# MACHINE LEARNING

(Prediction of Acceptance of Admission into a University)

Summer Internship Report Submitted in partial fulfilment of the

requirement for undergraduate degree of

**Bachelor of Technology**

In

**Computer Science Engineering**

By

**Janjam Gayathri**

**221710313018**

Under the Guidance of

Assistant Professor

GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU

Department of Computer Science engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

# DECLARATION

I submit this industrial training work entitled "**PREDICTION OF ACCEPTANCE OF ADMISSION INTO A UNIVERSITY**" to GITAM (Deemed To Be University), Hyderabad in partial fulfilment 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 Asst. Professor, 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 — Janjam Gayathri

Date: — 221710313018

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:

# CERTIFICATE

This is to certify that the Industrial Training Report entitled "**PREDICTION OF ACCEPTANCE OF ADMISSION INTO A UNIVERSITY**" is being submitted by Janjam Gayathri (221710313018) in partial fulfilment of the requirement for the award of **Bachelor of Technology in Computer Science Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2020.

It is faithful record work carried out by her at the **Computer Science Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Assistant Professor
Department of CSE

Head of Department
Department of CSE

# ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank **Dr. N. Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad.

I would like to thank respected **Dr. Phani Kumar**, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties who helped me to make this internship a successful accomplishment

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Janjam Gayathri
221710313018

# ABSTRACT

Machine learning algorithms are used to predict the values from the data set by Splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on Cereals data set. My perception of understanding the given data set has been in the view of undertaking a students requirement of for joining into a university

The aim of this research is to develop a system using machine learning algorithms, we will name it as Prediction of Acceptance of Admission into a University.. It will help the students to identify the chances of their application to a university being accepted. Also it will help them in identifying the universities which are best suitable for their profile and also provide them with the details of those universities. A simple user interface will be developed for the users to access the SAP system. Keywords: Student Admission Predictor; Machine Learning; Data Mining

# TABLE OF CONTENTS:

# LIST OF FIGURES

# CHAPTER 1
# MACHINE LEARNING

## 1.1 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).

## 1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber 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


Figure 1 : The Process Flow

## 1.3 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.

## 1.4 TYPES OF 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.

### 1.4.1 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 labelled training dataset – that is, a dataset 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 classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

### 1.4.2 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.

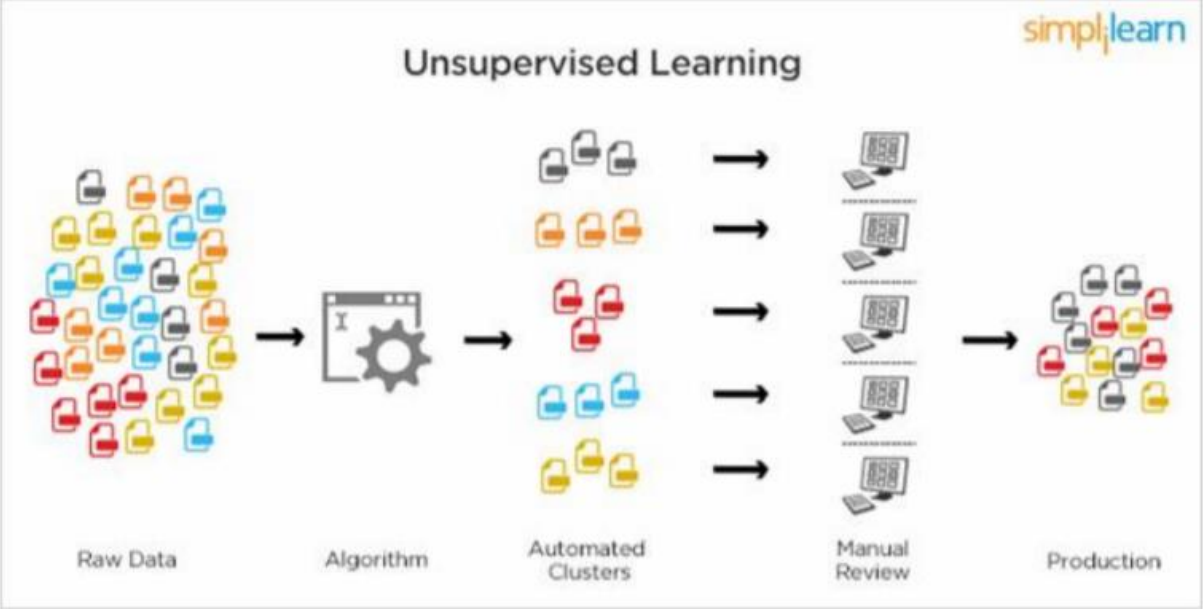


Figure 2 : Unsupervised Learning

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.

