MACHINE LEARNING
(CHURN PREDICTION)

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

Bachelor of Technology
In
COMPUTER SCIENCE AND ENGINEERING
By

VARKOOR SANDEEP SAGAR
221710309061

Under the Guidance of

Department of COMPUTER SCIENCE AND ENGINEERING
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

DECLARATION

I submit this industrial training work entitled “CHURN PREDICTION” 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 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. Varkoor Sandeep Sagar.
DATE: 15-07-2020. 221710309061

GITAM (DEEMED TO BE UNIVERSITY)
Hyderabad-502329, India
Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled "CHURN PREDICTION" is being submitted by V. SANDEEP SAGAR (221710309061) 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 2020-21.

It is faithful record work carried out by her 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 competition 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.

V. SANDEEP SAGAR.
221710309061

ABSTRACT

Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenues of the companies, especially in the telecom field, companies are seeking to develop means to predict potential customer to churn.

Therefore, finding factors that predict customer churn is important. The main contribution of our work is to develop a churn prediction model which assists telecom operators to predict customers who are most likely subject to churn. The model developed in this work uses machine learning techniques builds a new way of features engineering and selection.

Finally, it is observed through simulations that our proposed approach based on LogisticRegression, RandomForestClassifier and KNeighborsClassifier, performs quite well for predicting churners and therefore can be beneficial for highly competitive telecommunication industry.

Table of Contents:

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).

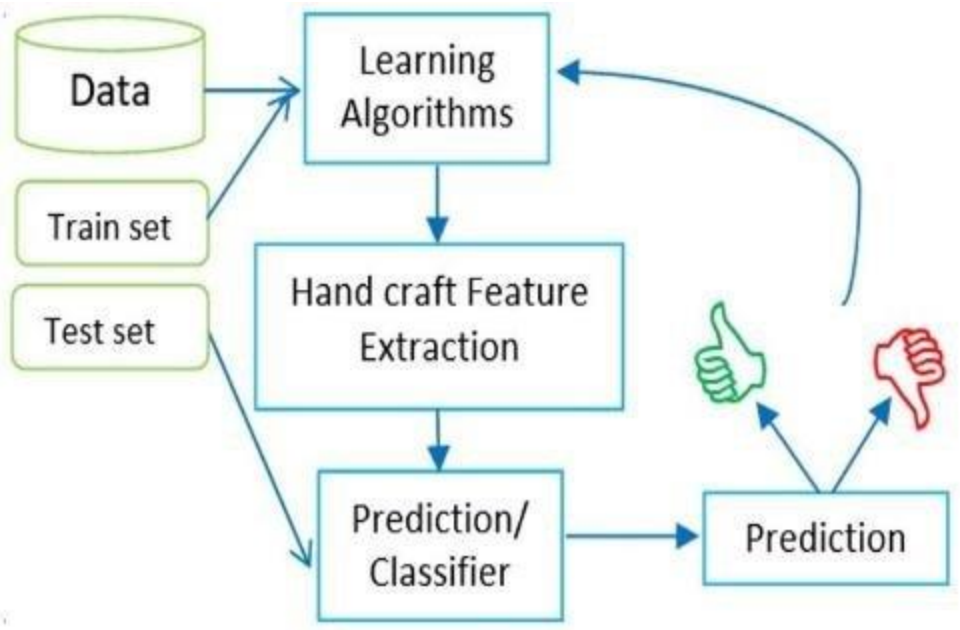
1.2 IMPORTANCE OF MACHINE LEARNING:

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



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.

