DATA SCIENCE

(Fake News Classifier)

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

In

Computer Science and Engineering

GAMPA KAVYASREE

221710306018

Under the Guidance of

Assistant Professor

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

Department Of Computer Science and Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
JULY 2020

DECLARATION

I submit this industrial training work entitled **"FAKE NEWS CLASSIFIER**" 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 , 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 Gampa Kavyasree

Date: 221710306018

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

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"FAKE NEWS CLASSIFIER"** is being submitted by Gampa Kavyasree (221710306018) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science & Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2018-19

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

Mr.
Assistant Professor
Department of CSE

Dr.S.Phani kumar
Professor and HOD
Department of CSE

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 **Dr.S.Phani Kumar,** Head of the Department of Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a 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.

Gampa Kavyasree
221710306018

ABSTRACT

This paper helps us to detect the accuracy of the fake news using Naive Bayes classification. Here the data is divided into test dataset and train dataset and the train dataset is divided into groups of similar information. Test data is later matched with these groups and accuracy is found using Naive Bayes classifier. It helps in knowing whether a given news is fake or real. It provides maximum accuracy and helps to determine the fake news.

Keywords—Machine Learning, Fake News Classification, Probability

Table of Contents:

LIST OF FIGURES:

CHAPTER 1

DATA SCIENCE

1.1 INTRODUCTION:

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data.[1][2] Data science is related to data mining, deep learning and big data.

Data science is a "concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods" in order to "understand and analyze actual phenomena" with data.[3] It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge and information science. Turing award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.

Data science is an interdisciplinary field focused on extracting knowledge from data sets, which are typically large (see big data). The field encompasses analysis, preparing data for analysis, and presenting findings to inform high-level decisions in an organization. As such, it incorporates skills from computer science, mathematics, statistics, information visualization, graphic design, and business.[7][8] Statistician Nathan Yau, drawing on Ben Fry, also links data science to human-computer interaction: users should be able to intuitively control and explore data. In 2015, the American Statistical Association identified database management, statistics and machine learning, and distributed and parallel systems as the three emerging foundational professional communities

1.2 Need of Data Science:

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

Data scientists help improve how humans make decisions and how algorithms optimize outcomes" I believe that data science has the power to improve the human condition by helping

us investigate phenomena, acquire new knowledge and integrate previous knowledge with new ideas.

Data science methodologies can explore historicals, make comparisons to competition, analyze the market, and ultimately, make recommendations of when and where your product or services will sell best. This can help a company understand how their product helps others and, as needed, question existing business processes.

Big data. Virtually every organization has it and most want to find ways to use it to help them grow their business. That's where data scientists come in. Data scientists know how to use their skills in math, statistics, programming, and other related subjects to organize large data sets Fresh college graduates who are curious to learn quantitative/statistical analysis, and have a fair knowledge of programming can pursue a career in Data Science. One doesn't have to necessarily possess a degree, or a Ph. D. to become a Data Science professional

1.3 USES OF DATA SCIENCE:

1.3.1 Fraud and Risk Detection:

The earliest applications of data science were in Finance. Companies were fed up of bad debts and losses every year. However, they had a lot of data which use to get collected during the initial paperwork while sanctioning loans. They decided to bring in *data scientists* in order to rescue them out of losses.

Over the years, banking companies learned to divide and conquer data via customer profiling, past expenditures, and other essential variables to analyze the probabilities of risk and default. Moreover, it also helped them to push their banking products based on customer's purchasing power.

1.3.2 Healthcare:

The healthcare sector, especially, receives great benefits from data science applications.

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.

1.3.3 Speech Recognition:

Some of the best examples of speech recognition products are Google Voice, Siri, Cortana etc. Using speech-recognition feature, even if you aren't in a position to type a message, your life wouldn't stop. Simply speak out the message and it will be converted to text. However, at times, you would realize, speech recognition doesn't perform accurately.