### 1.4.3 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.

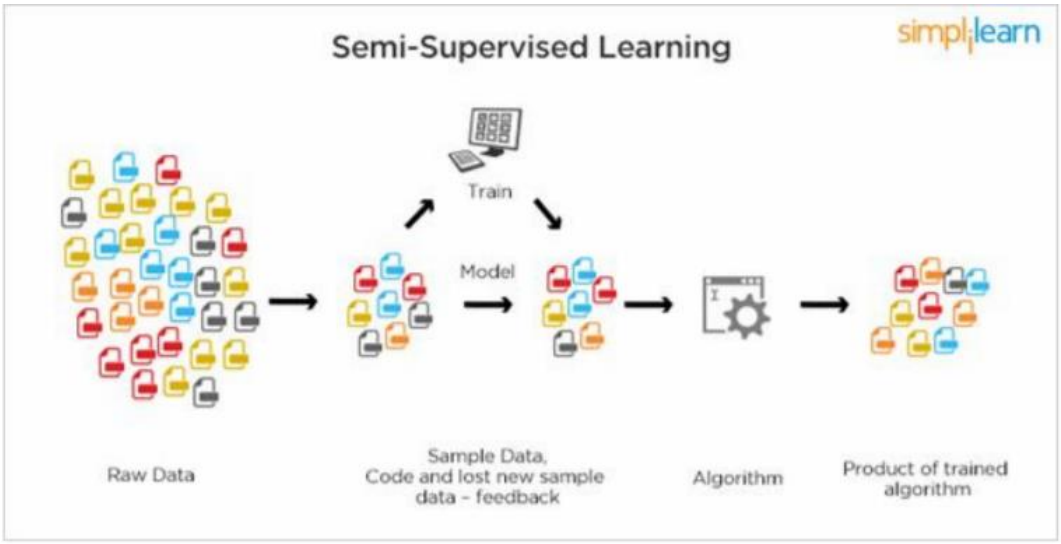


Figure 3 : Semi Supervised Learning

# CHAPTER 2

# DEEP LEARNING

## 2.1 INTRODUCTION:

In statistical machine learning, a major issue is the selection of an appropriate feature space where input instances have desired properties for solving a particular problem. For example, in the context of supervised learning for binary classification, it is often required that the two classes are separable by a hyperplane.

## 2.2 IMPORTANCE OF DEEP LEARNING:

The ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data. However, deep learning algorithms can be overkill for less complex problems because they require access to a vast amount of data to be effective. For instance, ImageNet, the common benchmark for training deep learning models for comprehensive image recognition, has access to over 14 million images.

If the data is too simple or incomplete, it is very easy for a deep learning model to become overfitted and fail to generalize well to new data. As a result, deep learning models are not as effective as other techniques (such as boosted decision trees or linear models) for most practical business problems such as understanding customer churn, detecting fraudulent transactions and other cases with smaller datasets and fewer features. In certain cases like multiclass classification, deep learning can work for smaller, structured datasets.

## 2.3 RELATION BETWEEN DATA MINING, MACHINE LEARNING, 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 discovered 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.

# CHAPTER 3

# PYTHON

Basic programming language used for machine learning is : PYTHON

## 3.1 INTRODUCTION TO PYTHON:

- 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 an 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 HOW TO SETUP PYTHON:

● 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.

