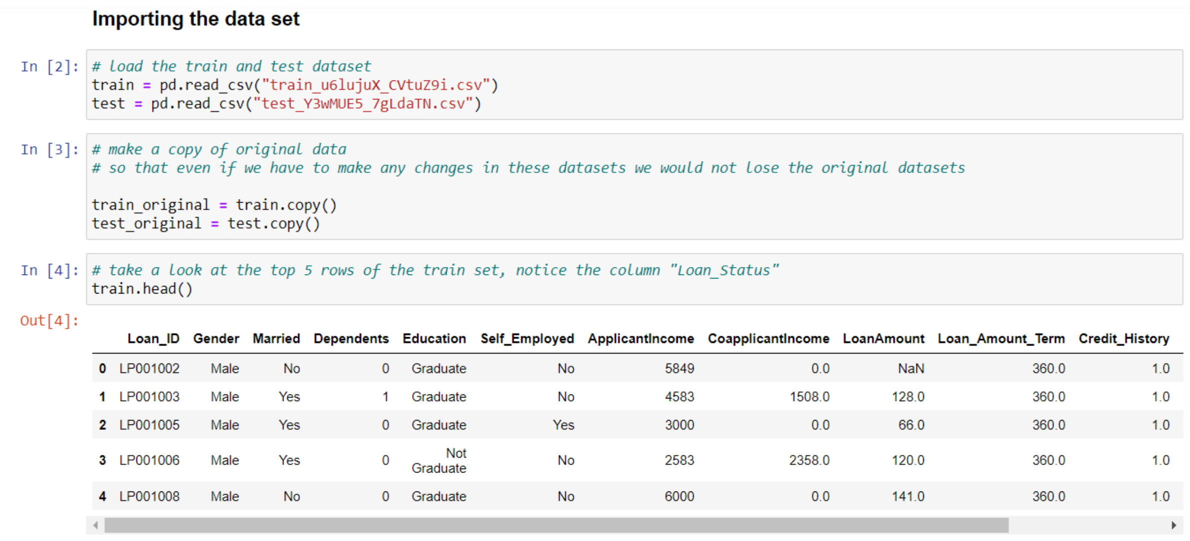
Libraries

In [1]:
```python
# import libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Fig 5.1.2.1: Importing Libraries

5.1.3 Importing The Data-Set

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire data set from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the data frame. Any missing value or Nan value have to be cleaned.



Importing the data set

```
In [2]: # load the train and test dataset
        train = pd.read_csv("train_u6lujuX_CVtuZ9i.csv")
        test = pd.read_csv("test_Y3wMUE5_7gLdaTN.csv")
```

```
In [3]: # make a copy of original data
        # so that even if we have to make any changes in these datasets we would not lose the original datasets

        train_original = train.copy()
        test_original = test.copy()
```

```
In [4]: # take a look at the top 5 rows of the train set, notice the column "Loan_Status"
        train.head()
```

Out[4]:

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

Fig 5.1.3.1: Reading the data set

5.1.4 Handling Missing Values:

In my Data set there are no missing values.

Missing values can be handled in many ways using some inbuilt methods.

5.1.5 Categorical Data

- Machine Learning models are based on equations; we need to replace the text by numbers. So that we can include the numbers in the equations

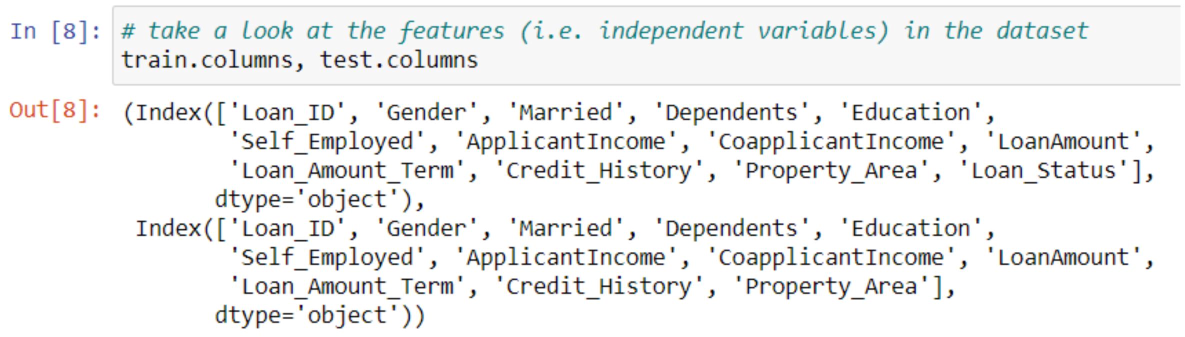
● Categorical Variables are of two types: Nominal and Ordinal

● **Nominal**: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any color

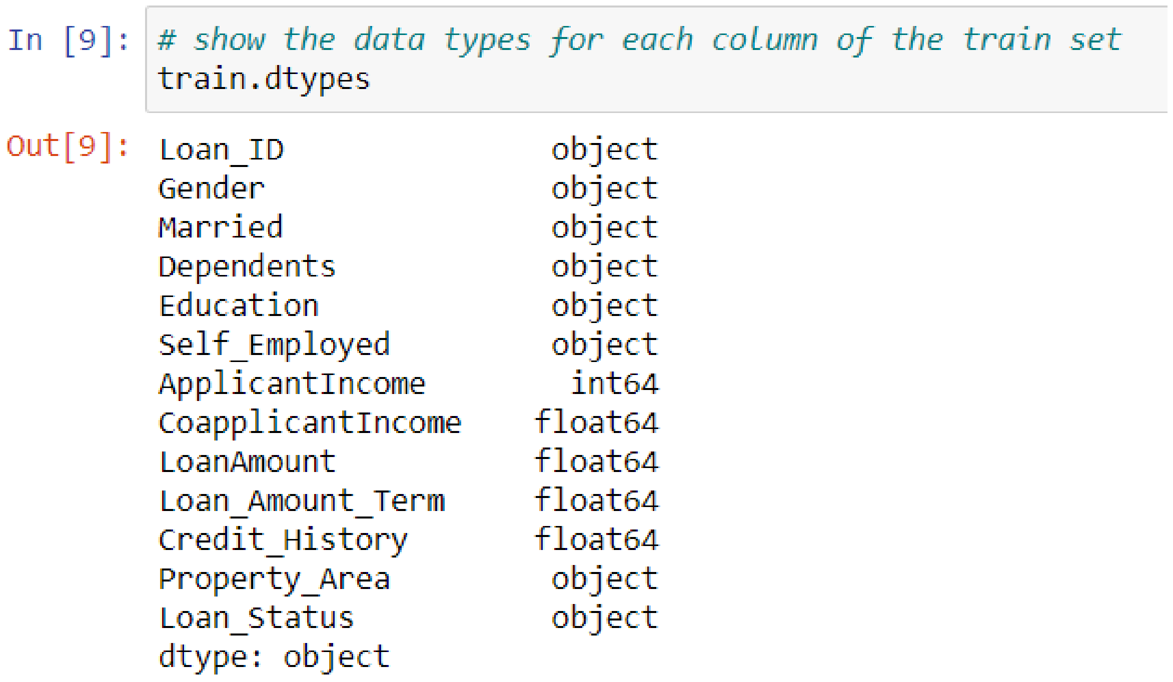
● **Ordinal**: The categories have a numerical ordering in between them. Example:

Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer

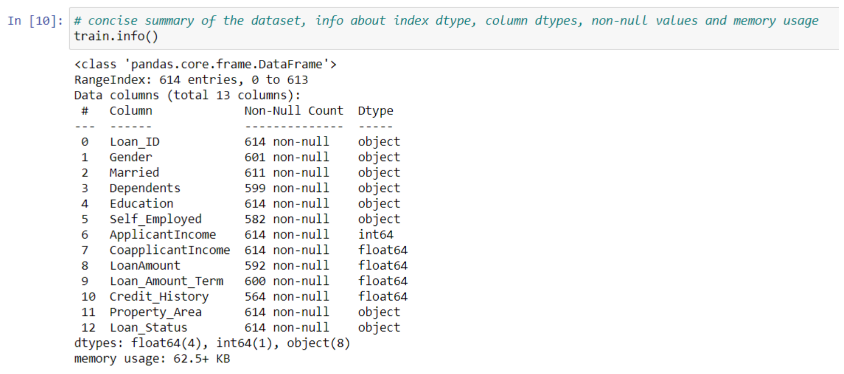
satisfaction survey, high low medium