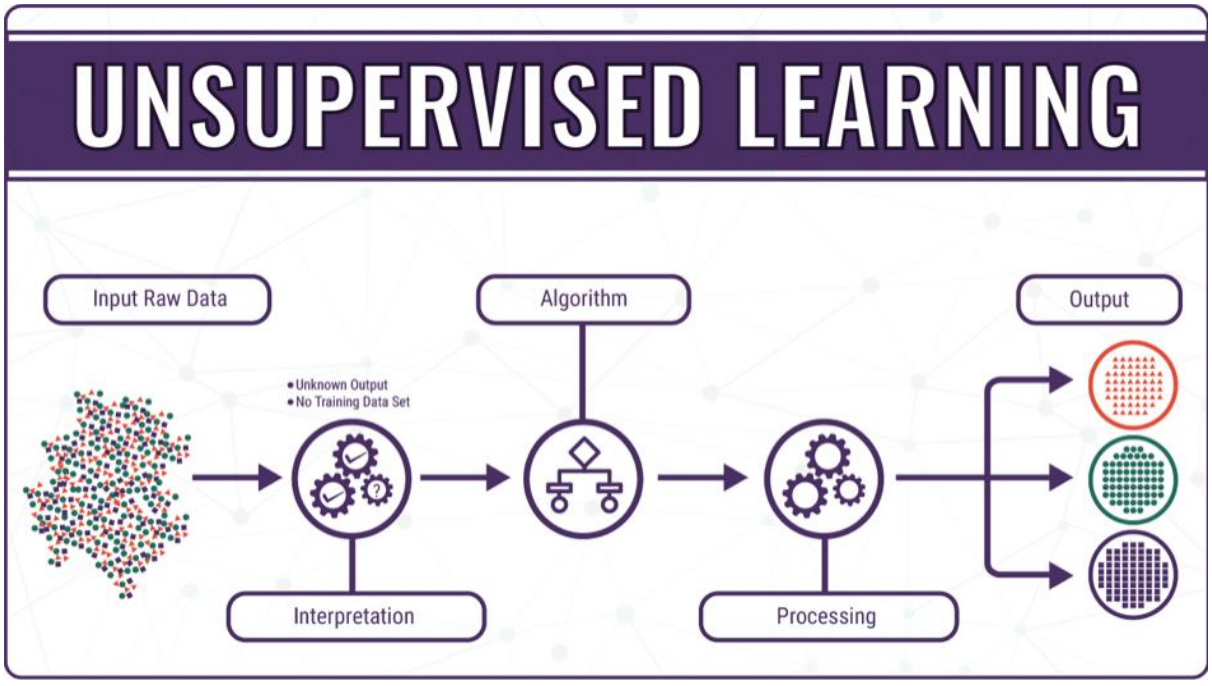
Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem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

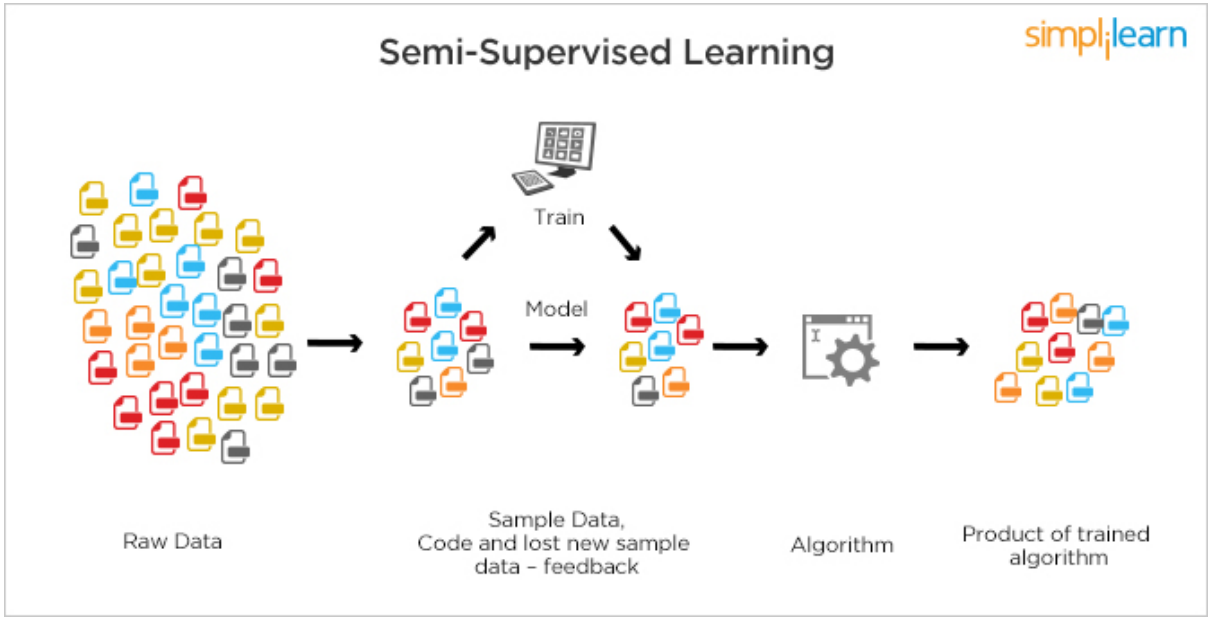
When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values.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