1.3.4 Airline Route Planning:

Airline Industry across the world is known to bear heavy losses. Except for a few airline service providers, companies are struggling to maintain their occupancy ratio and operating profits. With high rise in air-fuel prices and need to offer heavy discounts to customers has further made the situation worse. It wasn't for long when airlines companies started using data science to identify the strategic areas of improvements. Now using data science, the airline companies can:

1. Predict flight delay
2. Decide which class of airplanes to buy
3. Whether to directly land at the destination or take a halt in between (For example, A flight can have a direct route from New Delhi to New York. Alternatively, it can also choose to halt in any country.)
4. Effectively drive customer loyalty programs

   Southwest Airlines, Alaska Airlines are among the top companies who've embraced data science to bring changes in their way of working.

You can get a better insight into it by referring to this video by our team, which vividly speaks of all the various fields conquered by Data Science Applications.

1.3.5 Gaming:

Games are now designed using machine learning algorithms which improve/upgrade themselves as the player moves up to a higher level. In motion gaming also, your opponent (computer) analyzes your previous moves and accordingly shapes up its game. EA Sports, Zynga, Sony, Nintendo, ActivisionBlizzard have led gaming experience to the next level using data science.

1.3.6Augmented Reality

This is the final of the data science applications which seems most exciting in the future. Augmentedreality.

Data Science and Virtual Reality do have a relationship, considering a VR headset contains computing knowledge, algorithms and data to provide you with the best viewing experience. A very small step towards this is the high trending game of *Pokemon GO*. The ability to walk around things and look at

Pokemon on walls, streets, things that aren't really there. The creators of this game used the data from Ingress, the last app from the same company, to choose the locations of the Pokemon and gyms.

CHAPTER 2

MACHINE LEARNING

2.1 INTRODUCTION:

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

2.2 IMPORTANCE OF MACHINE LEARNING:

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

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

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

Figure 2.2: The Process Flow

2.3 USES OF MACHINE LEARNING:

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

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

2.4 TYPES OF LEARNING ALGORITHMS:

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

2.4.1 SUPERVISED LEARNING :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the

correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a 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.

2.4.2 UNSUPERVISED LEARNING:

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

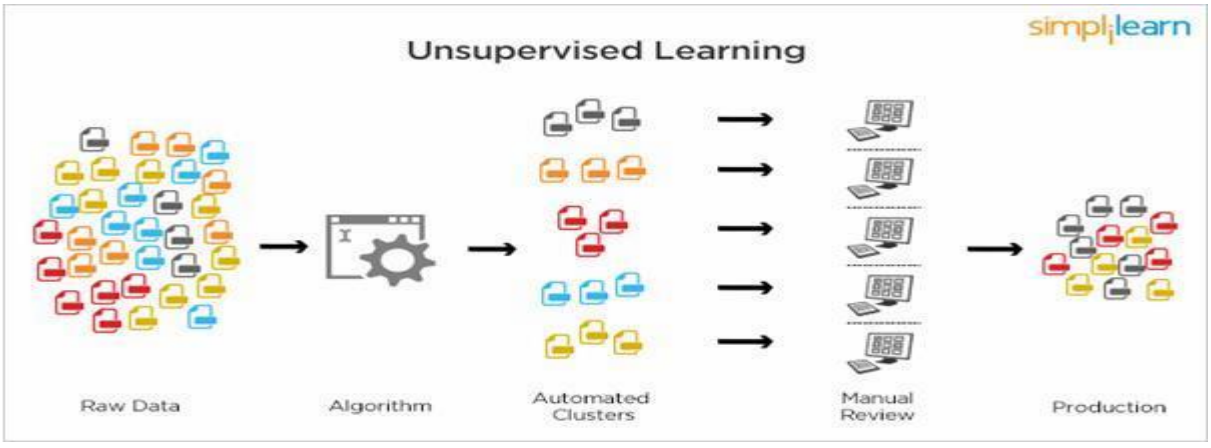


Figure 2.4.2: Unsupervised Learning

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.