```
In [8]: # take a look at the features (i.e. independent variables) in the dataset
        train.columns, test.columns

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

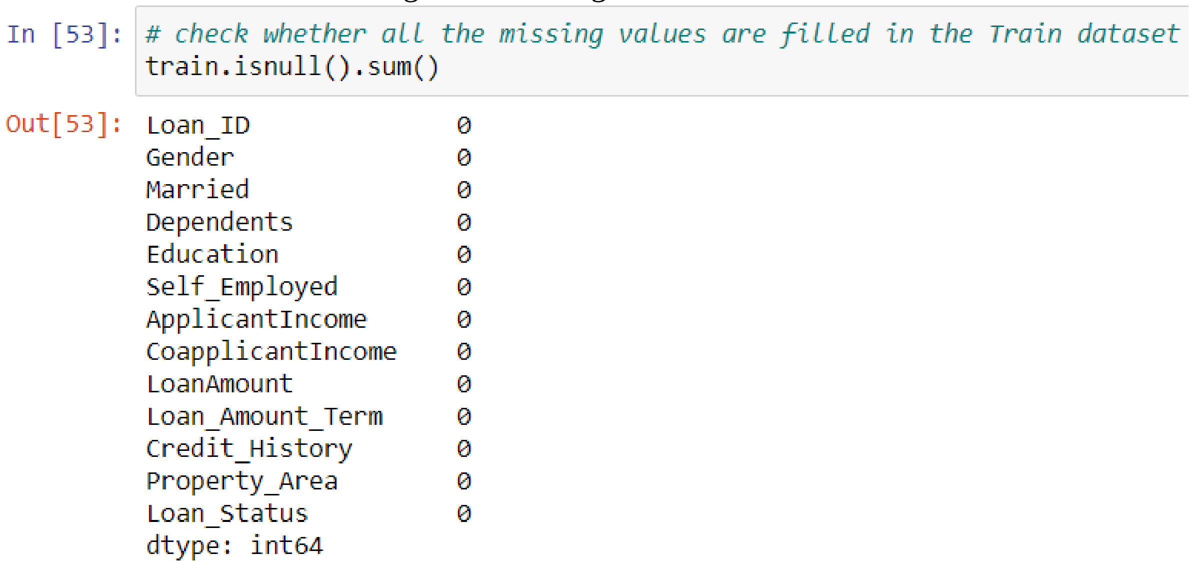


```
In [9]: # show the data types for each column of the train set
        train.dtypes

Out[9]: Loan_ID               object
        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 [10]: # concise summary of the dataset, info about index dtype, column dtypes, non-null values and memory usage
         train.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
```

Fig 5.1.5.2: Categorical Data



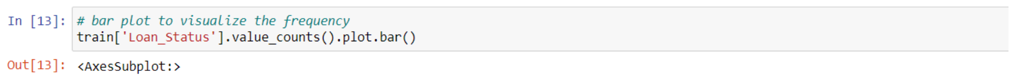
```
In [53]: # check whether all the missing values are filled in the Train dataset
         train.isnull().sum()

Out[53]: Loan_ID              0
         Gender               0
         Married              0
         Dependents           0
         Education            0
         Self_Employed        0
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount           0
         Loan_Amount_Term     0
         Credit_History       0
         Property_Area        0
         Loan_Status          0
         dtype: int64
```

Fig : 5.1.5.3: Getting count of Null values

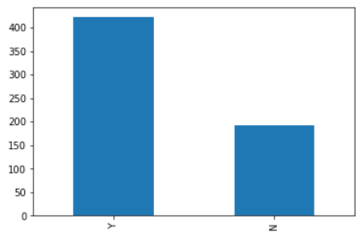
5.2 Generating Plots

5.2.1 Visualize the data between all the Features

Using inline matplotlib I generated the graphs (Histograms) to get count of each label inside categorical feature:

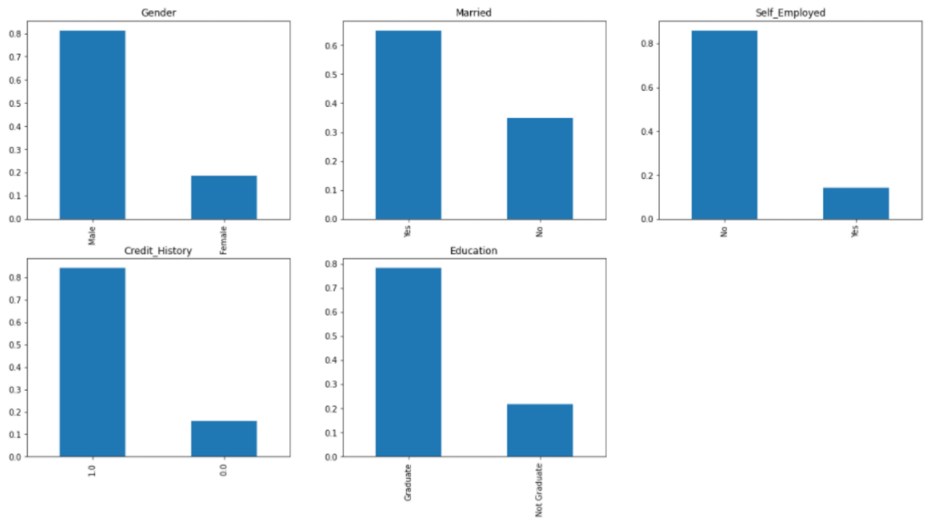


```
In [13]: # bar plot to visualize the frequency
         train['Loan_Status'].value_counts().plot.bar()

Out[13]: <AxesSubplot:>
```