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.



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 word. 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 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 SETUP OF 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.

3.2.1 Installation(using python IDLE):

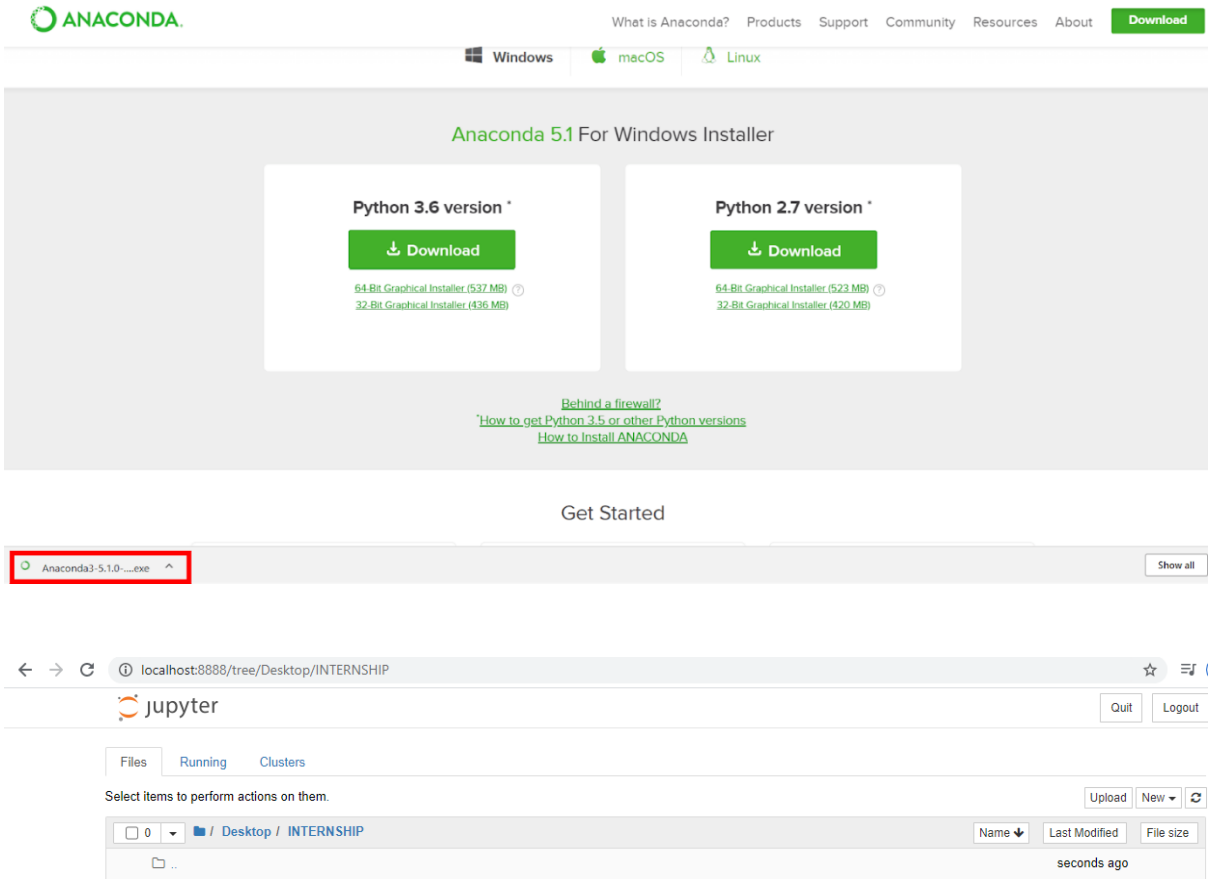
- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

3.2.2 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:
● 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 python 3.4)

  next click install and next click finish.
- ● Step 5: Open jupyter notebook (it opens in default browser)



3.3 FEATURES OF PYTHON:

● Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
● Easy-to-read: Python code is more clearly defined and visible to the eyes
● Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh
● Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
● Databases: Python provides interfaces to all major commercial databases.
● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 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

3.4.1 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).

