

**MACHINE LEARNING  
(CHURN PREDICTION)**

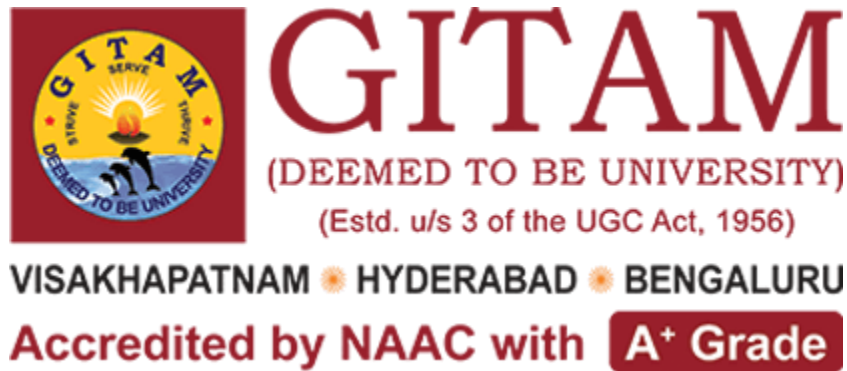
*Summer Internship Report Submitted in partial fulfillment of the  
requirement for undergraduate degree of*

**Bachelor of Technology In  
COMPUTER SCIENCE AND ENGINEERING**

**By**

**M.JASWANTH GOUD  
221710307036**

*Under the Guidance of*



**Department of COMPUTER SCIENCE AND ENGINEERING**

**GITAM School of Technology**

**GITAM (Deemed to be University) Hyderabad-502329**

**July 2020**

## **DECLARATION**

I submit this industrial training work entitled “CHURN PREDICTION” 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 and Engineering”. I declare that it was carried out independently by me under the guidance of, 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

M.JASWANTH GOUD

DATE:30-07-2020221710307036



GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329

Dated:

## **CERTIFICATE**

This is to certify that the Industrial Training Report entitled “CHURN PREDICTION” is being submitted by M.JASWANTH GOUD (221710307036) in partial fulfillment 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 2019-2020.

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

## 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 completion of this internship.

I would like to thank respected **Dr. N. Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad and **Prof.N.Seeta Ramaiah**, Principal, GITAM Hyderabad

I would like to thank respected **Mr.S.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 of 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.

M.JASWANTH GOUD

221710307036

## **ABSTRACT**

With the rapid growth of digital systems and associated information technologies, there is an emerging trend in the global economy to build digital customer relationship management (CRM) systems. This trend is more obvious in the telecommunications industry, where companies become increasingly digitalized. Customer churn prediction is a main feature of in modern telecom communication CRM systems. This research conducts a real-world study on customer churn prediction and proposes the use of boosting to enhance a customer churn prediction model.

Unlike most research that uses boosting as a method to boost the accuracy of a given basis learner, this paper tries to separate customers into two clusters based on the weight assigned by the boosting algorithm. As a result, a higher risk customer cluster has been identified. Logistic regression is used in this research as a basis learner, and a churn prediction model is built on each cluster, respectively. The result is compared with a single logistic regression model. Experimental evaluation reveals that boosting also provides a good separation of churn data; thus, boosting is suggested for churn prediction analysis.

## Table of Contents

<b>CHAPTER 1 .....</b>	<b>1</b>
<b>MACHINE LEARNING.....</b>	<b>1</b>
1.1. INTRODUCTION: .....	1
1.2. IMPORTANCE OF MACHINELEARNING: .....	1
1.3. USES OF MACHINELEARNING: .....	2
1.4. TYPES OF LEARNINGALGORITHMS:.....	2
1.4.1. Supervised Learning:.....	2
1.4.2. Unsupervised Learning: .....	3
1.4.3. Semi Supervised Learning: .....	4
1.5. RELATION BETWEEN DATA MINING, MACHINEAND DEEP LEARNING: .....	4
<b>CHAPTER 2 INFORMATION ABOUT DEEP LEARNING .....</b>	<b>5</b>
2.1 IMPORTANCE OF DEEPLARNING: .....	5
2.2USES OF MACHINELEARNING: .....	5
2.3RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEPLARNING:.....	6
<b>CHAPTER 3 .....</b>	<b>7</b>
<b>PYTHON.....</b>	<b>7</b>
3.1INTRODUCTION TOPYTHON:.....	7
3.2HISTORY OF PYTHON: .....	7
3.3FEATURES OF PYTHON: .....	7
3.4HOW TO SETUP PYTHON: .....	8
3.4.1INSTALLATION (using python IDLE):.....	8
3.4.2Installation (using Anaconda): .....	8
3.5PYTHON VARIABLE TYPES: .....	9
3.5.1Python Numbers: .....	10
3.5.2Python Strings:.....	10
3.5.3Python Lists: .....	11
3.5.4PythonTuples: .....	11
3.5.5 Python Dictionary: .....	11
3.6PYTHON FUNCTION: .....	12
3.6.1Defining a Function: .....	12
3.6.2Calling a Function:.....	12
3.7 PYTHON USING OOPsCONCEPTS: .....	12
3.7.1 Class: .....	12
3.7.2__init__ method in Class: .....	13
<b>CHAPTER 4 .....</b>	<b>14</b>
<b>CASE STUDY.....</b>	<b>14</b>
4.1PROBLEMSTATEMENT: .....	16

4.2 DATA SET: .....	17
4.3 OBJECTIVE OF THE CASE STUDY: .....	17
<b>CHAPTER 5 .....</b>	<b>18</b>
<b>MODEL BUILDING.....</b>	<b>18</b>
5.1 PREPROCESSING OF THE DATA: .....	18
5.1.1 GETTING THE DATASET: .....	18
5.1.2 IMPORTING THE LIBRARIES: .....	18
5.1.3 IMPORTING THE DATA-SET: .....	19
5.1.4 HANDLING MISSING VALUES: .....	20
5.1.4 CATEGORICAL DATA: .....	21
<b>CHAPTER 6 .....</b>	<b>22</b>
<b>DATA PREPROCESSING/FEATURE .....</b>	<b>22</b>
<b>ENGINEERING AND EDA .....</b>	<b>22</b>
6.1 STATISTICAL ANALYSIS: .....	22
6.2 GENERATING PLOTS: .....	22
6.2.1 VISUALIZE THE DATA BETWEEN TARGET AND THE FEATURES .....	23
6.3 DATA TYPE CONVERSIONS: .....	24
6.4 DETECTION OF OUTLIERS: .....	25
6.6 HANDLING MISSING VALUES: .....	25
6.7 ENCODING CATEGORICAL DATA: .....	25
<b>CHAPTER 7 .....</b>	<b>27</b>
<b>FEATURE SELECTION .....</b>	<b>27</b>
7.1 SELECT RELEVANT FEATURES FOR THE ANALYSIS: .....	27
7.2 DROP IRRELEVANT FEATURES: .....	27
7.3 TRAIN-TEST-SPLIT: .....	27
<b>CHAPTER 8 .....</b>	<b>28</b>
<b>MODEL BUILDING AND EVALUATION .....</b>	<b>28</b>
8.1 BRIEF ABOUT THE ALGORITHMS USED .....	28
8.2 TRAIN THE MODELS.....	29
8.2.1 K-nearest neighbors classifier: .....	29
8.2.2 Make Predictions: .....	29
8.2.2.1 Predictions from: .....	30
1. Raw Data .....	30
2. FOR TRAIN DATA: .....	30
3. FOR TEST DATA: .....	31
8.2.3 Random Forest Classifier: .....	32
8.2.3.1 Make Predictions: .....	32
8.2.3.2 Predictions from .....	33
1. Raw data: .....	33
2. FOR TRAIN DATA: .....	33
3. FOR TEST DATA: .....	34
8.2.4 LOGISTIC REGRESSION: .....	35