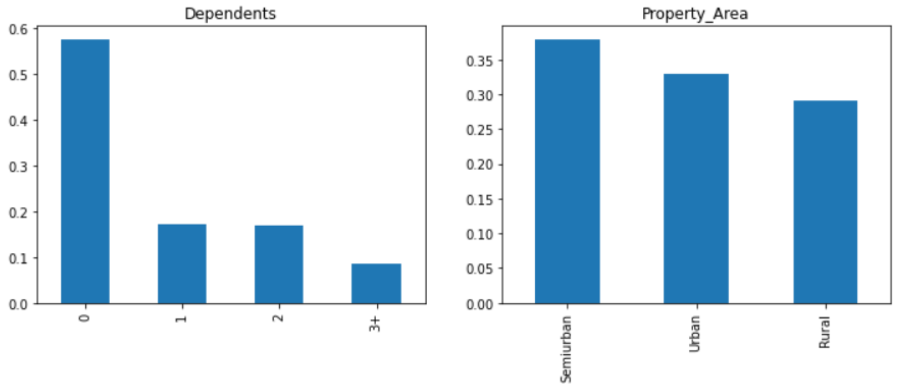


Note: The loan of 422 (around 69%) people out of 614 was approved. There is no imbalanced classes issue in this dataset, thus accuracy as an evaluation metric should be appropriate. On the other hand, if there are imbalanced or skewed classes, then we might need to use precision and recall as evaluation metrics.

```
In [14]: # Visualizing categorical features
         # plt.figure(1)
         plt.subplot(231)
         train['Gender'].value_counts(normalize=True).plot.bar(figsize=(20,10), title= 'Gender')

         plt.subplot(232)
         train['Married'].value_counts(normalize=True).plot.bar(title= 'Married')

         plt.subplot(233)
         train['Self_Employed'].value_counts(normalize=True).plot.bar(title= 'Self_Employed')

         plt.subplot(234)
         train['Credit_History'].value_counts(normalize=True).plot.bar(title= 'Credit_History')

         plt.subplot(235)
         train['Education'].value_counts(normalize=True).plot.bar(title= 'Education')

         plt.show()
```



```
In [15]: # Visualizing remaining categorical features
         # plt.figure(1)
         plt.subplot(121)
         train['Dependents'].value_counts(normalize=True).plot.bar(figsize=(12,4), title= 'Dependents')

         plt.subplot(122)
         train['Property_Area'].value_counts(normalize=True).plot.bar(title= 'Property_Area')

         plt.show()
```



```
Name: Loan_Amount_Term, dtype: int64
```

```
In [21]: # plot bar chart
         train['Loan_Amount_Term'].value_counts(normalize=True).plot.bar(title= 'Loan_Amount_Term')

Out[21]: <AxesSubplot:title={'center':'Loan_Amount_Term'}>
```

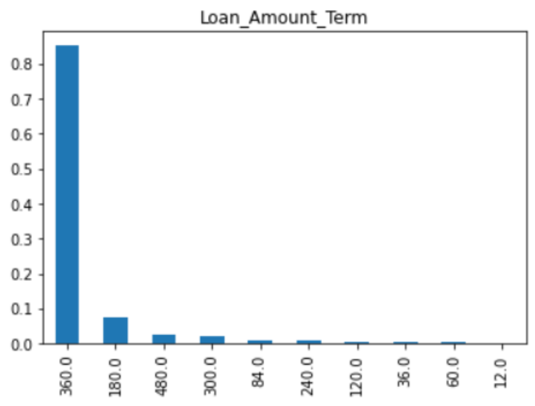
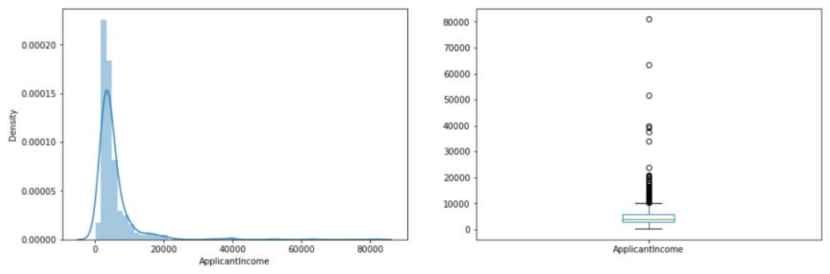


Fig: 5.2.1.1: visualization of data in between all the features

5.2.2 Visualize the data between Target and the Features

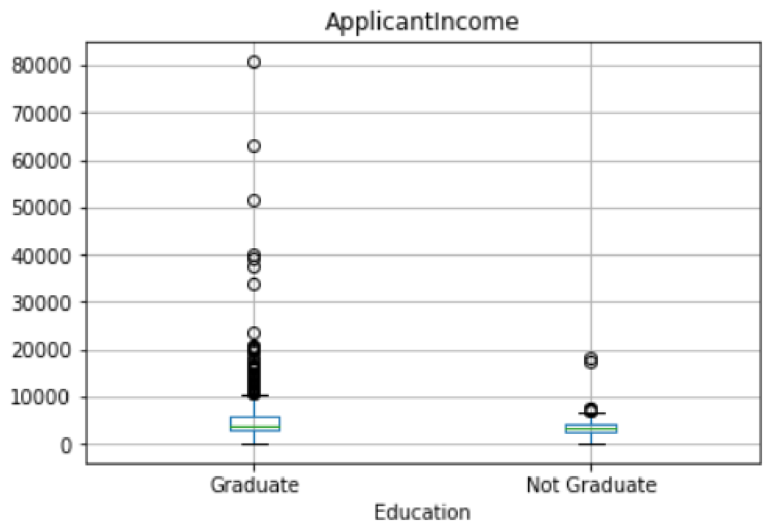
Now for visualization of data against the target value I used seaborn

```
In [16]: # Visualizing ApplicantIncome
         # plt.figure(1)
         plt.subplot(121)
         sns.distplot(train['ApplicantIncome']);

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

         plt.show()
```

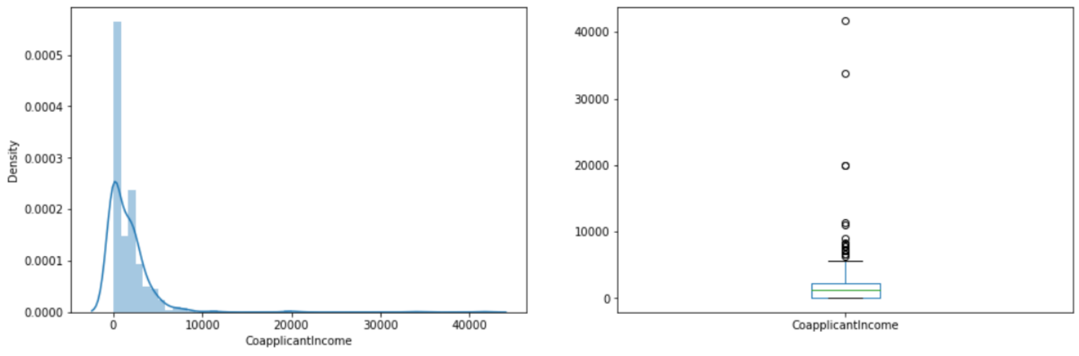


```
In [17]: train.boxplot(column='ApplicantIncome', by = 'Education')
         plt.suptitle("")
```

