DATA SCIENCE

(Predicting News item as Fake or Real)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Banda Akhila Shankar

221710313005

Under the Guidance of

Mr. _______

Assistant Professor

GITAM
(DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of the UGC Act, 1956)
VISAKHAPATNAM ✸ HYDERABAD ✸ BENGALURU

Department Of Electronics and Communication Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

DECLARATION

I submit this industrial training work entitled “**PREDICTING NEWS ITEM AS REAL OR FAKE**” 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 **Mr. ______**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD	Banda Akhila Shankar
Date:	221710313005

CERTIFICATE

This is to certify that the Industrial Training Report entitled “PREDICTING NEWS ITEM AS REAL OR FAKE” is being submitted by Banda Akhila Shankar (221710313005) in partial fulfillment of the requirement for the award of Bachelor of Technology in Computer Science and 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 and Engineering Department, GITAM University Hyderabad Campus under my guidance and supervision.

Mr.______________

Dr. S.Phani Kumar,
Professor and HOD,
CSE Dept.

certificate

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 Dr. CH. Sanjay, Principal, GITAM Hyderabad.

I would like to thank respected Prof. 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 the internship report. It helped me alot to realize what we study for.

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

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

Banda Akhila Shankar
221710313005

ABSTRACT

Now-a-days Social media and news outlets publish fake news to increase readership or as part of psychological warfare. Ingeneral, the goal is profiting through clickbaits. Clickbaits lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. The purpose of the work is to come up with a solution that can be utilized by users to detect and filter out sites containing false and misleading information.

Machine Learning technique is used to deal with this problem. The result of the prediction is whether the news item is fake or real.

Machine learning algorithms are used to predict the values from the dataset by splitting the dataset into train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on news dataset.

Table of Contents:

CHAPTER 1: INFORMATION ABOUT DATA SCIENCE

CHAPTER 2.INFORMATION ABOUT MACHINE LEARNING

CHAPTER 3. INFORMATION ABOUT PYTHON

CHAPTER 4. PROJECT: Prediction Of News as Real or Fake

CHAPTER 1

INFORMATION ABOUT DATA SCIENCE

1.1.1 WHAT IS DATA SCIENCE?

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. How is this different from what statisticians have been doing for years?

The answer lies in the difference between explaining and predicting.

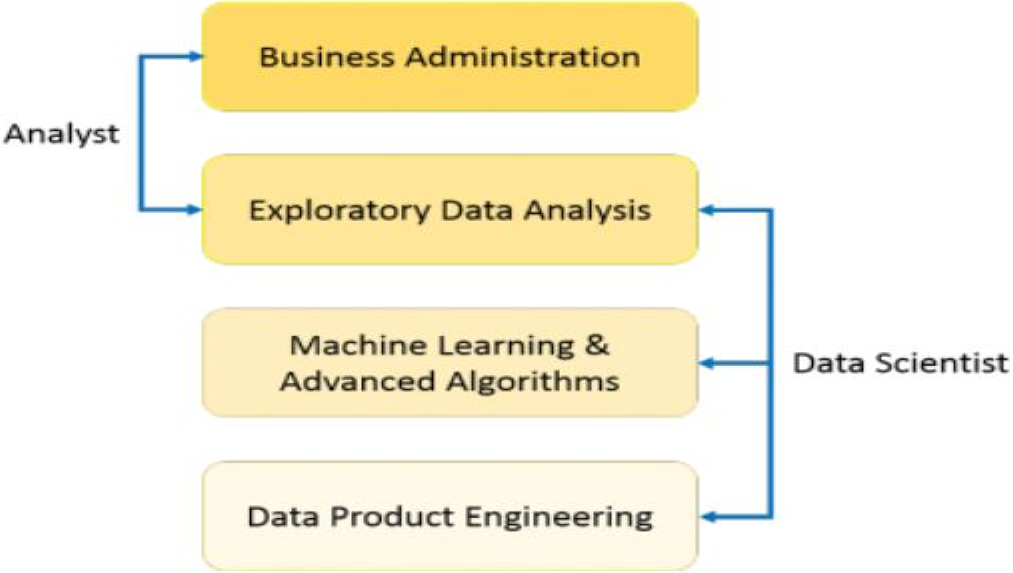


Fig.1.1 difference between explaining and predicting

As you can see from the above image, a Data Analyst usually explains what is going on by processing history of the data. On the other hand, Data Scientist not only does the exploratory analysis to discover insights from it, but also uses various advanced machine learning algorithms to identify the occurrence of a particular event in the future. A Data Scientist will look at the data from many angles, sometimes angles not known earlier.

1.1.2 NEED OF DATA SCIENCE

Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science) and machine learning.

- Predictive causal analytics – If you want a model which can predict the possibilities of a particular event in the future, you need to apply predictive causal analytics. Say, if you are providing money on credit, then the probability of customers making future credit payments on time is a matter of concern for you. Here, you can build a model which can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not.

- Prescriptive analytics: If you want a model which has the intelligence of taking its own decisions and the ability to modify it with dynamic parameters, you certainly need prescriptive analytics for it. This relatively new field is all about providing advice. In other terms, it not only predicts but suggests a range of prescribed actions and associated outcomes.
  The best example for this is Google's self-driving car which I had discussed earlier too. The data gathered by vehicles can be used to train self-driving cars. You can run

algorithms on this data to bring intelligence to it. This will enable your car to take decisions like when to turn, which path to take, when to slow down or speed up.

- Machine learning for making predictions — If you have transactional data of a finance company and need to build a model to determine the future trend, then machine learning algorithms are the best bet. This falls under the paradigm of supervised learning. It is called supervised because you already have the data based on which you can train your machines. For example, a fraud detection model can be trained using a historical record of fraudulent purchases.

- Machine learning for pattern discovery — If you don't have the parameters based on which you can make predictions, then you need to find out the hidden patterns within the dataset to be able to make meaningful predictions. This is nothing but the unsupervised model as you don't have any predefined labels for grouping. The most common algorithm used for pattern discovery is Clustering.
  Let's say you are working in a telephone company and you need to establish a network by putting towers in a region. Then, you can use the clustering technique to find those tower locations which will ensure that all the users receive optimum signal strength.

Let's see how the proportion of above-described approaches differ for Data Analysis as well as Data Science. As you can see in the image below, Data Analysis includes descriptive analytics and prediction to a certain extent. On the other hand, Data Science is more about Predictive Causal Analytics and Machine Learning.

1.1.3 USES OF DATA SCIENCE

1. Medical Image Analysis

Procedures such as detecting tumors, artery stenosis, organ delineation employ various different methods and frameworks like MapReduce to find optimal parameters for tasks like lung texture classification. It applies machine learning methods, support vector machines (SVM), content-based medical image indexing, and wavelet analysis for solid texture classification.

2. Genetics & Genomics

Data Science applications also enable an advanced level of treatment personalization through research in genetics and genomics. The goal is to understand the impact of the DNA on our health and find individual biological connections between genetics, diseases, and drug response. Data science techniques allow integration of different kinds of data with genomic data in the disease research, which provides a deeper understanding of genetic issues in reactions to particular drugs and diseases. As soon as we acquire reliable personal genome data, we will achieve a deeper understanding of the human DNA. The advanced genetic risk prediction will be a major step towards more individual care.

3. Drug Development

The drug discovery process is highly complicated and involves many disciplines. The greatest ideas are often bounded by billions of testing, huge financial and time expenditure. On average, it takes twelve years to make an official submission.

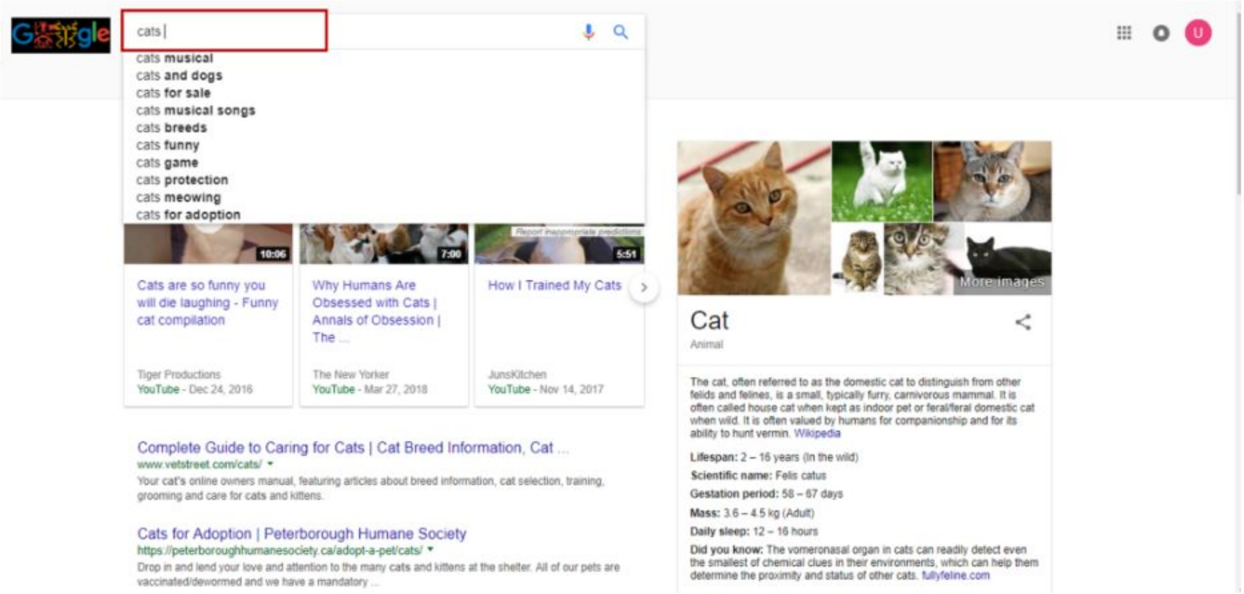
Data science applications and machine learning algorithms simplify and shorten this process, adding a perspective to each step from the initial screening of drug compounds to the prediction of the success rate based on the biological factors. Such algorithms can forecast how the compound will act in the body using advanced mathematical modeling and simulations instead of the “lab experiments”. The idea behind the computational drug discovery is to create computer model simulations as a biologically relevant network simplifying the prediction of future outcomes with high accuracy.

4. Virtual assistance for patients and customer support

Optimization of the clinical process builds upon the concept that for many cases it is not actually necessary for patients to visit doctors in person. A mobile application can give a more effective solution by *bringing the doctor to the patient instead*.The AI-powered mobile apps can provide basic healthcare support, usually as chatbots. You simply describe your symptoms, or ask questions, and then receive key information about your medical condition derived from a wide network linking symptoms to causes. Apps can remind you to take your medicine on time, and if necessary, assign an appointment with a doctor.This approach promotes a healthy lifestyle by encouraging patients to make healthy decisions, saves their time waiting in line for an appointment, and allows doctors to focus on more critical cases.

5.Internet Search

Now, this is probably the first thing that strikes your mind when you think Data Science Applications.When we speak of search, we think ‘Google’. Right? But there are many other search engines like Yahoo, Bing, Ask, AOL, and so on. All these search engines (including Google) make use of data science algorithms to deliver the best result for our searched query in a fraction of seconds. Considering the fact that, Google processes more than 20 petabytes of data every day.



Cats are so funny you will die laughing - Funny cat compilation
Tiger Productions
YouTube - Dec 24, 2016

Why Humans Are Obsessed with Cats | Annals of Obsession | The ...
The New Yorker
YouTube - Mar 27, 2018

How I Trained My Cats
JunsKitchen
YouTube - Nov 14, 2017

Complete Guide to Caring for Cats | Cat Breed Information, Cat ...
www.vetstreet.com/cats/
Your cat's online owners manual, featuring articles about breed information, cat selection, training, grooming and care for cats and kittens.

Cats for Adoption | Peterborough Humane Society
https://peterboroughhumanesociety.ca/adopt-a-pet/cats/
Drop in and lend your love and attention to the many cats and kittens at the shelter. All of our pets are vaccinated/dewormed and we have a mandatory ...

Cat
Animal

The cat, often referred to as the domestic cat to distinguish from other felids and felines, is a small, typically furry, carnivorous mammal. It is often called house cat when kept as indoor pet or feral/feral domestic cat when wild. It is often valued by humans for companionship and for its ability to hunt vermin. Wikipedia

Lifespan: 2 – 16 years (In the wild)
Scientific name: Felis catus
Gestation period: 58 – 67 days
Mass: 3.6 – 4.5 kg (Adult)
Daily sleep: 12 – 16 hours
Did you know: The vomeronasal organ in cats can readily detect even the smallest of chemical clues in their environments, which can help them determine the proximity and status of other cats. fullyfeline.com

Fig.1.2 internet search

6.Targeted Advertising

If you thought Search would have been the biggest of all data science applications, here is a challenger – the entire digital marketing spectrum. Starting from the display banners on various websites to the digital billboards at the airports – almost all of them are decided by using data science algorithms.

This is the reason why digital ads have been able to get a lot higher CTR (Call-Through Rate) than traditional advertisements. They can be targeted based on a user’s past behavior.

This is the reason why you might see ads of Data Science Training Programs while I see an ad of apparels in the same place at the same time.

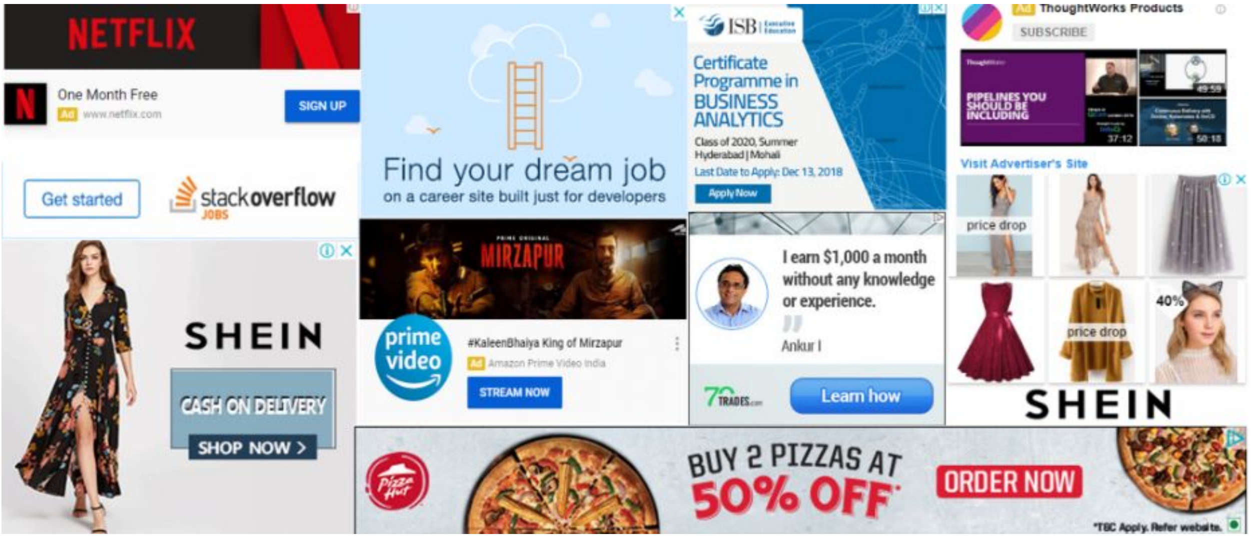


Fig 1.3 targeted advertising

CHAPTER 2

INFORMATION ABOUT MACHINE LEARNING

2.1.1 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).