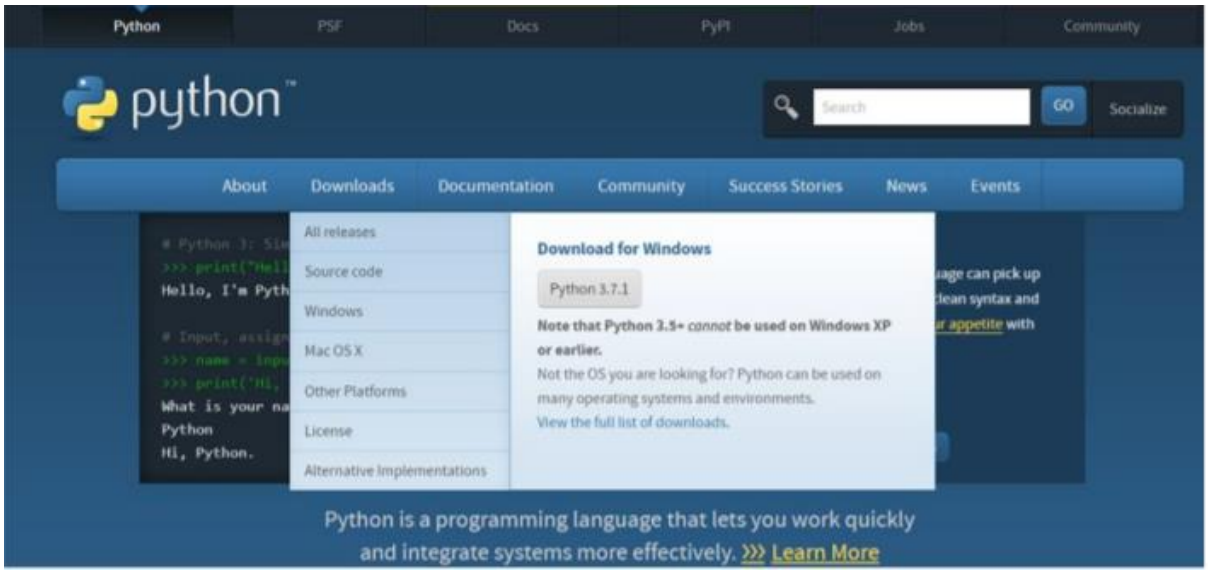


Figure 4: Python download

### 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 that quickly installs and manages packages.
- In WINDOWS:
- Step 1: Open Anaconda.com/downloads in a web browser.
- Step 2: Download python 3.4 version for (32-bits graphic 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)

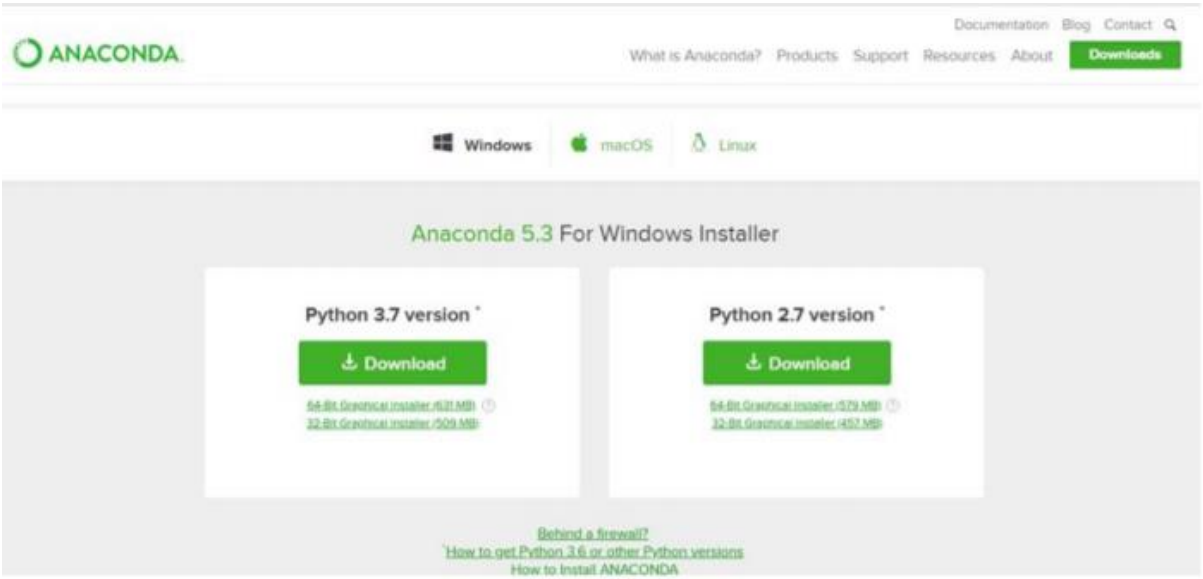


Figure 5 : Anaconda download


Figure 6 : Jupyter notebook

## 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 –

- o Numbers
- o Strings
- o Lists
- o Tuples
- o 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 types.

● 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 the lists and the 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 12 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 FUNCTION:

## 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 PYTHON USING 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
```

Figure 7 : 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__().

# CHAPTER 4
# CASE STUDY

## 4.1 PROBLEM STATEMENT:

To predict the Acceptance of Admission into a University using Machine Learning Algorithm called Multiple Linear Regression, Random Forest Regressor, Decision Tree Algorithm.

## 4.2 DATA SET:

The given data set consists of the following parameters:

A - Serial No.

B - GRE Score

C - TOEFL Score

D - University Rating

E - SOP

F - LOR

G - CGPA

H - Research

I - Chance of Admit

## 4.3 OBJECTIVE OF THE CASE STUDY:

The main objective of the project was to develop a system that can help the admission committee of the university to make better and faster decisions. Created a similar model to predict the enrolment of the student in the university based on the factors like GRE score, GPA score etc.

To get a better understanding and chalking out a plan of action for solution of the student, we have adapted the viewpoint of looking at categories and for further deep understanding of the problem, we have also considered SOP,LOR,Research features.

# CHAPTER 5
# MODEL BUILDING

## 5.1 PREPROCESSING OF THE DATA:

Preprocessing 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 the client.

### 5.1.2 IMPORTING THE LIBRARIES:

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

**Importing the Libraries**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

Figure 8 : 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 dataset from a comma separated values file and we can assign it to a DataFrame 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 accessed using the dataframe. Any missing value or NaN value has to be cleaned.

**Reading the Data Set**