8.2.4.1 MakePredictions: .....	35
8.2.4.2 Predictions from.....	36
1. Rawdata: .....	36
2. FOR TRAIN DATA: .....	36
3. FOR TEST DATA: .....	37
<b>CHAPTER 9 .....</b>	<b>39</b>
<b>COMPARING THE PERFORMANCES OF ALL THE MODELS.....</b>	<b>39</b>
<b>CONCLUSION.....</b>	<b>41</b>
<b>REFERENCES .....</b>	<b>42</b>



## List of Figures

Figure 1.2. 1: The Process Flow .....	1
Figure 1.4.2.1: Unsupervised Learning .....	3
Figure 1.4.3.1.Semi Supervised Learning .....	4
Figure 3.4.1.1. Python download .....	8
Figure 3.4.2.1: Anaconda download .....	9
Figure 3.4.2.2.Jupyter notebook.....	9
Figure 3.7.1.1: Defining a Class.....	13
Figure 5.1.2.1: Importing Libraries .....	18
Figure 5.1.3.1:Reading the dataset .....	19
Figure 5.1.3.2:Displaying the train data.....	<b>Error! Bookmark not defined.</b>
Figure 5.1.3 3: Displaying the test data.....	<b>Error! Bookmark not defined.</b>
Figure 5.1.4.1:Displaying train data after drop .....	<b>Error! Bookmark not defined.</b>
Figure 5.1.4.2: Checking for null values .....	<b>Error! Bookmark not defined.</b>
Figure 6. 1:Describing the data .....	22
Figure 6.2. 1:Visualization of the data .....	23
Figure 6.3. 1:Data type conversions.....	24
Figure 6.4. 1:Checking for null values.....	25
Figure 6.7. 1:Encoding the data .....	26
Figure 6.7. 2:After encoding displaying the data .....	26
Figure 7.2. 1:Dropping irrelevant features .....	27
Figure 7.2. 2:Importing train-test split .....	27
Figure 8.2.1: prediction on train and test data.....	29
Figure 8.2.2: Importing KNN and fitting the model .....	29
Figure 8.2.3: prediction on train and test data.....	29

Figure 8.2.2.2.1: train and test accuries.....	30
Figure 8.2.2.2.2: confusion matrix for train data .....	30
Figure 8.2.2.2.3: confusion matrix for test data .....	31
Figure 8.2.2.2.4: Classification report.....	32
Figure 8.2.3. 1: Importing random forest classifier.....	32
Figure 8.2.3.1. 1:prediction on train and test .....	32
Figure 8.2.3.2.1: Accuracy for test and train data .....	33
Figure 8.2.3.2.2: classification report.....	33
Figure 8.2.3.2.3: Confusion matrix for train data.....	33
Figure 8.2.3.2.4: confusion matrix for test data .....	34
Figure 8.2.3.2.5 : classification report.....	34
Figure 8.2.4. 1: Importing logistic regression method and fitting the model.....	35
Figure 8.2.4.1. 1: prediction on train and test .....	35
Figure 8.2.4.2. 1: train and test accuries.....	36
Figure 8.2.4.2.2: confusion matrix for train data .....	36
Figure 8.2.4.2.3: classification report.....	37
Figure 8.2.4.2.4: confusion matrix for test data .....	37
Figure 8.2.4.2.5: classification report.....	38
Figure 9. 1: Accuracies of all the 3 models.....	39
Figure 9. 2: Visualizing the train accuracies .....	39
Figure 9. 3 : Visualizing the test accuracies of all 3 models .....	40

# 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).

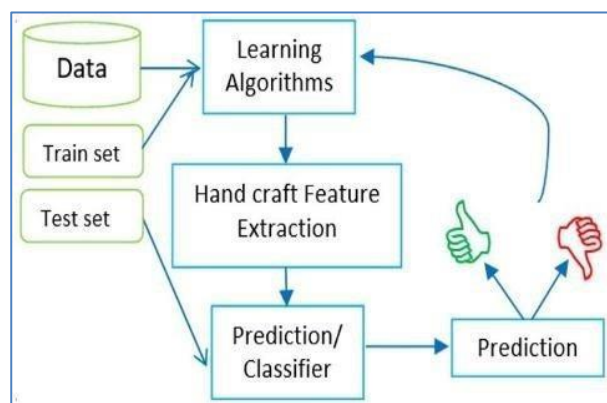
### 1.2. IMPORTANCE OF MACHINELEARNING:

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 Face book, 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 1.2.1: The Process Flow**

### **1.3. USES OF MACHINELEARNING:**

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 LEARNINGALGORITHMS:**

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.

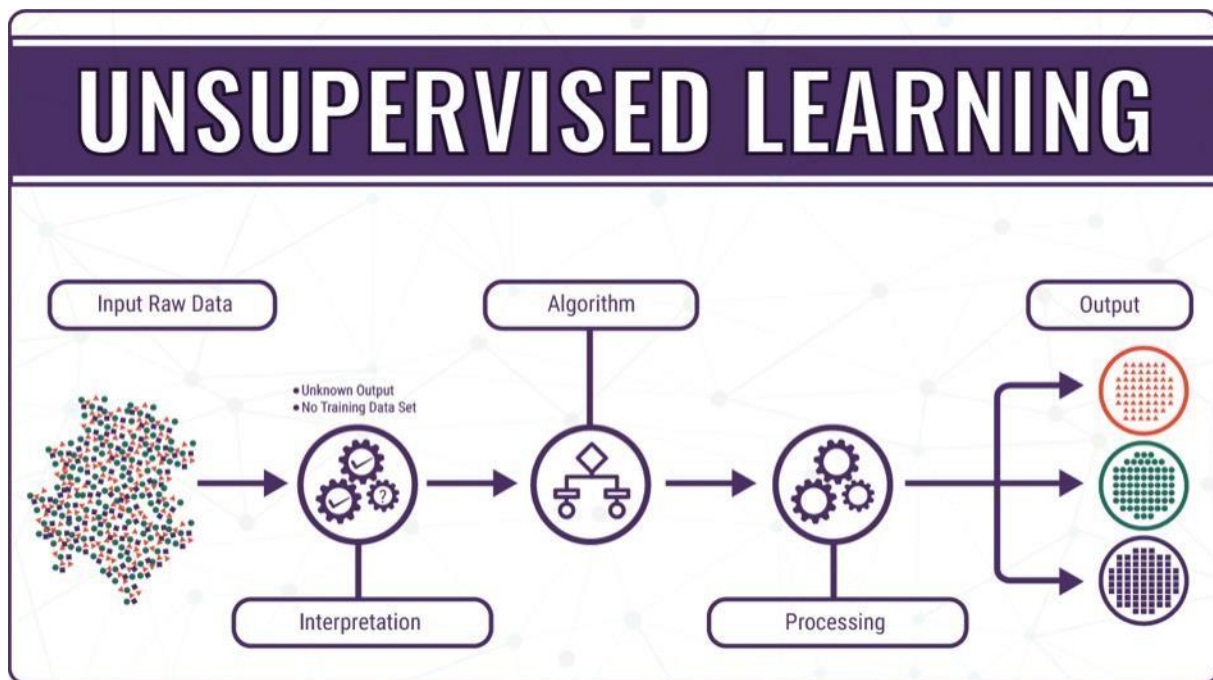
Supervised machine learning algorithms uncover insights, patterns, and relationships from a labeled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem 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 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 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.

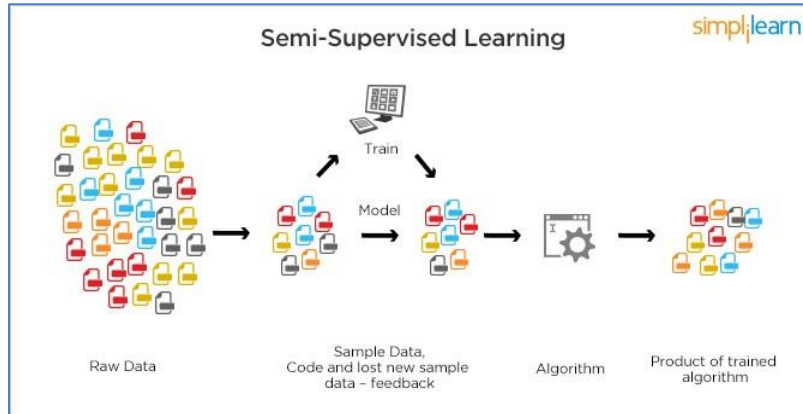


**Figure 1.4.2.1: Unsupervised Learning**

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.