2.1.2 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

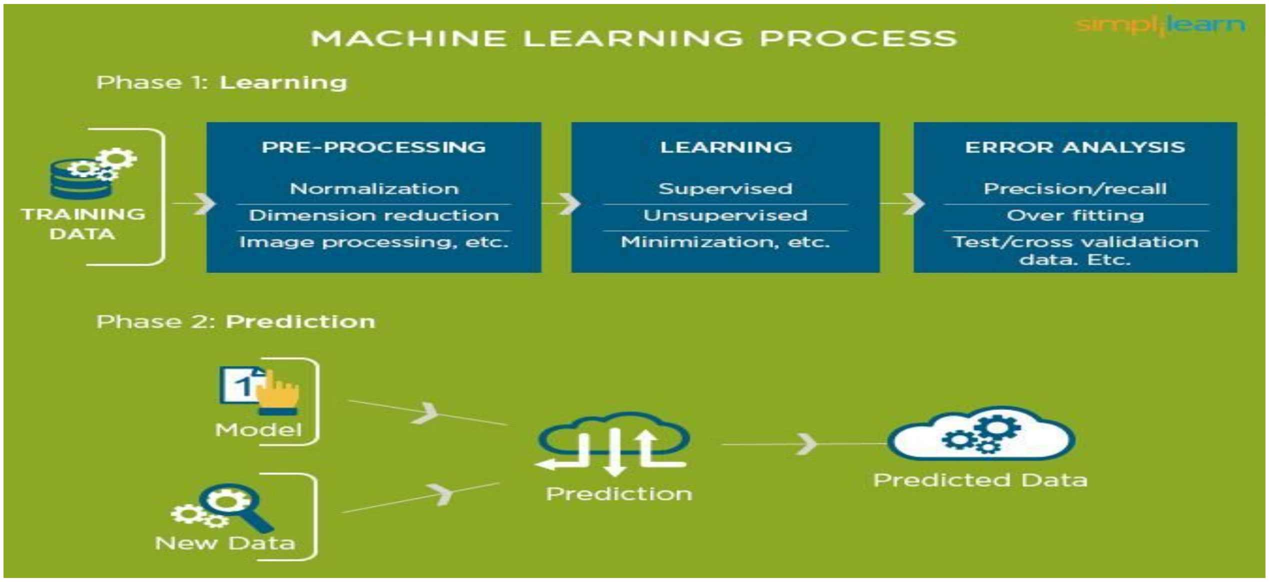


Fig 2.1 : The Process Flow

2.1.3 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.

2.1.4 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.

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.

Unsupervised Learning:

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

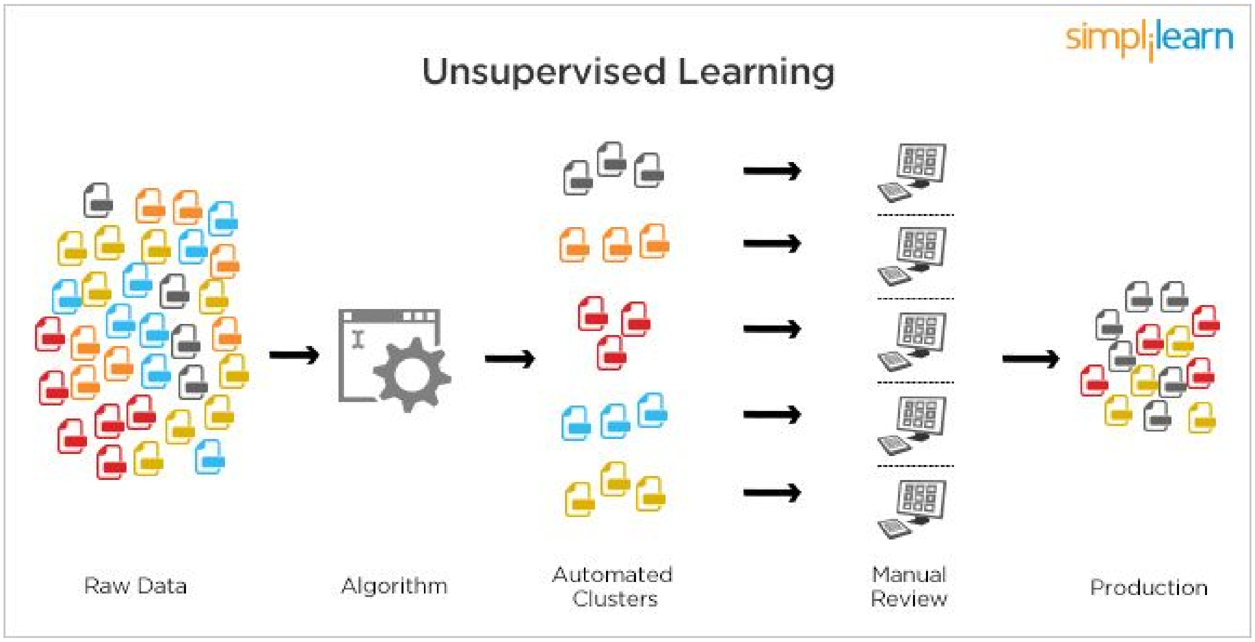


Fig 2.2 : Unsupervised Learning

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

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.

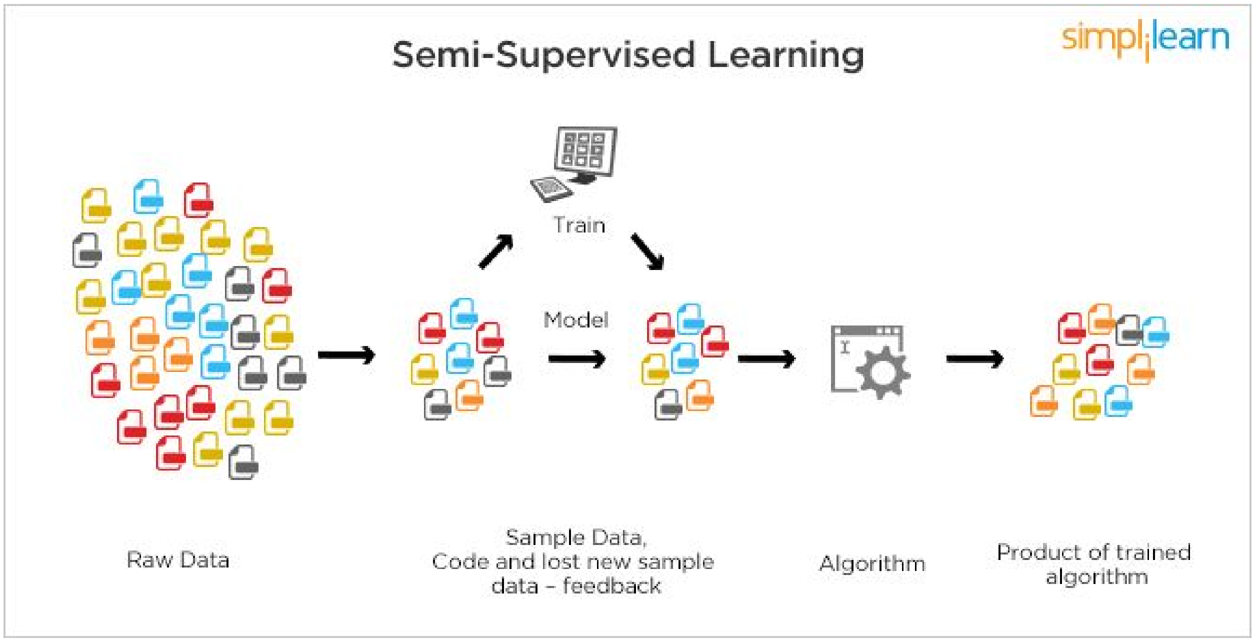


Fig 2.3 : Semi Supervised Learning

2.1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

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

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

CHAPTER 3
INFORMATION ABOUT PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 INTRODUCTION TO PYHTON:

● 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.

HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990’s

- Its latest version is 3.7 , it is generally called as python

3.2 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.

3.3 HOW TO SETUP PYTHON:

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

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

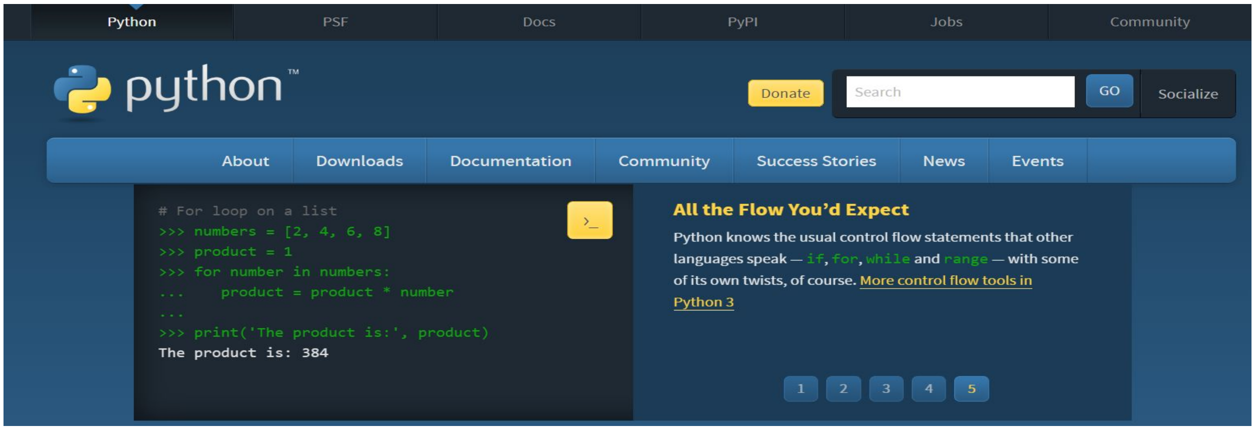
Installation(using python IDLE):