```
data = pd.read_csv("Admission_Predict.csv") #Data Loading
data.head()
```

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |

Figure 9 : Reading the dataset

## 5.1.4 HANDLING MISSING VALUES:

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

(a)dropna(): dropna() is a function which drops all the rows and columns which are having the missing values(i.e. NaN)

```
data.dropna()
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **395** | 396 | 324 | 110 | 3 | 3.5 | 3.5 | 9.04 | 1 | 0.82 |
| **396** | 397 | 325 | 107 | 3 | 3.0 | 3.5 | 9.11 | 1 | 0.84 |
| **397** | 398 | 330 | 116 | 4 | 5.0 | 4.5 | 9.45 | 1 | 0.91 |
| **398** | 399 | 312 | 103 | 3 | 3.5 | 4.0 | 8.78 | 0 | 0.67 |
| **399** | 400 | 333 | 117 | 4 | 5.0 | 4.0 | 9.66 | 1 | 0.95 |

400 rows × 9 columns

Figure 10 : data after using dropna()

- dropna() function has a parameter called how which works as follows
- if how = 'all' is passed then it drops the rows where all the columns of the Particular row are missing
- if how = 'any' is passed then it drops the rows where all the columns of the particular row are missing

(b)fillna(): fillna() is a function which replaces all the missing values using different ways.

```
data.fillna(0)
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **395** | 396 | 324 | 110 | 3 | 3.5 | 3.5 | 9.04 | 1 | 0.82 |
| **396** | 397 | 325 | 107 | 3 | 3.0 | 3.5 | 9.11 | 1 | 0.84 |
| **397** | 398 | 330 | 116 | 4 | 5.0 | 4.5 | 9.45 | 1 | 0.91 |
| **398** | 399 | 312 | 103 | 3 | 3.5 | 4.0 | 8.78 | 0 | 0.67 |
| **399** | 400 | 333 | 117 | 4 | 5.0 | 4.0 | 9.66 | 1 | 0.95 |

400 rows × 9 columns

Figure 11 : functioning of fillna(0)

- fillna() also have parameters called method and axis
- if we use method = ’ffill’ where ffill is a method called forward fill, which carry forwards the previous row’s value
- if we use method = ‘bfill’ where bfill is a method called backward fill, which carry backward the next row’s value
- if we use method = ‘ffill’ , axis = ‘columns’ then it carry forwards the previous column’s value
- if we use method = ‘bfill’ , axis = ‘columns’ then it carry backward the next column’s value

(c)interpolate(): interpolate() is a function which comes up with a guess value based on the other values in the dataset and fills those guess values in the place of missing values.

```
data.interpolate()
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 12 : functioning of interpolate()

(d)mean and median imputation

- mean and median imputation can be performed by using fillna().
- mean imputation calculates the mean for the entire column and replaces the missing values in that column with the calculated mean.
- median imputation calculates the median for the entire column and replaces the missing values in that column with the calculated median.

```
data.fillna(data.mean())
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 13 : mean imputation

```
data.fillna(data.median())
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 14 : median imputation

### 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 colour

● 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

● Categorical data can be handled by using dummy variables, which are also called as indicator variables.

● Handling categorical data using dummies: In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0’s and 1’s. Once these dummies got created we have to concat this dummy set to our data frame or we can add that dummy set to the dataframe.

## 5.2 Exploratory Data Analysis:

describe(): This method computes a summary of statistics like count, mean,standard ] deviation,min,max,and quartile.

```
data.describe()
```

| | no | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 | 400.000000 |
| **mean** | 200.500000 | 316.807500 | 107.410000 | 3.087500 | 3.400000 | 3.452500 | 8.598925 | 0.547500 | 0.724350 |
| **std** | 115.614301 | 11.473646 | 6.069514 | 1.143728 | 1.006869 | 0.898478 | 0.596317 | 0.498362 | 0.142609 |
| **min** | 1.000000 | 290.000000 | 92.000000 | 1.000000 | 1.000000 | 1.000000 | 6.800000 | 0.000000 | 0.340000 |
| **25%** | 100.750000 | 308.000000 | 103.000000 | 2.000000 | 2.500000 | 3.000000 | 8.170000 | 0.000000 | 0.640000 |
| **50%** | 200.500000 | 317.000000 | 107.000000 | 3.000000 | 3.500000 | 3.500000 | 8.610000 | 1.000000 | 0.730000 |
| **75%** | 300.250000 | 325.000000 | 112.000000 | 4.000000 | 4.000000 | 4.000000 | 9.062500 | 1.000000 | 0.830000 |
| **max** | 400.000000 | 340.000000 | 120.000000 | 5.000000 | 5.000000 | 5.000000 | 9.920000 | 1.000000 | 0.970000 |