### 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 1.4.3.1.Semi Supervised Learning**

## 1.5. RELATION BETWEEN DATA MINING, MACHINE 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 discovered previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making.

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 2**

### **INFORMATION ABOUT DEEP LEARNING**

#### **2.1 Importance of Deep Learning:**

Deep learning recently returned to the headlines when google's AlphaGo program crushed Lee Sedol, one of the highest-ranking Go players in the world. Google has invested heavily in deep learning and AlphaGo is just their latest deep learning project to make the news. Google's search engine, voice recognition system and self-driving cars all rely heavily on deep learning. They've used deep learning networks to build a program that picks out an alternative still from a YouTube video to use as a thumbnail. Late last year Google announced smart reply, a deep learning network that writes short email responses for you. Deep learning is clearly powerful, but it also may seem somewhat mysterious.

#### **2.2 Uses of Machine Learning:**

The value of machine learning technology has been recognized by companies across several industries that deal with huge volumes of data. By leveraging insights obtained from this data, companies are able work in an efficient manner to control costs as well as get an edge over their competitors. This is how some sectors / domains are implementing machine learning-

- **Financial Services**

Companies in the financial sector are able to identify key insights in financial data as well as prevent any occurrences of financial fraud, with the help of machine learning technology. The technology is also used to identify opportunities for investments and trade. Usage of cyber surveillance helps in identifying those individuals or institutions which are prone to financial risk, and take necessary actions in time to prevent fraud.

- **Marketing and Sales**

Companies are using machine learning technology to analyze the purchase history of their customers and make personalized product recommendations for their next purchase. This ability to capture, analyze, and use customer data to provide a personalized shopping experience is the future of sales and marketing.

- **Government**

Government agencies like utilities and public safety have a specific need FOR ML, as they have multiple data sources, which can be mined for identifying useful patterns and insights. For example sensor data can be analyzed to identify ways to minimize costs and increase efficiency. Furthermore, ML can also be used to minimize identity thefts and detect fraud.

- **Healthcare**

With the advent of wearable sensors and devices that use data to access health of a patient in real time, ML is becoming a fast-growing trend in healthcare.

Sensors in wearable provide real-time patient information, such as overall health condition, heartbeat, blood pressure and other vital parameters. Doctors and medical experts can use this information to analyze the health condition of an individual, draw a pattern from the patient history, and predict the occurrence of any ailments in the future. The technology also empowers medical experts to analyze data to identify trends that facilitate better diagnoses and treatment.

- **Transportation**

Based on the travel history and pattern of traveling across various routes, machine learning can help transportation companies predict potential problems that could arise on certain routes, and accordingly advise their customers to opt for a different route. Transportation firms and delivery organizations are increasingly using machine learning technology to carry out data analysis and data modeling to make informed decisions and help their customers make smart decisions when they travel.

- **Oil and Gas**

This is perhaps the industry that needs the application of machine learning the most. Right from analyzing underground minerals and finding new energy sources to streaming oil distribution, ML applications for this industry are vast and are still expanding.

## **2.3 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEPLARNING:**

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

### **3.1INTRODUCTION TOPYTHON:**

- 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 andPHP.
- 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.

### **3.2HISTORY 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

### **3.3FEATURES 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 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OSX. 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 MacOS distributions include a recent python.
- Download python from [www.python.org](http://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 3.4.1.1. Python download

#### 3.4.2 Installation (using Anaconda):

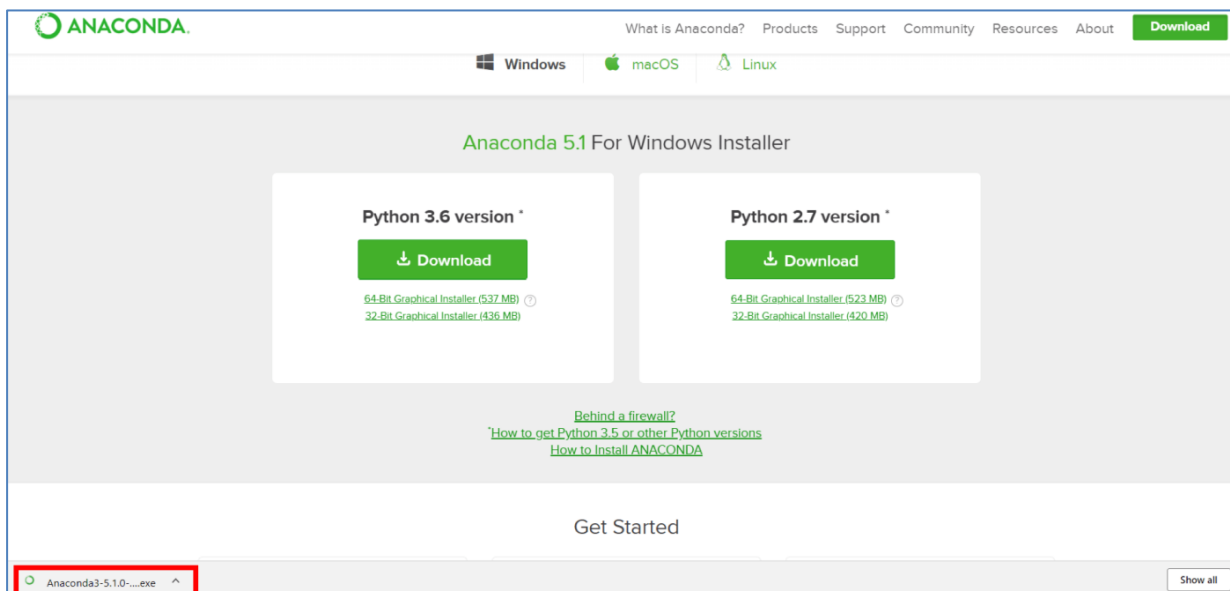
- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive

analytics and scientific computing.

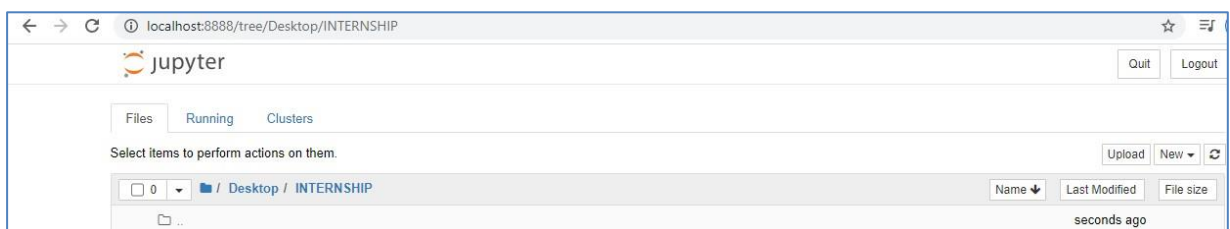
- Conda is a package manager that quickly installs and manages packages.

### In WINDOWS:

- Step 1: Open Anaconda.com/downloads in 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 python3.4) next click install and next click finish.
- Step 5: Open Jupyter notebook ( it opens in default browser)



**Figure 3.4.2.1: Anaconda download**



**Figure 3.4.2.2.Jupyter notebook**

## 3.5PYTHON 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 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

### **3.5.1Python 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).

### **3.5.2Python 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 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 a 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.

## **3.6PYTHON FUNCTION:**

### **3.6.1Defining 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`.

### **3.6.2Calling 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 OOPsCONCEPTS:**

### **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:**

- We define a class in a very similar way how we define a function.
- 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 3.7.1.1: 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 ().

# **CHAPTER 4**

## **CASE STUDY**

### **PROJECT NAME-CHURN PREDICTION**

#### **Packages used:**

##### **1.Pandas-**

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

##### **2.Numpy-**

NumPy is a python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

##### **3.Matplotlib.pyplot-**

It is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

##### **4.Seaborn-**

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

##### **5.sklearn.preprocessing import LabelEncoder-**

In machine learning, we usually deal with datasets which contains multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human readable form, the training data is often labeled in words.