2.4.3 SEMI SUPERVISED LEARNING:

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

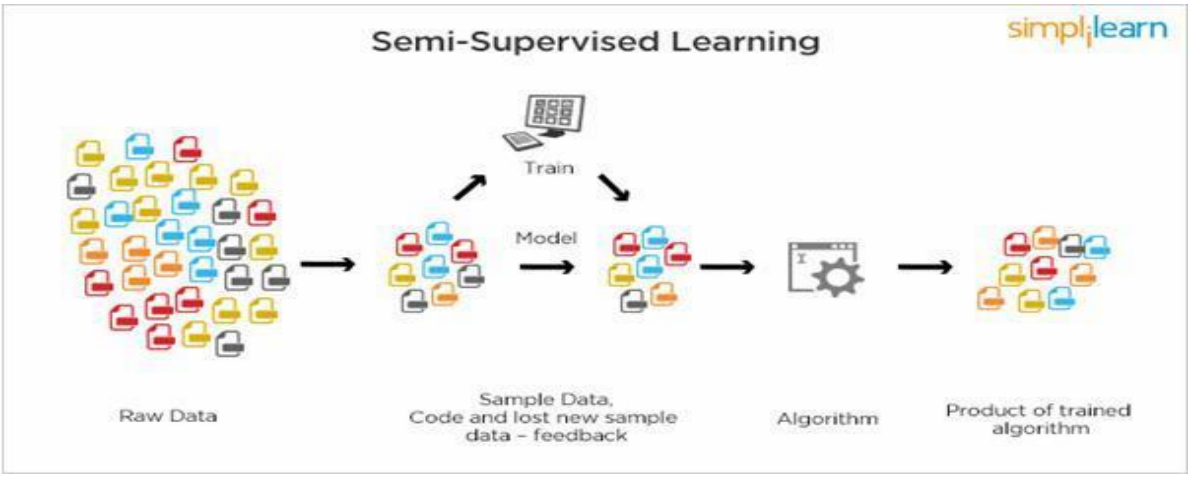


Figure 2.4.3: Semi Supervised Learning

2.5 RELATION BETWEEN DATA MINING MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions. Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 3

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

3.2 HISTORY OF PYTHON:

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

3.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax,This allows the student to pick up the language quickl.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform
- compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same
- interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter.
- These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.

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

3.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and
- Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation,
- news, etc., is
- available on the official website of Python.

3.4.1 Installation(using python IDLE):

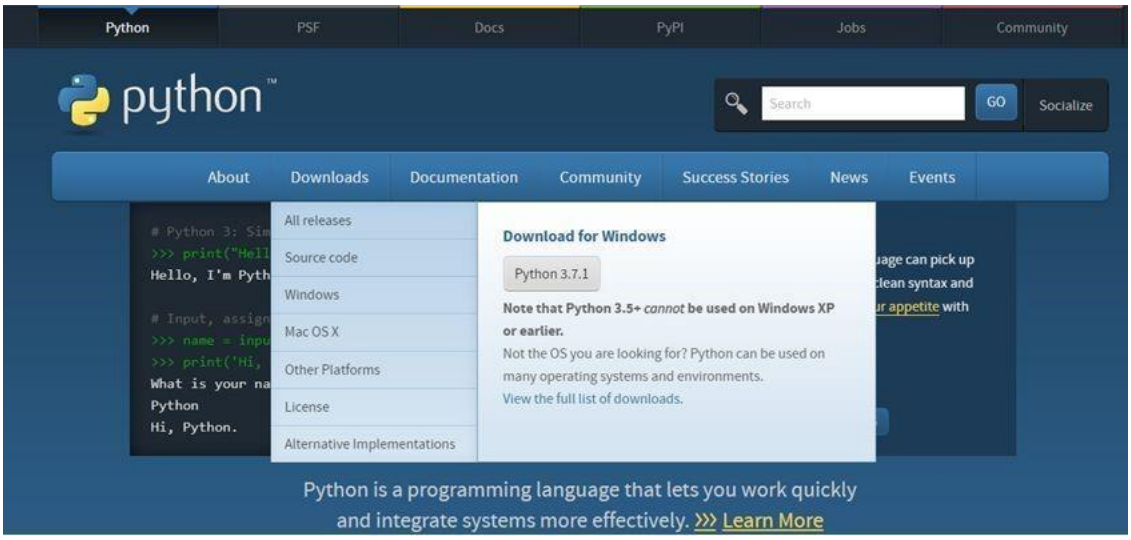
- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is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