Figure 15 : describing the data

shape: This gives the number of columns and rows.

```
data.shape
```

```
(400, 9)
```

Figure 16 : showing no.of columns and rows

columns: This gives all the column names of the Data set.

```
data.columns
```

```
Index(['no', 'gre', 'toefl', 'rating', 'sop', 'lor', 'gpa', 'research',
       'chance'],
      dtype='object')
```

Figure 17 : column names of data set

## 5.3 TRAINING THE MODEL:

### 5.3.1 Method 1:

- Splitting the data : after the preprocessing is done then the data is split into train and test sets
- In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.
- training set - a subset to train a model.(Model learns patterns between Input and Output).
- test set - a subset to test the trained model. (To test whether the model has correctly learnt ).
- The amount or percentage of Splitting can be taken as specified (i.e. train data = 75% , test data =25% or train data = 80% , test data= 20%).
- First we need to identify the input and output variables and we need to separate the input set and output set.
- In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method.
- This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables) 29 Figure 21 : importing train_test_split.
- Then we need to import linear regression.

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test =train_test_split(X,Y,test_size=0.2,random_state=42)
```

Figure 18 : importing train_test_split

- Then we need to import linear regression method from linear_model package from scikit learn library
- We need to train the model based on our train set (that we have obtained from splitting)
- Then we have to test the model for the test set ,that is done as follows

● We have a method called predict , using this method we need to predict the output for input test set and we need to compare the out but with the output test data.

● If the predicted values and the original values are close then we can say that the model is trained with good accuracy.

Correlation:

● Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. Correlation is described as the analysis which lets us know the association or the absence of the relationship between two variables ‘x’ and ‘y’. It is a statistical measure that represents the strength of the connection between pairs of variables.

● We can also find the column which is affecting the R-Squared value so that we can perform operations on that specific column or we can remove that column , this can be done using pair plot.

● In pair plot we need to find the correlation between two variables and we can do some 33 operations on that variables

● pairplot is a method which is available in seaborn library, so to use this pairplot method we have to import seaborn library.

```
corelation = data.corr()
```

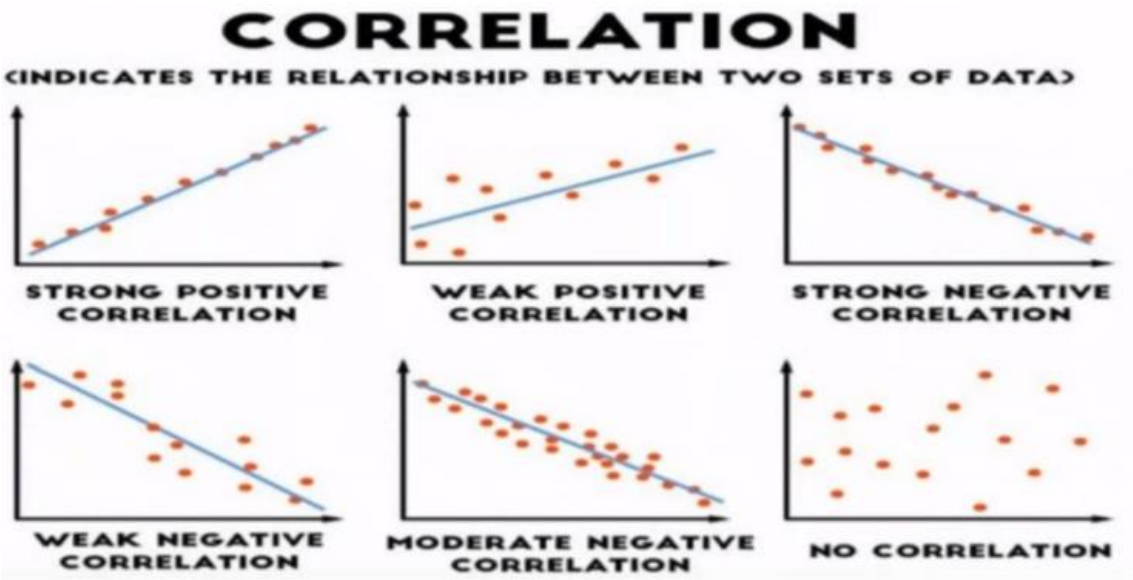


Figure 19 : Correlation

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x1f2ce662f08>
```