3.4.2 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.

3.4.3 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.

3.4.4 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.

3.4.5 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.

3.5 PYTHON FUNCTION:

3.5.1 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.

3.5.2 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.

3.6 PYTHON USING OOPs CONCEPTS:

3.6.1 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
```

3.6.2 __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

4.1 Project Requirements

4.1.1 Packages used:

1. Pandas
2. Numpy
3. Matplotlib.pyplot
4. Seaborn
5. sklearn.preprocessing import LabelEncoder
6. sklearn.preprocessing import StandardScaler
7. sklearn.linear_model import LogisticRegression
8. sklearn.metrics import classification_report,confusion_matrix
9. sklearn.neighbors import KNeighborsClassifier
10. sklearn.ensemble import RandomForestClassifier

4.1.2 Algorithms used:

1.K-NEAREST NEIGHBORS

2.LOGISTIC REGRESSION ALGORITHM

3.RANDOM FOREST CLASSIFIER ALGORITHM

4.2 PROBLEM STATEMENT:

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

4.3 DATASET DESCRIPTION:

1. State
2. Account length
3. Area code
4. International plan
5. Voice mail pan
6. Number vmail messages
7. Total day minutes
8. Total day calls
9. Total day charge
10. Total eve minutes
11. Total eve calls
12. Total eve charge
13. Total night minutes
14. Total night calls
15. Total night charge Total day minutes
16. Total intl calls
17. Total intl charge
18. Customer service calls
19. Churn

4.4 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
MODEL BUILDING

5.1 PREPROCESSING OF THE DATA:

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

5.2 GETTING THE DATASET:

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

5.3 IMPORTING THE LIBRARIES:

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

Importing the libraries:

```
In [1]: #import req libs:
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

5.4 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 access using the dataframe. Any missing value or NaN value have to be cleaned.

Acquring 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)
```

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

Out[3]:

	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
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122

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

Out[4]:

	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
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	80
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	83
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	97
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	102
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	109

5.5 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

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

```
train.head() #displaying head after drop
```

	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	To nig minut
0	128	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244
1	107	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254
2	137	No	No	0	243.4	114	41.38	121.2	110	10.30	162
3	84	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196
4	75	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186

Checking for Null values:

```
train.isnull().sum() #no 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
```