● 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.



```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

All the Flow You'd Expect

Python knows the usual control flow statements that other languages speak — if, for, while and range — with some of its own twists, of course. More control flow tools in Python 3

Fig 3.1 : Python download

Installation(using Anaconda):

- Python programs are also executed using Anaconda.

- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.

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

● In WINDOWS:

- In windows
  - Step 1: Open Anaconda.com/downloads in web browser.
  - Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
  - Step 3: select installation type(all users)
  - Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
  - Step 5: Open jupyter notebook (it opens in default browser).

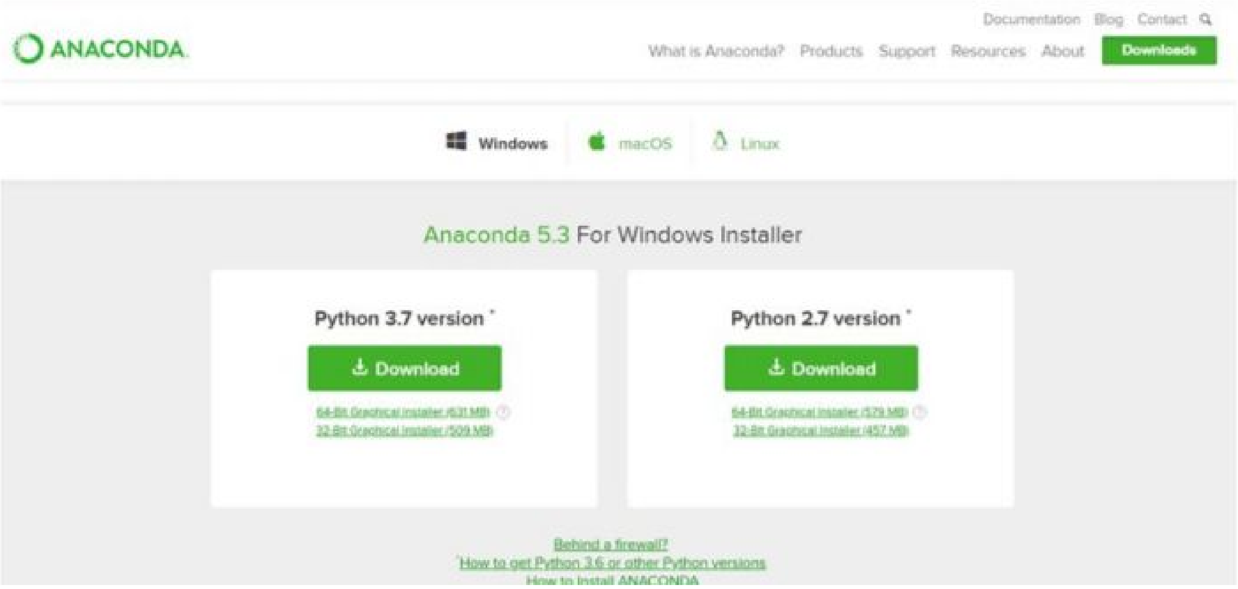
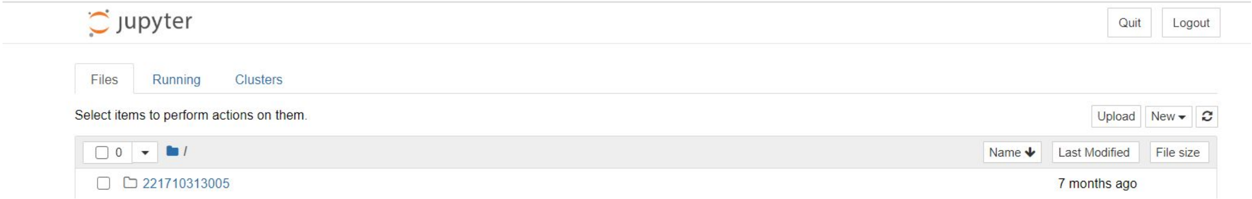

Documentation Blog Contact
ANACONDA.
What is Anaconda? Products Support Resources About
Downloads
Windows
macOS
Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
*How to get Python 3.6 or other Python versions
How to Install ANACONDA

Fig 3.2 : Anaconda download



Select items to perform actions on them.

221710313005 7 months ago

Fig 3.3 : Jupyter notebook

3.4 PYTHON VARIABLE TYPES:

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

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

o Numbers

o Strings

o Lists

o Tuples

o Dictionary

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).

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.

Python Lists:

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

. ● A list contains items separated by commas and enclosed within square brackets.

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

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

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

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.

Python Dictionary:

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

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

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

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

3.5 PYTHON FUNCTION:

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.

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 OOP’s CONCEPTS:

Class:

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

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

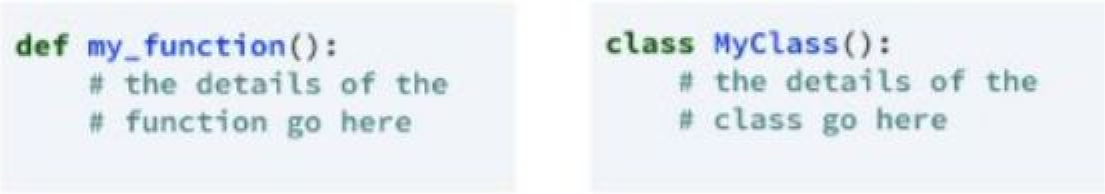
● Data member: A class variable or instance variable that holds data associated with a class and its objects.

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

- Defining a Class:

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

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



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

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

Fig 3.4 : Defining a Class

__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

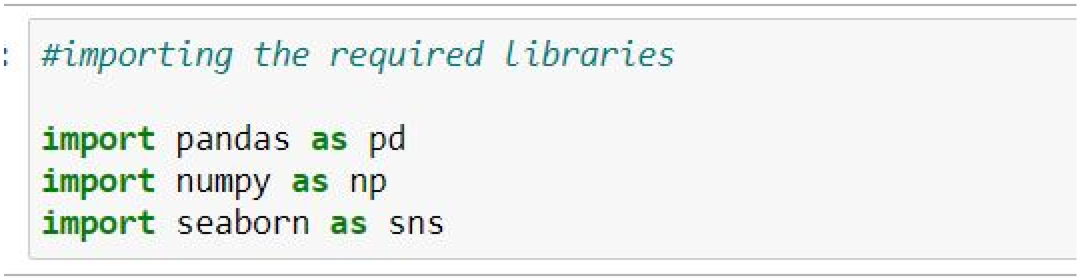
PROJECT NAME(INFORMATION ABOUT THE PROJECT)

4.1 PROJECT REQUIREMENTS

4.1.1 Packages Used

The packages used are:

- pandas
- numpy
- seaborn

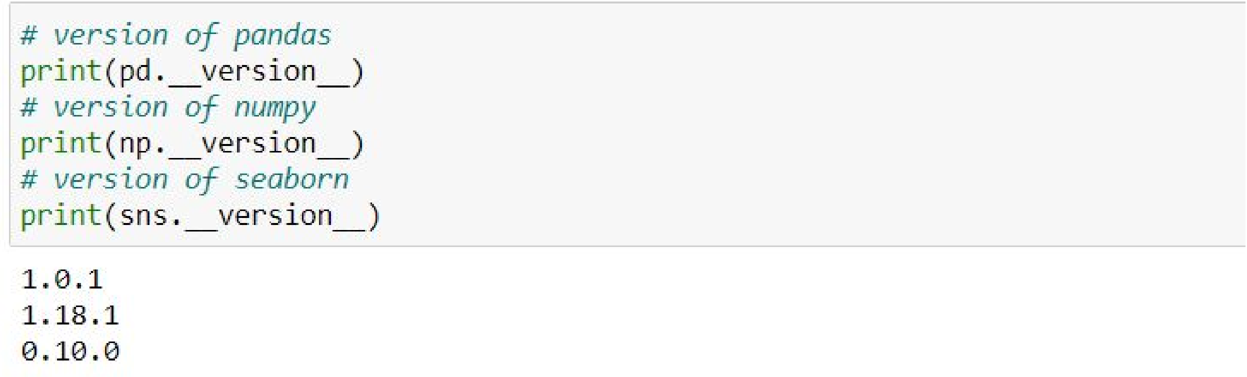


```
#importing the required libraries

import pandas as pd
import numpy as np
import seaborn as sns
```

Fig 4.1.1 importing packages

4.1.2 Versions of the Packages

- pandas version 1.0.1
- numpy version 1.18.1
- seaborn version 0.10.0



```
# version of pandas
print(pd.__version__)
# version of numpy
print(np.__version__)
# version of seaborn
print(sns.__version__)
```

```
1.0.1
1.18.1
0.10.0
```

Fig 4.1.2 versions

4.1.3 Algorithms Used

- Naive Bayes Classifier (CountVectorizer and Tfidf)
- Logistic Regression

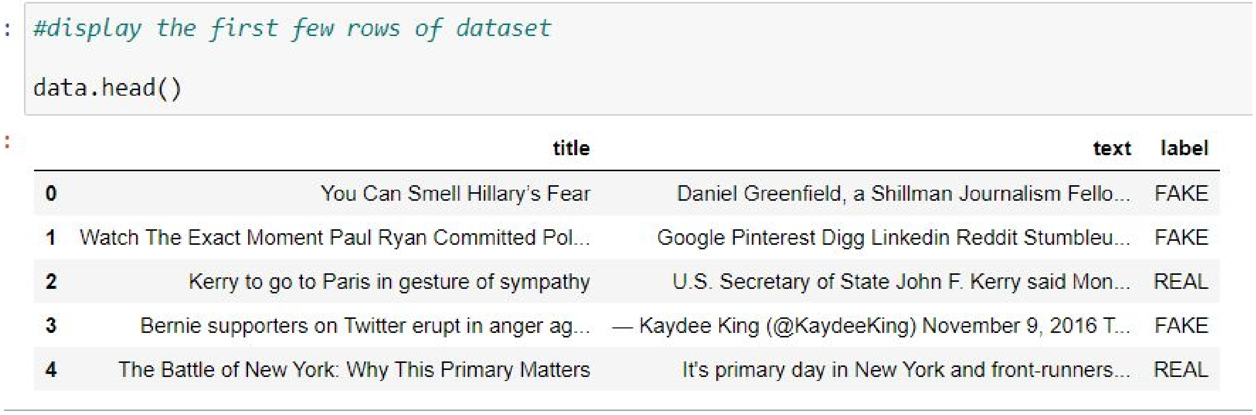
4.2 PROBLEM STATEMENT

The problem we are going to solve here is the classification of a news element as either real or fake, taking into consideration the features of the dataset NewsData.

4.3 DATASET DESCRIPTION

The dataset consists of the following features:

- title: which gives the brief description about the news element.
- text: lines of text of a news element
- label: this feature indicates whether the news element is FAKE or REAL



```
#display the first few rows of dataset

data.head()
```

	title	text	label
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Fig 4.3 dataset description

4.4 OBJECTIVE OF CASE STUDY

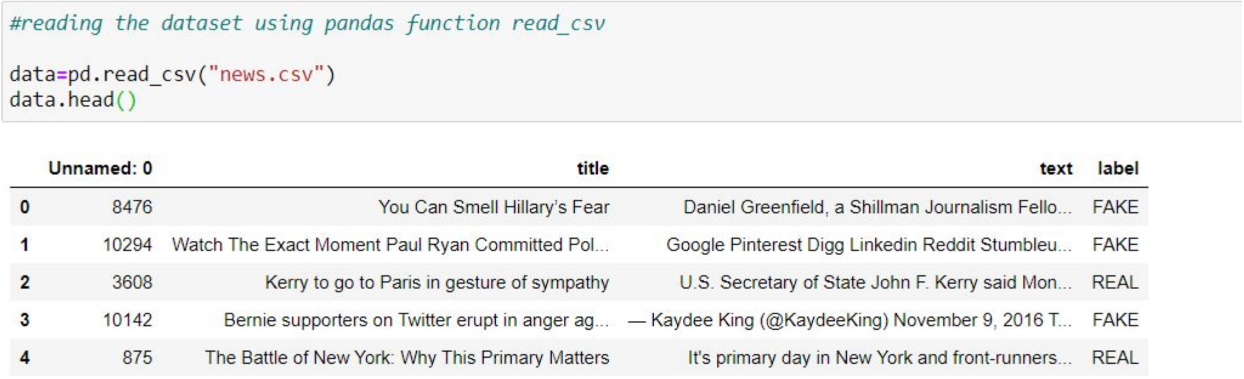
To get a better understanding of whether a news element read anywhere on the internet is FAKE or REAL by considering the features of the data and provide the client with desired results**.**

CHAPTER 5

DATA PREPROCESSING

5.1 READING THE DATASET

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.

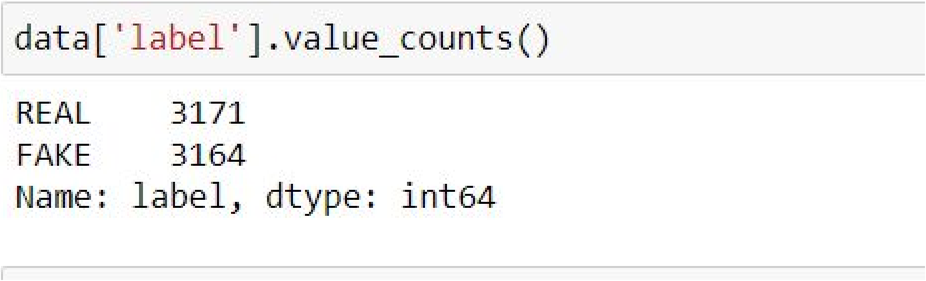


```
#reading the dataset using pandas function read_csv

data=pd.read_csv("news.csv")
data.head()
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Fig 5.1 reading data

Check for the number of FAKE and REAL news items using value_counts()



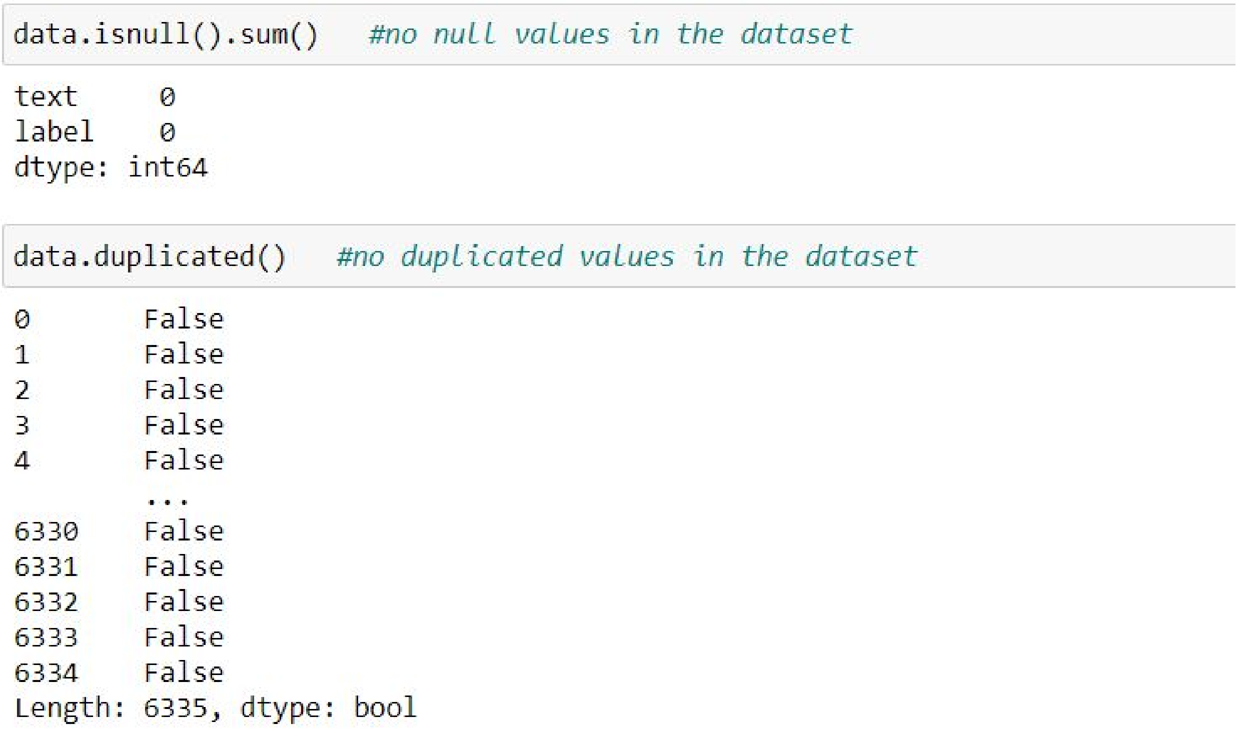
```
data['label'].value_counts()
```

```
REAL    3171
FAKE    3164
Name: label, dtype: int64
```

Fig 5.1.2 value counts

5.2 HANDLING MISSING VALUES AND DUPLICATE VALUES

We can find the number of missing values and duplicated values in each column using isnull() and duplicated() functions respectively. For our dataset no missing values or duplicated values were found.



```
data.isnull().sum()   #no null values in the dataset

text     0
label    0
dtype: int64
```

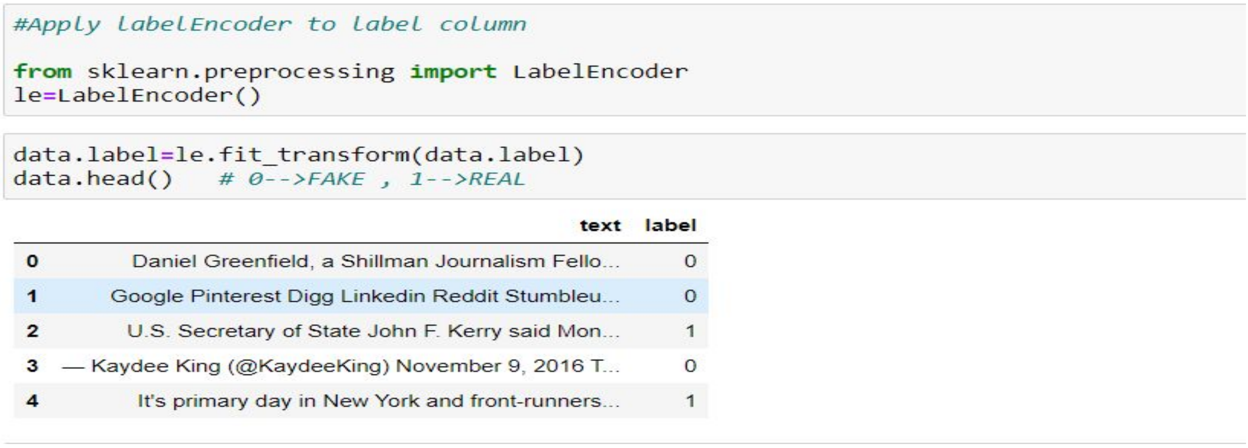
```
data.duplicated()   #no duplicated values in the dataset

0       False
1       False
2       False
3       False
4       False
        ...  
6330    False
6331    False
6332    False
6333    False
6334    False
Length: 6335, dtype: bool
```

Fig 5.2 handling missing values and duplicates

5.3 ENCODING CATEGORICAL DATA

LabelEncoder is imported from sklearn.preprocessing and an object is created for that.