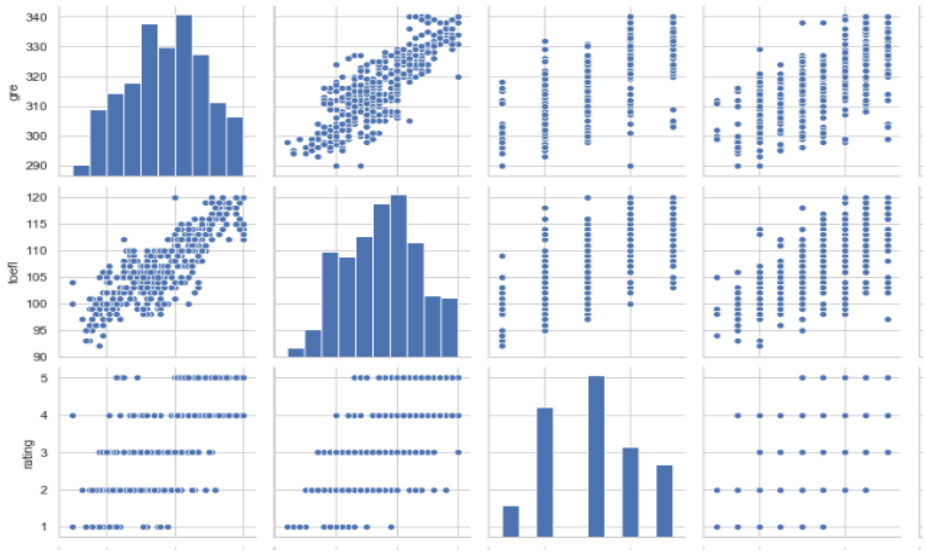


Figure 20 : pairplot

## 5.4 EVALUATING THE CASE STUDY:

● MULTIPLE LINEAR REGRESSION: A multiple linear regression model allows us to capture the relationship between multiple feature columns and the target column. Here's what the formula looks like:

y^=a0+a1x1+a2x2+...+anxn

● Importing the required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
```

Figure 21 : Importing the libraries

- Reading the Data-Set

**Reading the Data Set**

```
data = pd.read_csv("Admission_Predict.csv") #Data Loading
data.head()
```

| | Serial No. | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of Admit |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 2 | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 3 | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 4 | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 5 | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |

Figure 22 : Reading the Data-Set

- Handling the missing values. There is a method called isnull() which gives the number of missing values in each and every column.

```
data.isnull().sum() #There are no missing values in the data.
```

```
gre         0
toefl       0
rating      0
sop         0
lor         0
gpa         0
research    0
chance      0
dtype: int64
```

Figure 23 : After handling the missing values

● Removing the columns which are not required using drop method

```
data = data.drop("no", axis=1) #drop the  serial number column
data.head()
```

| | gre | toefl | rating | sop | lor | gpa | research | chance |
|---|---|---|---|---|---|---|---|---|
| **0** | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 0.92 |
| **1** | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 0.76 |
| **2** | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 0.72 |
| **3** | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 0.80 |
| **4** | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 0.65 |

Figure 24 : dropping the columns which are not required

● Heatmap: The heatmap is a way of representing data in a two dimensional form. The data values are represented in colours in the graph. The aim of heatmap is to provide a coloured visual summary of the information.

```
sns.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f2ce57ce88>
```


Figure 25 : Heatmap Visualization

- Data normalization is important in order to represent data in comparable scales.

```
from sklearn.preprocessing import MinMaxScaler
xs=MinMaxScaler()
X_train[X_train.columns] = xs.fit_transform(X_train[X_train.columns])
X_test[X_test.columns] = xs.transform(X_test[X_test.columns])
```

```
C:\Users\jsvgp\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\jsvgp\anaconda3\lib\site-packages\pandas\core\frame.py:2969: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

Figure 26 : Data Normalization

### 5.4.1 Building the model (using splitting):

- First we have to retrieve the input and output sets from the given dataset
- Import the train_test_split from model_selection package from scikit learn library
- Then assigning the output to four different variables, before assigning we have to mention the train size or test size as a parameter to train_test_split. Then this method will split according to the size and assigns it to four variables.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =train_test_split(X,Y,test_size=0.2,random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(320, 1)
(80, 1)
(320,)
(80,)
```

Figure 27 : Splitting the data

● Import linear regression method which is available in linear_model package from scikit learn library

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
clf = LinearRegression(normalize=True)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print(r2_score(y_test,y_pred))
```

```
0.7806138920790952
```

Figure 28 : Importing linear regression

● Once the model is built we need to check for accuracy.