Figure 3.4.1: Python download

3.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- In windows
- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path (i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

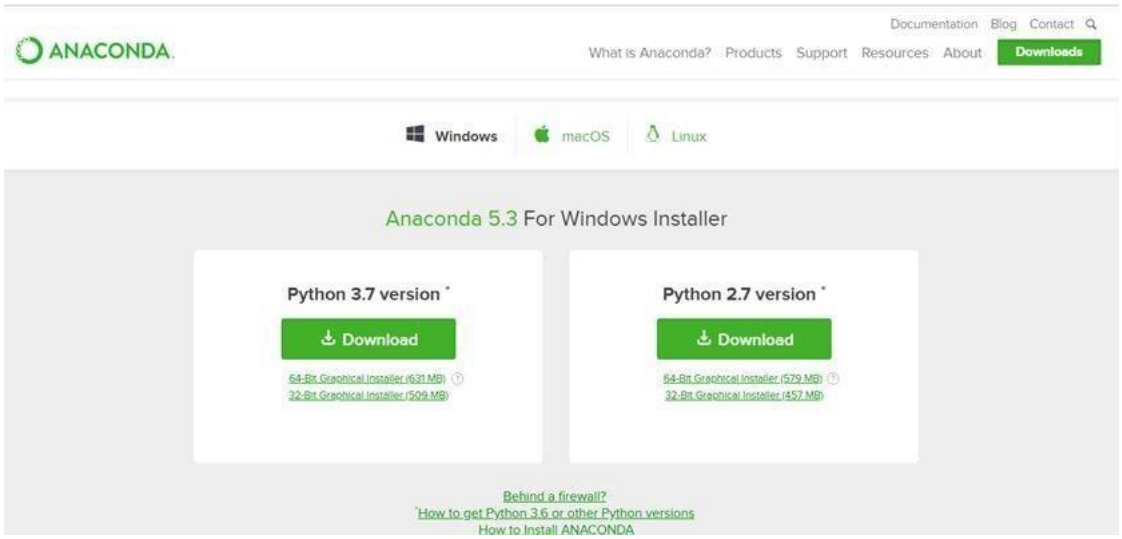


Figure 3.4.2: Anaconda download

Figure 3.4.2: Jupyter notebook

3.5 PYTHON VARIABLE TYPES:

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

3.5.1 Python Numbers:

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

3.5.2 Python Strings:

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

3.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).

- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-onllists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

3.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 PYTHON FUNCTION:

3.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses.You can also define parameters inside these parentheses

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

3.6.1Calling 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.

3.7PYTHON USING OOP's CONCEPTS:

3.7.1Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.
- Defining a Class:
- We define a class in a very similar way how we define a function.
- Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is .indented like a functions body is.

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

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

Figure 3.7.1: Defining a Class

- ## 3.7.2__init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 4

CASE STUDY

4.1 PROBLEM STATEMENT:

To check if the news is fake or real NAÏVE BAYES can be used .It is a kind of algorithm is used in text classification.

```
In [1]: import pandas as pd

        print(pd.__version__)
        # 0.22.0

0.25.1
```

```
In [2]: import numpy as np
        print(pd.__version__)
        # 0.22.0

0.25.1
```

```
In [3]: import seaborn as sns
        print(pd.__version__)
        # 0.22.0

0.25.1
```

Figure 4.1: Versions

4.2OBJECTIVE OF THE CASE STUDY:

Therefore by using naïve Bayes theorem we can conclude that any news from a large or small dataset can be classified as fake or real news by matching it with the previous dataset values in less time which in turn helps the users to believe in a particular news.