**Label Encoding** refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

## **6.sklearn.preprocessing import StandardScaler-**

Standardize features by removing the mean and scaling to unit variance

### **\* Parameters:**

-Copyboolean:

If False, try to avoid a copy and do inplace scaling instead. This is not guaranteed to always work inplace; e.g. if the data is not a NumPy array or scipy.sparse CSR matrix, a copy may still be returned.

-with\_meanboolean,

If True, center the data before scaling. This does not work (and will raise an exception) when attempted on sparse matrices, because centering them entails building a dense matrix which in common use cases is likely to be too large to fit in memory.

-with\_stdboolean:

If True, scale the data to unit variance (or equivalently, unit standard deviation).

## **7.sklearn.linear\_model import LogisticRegression-**

One of the most amazing things about Python's scikit-learn library is that it has a 4-step modeling pattern that makes it easy to code a machine learning classifier. While this tutorial uses a classifier called Logistic Regression, the coding process in this tutorial applies to other classifiers in sklearn (Decision Tree, K-Nearest Neighbors etc). In this tutorial, we use Logistic Regression to predict digit labels based on images. The image above shows a bunch of training digits (observations) from the MNIST dataset whose category membership is known (labels 0–9). After training a model with logistic regression, it can be used to predict an image label (labels 0–9) given an image.

### **\*Parameters-**

-dualbool

-tolfloat

-Cfloat

-fit\_interceptbool

## **8.sklearn.metrics import classification\_report, confusion\_matrix-**

### **\*Parameters-**

-y\_true  
-y\_pred  
-target\_nameslist of string  
-digitsint

#### 9.sklearn.neighbors importKNeighborsClassifier-

Sklearn.neighbors provides functionality for unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering. Supervised neighbors-based learning comes in two flavors: classification for data with discrete labels, and regression for data with continuous labels.

##### **\*Parameters-**

-n\_neighborsint  
-leaf\_sizeint  
-pint

#### 10.sklearn.ensemble importRandomForestClassifier-

A random forest classifier.A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

#### **Algorithms used:**

1. NAIVE BAYESALGORITHM
2. LOGISTIC REGRESSIONALGORITHM
3. RANDOM FOREST CLASSIFIERALGORITHM

### **4.1PROBLEMSTATEMENT:**

To Predict Customer Churn Model based on various Variables like Customer Profile, Customer Account Information & particular services calls etc.

## **4.2 DATA SET:**

1. State
2. Accountlength
3. Areacode
4. Internationalplan
5. Voice mailpan
6. Number vmailmessages
7. Total dayminutes
8. Total daycalls
9. Total daycharge
10. Total eveminutes
11. Total evecalls
12. Total evecharge
13. Total nightminutes
14. Total nightcalls
15. Total night charge Total dayminutes
16. Total intlcalls
17. Total intlcharge
18. Customer servicecalls
19. Churn

## **4.3 OBJECTIVE OF THE CASE STUDY:**

In an Online business, with multiple competitors in the same business it's really important to re-engage existing customers and keep them from churning. This project is my attempt to make a sample model for a company to predict customer's behaviour and prevent them from abandoning their product.

## CHAPTER 5

### MODELBUILDING

#### 5.1PREPROCESSING OF THE DATA:

**Data preprocessing** in **Machine Learning** refers to the technique of preparing (cleaning and organizing) the raw **data** to make it suitable for a building and training **machine learning** models.

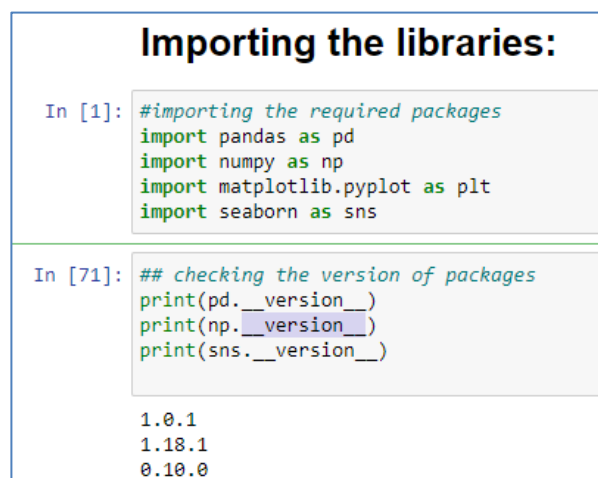
Pre-processing of the data actually involves the following steps:

##### 5.1.1GETTING THEDATASET:

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

##### 5.1.2IMPORTING THELIBRARIES:

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



```
Importing the libraries:

In [1]: #importing the required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

In [71]: ## checking the version of packages
print(pd.__version__)
print(np.__version__)
print(sns.__version__)

1.0.1
1.18.1
0.10.0
```

**Figure 5.1.2.1: Importing Libraries**

The versions of the packages used are:

- Pandas:1.0.1
- Numpy:1.18.1
- Seaborn:0.10.0

### 5.1.3IMPORTING THEDATA-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 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 dataframe. Any missing value or NaNvalue have to be cleaned.

#### Acquiring the data:

```
In [2]: #Read the data sets:
train=pd.read_csv('churn-bigml-train.csv')
test=pd.read_csv('churn-bigml-test.csv')
print(train.shape) #to know the shape of train data
print(test.shape) #to know the shape of test data

(2666, 20)
(667, 20)
```

Figure 5.1.3.1:Reading the dataset

```
[3]: train.head() #displaying train data
```

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Cus s
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35	
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83	19.42	208.8	111	9.40	12.7	6	3.43	
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97	27.01	160.6	128	7.23	5.4	9	1.46	
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102	11.67	189.6	105	8.53	7.7	6	2.08	
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109	18.28	178.7	90	8.04	11.1	1	3.00	

Figure 5.1.3.2: displaying the train data

```
[4]: test.head() #displaying the test data
```

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Cus s
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	

Figure 5.1.3.3 : Displaying the test data

## 5.1.4 HANDLING MISSING VALUES:

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

- a) \*dropna()
- b) \*fillna()
- c) \*interpolate()
- d) \*mean imputation and median imputation.

```
n [8]: train.drop(['Area code', 'State'], axis=1, inplace=True)
       test.drop(['Area code', 'State'], axis=1, inplace=True)

n [9]: train.head() #displaying after drop
ut[9]:
```

	Account length	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
0	117	No	No	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35	1	False
1	65	No	No	0	129.1	137	21.95	228.5	83	19.42	208.8	111	9.40	12.7	6	3.43	4	True
2	161	No	No	0	332.9	67	56.59	317.8	97	27.01	160.6	128	7.23	5.4	9	1.46	4	True
3	111	No	No	0	110.4	103	18.77	137.3	102	11.67	189.6	105	8.53	7.7	6	2.08	2	False
4	49	No	No	0	119.3	117	20.28	215.1	109	18.28	178.7	90	8.04	11.1	1	3.00	1	False

Figure 5.1.4.1 : displaying train data after drop

### Checking for Null values:

```
] : train.isnull().sum() #none null values in train

]: Account length           0
   International plan       0
   Voice mail plan          0
   Number vmail messages   0
   Total day minutes        0
   Total day calls          0
   Total day charge         0
   Total eve minutes        0
   Total eve calls          0
   Total eve charge         0
   Total night minutes      0
   Total night calls        0
   Total night charge       0
   Total intl minutes       0
   Total intl calls         0
   Total intl charge        0
   Customer service calls   0
   Churn                    0
   dtype: int64
```

Figure 5.1.4.2: Checking for null values

```
|: test.isnull().sum() #none null value in test

|: Account length           0
   International plan      0
   Voice mail plan         0
   Number vmail messages   0
   Total day minutes       0
   Total day calls         0
   Total day charge        0
   Total eve minutes       0
   Total eve calls         0
   Total eve charge        0
   Total night minutes     0
   Total night calls       0
   Total night charge      0
   Total intl minutes      0
   Total intl calls        0
   Total intl charge       0
   Customer service calls  0
   Churn                   0
dtype: int64
```

#### 5.1.4 CATEGORICALDATA:

- 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.
- Categorical data can be handled by using dummy variables, which are also called as indicatorvariables
- 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 dataframe or we can add that dummy set to thedataframe.