We applied LabelEncoder to the label column and converted fake as 0 and real as 1.



```
#Apply LabelEncoder to label column

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
data.label=le.fit_transform(data.label)
data.head()    # 0-->FAKE , 1-->REAL
```

	text	label
0	Daniel Greenfield, a Shillman Journalism Fello...	0
1	Google Pinterest Digg Linkedin Reddit Stumbleu...	0
2	U.S. Secretary of State John F. Kerry said Mon...	1
3	— Kaydee King (@KaydeeKing) November 9, 2016 T...	0
4	It's primary day in New York and front-runners...	1

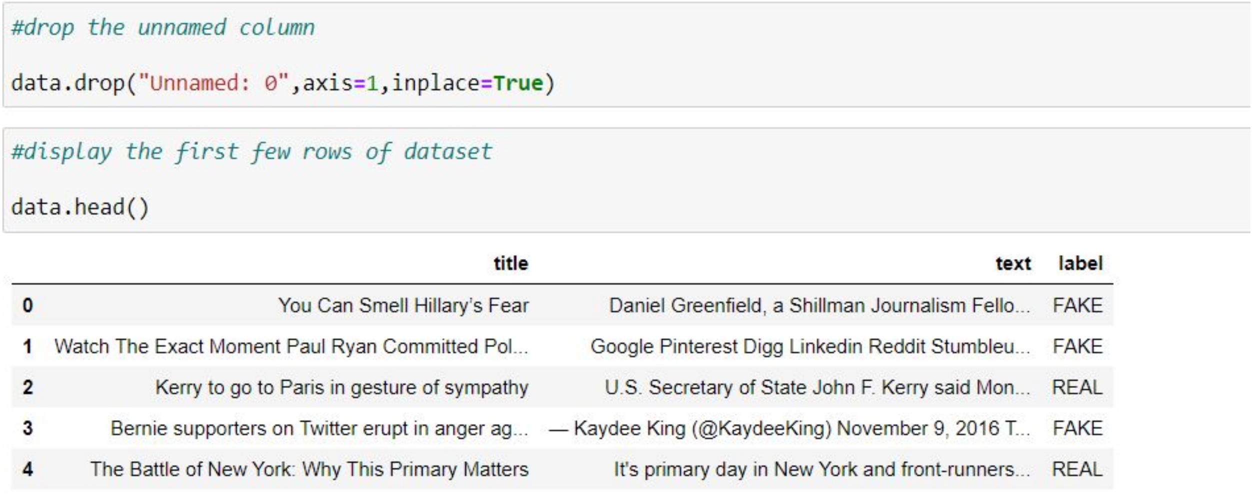
Fig 5.3 encoding categorical data

CHAPTER 6

FEATURE SELECTION

6.1 DROP IRRELEVANT FEATURES

We can observe an unnamed column in the dataset which is of no use or may cause trouble in the further operations, therefore the unnamed column has been dropped.



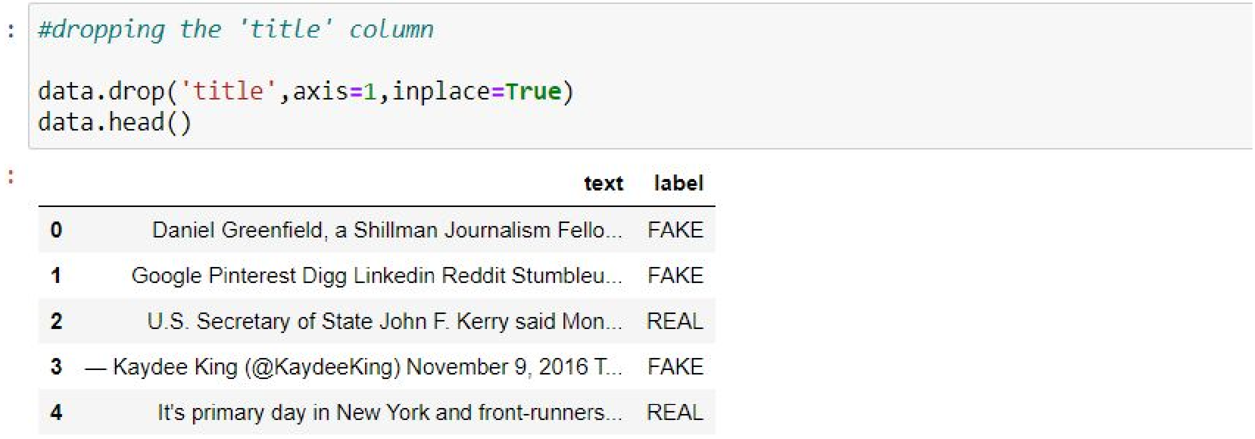
```
#drop the unnamed column

data.drop("Unnamed: 0",axis=1,inplace=True)
```

```
#display the first few rows of dataset

data.head()
```

	title	text	label
0	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Fig 6.1.1. Dropping unnamed column

Also, it can be noticed that the title column does not contribute much to the prediction of a news element as either real or fake, hence drop the title column.



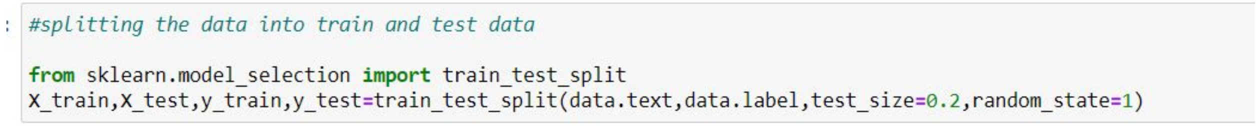
```
#dropping the 'title' column

data.drop('title',axis=1,inplace=True)
data.head()
```

	text	label
0	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	It's primary day in New York and front-runners...	REAL

Fig 6.1.2. Dropping title column

6.2 TRAIN-TEST-SPLIT

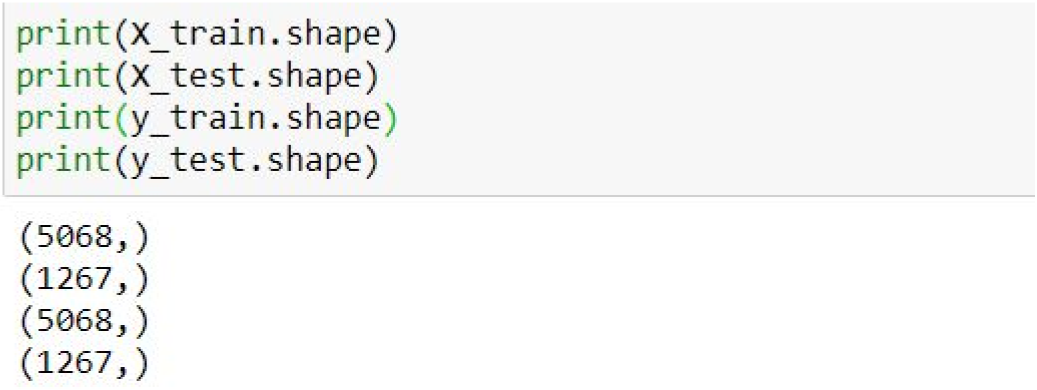
From sklearn.model_selection import TrainTestSplit and split the data by giving text as input and label as output in this problem we took the test size as 20% and random state as 1.



```
#splitting the data into train and test data

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data.text,data.label,test_size=0.2,random_state=1)
```

Fig 6.2.1. Splitting the data

When we display the size of X_train, y_train, X_test and y_test the output is as follows



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

```
(5068,)
(1267,)
(5068,)
(1267,)
```

Fig 6.2.2. Shapes of training and testing data

CHAPTER 7
MODEL BUILDING AND EVALUATION

Approach I : Naive Bayes Classifier

7.1.1 Brief about the algorithms used

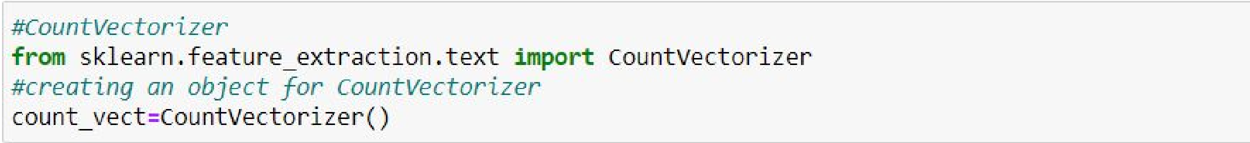
In statistics, Naïve Bayes classifiers are a family of simple "probabilistic classifier" based on applying Bayes’ theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernal density estimation and achieve higher accuracy levels.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. maximum-liklihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Baeysian method.

I. Using CountVectorizer

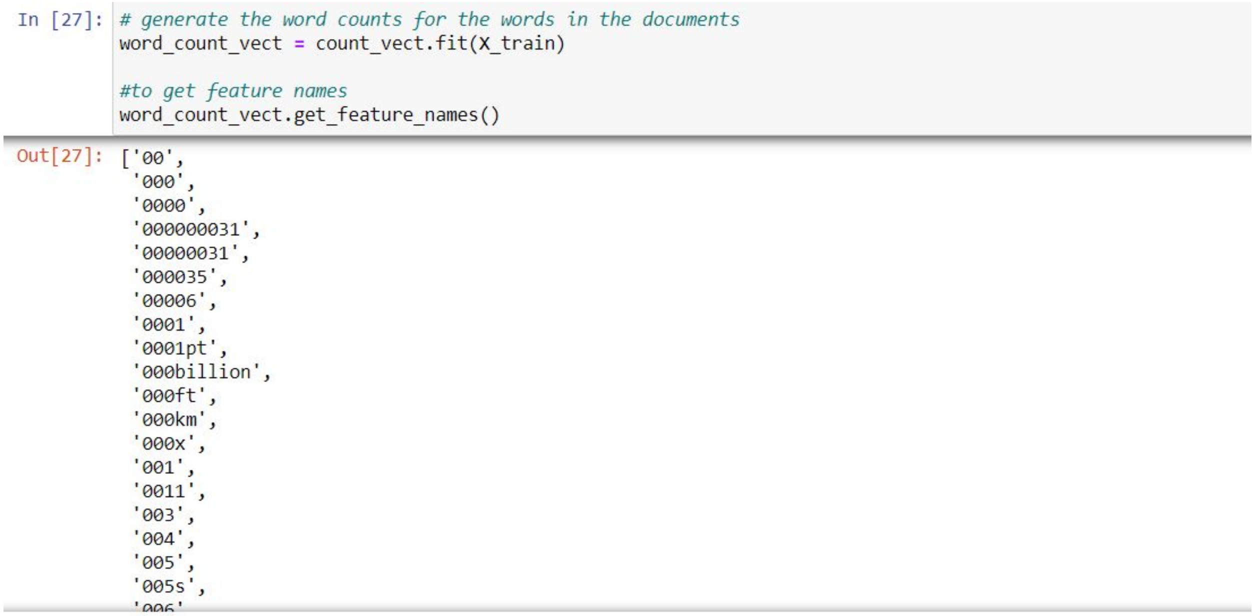
Import CountVectorizer from sklearn.feature_extraction.text and create an object for that class.



```
#CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
#creating an object for CountVectorizer
count_vect=CountVectorizer()
```

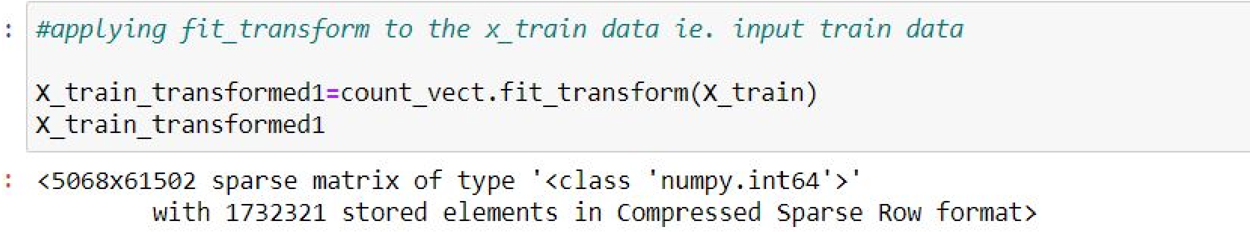
Fig 7.1.1 importing countvectorizer

Generate the word counts and get feature names



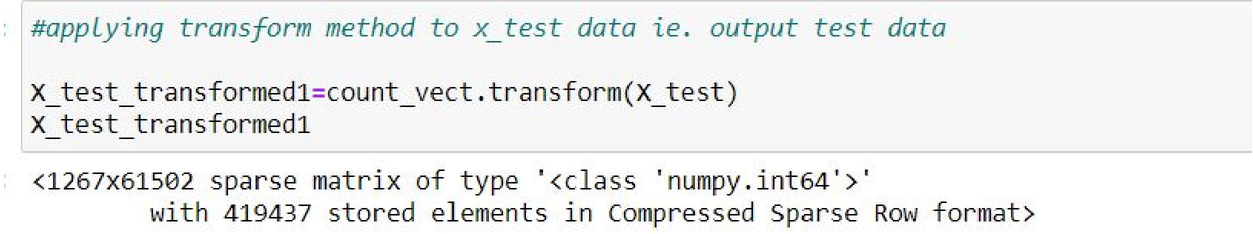
```
In [27]: # generate the word counts for the words in the documents
         word_count_vect = count_vect.fit(X_train)

         #to get feature names
         word_count_vect.get_feature_names()

Out[27]: ['00',
          '000',
          '0000',
          '000000031',
          '00000031',
          '000035',
          '00006',
          '0001',
          '0001pt',
          '000billion',
          '000ft',
          '000km',
          '000x',
          '001',
          '0011',
          '003',
          '004',
          '005',
          '005s',
          '006'
```