CHAPTER 5

MODEL BUILDING

5.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

5.1.1 GETTING THE DATASET:

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

5.1.2 IMPORTING THE LIBRARIES:

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

```
#importing the required libraries

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

Figure 5.1.2 : Importing Libraries

5.2 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method news.csv(). The news.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.Read data from data set.

```
#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

Figure 5.2: Reading the dataset

CHAPTER 6

EVALUATING THE CASE STUDY

6.1 DROP THE UNNAMED COLUMN:

To remove the unnamed columns we can use two different methods; loc and drop,together with other pandas dataframes methods.when using the drop method we can use the inplace parameter and get a dataframe without unnamed columns.

```
#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

Figure 6.1: Droping the unnamed column

6.2DATA SHAPE:

The shape attribute of pandas. DataFrame stores the number of rows and columns as a tuple (number of rows, number of columns) . It is also possible to unpack and store them in separate variables

```
data.shape
```

```
(6335, 3)
```

Figure 6.2:Data shape

6.3VALUE_COUNT():

The value_counts() method returns a series containing the counts of unique values. This means,for any column in a dataframe,this method returns the count of unique entries in that column.

```
In [6]: data['label'].value_counts()

Out[6]: REAL    3171
        FAKE    3164
        Name: label, dtype: int64
```

Figure 6.3:value_counts

6.4NO NULL VALUES IN THE DATASET:

```
In [7]: data.isnull().sum()   #no null values in the dataset

Out[7]: title    0
        text     0
        label    0
        dtype: int64
```

Figure 6.4: No Null Values In Dataset.

6.4.1NO DUPLICATED VALUES IN THE DATASET:

```
In [8]: data.duplicated()   #no duplicated values in the dataset

Out[8]: 0       False
        1       False
        2       False
        3       False
        4       False
                ...
        6330    False
        6331    False
        6332    False
        6333    False
        6334    False
        Length: 6335, dtype: bool
```

Figure 6.4.1: no duplicate values.

6.5 LABELENCODER TO LABLE COLUMN:

LabelEncoder() , fit_transform(y) fits the label encoder, or assigns labels, for a single column and transforms the values in that column to the correct labels. Use pandas. DataFrame.

```
In [9]: #Apply labelEncoder to label column

        from sklearn.preprocessing import LabelEncoder
        le=LabelEncoder()

In [10]: data.label=le.fit_transform(data.label)
         data.head()   # 0-->FAKE , 1-->REAL
```

Out[10]:

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

Figure 6.5 :Label encoder to lable column

6.6 SPLITTING THE DATA INTO TRAIN AND TEST DATA:

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets:for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

```
In [11]: #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)

In [12]: print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)

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

Figure 6.6:Splitting The Data Into Train And Test Data

6.7 COUNTVECTORIZER:

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. ... Call the fit() function in order to learn a vocabulary from one or more documents.

```
In [13]: #CountVectorizer
         from sklearn.feature_extraction.text import CountVectorizer
         #creating an object for CountVectorizer
         cvect=CountVectorizer()

In [14]: # generate the word counts for the words in the documents
         word_count_vect = cvect.fit(X_train)

         #to get feature names
         word_count_vect.get_feature_names()

Out[14]: ['00',
          '000',
          '0000',
          '000000031',
          '00000031',
          '000035',
          '00006',
          '0001',
          '0001pt',
          '000billion',
          '000ft',
          '000km',
          '000x',
          '001',
          '0011',
          '003',
          '004',
          '005',
          '005s',
```