# CHAPTER 6

## DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

### 6.1 Statistical Analysis:

-In statistical analysis, one of the possible analyses that can be conducted is to verify that the data fits a specific distribution, in other words that the data matches a specific theoretical model.

-It is also called as Distribution fitting.

The describing of data is done as follow:

Describing the data:

```
] train.describe()
]:
```

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total
count	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000
mean	102.841079	436.157421	8.407796	180.948126	100.937031	30.761769	203.355322	100.476762	17.285262	199.685307	100.113943	8.985907	10.000000
std	40.819480	41.783305	13.994480	55.508628	20.396790	9.436463	49.719268	18.948262	4.226160	49.759931	20.172505	2.239429	2.000000
min	1.000000	408.000000	0.000000	25.900000	30.000000	4.400000	48.100000	37.000000	4.090000	23.200000	42.000000	1.040000	0.000000
25%	76.000000	408.000000	0.000000	146.250000	87.500000	24.860000	171.050000	88.000000	14.540000	167.950000	86.000000	7.560000	8.000000
50%	102.000000	415.000000	0.000000	178.300000	101.000000	30.310000	203.700000	101.000000	17.310000	201.600000	100.000000	9.070000	10.000000
75%	128.000000	415.000000	20.000000	220.700000	115.000000	37.520000	236.450000	113.000000	20.095000	231.500000	113.500000	10.420000	12.000000
max	232.000000	510.000000	51.000000	334.300000	165.000000	56.830000	361.800000	168.000000	30.750000	367.700000	175.000000	16.550000	18.000000

Figure 6.1: Describing the data

### 6.2 Generating Plots:

- Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
- By matplotlib.pyplot we can display a bar chart, line chart, Scatter plots, histograms.



## 6.2.1 Visualize the data between Target and the Features

### Visualizations:

```
In [37]: plt.subplots(figsize=(8,6))  
sns.heatmap(train.corr(),annot=True,fmt=".3f")
```

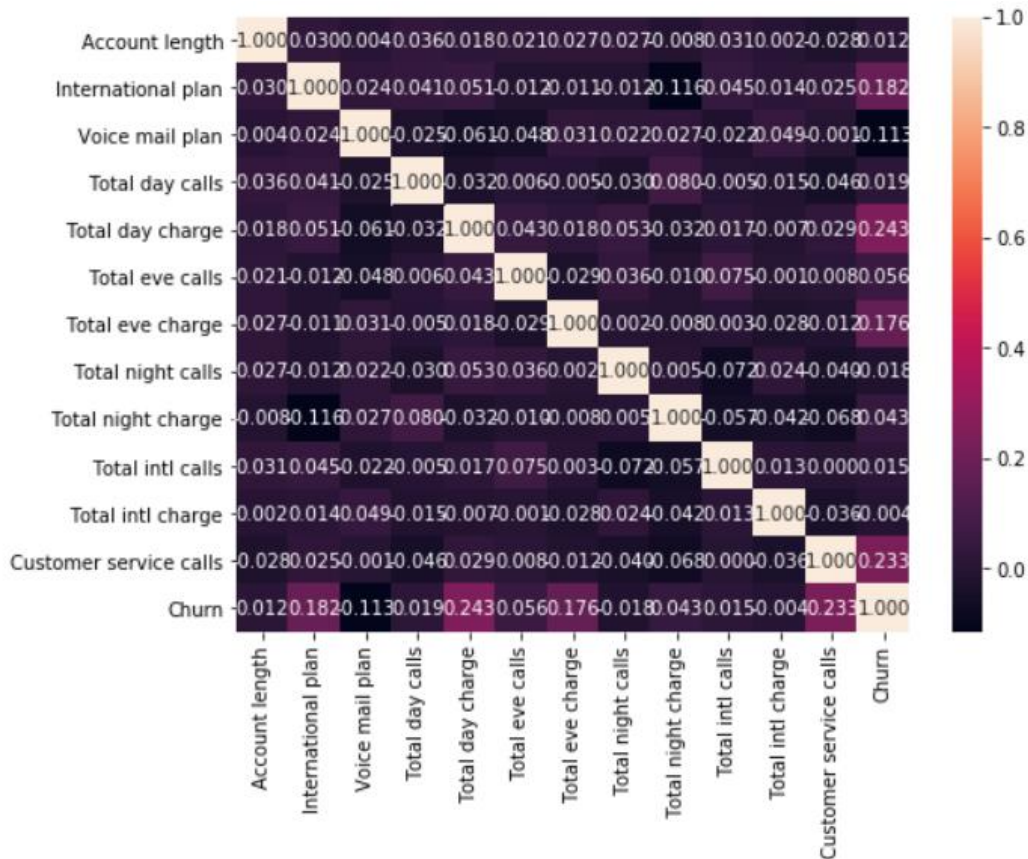


Figure 6.2.1: Visualization of the data

## 6.3 Data TypeConversions:

In my dataset I have two columns namely international plan, voice mail plan of type string object and my target column (i.e.,churn) of type Boolean and the rest are of type integer and float.

```
: train.dtypes
: State                object
: Account length      int64
: Area code           int64
: International plan   object
: Voice mail plan      object
: Number vmail messages int64
: Total day minutes    float64
: Total day calls      int64
: Total day charge     float64
: Total eve minutes    float64
: Total eve calls      int64
: Total eve charge     float64
: Total night minutes  float64
: Total night calls    int64
: Total night charge   float64
: Total intl minutes   float64
: Total intl calls     int64
: Total intl charge    float64
: Customer service calls int64
: Churn               bool
dtype: object
```

**Figure 6.3.1:Data type conversions**

## 6.4 Detection of Outliers:

We don't have any outliers in the data.

## 6.6 Handling Missing Values:

We don't have any missing values in the given data.

### Checking for Null values:

```
: train.isnull().sum() #none null values in train
: Account length          0
: International plan       0
: Voice mail plan         0
: Number vmail messages   0
: Total day minutes       0
: Total day calls         0
: Total day charge        0
: Total eve minutes       0
: Total eve calls         0
: Total eve charge        0
: Total night minutes     0
: Total night calls       0
: Total night charge      0
: Total intl minutes      0
: Total intl calls        0
: Total intl charge       0
: Customer service calls  0
: Churn                   0
dtype: int64
```

**Figure 6.4.1: Checking for null values**

## 6.7 Encoding Categorical Data:

- The categorical columns in my dataset are: International plan, voice mail plan and target column.
- I have used LabelEncoder and fit\_transform method to encode my categorical data into zeros and ones.

## Encoding the given dataset using label encoding:

```
In [15]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
#encoding international plan
train["International plan"]=le.fit_transform(train["International plan"])
test["International plan"]=le.fit_transform(test["International plan"])
#encoding voice mail plan
train["Voice mail plan"]=le.fit_transform(train["Voice mail plan"])
test["Voice mail plan"]=le.fit_transform(test["Voice mail plan"])
#encoding churn
train["Churn"]=le.fit_transform(train["Churn"])
test["Churn"]=le.fit_transform(test["Churn"])
train.head()
```

Figure 6.7.1:Encoding the data

	Account length	International plan	Voice mail plan	Total day calls	Total day charge	Total eve calls	Total eve charge	Total night calls	Total night charge	Total intl calls	Total intl charge	Customer service calls	Churn
0	117	0	0	97	31.37	80	29.89	90	9.71	4	2.35	1	0
1	65	0	0	137	21.95	83	19.42	111	9.40	6	3.43	4	1
2	161	0	0	67	56.59	97	27.01	128	7.23	9	1.46	4	1
3	111	0	0	103	18.77	102	11.67	105	8.53	6	2.08	2	0
4	49	0	0	117	20.28	109	18.28	90	8.04	1	3.00	1	0

Figure 6.7.2:After encoding displaying the data

## CHAPTER 7

### FEATURE SELECTION

- Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model.
- The data features that you use to train your machine learning models have a huge influence on the performance you can achieve.
- Irrelevant or partially relevant features can negatively impact model performance.