```
In [9]: test.isnull().sum() #no null value in test
```

```
Out[9]: 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
```

There are no missing values in the given data set

5.6 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 dataframe or we can add that dummy set to the dataframe.

International plan	Voice mail plan
No	Yes
No	Yes
No	No
Yes	No
Yes	No

CHAPTER 6
DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

6.1 Statistical Analysis:

Describing the data:

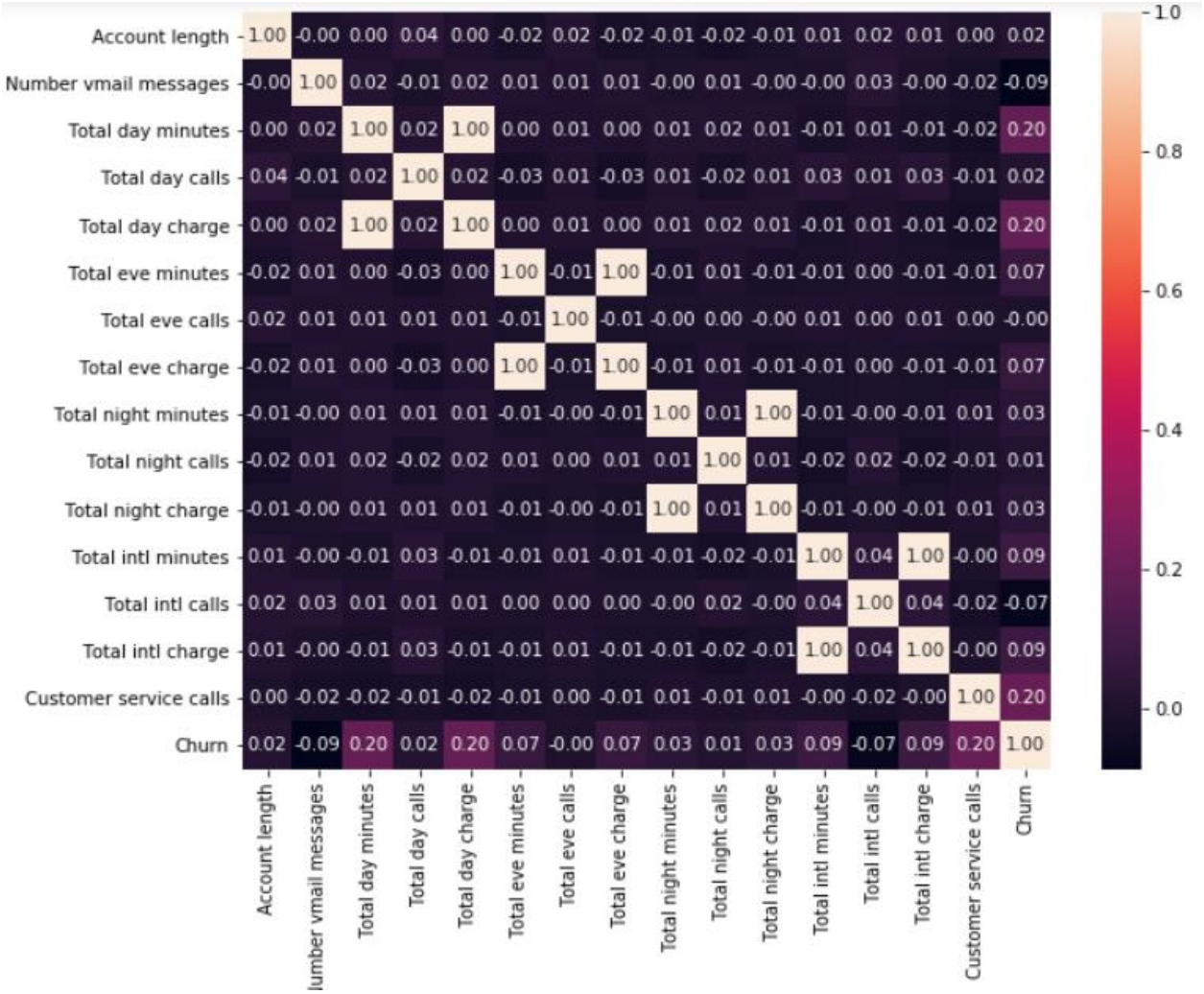
```
train.describe()
```

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes
count	2666.000000	2666.000000	2666.000000	2666.00000	2666.000000	2666.000000	2666.000000
mean	100.620405	437.438860	8.021755	179.48162	100.310203	30.512404	200.386159
std	39.563974	42.521018	13.612277	54.21035	19.988162	9.215733	50.951515
min	1.000000	408.000000	0.000000	0.00000	0.000000	0.000000	0.000000
25%	73.000000	408.000000	0.000000	143.40000	87.000000	24.380000	165.300000
50%	100.000000	415.000000	0.000000	179.95000	101.000000	30.590000	200.900000
75%	127.000000	510.000000	19.000000	215.90000	114.000000	36.700000	235.100000
max	243.000000	510.000000	50.000000	350.80000	160.000000	59.640000	363.700000

6.2 Generating Plots:

Visualizations:

```
In [16]: plt.subplots(figsize=(10,8))
         sns.heatmap(train.corr(),annot=True,fmt=".2f")
```



6.3 Data Type Conversions:

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.