Figure 6.7: Countvectorizer

```
In [15]: doc_array=word_count_vect.transform(X_train)
         doc_array

Out[15]: <5068x61502 sparse matrix of type '<class 'numpy.int64'>'
                with 1732321 stored elements in Compressed Sparse Row format>

In [16]: #converting to an array
         doc_array=word_count_vect.transform(X_train).toarray()
         doc_array

Out[16]: array([[ 0, 13,  0, ...,  0,  0,  0],
                [ 0,  0,  0, ...,  0,  0,  0],
                [ 0,  0,  0, ...,  0,  0,  0],
                ...,
                [ 0,  0,  0, ...,  0,  0,  0],
                [ 0,  0,  0, ...,  0,  0,  0],
                [ 0,  0,  0, ...,  0,  0,  0]], dtype=int64)
```

Figure 6.7.1: Converting To An Array

```
In [17]: #converting to a dataframe
         feature_matrix=pd.DataFrame(doc_array,columns=word_count_vect.get_feature_names())
         feature_matrix

Out[17]:
```

	00	000	0000	000000031	00000031	000035	00006	0001	0001pt	000billion	...	תאמצנה	תוצאה	תחל	תיירות	תנותק	תעודת	תתרכז	القادمون	عربي	ยงade
0	0	13	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5063	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5064	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5065	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5066	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5067	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5068 rows × 61502 columns

Figure 6.7.2 : Converting To An Dataframe

```
In [18]: #applying fit_transform to the x_train

         X_train_transformed=cvect.fit_transform(X_train)
         X_train_transformed

Out[18]: <5068x61502 sparse matrix of type '<class 'numpy.int64'>'
                with 1732321 stored elements in Compressed Sparse Row format>

In [19]: #applying transform method to x_test

         X_test_transformed=cvect.transform(X_test)
         X_test_transformed

Out[19]: <1267x61502 sparse matrix of type '<class 'numpy.int64'>'
                with 419437 stored elements in Compressed Sparse Row format>
```

Figure 6.7.3: Applying Fit_Transform To The X_Train

6.8 NAIVVE BAYES ALGORITHM:

```
In [21]: #apply naivve bayes algorithm

         from sklearn.naive_bayes import BernoulliNB

         #creating an object
         BernNB=BernoulliNB()

In [22]: #applying algorithm to data

         BernNB.fit(X_train_transformed,y_train)

Out[22]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Figure 6.8 : Applying Naïve Bayes Algorithm

6.9 PREDICTING ON TRAIN DATA:

```
In [23]: #prediction on train data

         y_train_pred=BernNB.predict(X_train_transformed)
```

```
In [24]: #compare the actual values(y_train) with predicted values(y_train_pred1)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_train,y_train_pred)

Out[24]: array([[2324,  189],
                [ 595, 1960]], dtype=int64)
```

```
In [25]: #visualizing confusion matrix
         sns.heatmap(confusion_matrix(y_train,y_train_pred),annot=True,fmt='3.0f')

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x16635e779c8>
```

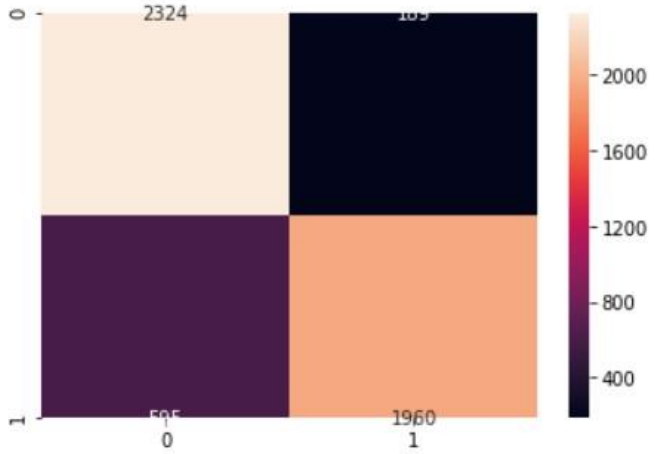


Figure 6.9: Prediction On Train Data