#### 7.1 Select relevant features for the analysis:

The irrelevant features in my dataset are area code and state. These features are considered does not effect my target column. Rest of the columns are all relevant to the target column.

#### 7.2 Drop irrelevant features:

```
In [8]: train.drop(['Area code', 'State'], axis=1, inplace=True)
        test.drop(['Area code', 'State'], axis=1, inplace=True)
```

**Figure 7.2.1: Dropping irrelevant features**

#### 7.3 Train-Test-Split:

- Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation and `next(ShuffleSplit().split(X, y))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

##### **Train-Test Split:**

```
In [17]: X_train = train.drop(["Churn"], axis=1)
        y_train = train.Churn
        X_test = test.drop(["Churn"], axis=1)
        y_test = test.Churn
```

**Figure 7.2.2: Importing train-test split**

## CHAPTER 8

### MODEL BUILDING AND EVALUATION

#### 8.1 Brief about the algorithms used

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its  $K$  nearest neighbors measured by a distance function. If  $K = 1$ , then the case is simply assigned to the class of its nearest neighbor.

In pattern recognition, the ***k*-nearest neighbors algorithm (*k*-NN)** is a non parametric method proposed by Thomas Cover used for classification and regression. In both cases, the input consists of the  $k$  closest training examples in the feature space. The output depends on whether  $k$ -NN is used for classification or regression:

- In *k*-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its  $k$  nearest neighbors ( $k$  is a positive integer, typically small). If  $k = 1$ , then the object is simply assigned to the class of that single nearest neighbor.
- In *k*-NN regression, the output is the property value for the object. This value is the average of the values of  $k$  nearest neighbors.

$k$ -NN is a type of **instance-based learning**, or **lazy learning**, where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, **normalizing** the training data can improve its accuracy dramatically.

## 8.2 Train the Models

### 8.2.1 K-nearest neighbors classifier:

- I have used “KNeighborsClassifier()” and created an object for my model.
- I have fitted the training dataset to the model by using “fit()”.

```
knn = KNeighborsClassifier(n_neighbors=20, metric='euclidean')
knn.fit(X_train,y_train) #training the model
knn_train_pred=knn.predict(X_train) #predicting on train data
knn_test_pred=knn.predict(X_test) #predicting on test data
```

Figure 8.2.1: prediction on train and test data

- I have used “KNeighborsClassifier()” and created an object for my model.
- I have fitted the training dataset to the model by using “fit()”.

```
: from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=20, metric='euclidean')
knn.fit(X_train,y_train) #training the model
knn_train_pred=knn.predict(X_train) #predicting on train data
knn_test_pred=knn.predict(X_test) #predicting on test data
```

Figure 8.2.2: Importing KNN and fitting the model

### 8.2.2 Make Predictions:

- I have sent X\_train (i.e., my training dataset) to the predict() as argument and I have stored the results in knn\_train\_predict.
- I have sent X\_test (i.e., my testing dataset) to the predict() as argument and I have stored the results in knn\_test\_predict.

```
knn_train_pred=knn.predict(X_train) #predicting on train data
knn_test_pred=knn.predict(X_test) #predicting on test data
```

Figure 8.2.3: prediction on train and test data

### 8.2.2.1 Predictions from: 1.Raw Data

```
In [19]: from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=20, metric='euclidean')
knn.fit(X_train,y_train) #training the model
knn_train_pred=knn.predict(X_train) #predicting on train data
knn_test_pred=knn.predict(X_test) #predicting on test data
#Checking the metrics
from sklearn.metrics import accuracy_score
train_accu[0]=accuracy_score(y_train,knn_train_pred)
test_accu[0]=accuracy_score(y_test,knn_test_pred)
print("LogisticRegression Train Accuracy: ",train_accu[0])
print("LogisticRegression Test Accuracy: ",test_accu[0])
```

LogisticRegression Train Accuracy: 0.8575712143928036  
LogisticRegression Test Accuracy: 0.854463615903976

Figure 8.2.2.2.1: train and test accuries

### 2.FOR TRAIN DATA:

```
0]: from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_train,knn_train_pred),annot=True,fmt='.1f')

0]: <matplotlib.axes._subplots.AxesSubplot at 0x221864c0388>
```

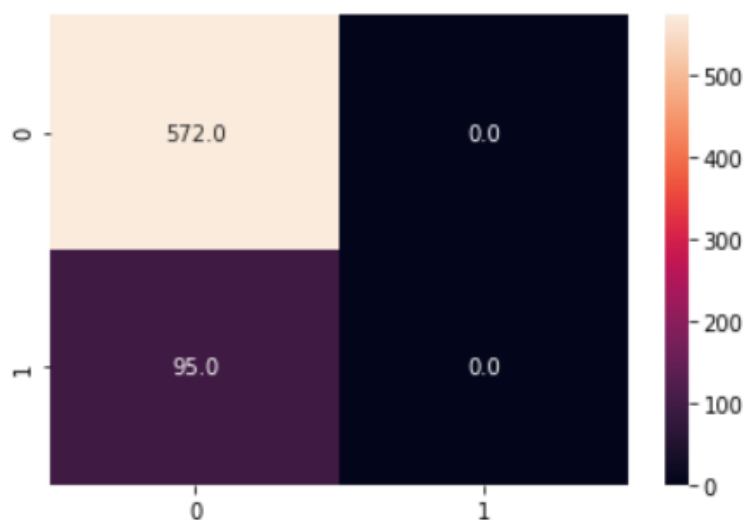


Figure 8.2.2.2.2: confusion matrix for train data



```
] : print(classification_report(y_train,knn_train_pred))
```

	precision	recall	f1-score	support
0	0.86	1.00	0.92	572
1	0.00	0.00	0.00	95
accuracy			0.86	667
macro avg	0.43	0.50	0.46	667
weighted avg	0.74	0.86	0.79	667

Figure8.2.2.2.3: classification report

### 3.FOR TEST DATA:

```
] : from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_test,knn_test_pred),annot=True,fmt='.1f')
]: <matplotlib.axes._subplots.AxesSubplot at 0x22186a29b88>
```

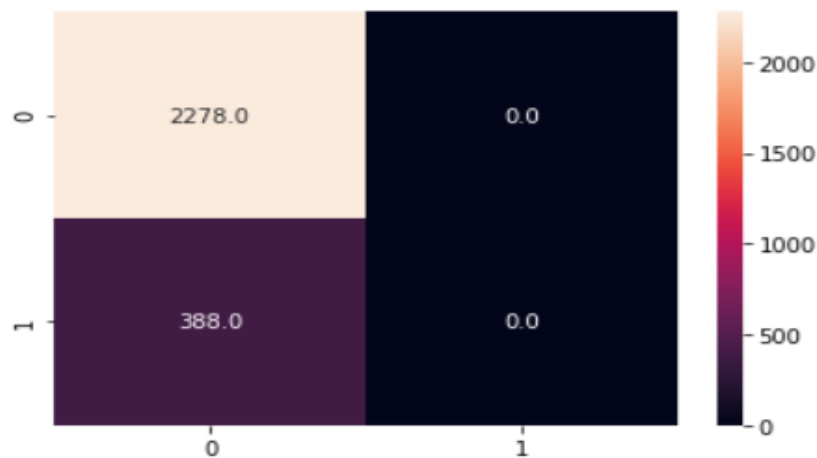


Figure 8.2.2.2.3: confusion matrix for test data

```
print(classification_report(y_test,knn_test_pred))
```

	precision	recall	f1-score	support
0	0.85	1.00	0.92	2278
1	0.00	0.00	0.00	388
accuracy			0.85	2666
macro avg	0.43	0.50	0.46	2666
weighted avg	0.73	0.85	0.79	2666

**Figure 8.2.2.2.4: Classification report**

### 8.2.3 Random Forest Classifier:

- I have used “RandomForestClassifier ()” and created an object for my model.
- I have fitted the training dataset to the model by using “fit()”.

```
In [24]: from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
clf_train_pred=clf.predict(X_train)
clf_test_pred=clf.predict(X_test)
```

**Figure 8.2.3.1: Importing random forest classifier**

#### 8.2.3.1 Make Predictions:

I have sent X\_train (i.e., my training dataset) to the predict() as argument and I have stored the results in clf\_train\_predict.