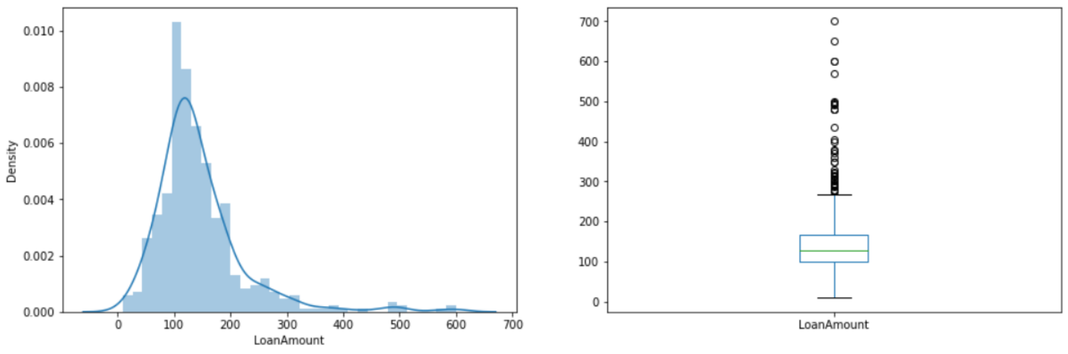
```
Out[17]: Text(0.5, 0.98, '')
```



```
In [18]: # plt.figure(1)
         plt.subplot(121)
         sns.distplot(train['CoapplicantIncome']);

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

         plt.show()
```



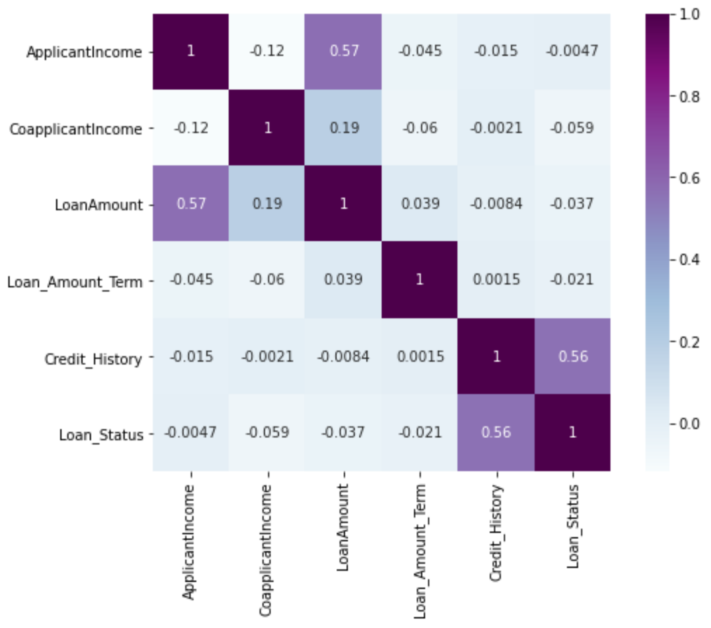
```
In [19]: # plt.figure(1)
         plt.subplot(121)
         df=train.dropna()
         sns.distplot(df['LoanAmount']);

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

         plt.show()
```



```
In [47]: # calculate and visualize correlation matrix
         matrix = train.corr()
         f, ax = plt.subplots(figsize=(9, 6))
         sns.heatmap(matrix, vmax=1, square=True, cmap="BuPu", annot=True)

         matrix
```

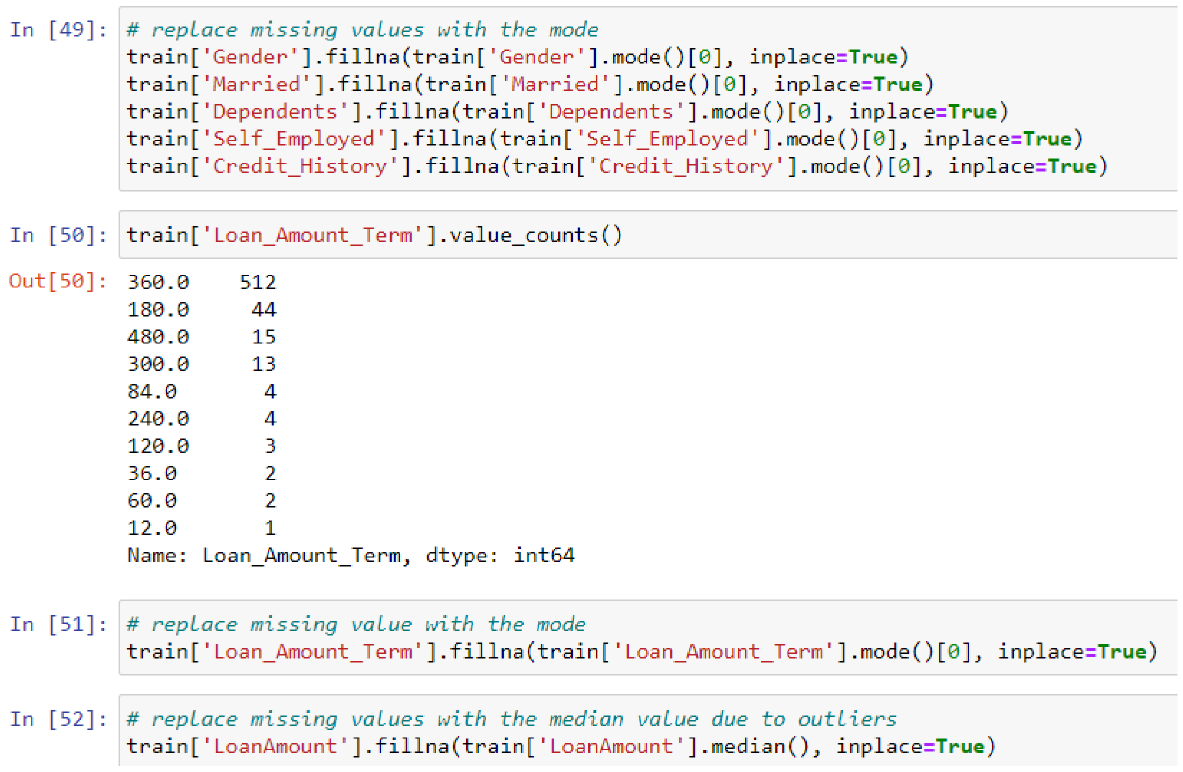
Out[47]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715	-0.004710
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056	-0.059187
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433	-0.037318
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470	-0.021268
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000	0.561678
Loan_Status	-0.004710	-0.059187	-0.037318	-0.021268	0.561678	1.000000



5.3 Data Cleaning:

I am Standardizing the features (No of Doors, and No of persons) to value 5.