● This can be done using the predict method which is used to predict the output for input test set, and compare the predicted output with original output test set.

```
pred=clf.predict(X_test)
pred
```

```
array([0.63018303, 0.73589007, 0.93104153, 0.82126883, 0.58139516,
       0.90054912, 0.58342799, 0.53667295, 0.69726634, 0.87412236,
       0.72775876, 0.91884457, 0.48381943, 0.90054912, 0.71962745,
       0.70743048, 0.70539765, 0.48991791, 0.67287241, 0.97576374,
       0.6505113 , 0.66677393, 0.7379229 , 0.57732951, 0.94120567,
       0.83143297, 0.69726634, 0.61188758, 0.67897089, 0.81923601,
       0.81720318, 0.91681174, 0.62408454, 0.52447598, 0.66067544,
       0.73182441, 0.71962745, 0.65457696, 0.63424868, 0.88225367,
       0.74198855, 0.54480426, 0.75215269, 0.93104153, 0.85379408,
       0.83956428, 0.97169809, 0.69116786, 0.91071326, 0.85582691,
       0.8700567 , 0.69523351, 0.75621835, 0.93917284, 0.53464012,
       0.54480426, 0.69726634, 0.83549863, 0.62611737, 0.82126883,
       0.701332  , 0.67083958, 0.68100372, 0.51227902, 0.63018303,
       0.66677393, 0.61798606, 0.83753146, 0.85582691, 0.81720318,
       0.70743048, 0.83956428, 0.87818801, 0.82940015, 0.56513254,
       0.75418552, 0.71352896, 0.61188758, 0.84159711, 0.80907187])
```

Figure 29 : Predicting the output

```
np.mean(pred)
```

```
0.73657614913454
```

```
np.mean(y_test)
```

```
0.720625
```

Figure 30 : Comparing with original output

● Computing the metrics for mean absolute error, mean squared error, root mean squared error, and R-squared, and put them into a DataFrame.

```
## Test the model on testing data
y_pred = clf.predict(X_test)  # test data--> unseen data
y_pred
## We need to compare the actual values(y_test) and the predicted
                                                    #values(y_test_pred)

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
print('R^2:', r2_score(y_test,y_pred))

print('Adjusted R^2:', 1- (1-r2_score(y_test, y_pred))*(len(X_test)-1)/
                                (len(X_test)-X_test.shape[1]-1))


print('MAE:', mean_absolute_error(y_test, y_pred))

print('MSE:', mean_squared_error(y_test, y_pred))
```

```
R^2: 0.7806138920790952
Adjusted R^2: 0.7778012496698528
MAE: 0.05369334636146487
MSE: 0.0056652863067240605
```

Figure 31 : Getting R-squared value

## Linear Regression Accuracy

```
#Linear Regression Accuracy
clf_score = (clf.score(X_test,y_test))*100
clf_score
```

```
78.06138920790951
```

Figure 32: Linear Regression Accuracy

### 5.4.2 Random Forest:

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

I have used a random forest algorithm for solving this regression problem. The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

n_estimators is the number of trees to be used in the forest. Since Random Forest is an ensemble method consisting of creating multiple decision trees, this parameter is used to control the number of trees to be used in the process.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
```

Figure 33 : Importing Random Forest Regressor

```
from sklearn import metrics
print('MAE:', mean_absolute_error(y_test, y_pred))

print('MSE:', mean_squared_error(y_test, y_pred))

print('RMSE', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
MAE: 0.06130149607683981
MSE: 0.0068883014827861595
RMSE 0.08299579195830449
```

## Random Forest Accuracy

```
#Random forest accuracy
rf_score = (rf.score(X_test, y_test))*100
rf_score
```

```
73.32530836613445
```

Figure 34: Random Forest Accuracy

### 5.4.3 Decision Tree:

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility. Decision-tree algorithms fall under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

**Decision Tree Accuracy**

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
print('R^2:', r2_score(y_test,y_test_pred))

print('Adjusted R^2:', 1- (1-r2_score(y_test, y_test_pred))*(len(X_test)-1)/
                                    (len(X_test)-X_test.shape[1]-1))


print('MAE:', mean_absolute_error(y_test, y_test_pred))

print('MSE:', mean_squared_error(y_test, y_test_pred))

print('RMSE', np.sqrt(mean_squared_error(y_test, y_test_pred)))



dtree_score = (dtree.score(X_test, y_test))*100
dtree_score
```

```
R^2: 0.6793909879856168
Adjusted R^2: 0.6752806160367144
MAE: 0.06349791666666667
MSE: 0.008279201736111108
RMSE 0.09099011889271882

67.93909879856169
```