I have sent X\_test (i.e., my testing dataset) to the predict() as argument and I have stored the results in clf\_test\_predict.

```
lr_train_pred=lr.predict(X_train) #predicting on train data
lr_test_pred=lr.predict(X_test) #predicting on test data
```

**Figure 8.2.3.1.1: prediction on train and test**

### 8.2.3.2 Predictions from

#### 1. Rawdata:

```
#Checking the metrics
train_accu[1]=accuracy_score(y_train,clf_train_pred)
test_accu[1]=accuracy_score(y_test,clf_test_pred)
print("LogisticRegression Train Accuracy: ",train_accu[1])
print("LogisticRegression Test Accuracy: ",test_accu[1])
```

```
LogisticRegression Train Accuracy: 1.0
LogisticRegression Test Accuracy: 0.9017254313578394
```

Figure 8.2.3.2.1: Accuracy for test and train data

#### 2.FOR TRAIN DATA:

```
print(classification_report(y_train,clf_train_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	572
1	1.00	1.00	1.00	95
accuracy			1.00	667
macro avg	1.00	1.00	1.00	667
weighted avg	1.00	1.00	1.00	667

Figure 8.2.3.2.2: classification report

```
] from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_train,clf_train_pred),annot=True,fmt='.1f')
]: <matplotlib.axes._subplots.AxesSubplot at 0x22186756dc8>
```

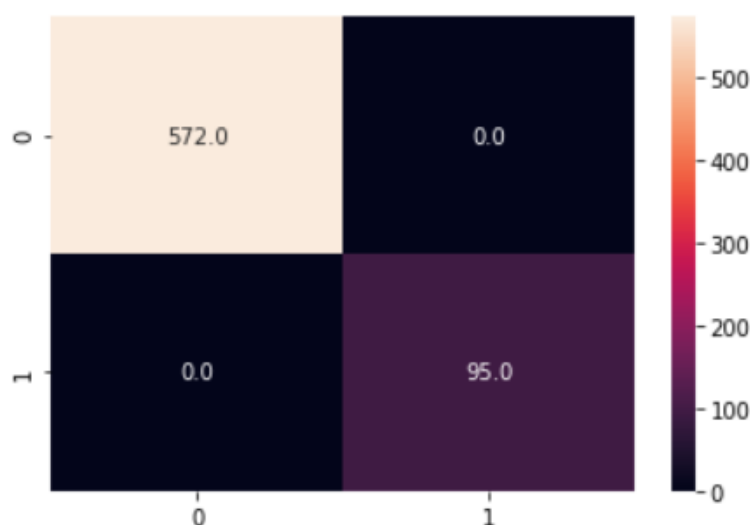


Figure 8.2.3.2.3: Confusion matrix for train data

### 3.FOR TEST DATA:

```
In [27]: from sklearn.metrics import classification_report, confusion_matrix  
sns.heatmap(confusion_matrix(y_test, clf_test_pred), annot=True, fmt='.1f')
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x221867ef9c8>
```

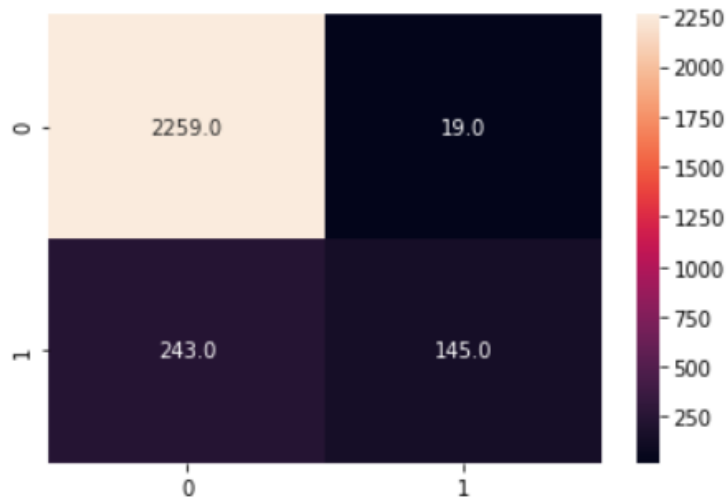


Figure 8.2.3.2.4: confusion matrix for test data

```
In [28]: print(classification_report(y_test, clf_test_pred))
```

	precision	recall	f1-score	support
0	0.90	0.99	0.95	2278
1	0.88	0.37	0.53	388
accuracy			0.90	2666
macro avg	0.89	0.68	0.74	2666
weighted avg	0.90	0.90	0.88	2666

Figure 8.2.3.2.5: classification report

## 8.2.4 LOGISTIC REGRESSION:

### Brief about the algorithms used.

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

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).

This class implements regularized logistic regression using the ‘liblinear’ library, ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ solvers. Note that regularization is applied by default. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

- I have used “LogisticRegression()” and created an object for my model.
- I have fitted the training dataset to the model by using “fit()”.

```
In [29]: from sklearn.linear_model import LogisticRegression #importing the model
lr = LogisticRegression() #creating an object for the model
lr.fit(X_train,y_train) #training the model
lr_train_pred=lr.predict(X_train) #predicting on train data
lr_test_pred=lr.predict(X_test) #predicting on test data
```

**Figure 8.2.4.1: Importing logistic regression method and fitting the model**

### 8.2.4.1 Make Predictions:

I have sent X\_train (i.e., my training dataset) to the predict() as argument and I have stored the results in lr\_train\_predict.

I have sent X\_test (i.e., my testing dataset) to the predict() as argument and I have stored the results in lr\_test\_predict.

```
lr_train_pred=lr.predict(X_train) #predicting on train data
lr_test_pred=lr.predict(X_test) #predicting on test data
```

**Figure 8.2.4.1.1: prediction on train and test**

## 8.2.4.2 Predictions from

### 1. Rawdata:

My model has achieved an accuracy score of 86% for training data and 85% from testing.

```
#Checking the metrics
train_accu[2]=accuracy_score(y_train,lr_train_pred)
test_accu[2]=accuracy_score(y_test,lr_test_pred)
print("LogisticRegression Train Accuracy: ",train_accu[2])
print("LogisticRegression Test Accuracy: ",test_accu[2])
```

```
LogisticRegression Train Accuracy:  0.8590704647676162
LogisticRegression Test Accuracy:  0.8514628657164292
```

Figure 8.2.4.2.1: train and test accuries

### 2.FOR TRAIN DATA:

```
from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_train,lr_train_pred),annot=True,fmt='.1f')
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x221868b2448>

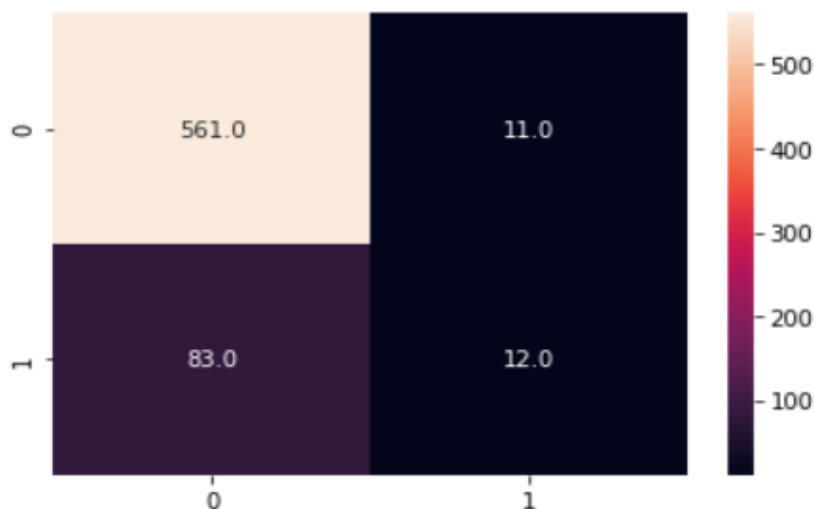


Figure 8.2.4.2.2: confusion matrix for train data

```
: print(classification_report(y_train,lr_train_pred))
```

	precision	recall	f1-score	support
0	0.87	0.98	0.92	572
1	0.52	0.13	0.20	95
accuracy			0.86	667
macro avg	0.70	0.55	0.56	667
weighted avg	0.82	0.86	0.82	667