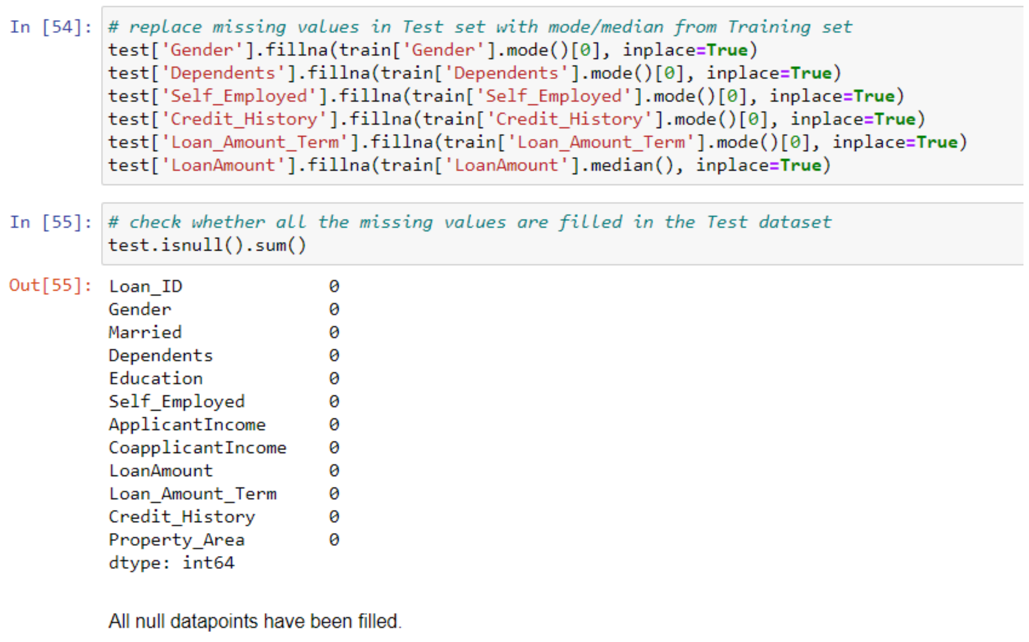


```
In [49]: # replace missing values with the mode
         train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
         train['Married'].fillna(train['Married'].mode()[0], inplace=True)
         train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
         train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
         train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)

In [50]: train['Loan_Amount_Term'].value_counts()

Out[50]: 360.0    512
         180.0     44
         480.0     15
         300.0     13
         84.0       4
         240.0      4
         120.0      3
         36.0       2
         60.0       2
         12.0       1
         Name: Loan_Amount_Term, dtype: int64

In [51]: # replace missing value with the mode
         train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)

In [52]: # replace missing values with the median value due to outliers
         train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

Fig 5.3.1: data cleaning

5.4 Encoding Categorical Data:

As Machine Learning can understand only numbers it’s time to encode the categorical data to its respective values in a traditional way without using label encoder or pandas.



```
In [54]: # replace missing values in Test set with mode/median from Training set
         test['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
         test['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
         test['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
         test['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
         test['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
         test['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)

In [55]: # check whether all the missing values are filled in the Test dataset
         test.isnull().sum()