```
In [24]: train.dtypes

Out[24]: 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
```

6.4 Detection of Outliers:

We don't have any outliers in the given data.

6.5 Handling Missing Values:

Checking for Null values:

```
train.isnull().sum() #no 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
```

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

6.6Encoding 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 data using label encoding:

In [13]:
```
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()
```

Out[13]:

	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	128	0	1	110	45.07	99	16.78	91	11.01	3	2.70	1	0
1	107	0	1	123	27.47	103	16.62	103	11.45	3	3.70	1	0
2	137	0	0	114	41.38	110	10.30	104	7.32	5	3.29	0	0
3	84	1	0	71	50.90	88	5.26	89	8.86	7	1.78	2	0
4	75	1	0	113	28.34	122	12.61	121	8.41	3	2.73	3	0

CHAPTER 7
FEATURE SELECTION

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)
```

7.3 Train-Test-Split:

Train-Test-Split:

In [15]:
```
X_train = train.drop(["Churn"],axis=1)
y_train = train.Churn
X_test = test.drop(["Churn"],axis=1)
y_test = test.Churn
```

7.4Feature Scaling:

Scaling the data using Standard Scalar:

In [16]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train=pd.DataFrame(data=ss.fit_transform(X_train),columns=X_train.columns)
X_test=pd.DataFrame(data=ss.fit_transform(X_test),columns=X_test.columns)
```

CHAPTER 8
MODEL BUILDING AND EVALUATION

8.1 LogisticRegression:

8.1.1 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).

8.1.2 Train the Models:

I have used "LogisticRegression()" and created an object for my model.

I have fitted the training dataset to the model by using "fit()".

```
In [18]: 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
```

8.1.3 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
```

8.1.4 Predictions from raw data:

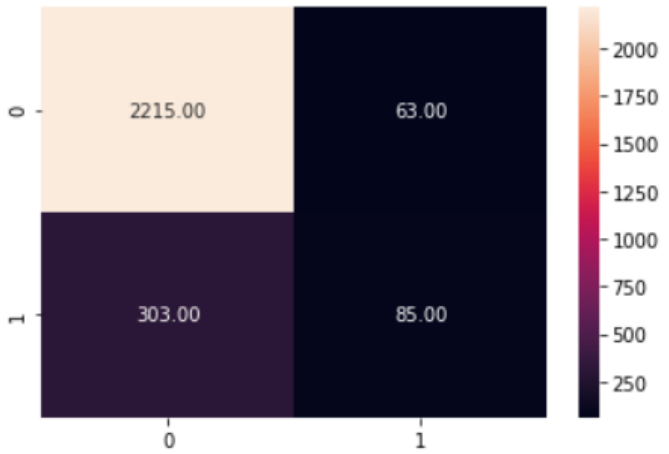
My model has achieved an accuracy score of 86% for training data and 85% for testing data as input.

```
#Checking the metrics
from sklearn.metrics import accuracy_score
train_accu[0]=accuracy_score(y_train,lr_train_pred)
test_accu[0]=accuracy_score(y_test,lr_test_pred)
print("LogisticRegression Train Accuracy: ",train_accu[0])
print("LogisticRegression Test Accuracy: ",test_accu[0])
```

```
LogisticRegression Train Accuracy:  0.8627156789197299
LogisticRegression Test Accuracy:  0.8515742128935532
```

```
In [19]: from sklearn.metrics import classification_report,confusion_matrix
         sns.heatmap(confusion_matrix(y_train,lr_train_pred),annot=True,fmt='.2f')
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f45dbed4510>
```




```
In [20]: print(classification_report(y_train,lr_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2278
           1       0.57      0.22      0.32       388

    accuracy                           0.86      2666
   macro avg       0.73      0.60      0.62      2666
weighted avg       0.84      0.86      0.84      2666
```

8.2 K-Nearest Neighbors Classifier:

8.2.1 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.

8.2.2 Train the Models

I have used “KNeighborsClassifier()” and created an object for my model.