7.1.2 Train the Models



```
: #applying fit_transform to the x_train data ie. input train data

  X_train_transformed1=count_vect.fit_transform(X_train)
  X_train_transformed1

: <5068x61502 sparse matrix of type '<class 'numpy.int64'>'
          with 1732321 stored elements in Compressed Sparse Row format>
```

Fig. 7.1.2.a fit_transform on train data

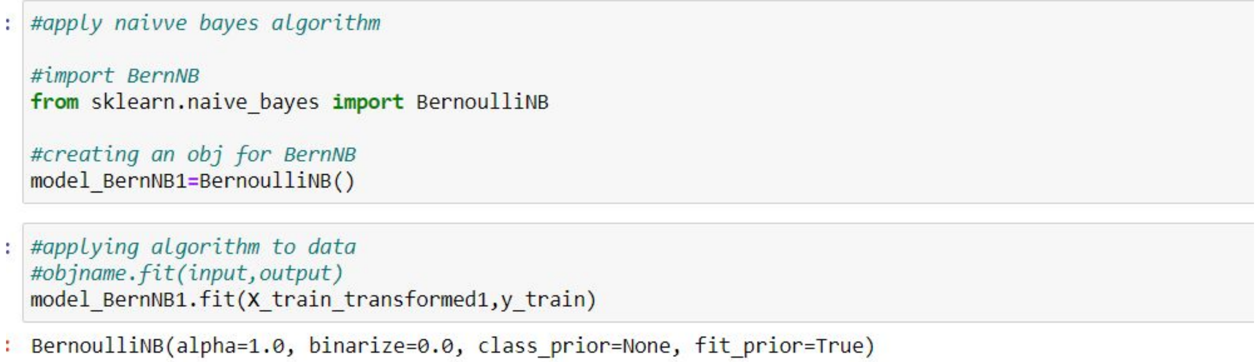


```
#applying transform method to x_test data ie. output test data

X_test_transformed1=count_vect.transform(X_test)
X_test_transformed1
```

```
<1267x61502 sparse matrix of type '<class 'numpy.int64'>'
        with 419437 stored elements in Compressed Sparse Row format>
```

Fig. 7.1.2.b transform on test data

Apply naive bayes algorithm to the data by importing BernoulliNB from sklearn.naive_bayes, then create an object and use the fit method on training data.



```
#apply naivve bayes algorithm

#import BernNB
from sklearn.naive_bayes import BernoulliNB

#creating an obj for BernNB
model_BernNB1=BernoulliNB()
```

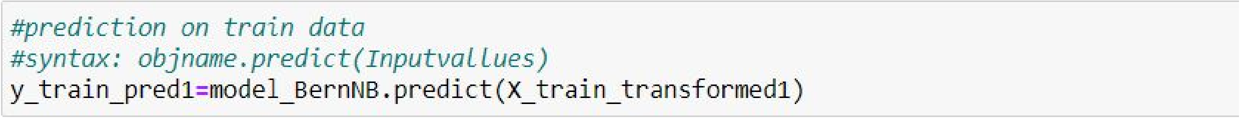
```
#applying algorithm to data
#objname.fit(input,output)
model_BernNB1.fit(X_train_transformed1,y_train)
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Fig 7.1.2.c applying naive bayes algorithm

7.1.3 Predicting on Train Data

We can predict the training data using predict function on X_train

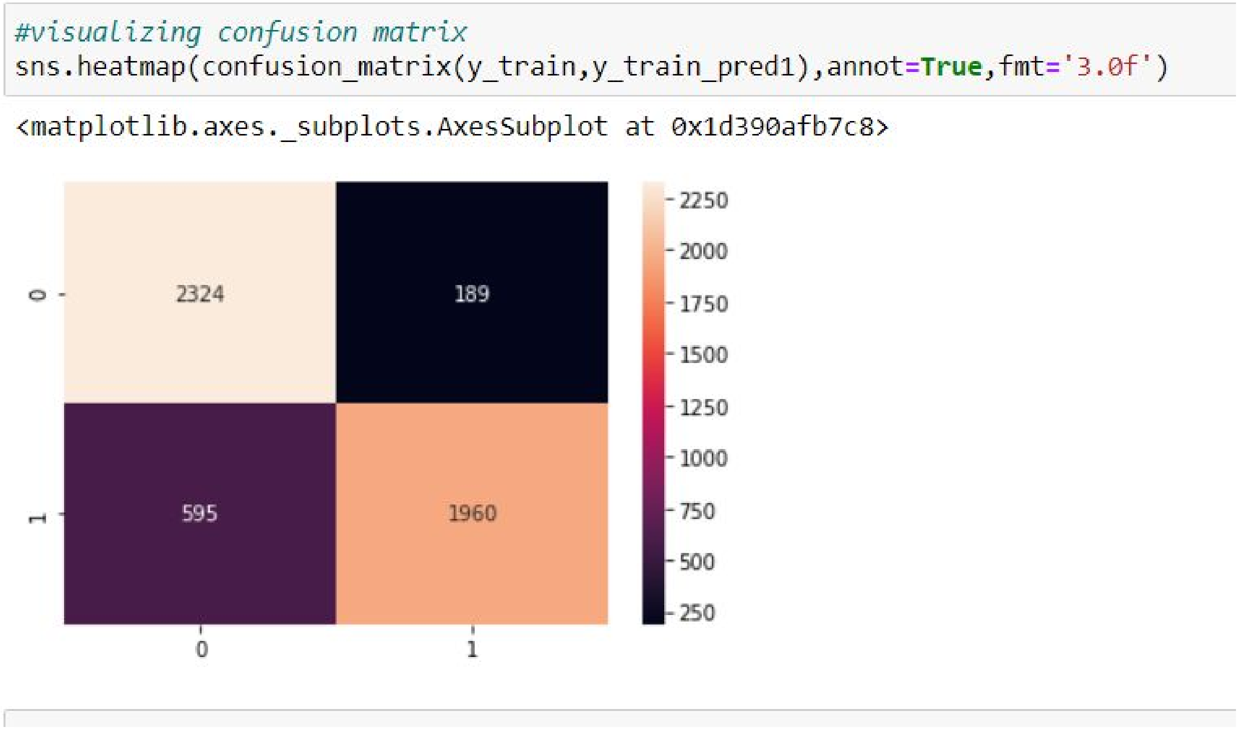


```
#prediction on train data
#syntax: objname.predict(Inputvallues)
y_train_pred1=model_BernNB.predict(X_train_transformed1)
```

Fig 7.1.3.a predicting on train data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heatmap

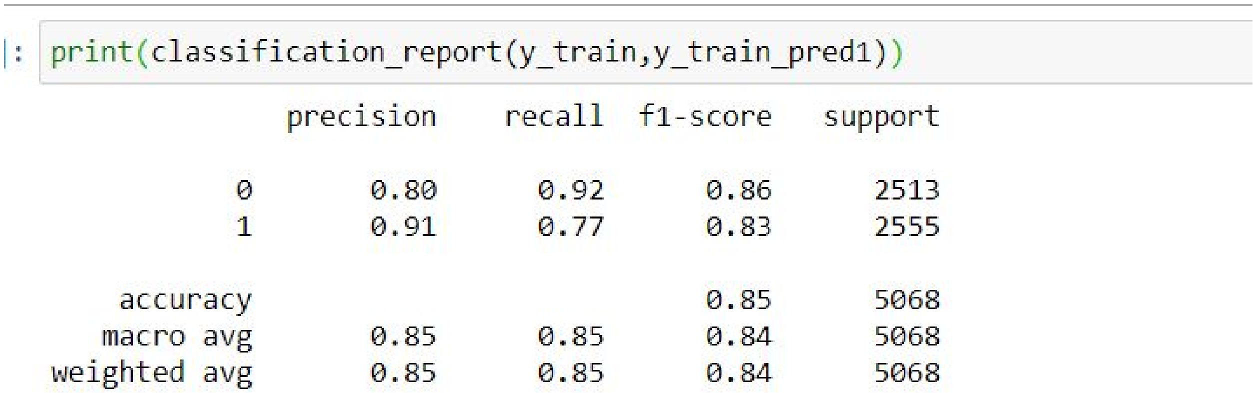


```
#visualizing confusion matrix
sns.heatmap(confusion_matrix(y_train,y_train_pred1),annot=True,fmt='3.0f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d390afb7c8>
```

Fig 7.1.3.b confusion matrix visualization of train data(CountVectorizer)

Print the classification report to know the accuracy score of training data



```
|: print(classification_report(y_train,y_train_pred1))
```

```
              precision    recall  f1-score   support

           0       0.80      0.92      0.86      2513
           1       0.91      0.77      0.83      2555

    accuracy                           0.85      5068
   macro avg       0.85      0.85      0.84      5068
weighted avg       0.85      0.85      0.84      5068
```

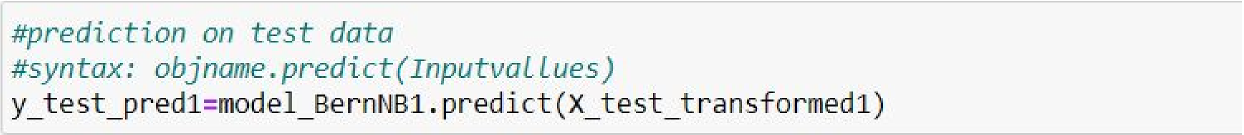
Fig 7.1.3.c classification report for training data(CountVectorizer)

We got an accuracy score of about 85% to the training data which is considered to be a good score.

Now let’s check the same for test data

7.1.4 Predicting on Test Data

We can predict the testing data using predict function on X_test

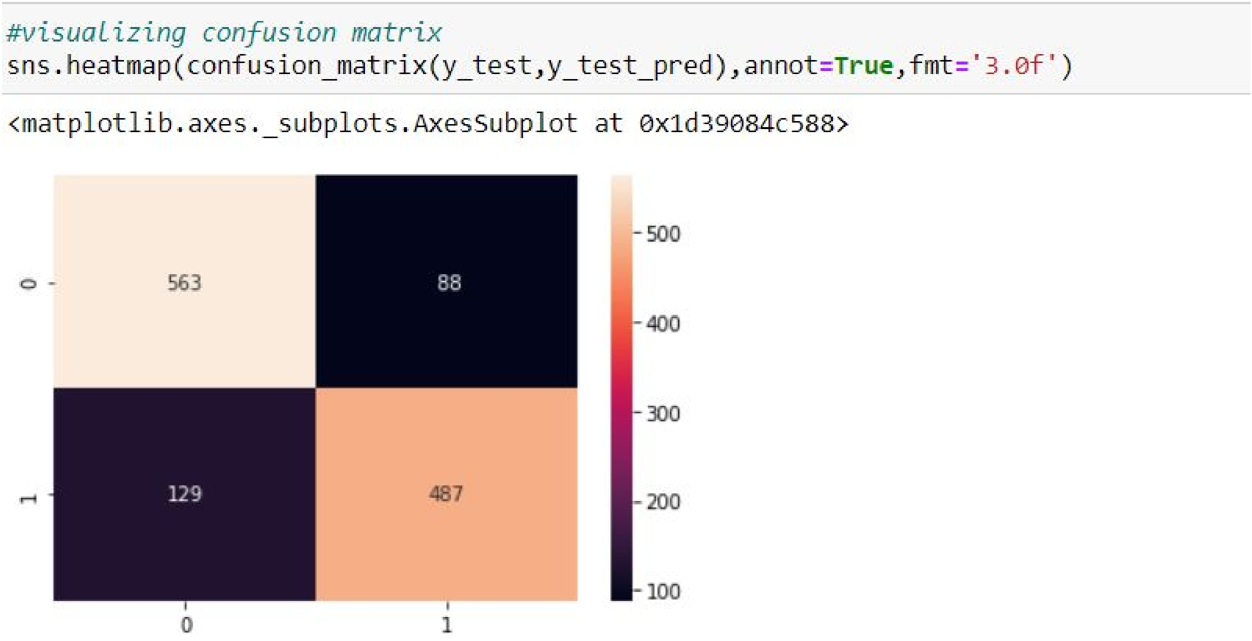


```
#prediction on test data
#syntax: objname.predict(Inputvallues)
y_test_pred1=model_BernNB1.predict(X_test_transformed1)
```

Fig 7.1.4.a predicting on test data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heatmap

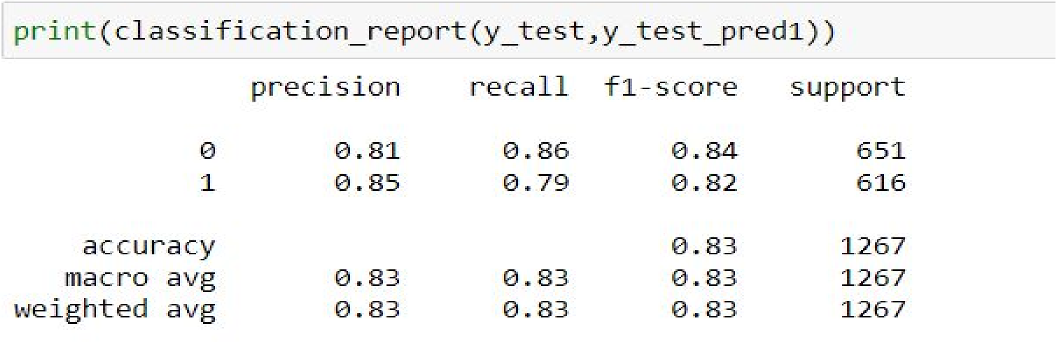