Out[55]: Loan_ID              0
         Gender               0
         Married              0
         Dependents           0
         Education            0
         Self_Employed        0
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount           0
         Loan_Amount_Term     0
         Credit_History       0
         Property_Area        0
         dtype: int64
```

All null datapoints have been filled.

Fig 5.4.1: mapping data

5.5 Data Flow Diagrams:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

UML Diagrams:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

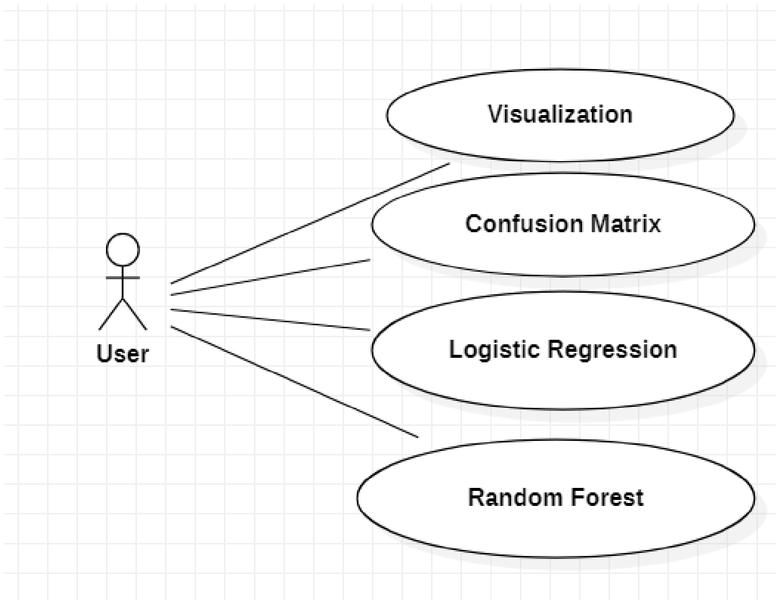
The Primary goals in the design of the UML are as follows:

1. +Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

6. Integrate best practices.

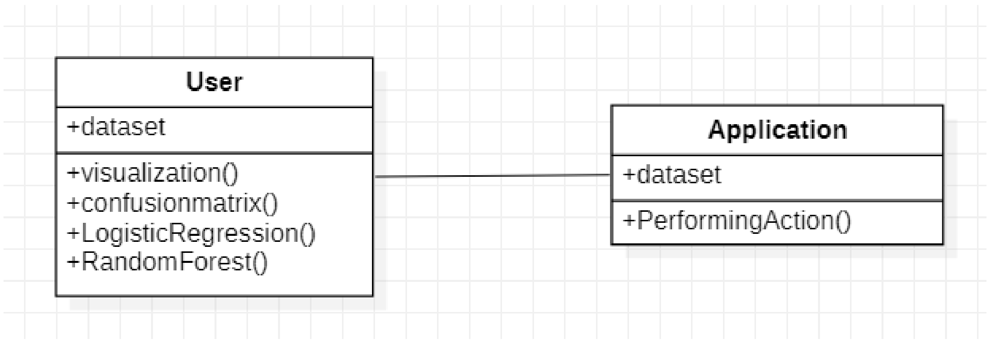
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



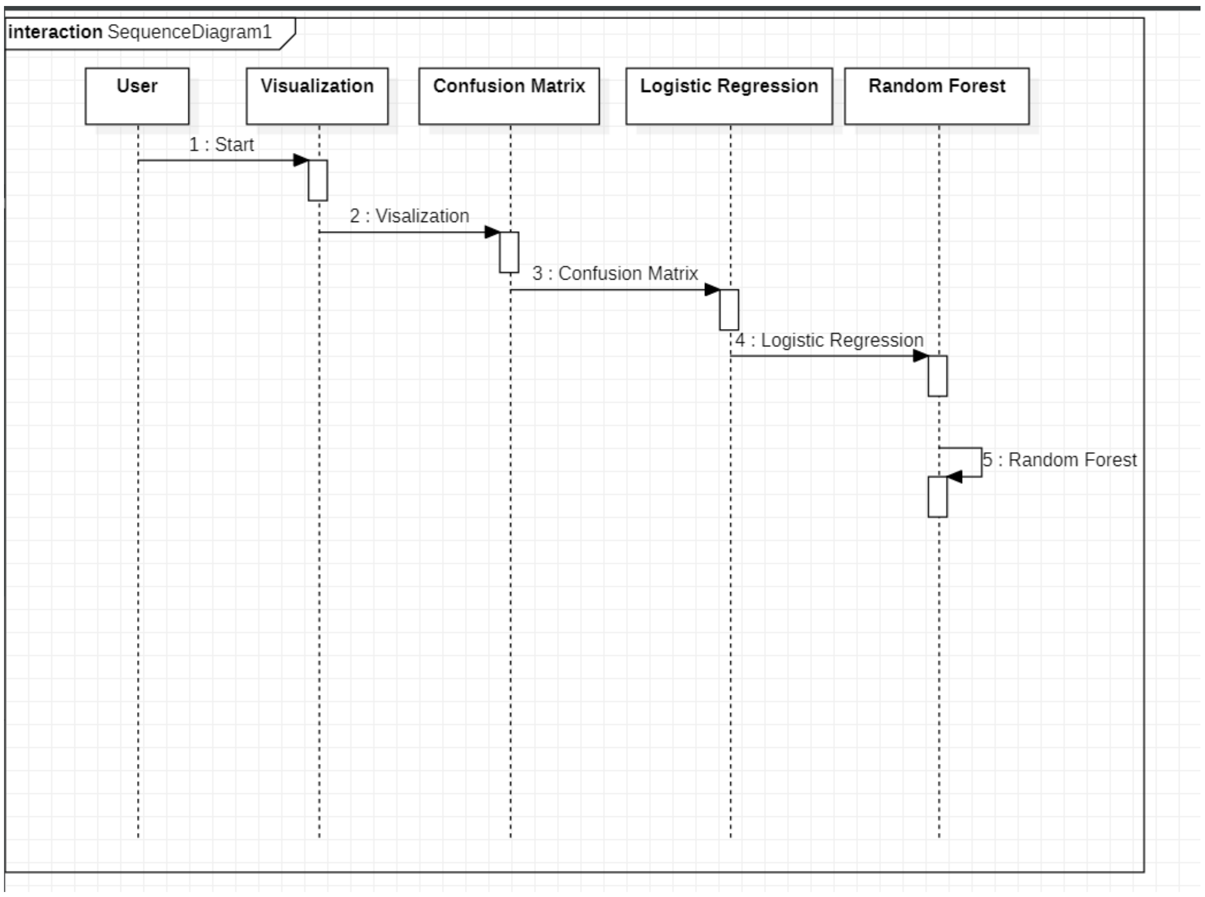
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



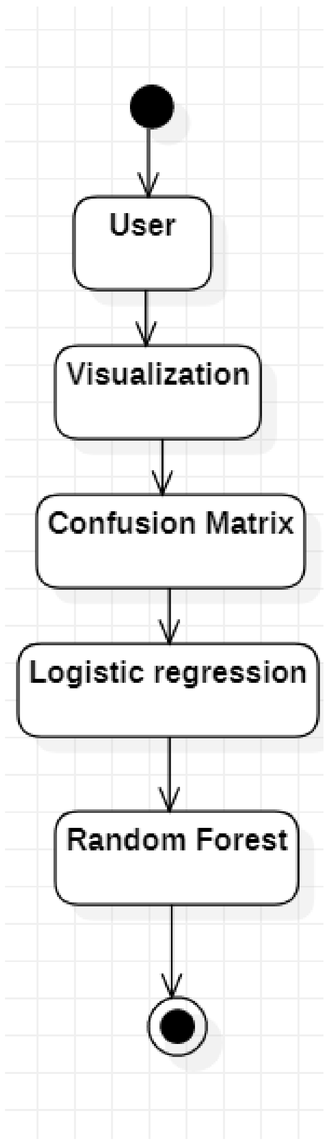
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Activity diagram:

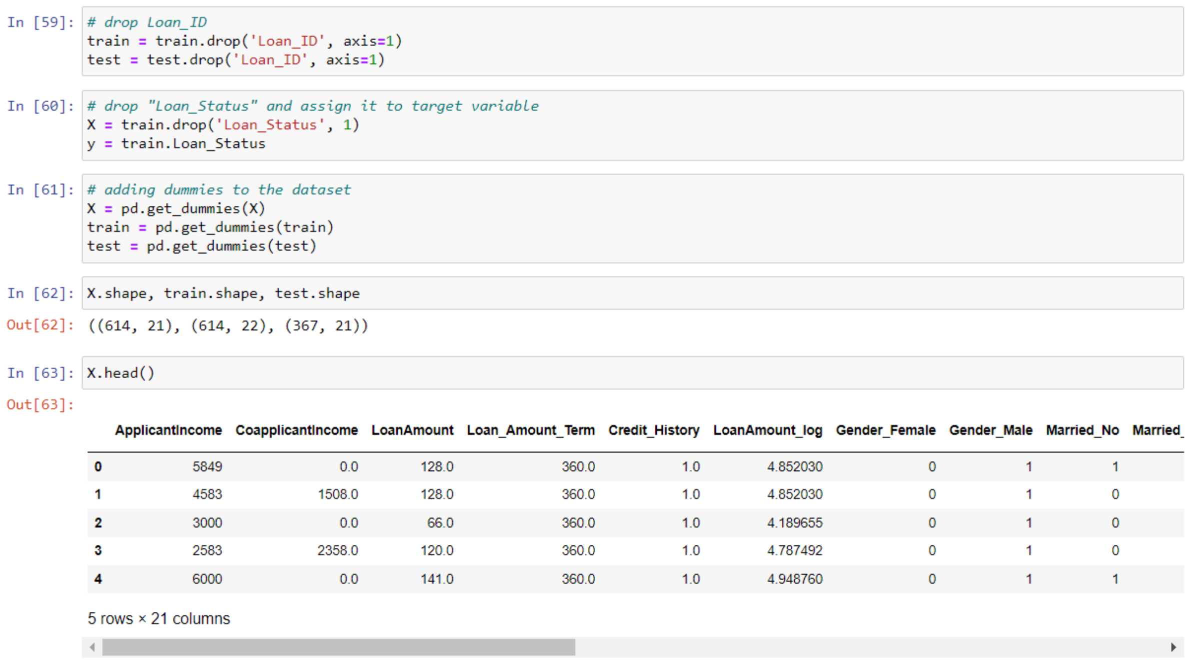
The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.


User
Visualization
Confusion Matrix
Logistic regression
Random Forest

6. Feature Selection

6.1 Train and Test Split:

Splitting data into Train and Test



```
In [59]: # drop Loan_ID
         train = train.drop('Loan_ID', axis=1)
         test = test.drop('Loan_ID', axis=1)

In [60]: # drop "Loan_Status" and assign it to target variable
         X = train.drop('Loan_Status', 1)
         y = train.Loan_Status