Figure 35: Decision Tree Accuracy

### 5.4.4 Comparing All The Accuracies:

Among the three Algorithms, Multiple Linear Regression gives the best accuracy result.

```
#Comparing Scores
Methods = ['Linear Regression','Random Forests','Decision Tree']
scores = np.array([clf_score, rf_score, dtree_score])
fig, ax = plt.subplots(figsize=(8,6))
sns.barplot(Methods,scores)
plt.title('Prediction')
plt.ylabel('Accuracy')
```

```
Text(0, 0.5, 'Accuracy')
```

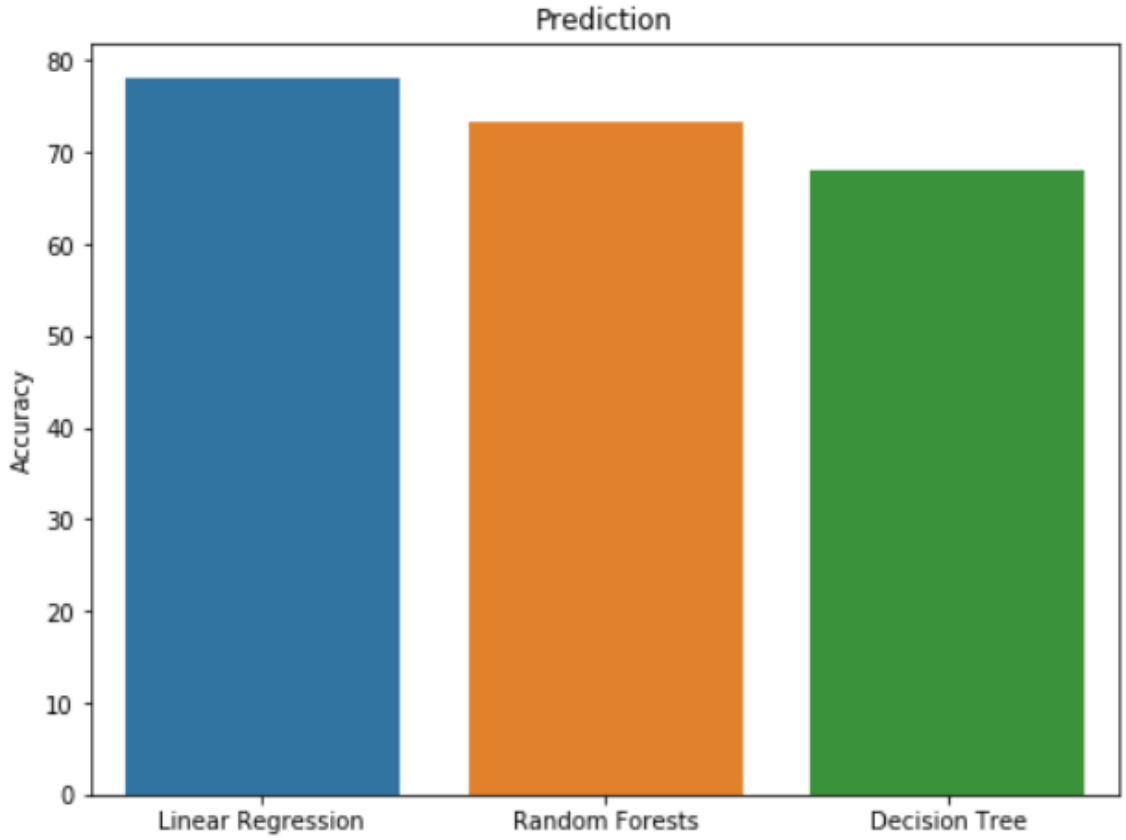


Figure 36: Comparison of Accuracies

# CONCLUSION:

It is concluded after performing thorough Exploratory Data analysis which include Splitting models which are computed to get accuracy and also Heat maps which are computed to get a clear understanding of the data set (which parameter has most abundant effect on the study case) and it's come to point of getting the solution for the problem statement being, Clearly CGPA is the most important criteria for graduate admission followed by GRE and TOEFL score.

As a quick summary, I used multiple linear regression algorithm and random forest algorithm to visualize the importance of each features for graduate admission. This could be of great help for students preparing for their higher studies.

# REFERENCES:


- https://towardsdatascience.com/graduate-admission-prediction-using-machine-learning-8e09ba1af359

- https://www.kaggle.com/utkucanozturk/predict-graduate-admissions-with-python

- https://moodle.dspsinstitute.com/pluginfile.php/6353/mod_resource/content/1/with%20page%20number.pdf