```
#visualizing confusion matrix
sns.heatmap(confusion_matrix(y_test,y_test_pred),annot=True,fmt='3.0f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d39084c588>
```

Fig 7.1.4.b confusion matrix visualization of test data(countvectorizer)

Print the classification report to know the accuracy score of training data



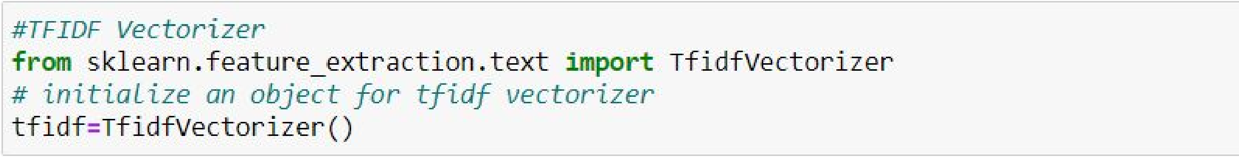
```
print(classification_report(y_test,y_test_pred1))
              precision    recall  f1-score   support

           0       0.81      0.86      0.84       651
           1       0.85      0.79      0.82       616

    accuracy                           0.83      1267
   macro avg       0.83      0.83      0.83      1267
weighted avg       0.83      0.83      0.83      1267
```

Fig 7.1.4.c classification report for testing data(countvectorizer)

We got an accuracy score of around 83%, therefore we can say that it is a best fit and the model predicted very well.

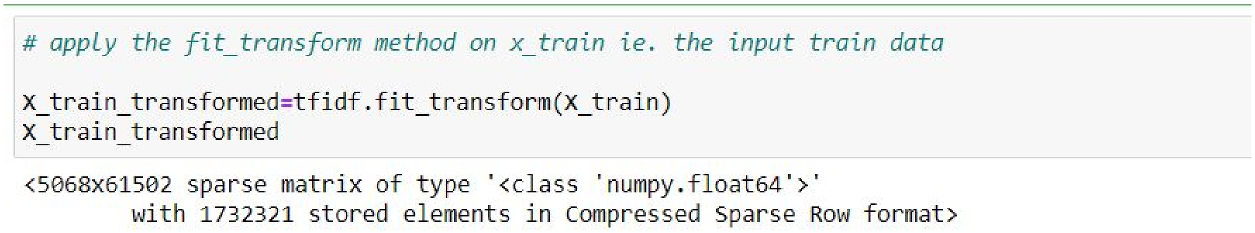
II. Using TfidfVectorizer

Import TfidfVectorizer from sklearn.feature_extraction.text and create an object for that class.



```
#TFIDF Vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
# initialize an object for tfidf vectorizer
tfidf=TfidfVectorizer()
```

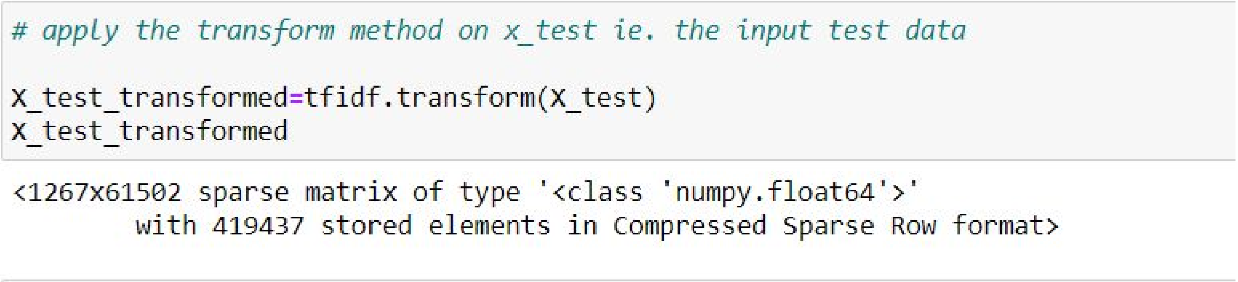
7.1.5 Train The Model



```
# apply the fit_transform method on x_train ie. the input train data

X_train_transformed=tfidf.fit_transform(X_train)
X_train_transformed
```

```
<5068x61502 sparse matrix of type '<class 'numpy.float64'>'
	with 1732321 stored elements in Compressed Sparse Row format>
```

Fig 7.1.5.a fit_transform on train data

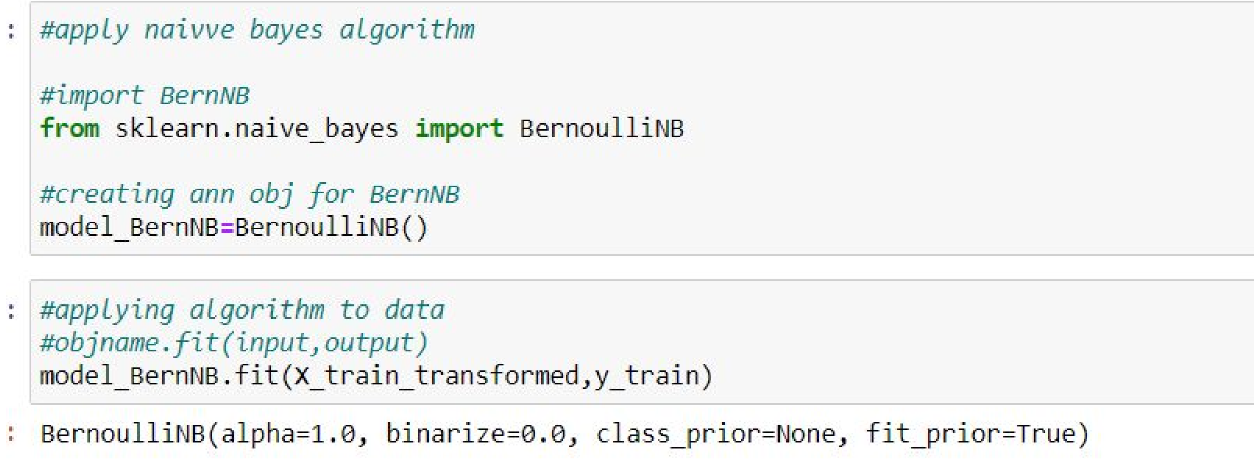


```
# apply the transform method on x_test ie. the input test data

X_test_transformed=tfidf.transform(X_test)
X_test_transformed
```

```
<1267x61502 sparse matrix of type '<class 'numpy.float64'>'
	with 419437 stored elements in Compressed Sparse Row format>
```

Fig 7.1.5.b transform on test data

Apply naive bayes algorithm to the data by importing BernoulliNB from sklearn.naive_bayes, then create an object and use the fit method on testing data.



```
: #apply naivve bayes algorithm

  #import BernNB
  from sklearn.naive_bayes import BernoulliNB

  #creating ann obj for BernNB
  model_BernNB=BernoulliNB()
```

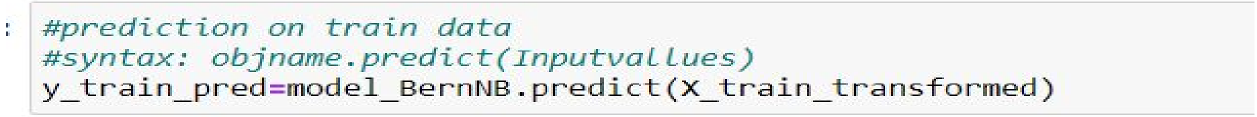
```
: #applying algorithm to data
  #objname.fit(input,output)
  model_BernNB.fit(X_train_transformed,y_train)
```

```
: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Fig 7.1.5.c applying naive bayes algorithm

7.1.6 Predicting on Train Data

We can predict the training data using predict function on X_train

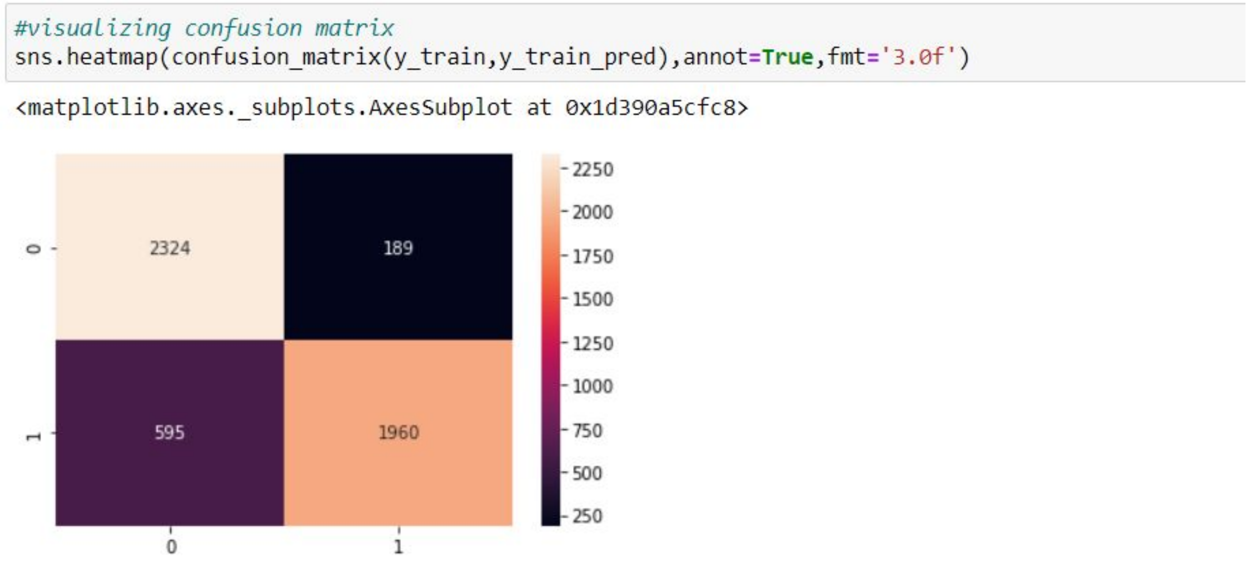


```
: #prediction on train data
  #syntax: objname.predict(Inputvallues)
  y_train_pred=model_BernNB.predict(X_train_transformed)
```

Fig 7.1.6.a predicting on train data(tfidf)

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heat map

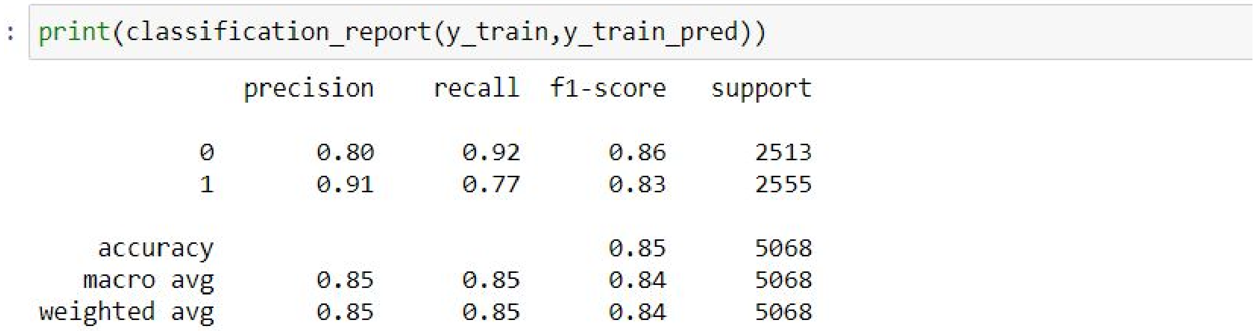


```
#visualizing confusion matrix
sns.heatmap(confusion_matrix(y_train,y_train_pred),annot=True,fmt='3.0f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d390a5cfc8>
```

Fig 7.1.6.b confusion matrix of train data(tfidf)

Print the classification report to know the accuracy score of training data



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

```
              precision    recall  f1-score   support

           0       0.80      0.92      0.86      2513
           1       0.91      0.77      0.83      2555

    accuracy                           0.85      5068
   macro avg       0.85      0.85      0.84      5068
weighted avg       0.85      0.85      0.84      5068
```

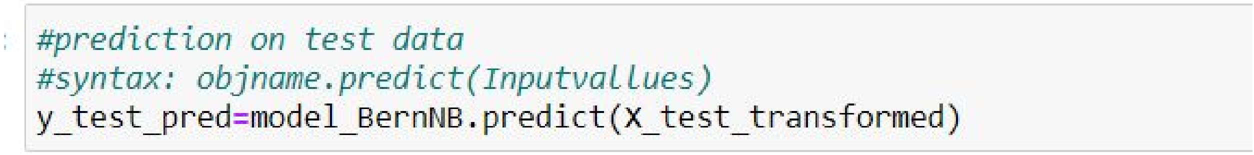
Fig 7.1.6.c classification report of training data(tfidf)

We got an accuracy score of about 85% to the training data which is considered to be a good score.