In [61]: # adding dummies to the dataset
         X = pd.get_dummies(X)
         train = pd.get_dummies(train)
         test = pd.get_dummies(test)

In [62]: X.shape, train.shape, test.shape

Out[62]: ((614, 21), (614, 22), (367, 21))

In [63]: X.head()

Out[63]:
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	LoanAmount_log	Gender_Female	Gender_Male	Married_No	Married_
0	5849	0.0	128.0	360.0	1.0	4.852030	0	1	1	
1	4583	1508.0	128.0	360.0	1.0	4.852030	0	1	0	
2	3000	0.0	66.0	360.0	1.0	4.189655	0	1	0	
3	2583	2358.0	120.0	360.0	1.0	4.787492	0	1	0	
4	6000	0.0	141.0	360.0	1.0	4.948760	0	1	1	

5 rows × 21 columns

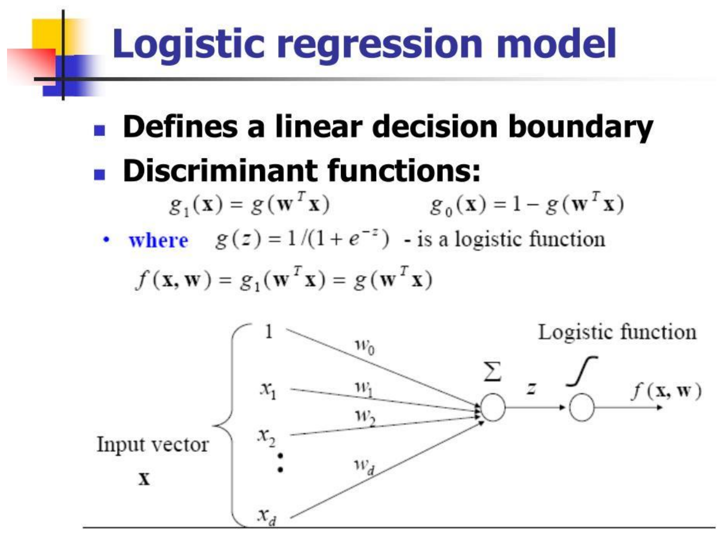
Fig 6.1.1: train & test split data

7. MODEL BUILDING AND EVALUATION

I Used 2 classifiers for predicting the output

7.1 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.
In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).



Logistic regression model

- **Defines a linear decision boundary**
- **Discriminant functions:**

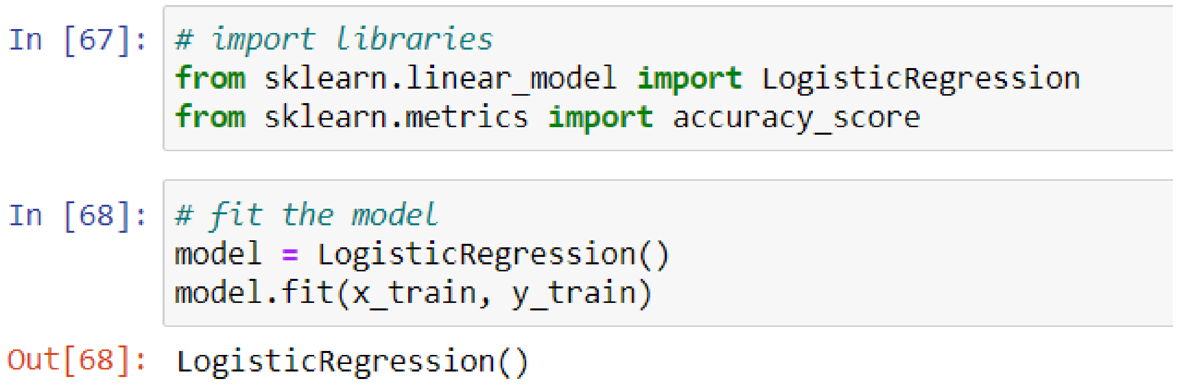
$$g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x}) \qquad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$$

- where $g(z) = 1/(1+e^{-z})$ - is a logistic function

$$f(\mathbf{x},\mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$

Fig 7.1.1 Logistic Regression model

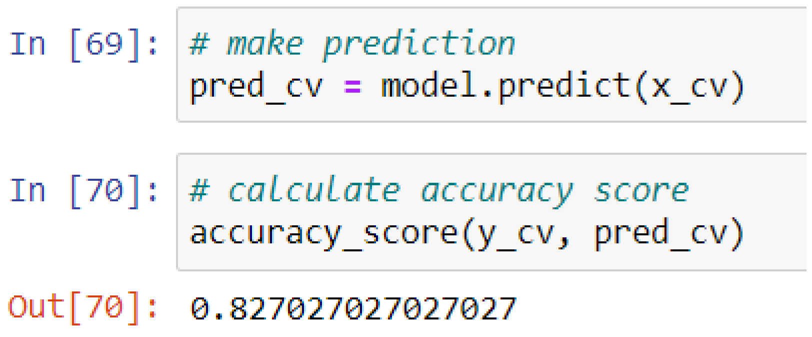
Model Creation:



```
In [67]: # import libraries
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score

In [68]: # fit the model
         model = LogisticRegression()
         model.fit(x_train, y_train)

Out[68]: LogisticRegression()
```

Fig 7.1.2 Logistic Regression(model creation and model fitting)

7.2 METRICS accuracy score:



```
In [69]: # make prediction
         pred_cv = model.predict(x_cv)

In [70]: # calculate accuracy score
         accuracy_score(y_cv, pred_cv)

Out[70]: 0.827027027027027
```

7.3 RANDOM FOREST CLASSIFIER

Random Forest Classifier

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

7.3.1 Ensemble Algorithm:

Ensemble algorithms are those which combines more than one algorithm of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.