Figure 8.2.4.2.3: classification report

### 3.FOR TEST DATA:

```
: from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_test,lr_test_pred),annot=True,fmt='.1f')
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x22186931788>
```

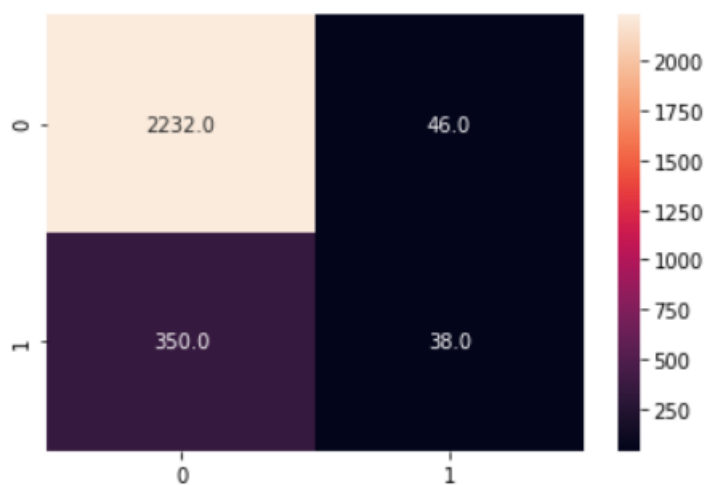


Figure 8.2.4.2.4: confusion matrix for test data

```
: print(classification_report(y_test,lr_test_pred))
```

	precision	recall	f1-score	support
0	0.86	0.98	0.92	2278
1	0.45	0.10	0.16	388
accuracy			0.85	2666
macro avg	0.66	0.54	0.54	2666
weighted avg	0.80	0.85	0.81	2666

**Figure 8.2.4.2.5: classification report**



## CHAPTER 9

### COMPARING THE PERFORMANCES OF ALL THE MODELS

Comapring the accuracies of all the three models:

```
4]: labels=['KNN','Random forest','Logistic Regression']
print("Train accuracies: ",train_accu)
print("Test Accuracies: ",test_accu)

Train accuracies: [0.85757121 1.          0.85907046]
Test Accuracies: [0.85446362 0.90172543 0.85146287]
```

Figure 9.1: Accuracies of all the 3 models

On comparing the accuracy score with respect to training dataset of all models it has been observed that Random Forest has better performance than Logistic Regression and KNN.

```
[35]: print("Accuracies Comparision for TRAIN data for all the models:\n")
plt.subplots(figsize=(7,7))
plt.bar(labels,train_accu,color=['red','green','orange'])
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Train Accuracies of various algorithms')
```

Accuracies Comparision for TRAIN data for all the models:

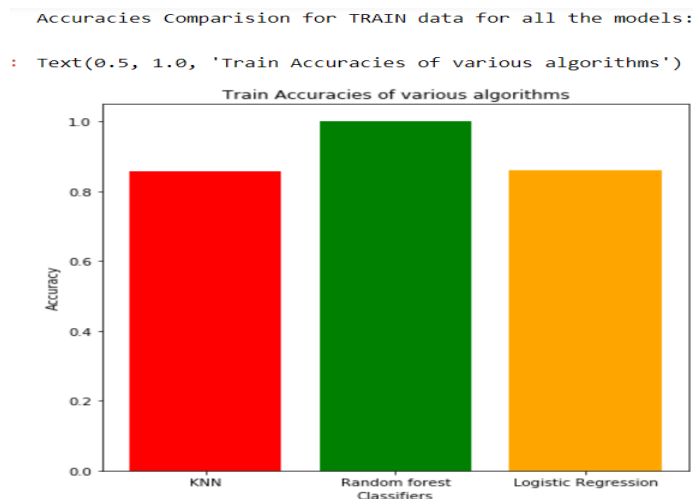


Figure 9.2: Visualizing the train accuracies

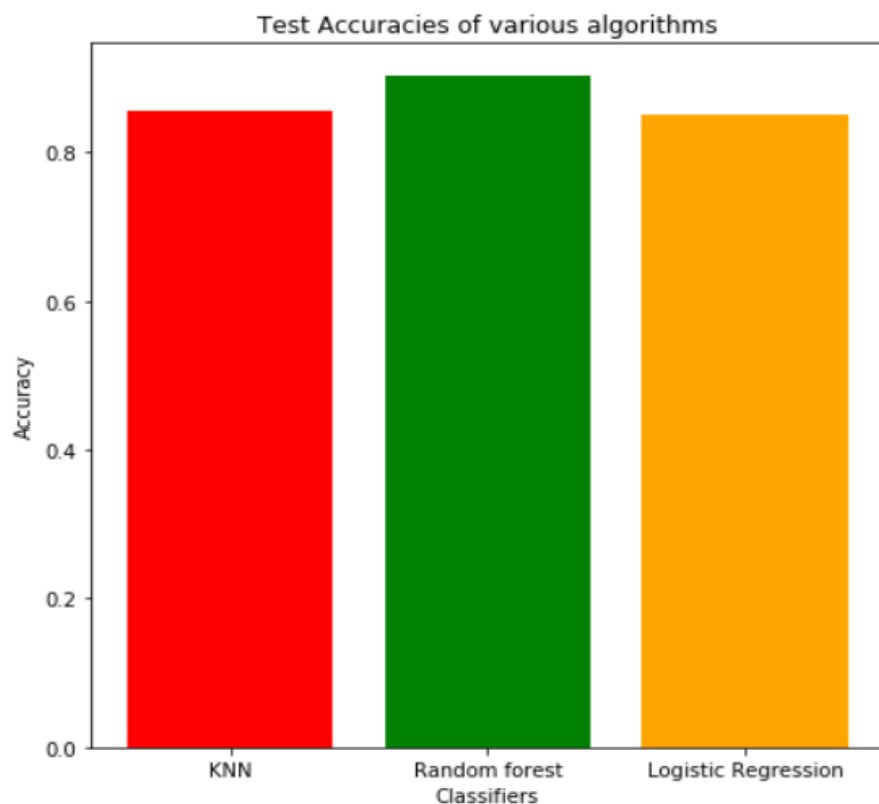
On comparing the accuracy score with respect to testing dataset of all models it has been observed that Random Forest has better performance than Logistic Regression and KNN.

```
[36]: print("Accuracies Comparision for TEST data for all the models:\n")
plt.subplots(figsize=(7,7))
plt.bar(labels,test_accu,color=['red','green','orange'])
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Test Accuracies of various algorithms')
```

Accuracies Comparision for TEST data for all the models:

Accuracies Comparision for TEST data for all the models:

```
: Text(0.5, 1.0, 'Test Accuracies of various algorithms')
```



**Figure 9.3: Visualizing the test accuracies of all 3 models**

## **CONCLUSION**

From the above results, we can infer that for our problem statement, the Random Forest Classifier has yielded us the best results from all of our experiments.

Hence, we conclude that the Random Forest Classifier works well for our classification.

## REFERENCES

1. <https://drive.google.com/drive/folders/1vZgoh1o5xT6vehsRdfAQEX4nA79yoSVe?usp=sharing>
2. <https://www.kaggle.com/mnassrib/telecom-churn-datasets>
3. [learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
4. [learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=t=random%20forest%20classifier#sklearn.ensemble.RandomForestClassifier](https://learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=t=random%20forest%20classifier#sklearn.ensemble.RandomForestClassifier)
5. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
6. <https://github.com/jaswanthgoudm/Churn-prediction-project>