Now let’s check the same for test data

7.1.7 Predicting on Test Data

We can predict the testing data using predict function on X_test

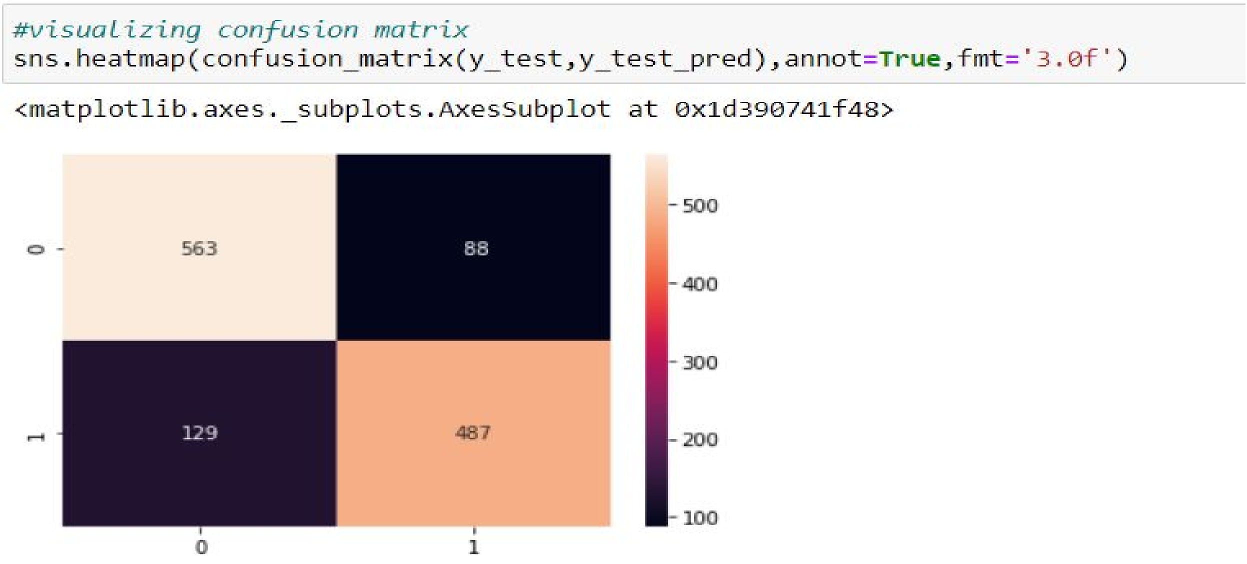


```
#prediction on test data
#syntax: objname.predict(Inputvallues)
y_test_pred=model_BernNB.predict(X_test_transformed)
```

Fig 7.1.7.a predicting on test data(tdidf)

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heat map

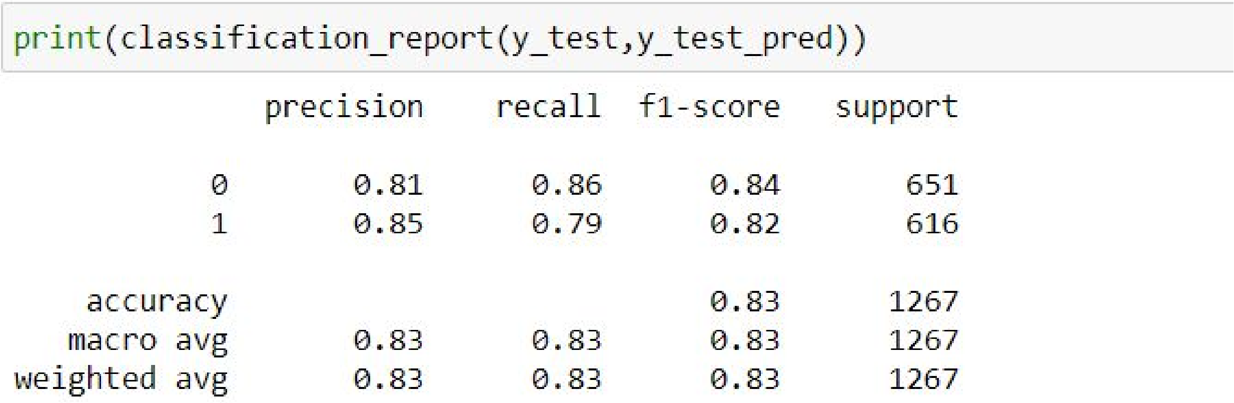


```
#visualizing confusion matrix
sns.heatmap(confusion_matrix(y_test,y_test_pred),annot=True,fmt='3.0f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d390741f48>
```

Fig 7.1.7.b confusion matrix visualization for testing data(tfidf)

Print classification report to know the accuracy of test data

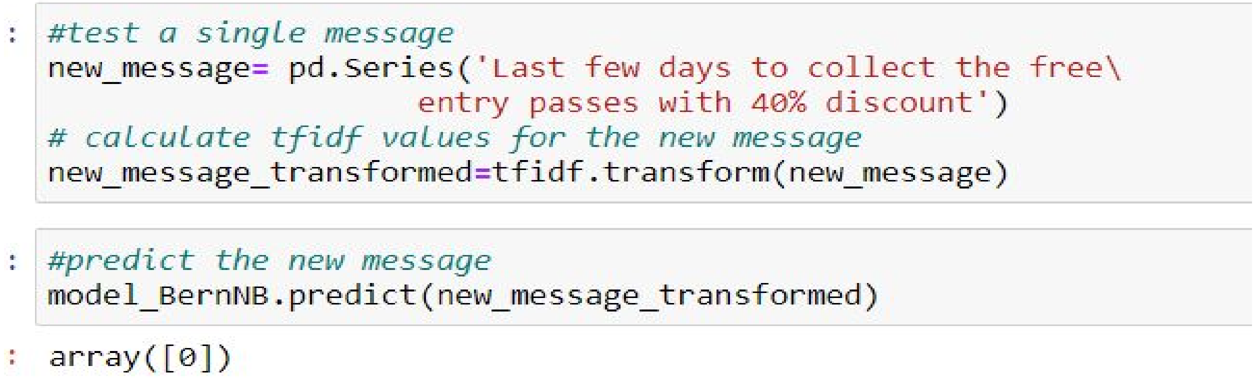


```
print(classification_report(y_test,y_test_pred))
              precision    recall  f1-score   support

           0       0.81      0.86      0.84       651
           1       0.85      0.79      0.82       616

    accuracy                           0.83      1267
   macro avg       0.83      0.83      0.83      1267
weighted avg       0.83      0.83      0.83      1267
```

Fig 7.1.7.c classification report for testing data(tfidf)

We got an accuracy score of around 83%, therefore we can say that it is a best fit and the model predicted very well.

Let us check the accuracy of the model by giving a sample fake news



```
: #test a single message
  new_message= pd.Series('Last few days to collect the free\
                    entry passes with 40% discount')
  # calculate tfidf values for the new message
  new_message_transformed=tfidf.transform(new_message)

: #predict the new message
  model_BernNB.predict(new_message_transformed)

: array([0])
```

Fig 7.1.7.d checking output for a sample news item

It gave the output as 0 indicating that the news is fake.

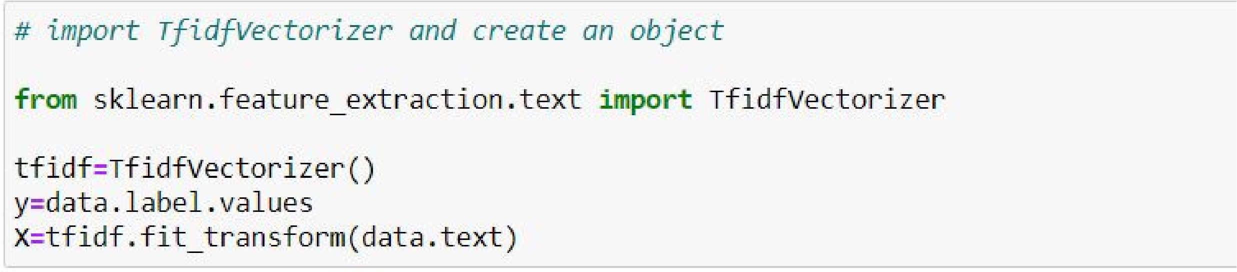
Approach II : Logistic Regression

7.2.1 Brief about the algorithms used

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

7.2.2 Train the model

Import TfidfVectorizer and create an object for that class



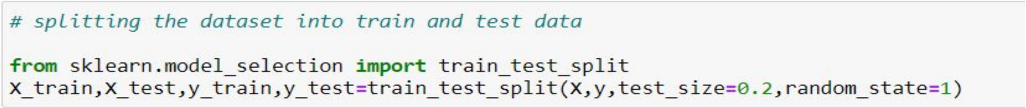
```
# import TfidfVectorizer and create an object

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer()
y=data.label.values
X=tfidf.fit_transform(data.text)
```

Fig 7.2.2.a importing Tfidf vectorizer

Take input data as text and apply fit_transform method to it, then take output as label column

Import train_test_split from sklearn.model_selection and split the data into train and test data.

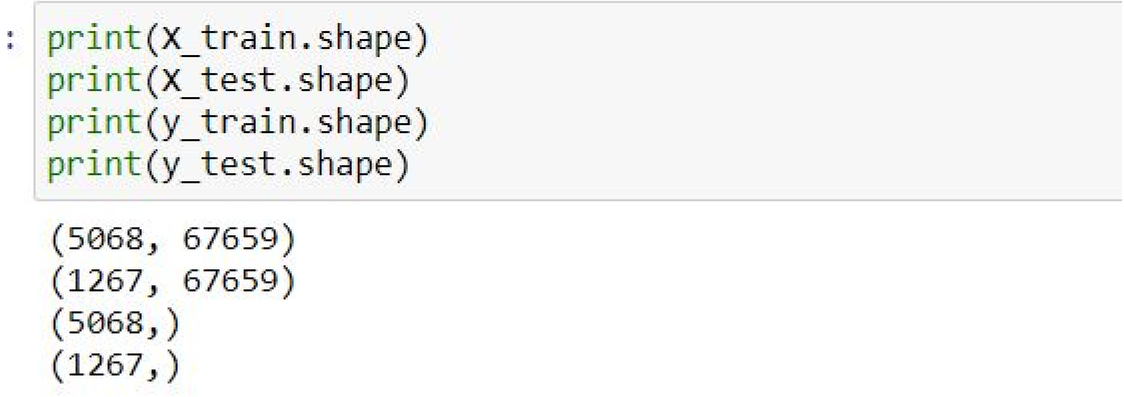
Here, we took the test size as 20% leaving 80% for training data and random state as 1.



```
# splitting the dataset into train and test data

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

Fig 7.2.2.b splitting the data

Print the sizes of X_train, X_test, y_train, y_test

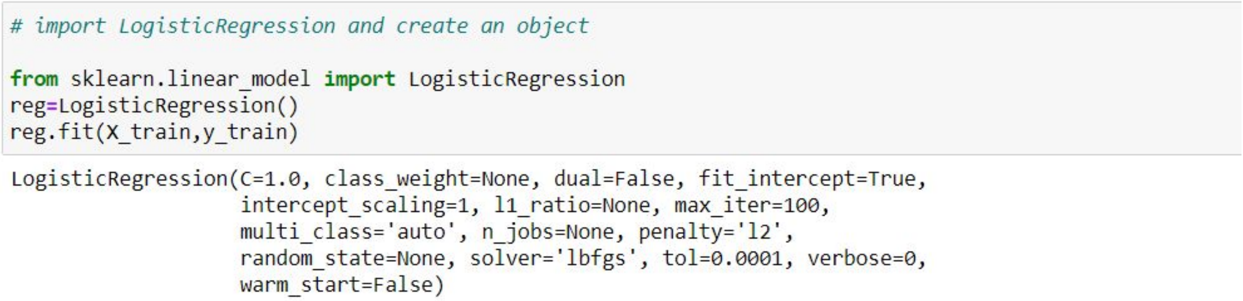


```
: print(X_train.shape)
  print(X_test.shape)
  print(y_train.shape)
  print(y_test.shape)
```

```
(5068, 67659)
(1267, 67659)
(5068,)
(1267,)
```

Fig 7.2.2.c shapes of train and test data

Import LogisticRegression from sklearn.linear_model and create an object. Fit the model on training data



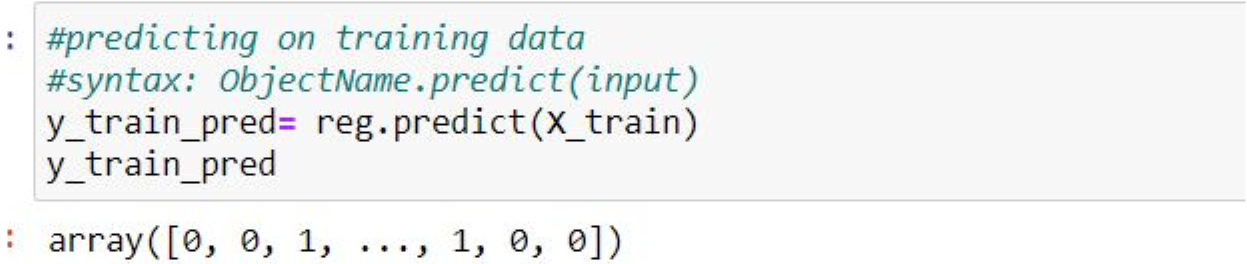
```
# import LogisticRegression and create an object

from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Fig 7.2.2.d importing logistic regression

7.2.3 Predicting on Train data

We can predict the training data using predict function on X_train

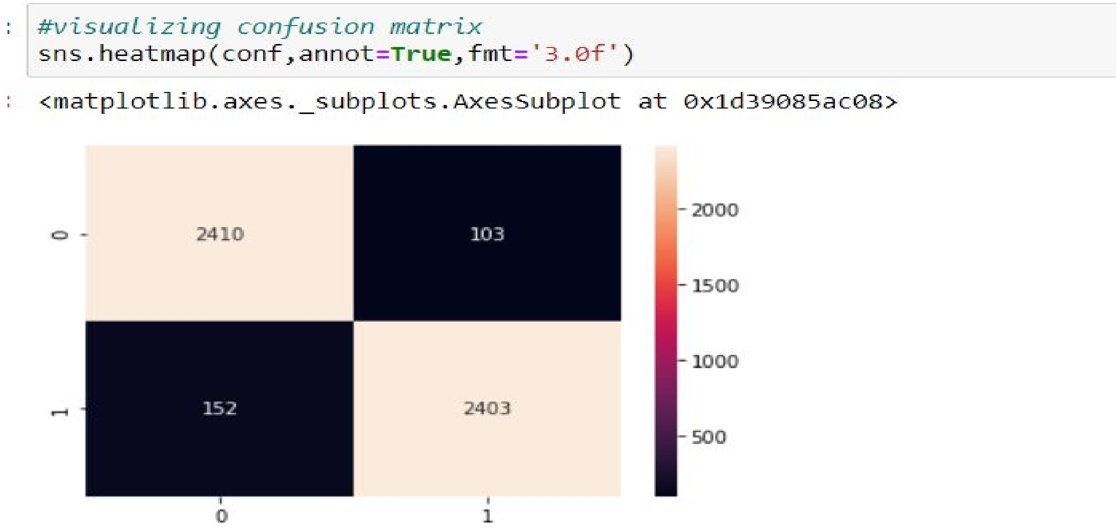


```
: #predicting on training data
  #syntax: ObjectName.predict(input)
  y_train_pred= reg.predict(X_train)
  y_train_pred

: array([0, 0, 1, ..., 1, 0, 0])
```