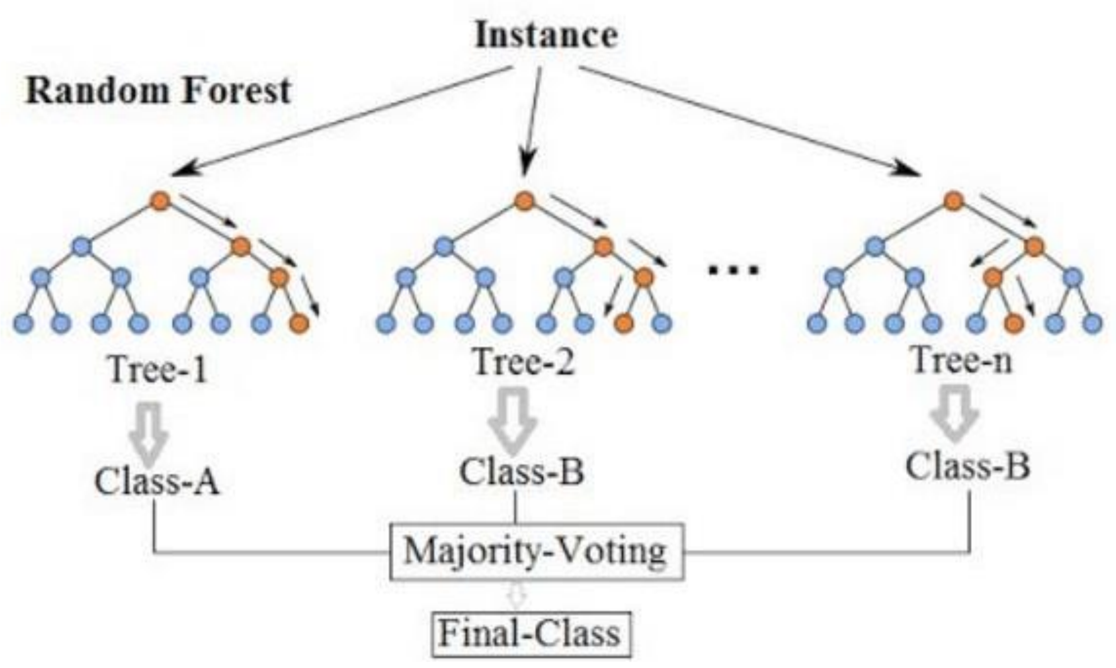
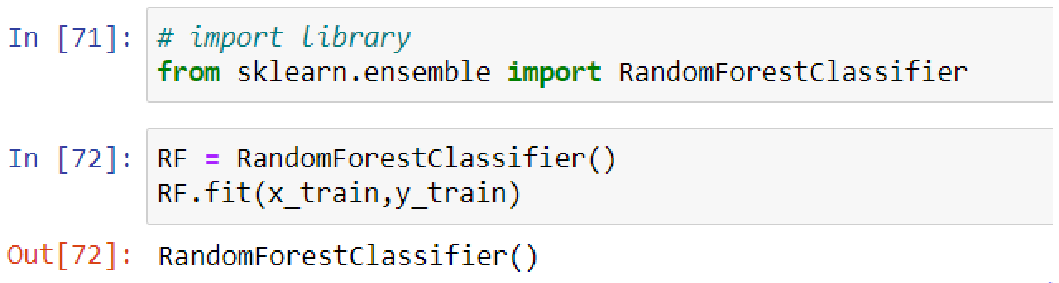


Fig7.2.1.1: Structure of Random Forest Classification

Features and Advantages of Random Forest:

1. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
2. It runs efficiently on large databases.
3. It can handle thousands of input variables without variable deletion.
4. It gives estimates of what variables that are important in the classification.
5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.
6. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

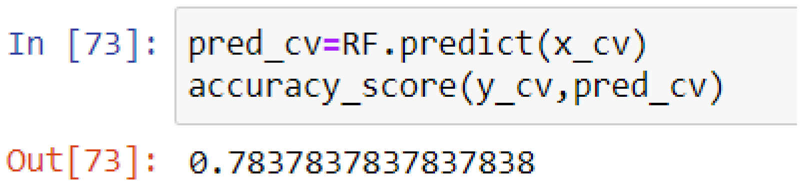
Model Creation:



```
In [71]: # import library
         from sklearn.ensemble import RandomForestClassifier

In [72]: RF = RandomForestClassifier()
         RF.fit(x_train,y_train)

Out[72]: RandomForestClassifier()
```

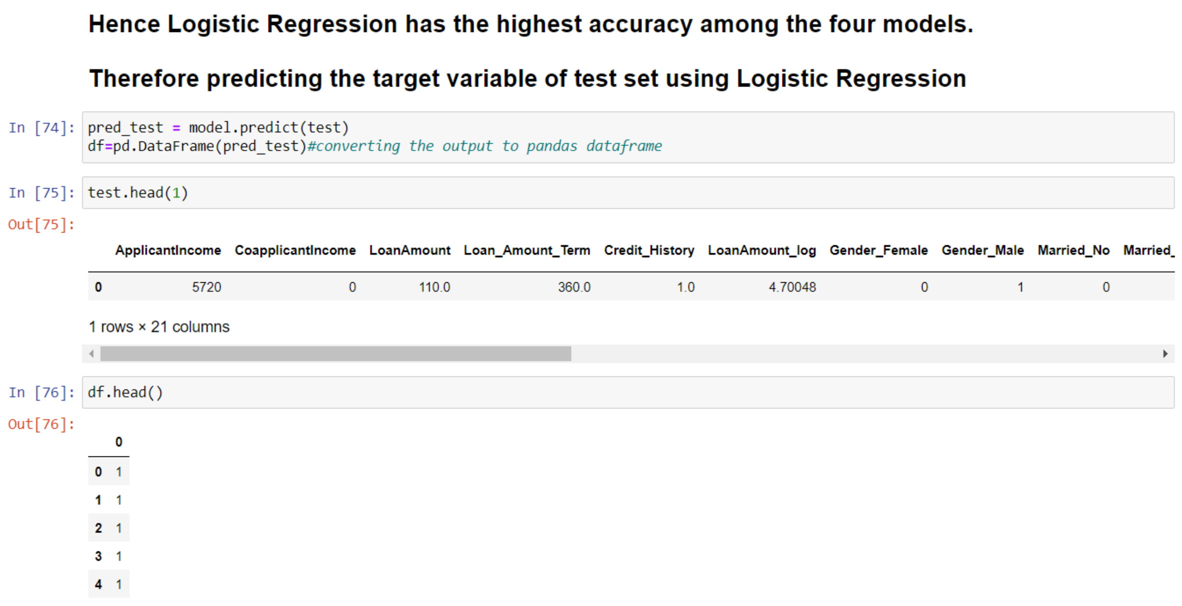
Fig7.2.1.4: random forest (model creation and model fitting)

7.3.2METRICS Accuracy Score:



```
In [73]: pred_cv=RF.predict(x_cv)
         accuracy_score(y_cv,pred_cv)

Out[73]: 0.7837837837837838
```

Fig 7.2.2.7: random forest (accuracy score)



Hence Logistic Regression has the highest accuracy among the four models.

Therefore predicting the target variable of test set using Logistic Regression

```
In [74]: pred_test = model.predict(test)
         df=pd.DataFrame(pred_test)#converting the output to pandas dataframe
```

```
In [75]: test.head(1)
```

Out[75]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	LoanAmount_log	Gender_Female	Gender_Male	Married_No	Married_
0	5720	0	110.0	360.0	1.0	4.70048	0	1	0	

1 rows × 21 columns

```
In [76]: df.head()
```

Out[76]:

	0
0	1
1	1
2	1
3	1
4	1

8. Conclusion

After trying and testing 4 different algorithms, the best accuracy on the public leaderboard is achieved by Logistic Regression (0.7847), followed by Random Forest (0.7778). While new features created via feature engineering helped in predicting the target variable, it did not improve the overall model accuracy much. Compared to using default parameter values. On the whole, a logistic regression classifier provides the best result in terms of accuracy for the given dataset, without any feature engineering needed. Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is often a good baseline that data scientists can use to measure the performance of other more complex algorithms. In this case, however, a basic Logistic Regression has already outperformed other more complex algorithms like Random Forest.