```
In [26]: 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

Figure 6.9.1: Prediction On Train Data Classification Report

6.10 PREDICTION ON TEST DATA:

```
In [27]: #prediction on test data
         #syntax: objname.predict(Inputvallues)
         y_test_pred=BernNB.predict(X_test_transformed)
```

```
In [28]: #compare the actual values(y_test) with predicted values(y_test_pred1)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_test,y_test_pred)

Out[28]: array([[563,  88],
                [129, 487]], dtype=int64)
```

```
In [29]: #visualizing confusion matrix
         sns.heatmap(confusion_matrix(y_test,y_test_pred),annot=True,fmt='3.0f')

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x16638a74948>
```

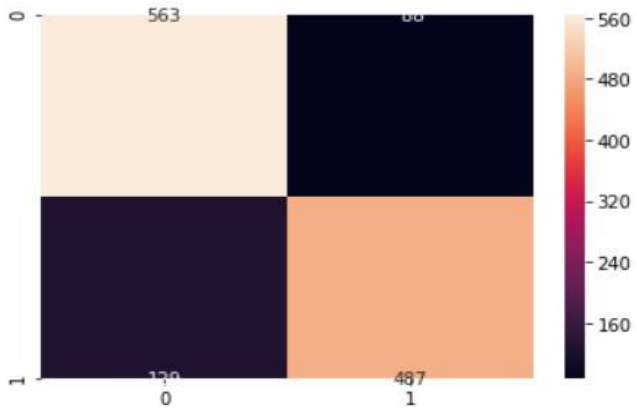


Figure 6.10 : Prediction On Test Data

```
In [30]: 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

Figure 6.10.1: Prediction On Test Data Claasification Report

6.11 TFIDF Vectorizer:

```
In [31]: #TFIDF Vectorizer
         from sklearn.feature_extraction.text import TfidfVectorizer
         # initialize an object for tfidf vectorizer
         tfidf=TfidfVectorizer()

In [32]: # apply the fit_transform method on x_train ie. the input train data

         X_train_tfidf=tfidf.fit_transform(X_train)
         X_train_tfidf

Out[32]: <5068x61502 sparse matrix of type '<class 'numpy.float64'>'
                with 1732321 stored elements in Compressed Sparse Row format>

In [33]: # apply the transform method on x_test ie. the input test data

         X_test_tfidf=tfidf.transform(X_test)
         X_test_tfidf

Out[33]: <1267x61502 sparse matrix of type '<class 'numpy.float64'>'
                with 419437 stored elements in Compressed Sparse Row format>
```

Figure6.11 : Tfidf Vectorizor

```
In [34]: #position of words in sparse matrix
         tfidf.vocabulary_
```

```
Out[34]: {'october': 38565,
          '31': 958,
          '2016': 671,
          'fort': 21842,
          'russ': 47312,
          'aleksandr': 3199,
          'khrolenko': 30528,
          'ia': 26845,
          'analytics': 3736,
          'translated': 55571,
          'by': 9031,
          'arnoldski': 4646,
          'hardly': 24908,
          'day': 14413,
          'goes': 23533,
          'without': 59980,
          'foreign': 21721,
          'media': 34553,
          'circulating': 10923,
          'the': 54430
```

```
In [35]: #IDF of the terms
         tfidf.idf_
```

```
Out[35]: array([5.57965512, 2.63420451, 8.83775166, ..., 8.83775166, 8.43228655,
                8.83775166])
```

Figure 6.11.1: Tfidf Vocabulary

```
In [36]: #apply naivve bayes algorithm

         from sklearn.naive_bayes import BernoulliNB

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

```
In [37]: #applying algorithm to data

         tfidf_BernNB.fit(X_train_tfidf,y_train)
```

```
Out[37]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Figure 6.11.2 :Apply Naivve Bayes Algorithm

6.12 PREDICTION ON TRAIN DATA:

```
In [38]: #prediction on train data

         y_train_pred1=tfidf_BernNB.predict(X_train_tfidf)
```

```
In [39]: #compare the actual values(y_train) with predicted values(y_train_pred)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_train,y_train_pred1)
```

```
Out[39]: array([[2324,  189],
                [ 595, 1960]], dtype=int64)
```

```
In [40]: #visualizing confusion matrix
         sns.heatmap(confusion_matrix(y_train,y_train_pred1),annot=True,fmt='3.0f')
```

```
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1663ae52608>
```

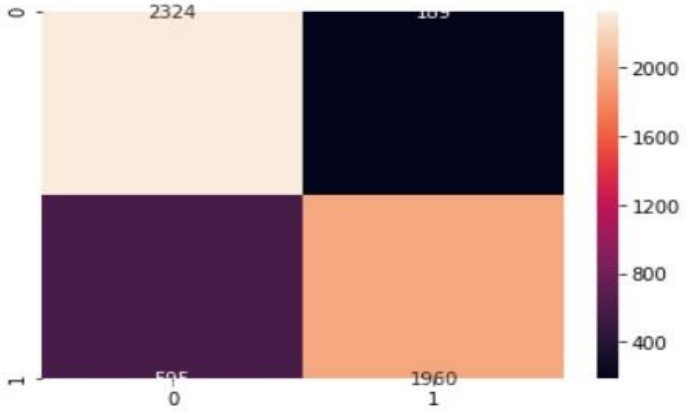


Figure 6.12: Prediction On Train Data

```
In [41]: 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
```

Figure 6.12.1: Prediction On Train Data Classification Report

6.13 prediction on test data:

```
In [42]: #prediction on test data

         y_test_pred1=tfidf_BernNB.predict(X_test_tfidf)
```

```
In [43]: #compare the actual values(y_test) with predicted values(y_test_pred)
         from sklearn.metrics import confusion_matrix, classification_report
         confusion_matrix(y_test,y_test_pred1)

Out[43]: array([[563,  88],
                [129, 487]], dtype=int64)
```

```
In [44]: #visualizing confusion matrix
         sns.heatmap(confusion_matrix(y_test,y_test_pred1),annot=True,fmt='3.0f')

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x1663aedf548>
```

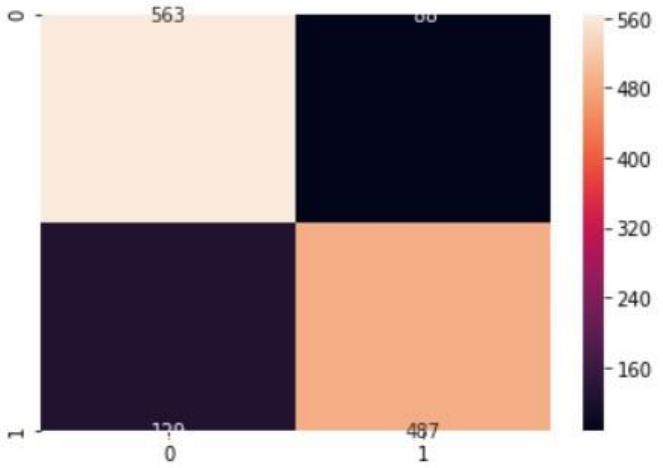


Figure 6.13: Prediction On Test Data

```
In [45]: 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

Figure 6.13.1: Prediction On Test Data Classification Report

CONCLUSION

Therefore by using naïve Bayes theorem we can conclude that any news from a large or small dataset can be classified as fake or real news by matching it with the previous dataset values in less time which in turn helps the users to believe in a particular news.

REFERENCES

- https://www.kaggle.com/semakulapaul/cereals-datase t

- https://medium.com/code-heroku/introduction-to-exploratory-dataanalysis-eda-c0257f888676

- https://en.wikipedia.org/wiki/Machine_learning

- https://www.edureka.co/blog/data-science-applications/