Fig 7.2.3.a predicting on train data

Import confusion_matrix and classification_report from sklearn.metrics

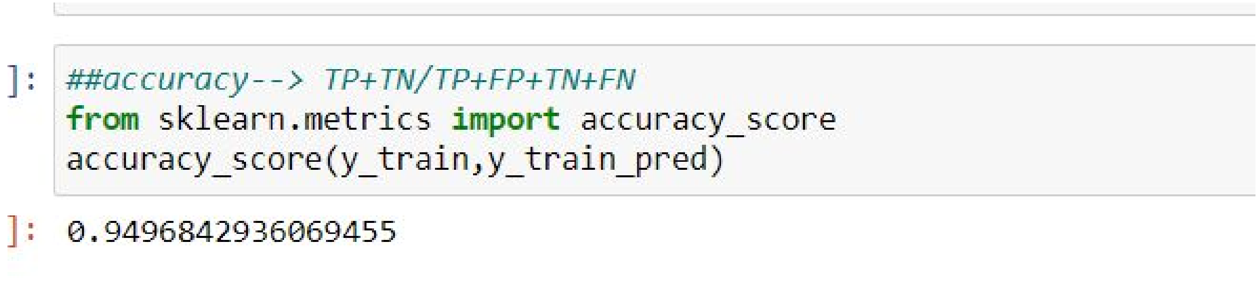
Visualizing the confusion matrix using heat map



```
: #visualizing confusion matrix
  sns.heatmap(conf,annot=True,fmt='3.0f')

: <matplotlib.axes._subplots.AxesSubplot at 0x1d39085ac08>
```

Fig 7.2.3.b visualizing confusion matrix of train data

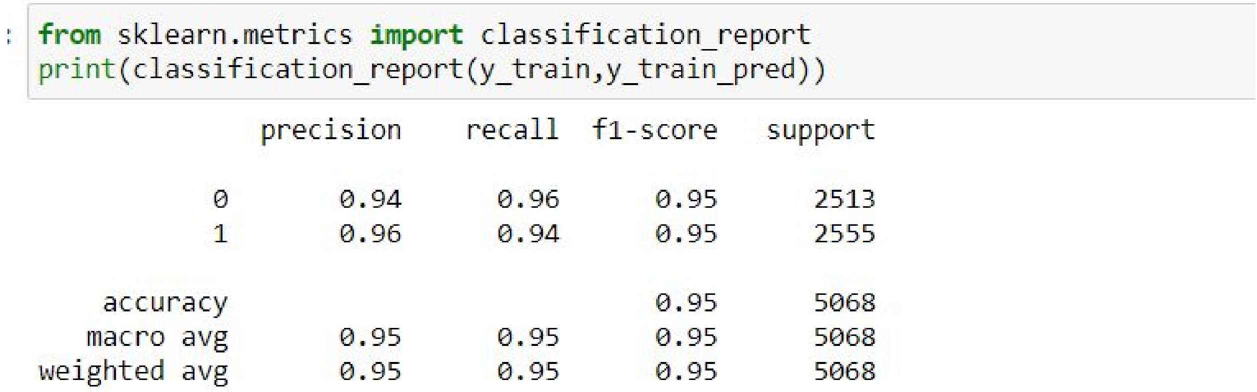
Find the accuracy score



```
]: ##accuracy--> TP+TN/TP+FP+TN+FN
   from sklearn.metrics import accuracy_score
   accuracy_score(y_train,y_train_pred)

]: 0.9496842936069455
```

Fig 7.2.3.c accuracy of train data

Print the classification report and check the accuracy of the training data



```
: from sklearn.metrics import classification_report
  print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      2513
           1       0.96      0.94      0.95      2555

    accuracy                           0.95      5068
   macro avg       0.95      0.95      0.95      5068
weighted avg       0.95      0.95      0.95      5068
```

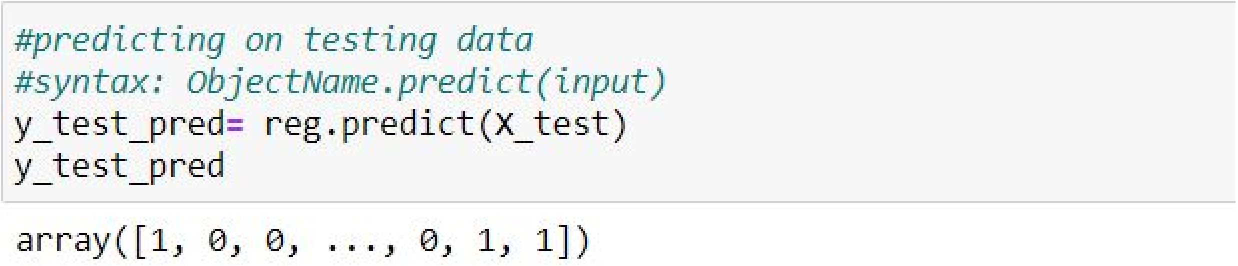
Fig 7.2.3.d classification report of train data

We got an accuracy score of around 95% for the training data which is considered to be a good score.

Now let’s check the same for test data.

7.2.4 Predicting on test data

We can predict the testing data using predict function on X_test



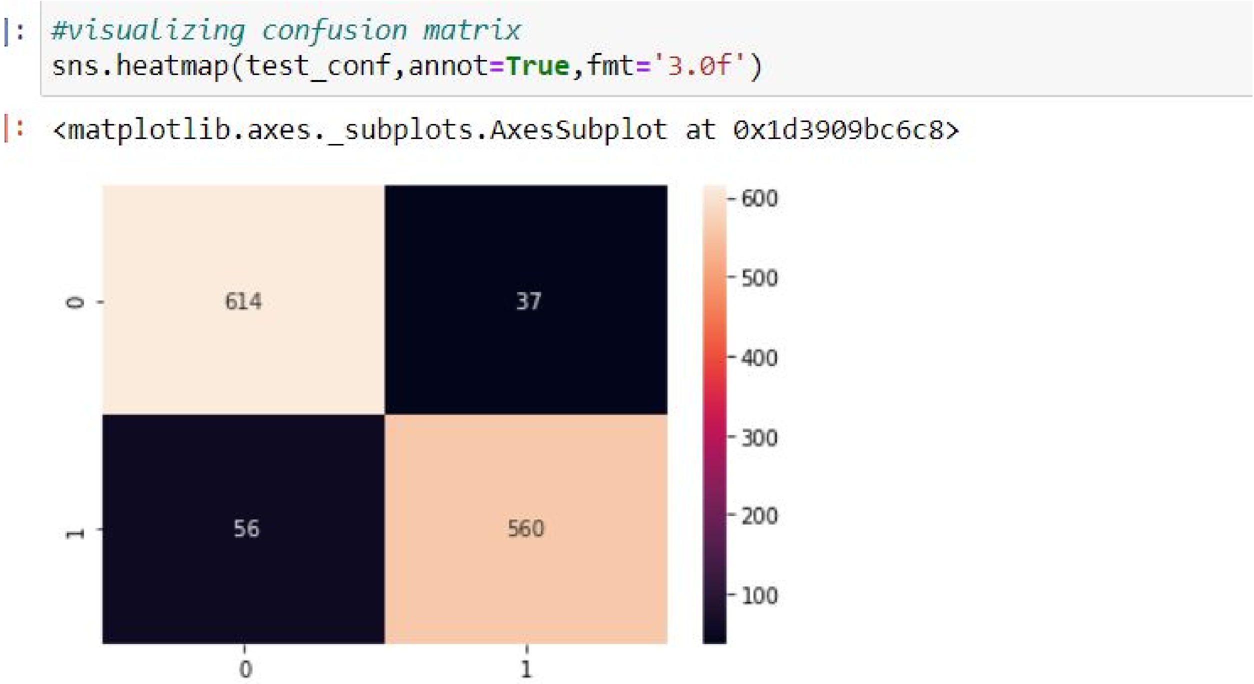
```
#predicting on testing data
#syntax: ObjectName.predict(input)
y_test_pred= reg.predict(X_test)
y_test_pred
```

```
array([1, 0, 0, ..., 0, 1, 1])
```

Fig 7.2.4.a predicting on test data

Import confusion_matrix and classification_report from sklearn.metrics

Visualizing the confusion matrix using heat map

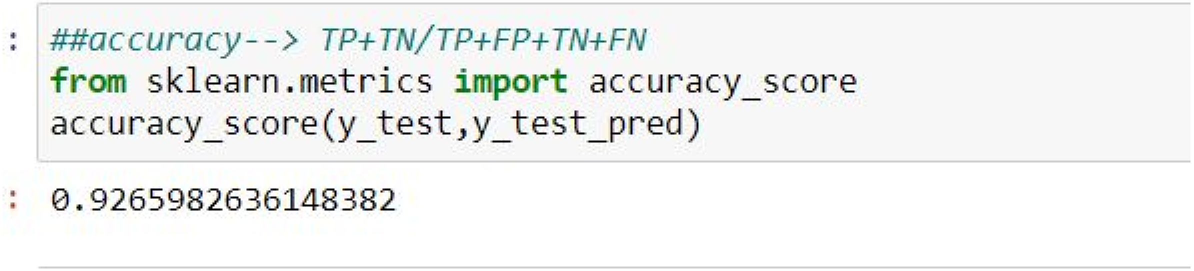


```
]: #visualizing confusion matrix
   sns.heatmap(test_conf,annot=True,fmt='3.0f')
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x1d3909bc6c8>
```

Fig 7.2.4.b visualizing confusion matrix of test data

Find the accuracy score

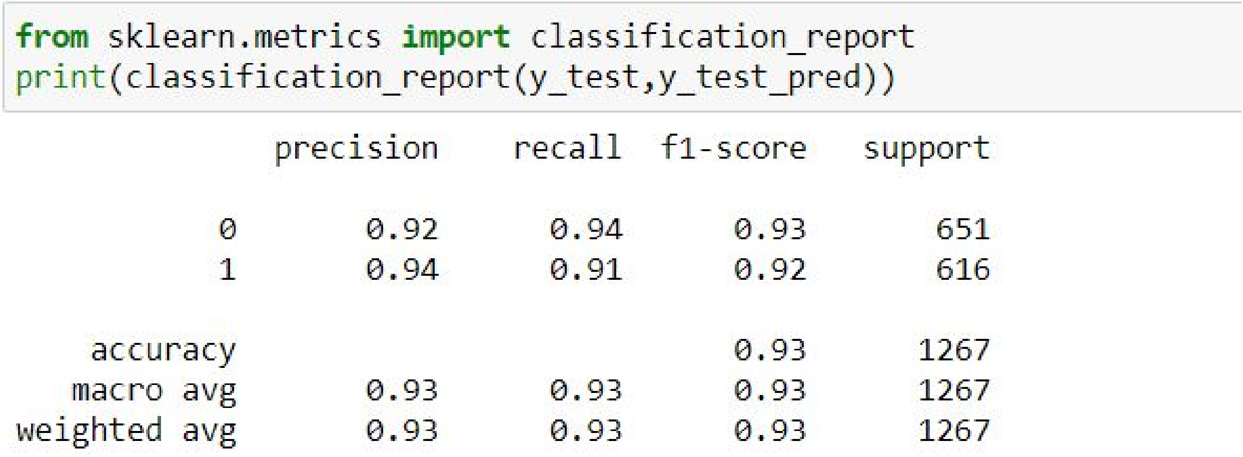


```
##accuracy--> TP+TN/TP+FP+TN+FN
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_test_pred)
```

```
0.9265982636148382
```

Fig 7.2.4.c accuracy score of test data

Print the classification report and check the accuracy of the training data



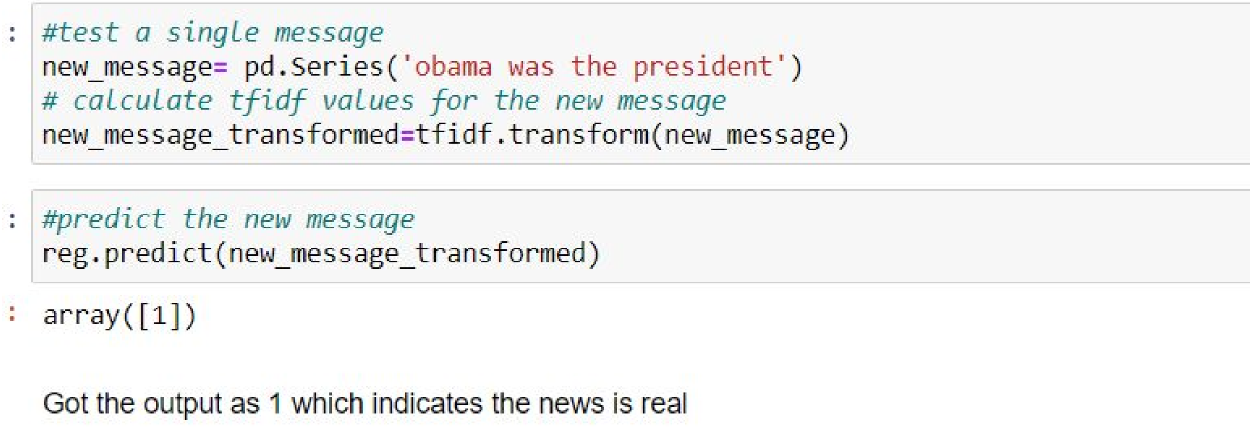
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       651
           1       0.94      0.91      0.92       616

    accuracy                           0.93      1267
   macro avg       0.93      0.93      0.93      1267
weighted avg       0.93      0.93      0.93      1267
```

Fig 7.2.4.d classification report of testing data

We got an accuracy score of around 93%, therefore we can say that it is a best fit and the model predicted very well.

Let us check the accuracy of the model by giving a sample real news



```
#test a single message
new_message= pd.Series('obama was the president')
# calculate tfidf values for the new message
new_message_transformed=tfidf.transform(new_message)
```

```
#predict the new message
reg.predict(new_message_transformed)
```

```
array([1])
```

Got the output as 1 which indicates the news is real

Fig 7.2.4.e checking output for a sample news item

It gave the output as 1 which indicates that the news is real.

8. CONCLUSION:

Comparing the two models we can clearly say that logistic regression performed well with an accuracy of around 93%-95% compared to naive bayes which gave an accuracy score of around 83%-85%.

9. REFERENCES:

1. https://medium.com/code-heroku/introduction-to-exploratory-data-analysis-eda-c0257f888676
2. https://en.wikipedia.org/wiki/Machine_learning
3. https://www.edureka.co/blog/what-is-data-science/
4. https://www.edureka.co/blog/data-science-applications/
5. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
6. https://en.wikipedia.org/wiki/Logistic_regression