I have fitted the training dataset to the model by using “fit()”.

```
In [22]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=20, metric='euclidean')
         knn.fit(X_train,y_train) #training the model
```

8.2.3 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
```

8.2.4 Predictions from raw data:

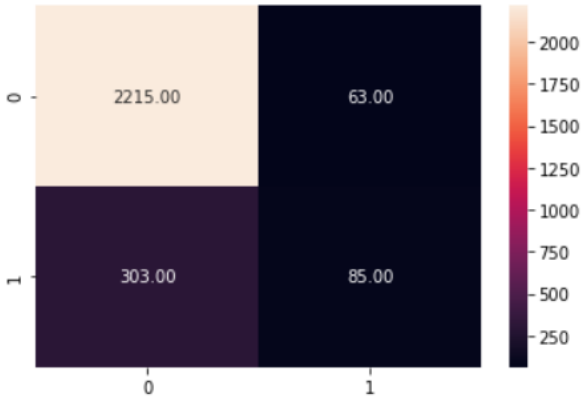
My model has achieved an accuracy score of 87% for training data and 87% for testing data as input.

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

```
LogisticRegression Train Accuracy:  0.8769692423105776
LogisticRegression Test Accuracy:  0.8785607196401799
```

```
In [23]: from sklearn.metrics import classification_report,confusion_matrix
         sns.heatmap(confusion_matrix(y_train,lr_train_pred),annot=True,fmt='.2f')
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1dce0ec7908>
```




```
In [24]: print(classification_report(y_train,lr_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2278
           1       0.57      0.22      0.32       388

    accuracy                           0.86      2666
   macro avg       0.73      0.60      0.62      2666
weighted avg       0.84      0.86      0.84      2666
```

8.3 Random Forest Classifier:

8.3.1 Brief about the algorithms used

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

8.3.2 Train the Models:

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

```
[n [25]: from sklearn.ensemble import RandomForestClassifier
         clf = RandomForestClassifier()
         clf.fit(X_train, y_train)
```

8.3.3 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.

```
clf_train_pred=clf.predict(X_train)
clf_test_pred=clf.predict(X_test)
```

8.3.4 Predictions from raw data:

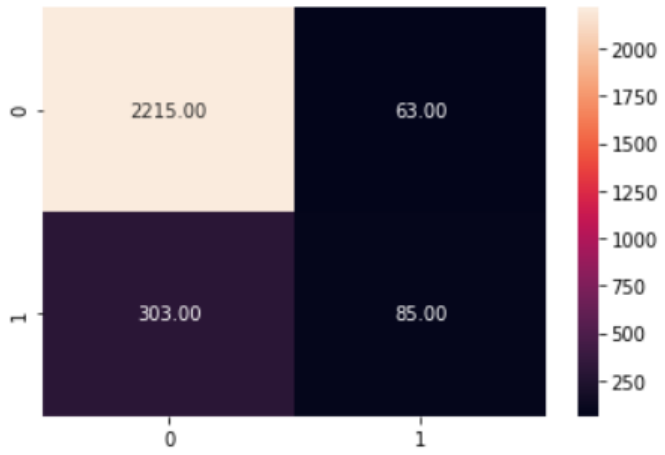
My model has achieved an accuracy score of 99 % for training data and 94% for testing data as input.

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

```
LogisticRegression Train Accuracy:  0.9996249062265566
LogisticRegression Test Accuracy:  0.9445277361319341
```

In [26]:
```
from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(y_train,lr_train_pred),annot=True,fmt='.2f')
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1dce107e908>



In [27]:
```
print(classification_report(y_train,lr_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2278
           1       0.57      0.22      0.32       388

    accuracy                           0.86      2666
   macro avg       0.73      0.60      0.62      2666
weighted avg       0.84      0.86      0.84      2666
```

CHAPTER 9

Comparing the performances of all the models:

Comparing the Accuracies of all the three models:

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

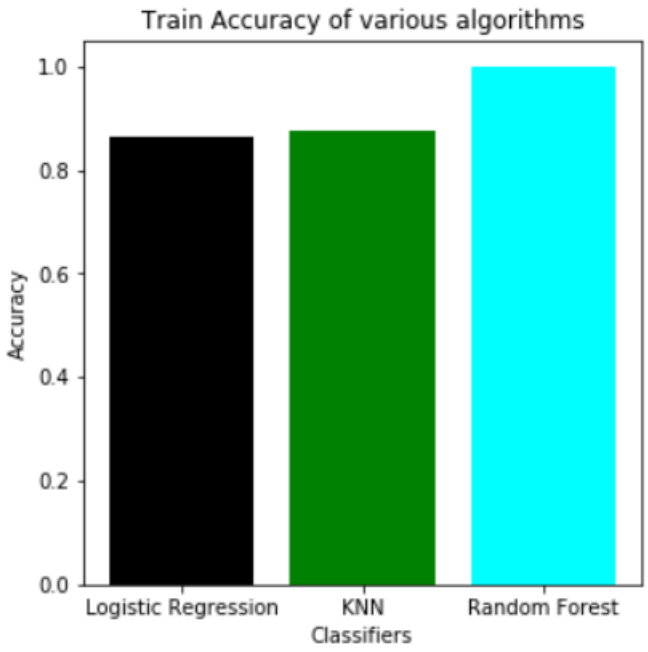
```
Train accuracies:  [0.86271568 0.87696924 1.        ]
Test Accuracies:  [0.85157421 0.87856072 0.94452774]
```

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.

In [31]:
```
print("Accuracy Comparision for TRAIN data for all the models:\n\n")
plt.subplots(figsize=(5,5))
plt.bar(labels,train_accu,color=['black','green','cyan'])
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Train Accuracy of various algorithms')
```

```
Accuracy Comparision for TRAIN data for all the models:
```

Out[31]:
```
Text(0.5, 1.0, 'Train Accuracy of various algorithms')
```

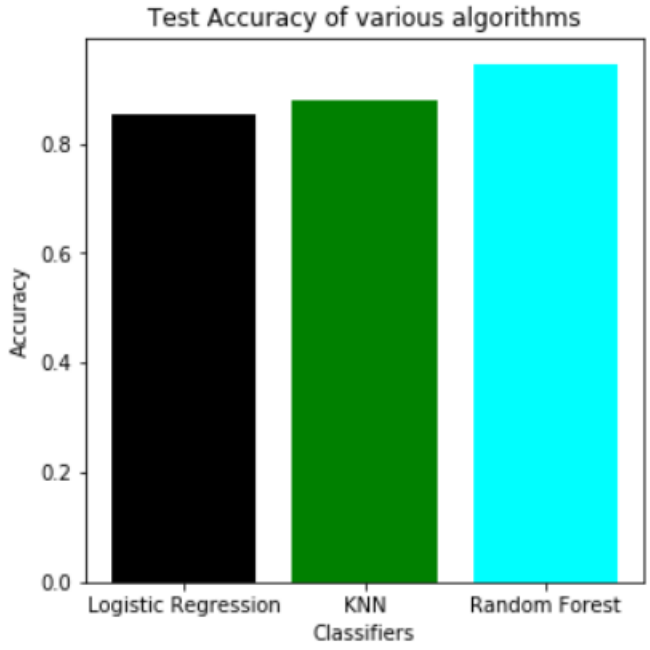


On comparing the accuracy score with respect to testing dataset of all models it has been observed that Random Forest has better performance than Logistic Regression and KNN.

In [33]:
```
print("Accuracy Comparision for TEST data for all the models:\n\n")
plt.subplots(figsize=(5,5))
plt.bar(labels,test_accu,color=['black','green','cyan'])
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Test Accuracy of various algorithms')
```

```
Accuracy Comparision for TEST data for all the models:
```

Out[33]:
```
Text(0.5, 1.0, 'Test Accuracy of various algorithms')
```



CHAPTER 10
CONCLUSION:

Here, as we can see out of all the three algorithms the maximum accuracy is obtained using RANDOM FOREST Classifier. Hence, we can conlude by our experiments that Random Forest approach towards Churn Prediction problem is best algorithm.

CHAPTER 11
REFERENCES:

https://www.kaggle.com/mnassrib/telecom-churn-datasets

learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest%20classifier#sklearn.ensemble.RandomForestClassifier

https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning