**Department of Computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**



***Declaration***

*We hereby declare that the work which is being presented in the Mini Project “****Fake News Detection and Analysis System”,****in partial fulfillment of the requirements for Mini-Project LAB, is an authentic record of our own work carried under the supervision of* ***Mr. Pawan KumarVerma, Assistant Professor, GLA University, Mathura****.*

**Name of Students with signature**



**Department of Computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**

**CERTIFICATE**

*This is to certify that the project entitled* ***“Fake News Detection and Analysis system”*** *carried out in Mini Project – ILab is a bonafide work done by* ***Rishabh Sengar (161500451), Ram Kamra (161500437) ,Priyank Kaushik (161500414) and Prashant Raghav (161500401)****and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).*

**Signature of Supervisor:**

**Name of Supervisor:**

**Date:**

**ACKNOWLEDGEMENT**

*It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.*

*Our heartiest thanks to* ***Dr. (Prof). Anand Singh Jalal,*** *Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.*

*We owe special debt of gratitude to* ***Mr****.* ***Pawan Kumar Verma,*** *Assistant Professor Department of CEA, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.*

*We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.*

Rishabh Sengar

Priyank Kaushik

Ram Kamra

Prashant Raghav

**Abstract**

Fake news, one of the biggest new-age problems has the potential to mold opinions and influence decisions. The explosion of fake news on social media and Internet is

Misleading people to an extent which needs to be stopped. The existing systems are inefficient in giving a accurate rating for any given news claim. Also, the restrictions on input and category of news make it less varied.This project proposes a system that classifies unreliable news into different categories. This system aims to use various NLP and Machine Learning techniquesto help achieve maximum accuracy. Along with this, we have tried to generate sentiments of the news so that we analyze the opinion provided by the news.

Since online news can be collected from different sources, manually determining the truthfulness of news is a challenging task, usually requiring annotators with domain expertise who performs a careful analysis of claims and additional evidence, context, and reports from authoritative sources. Existing public datasets of fake news are rather limited due to these challenges. To facilitate the research for fake news detection, this survey provides a usable datasets, named [**Fake News Net**](https://github.com/KaiDMML/FakeNewsNet)**, ISOT Fake News Dataset** and two other datasets picked from kaggle and github respectively which includes news content and social context features with reliable ground truth fake news labels and one dataset **Getting Real about Fake News,** which includes news content labeled of "bs" and "junksci", etc. do not constitute Truth.

Along with the above task performed we have created a front-end which should be interactive and well turned-out. The GUI should be capable of given the output information properly to the user so we made it using DASH.

**Table of Contents**

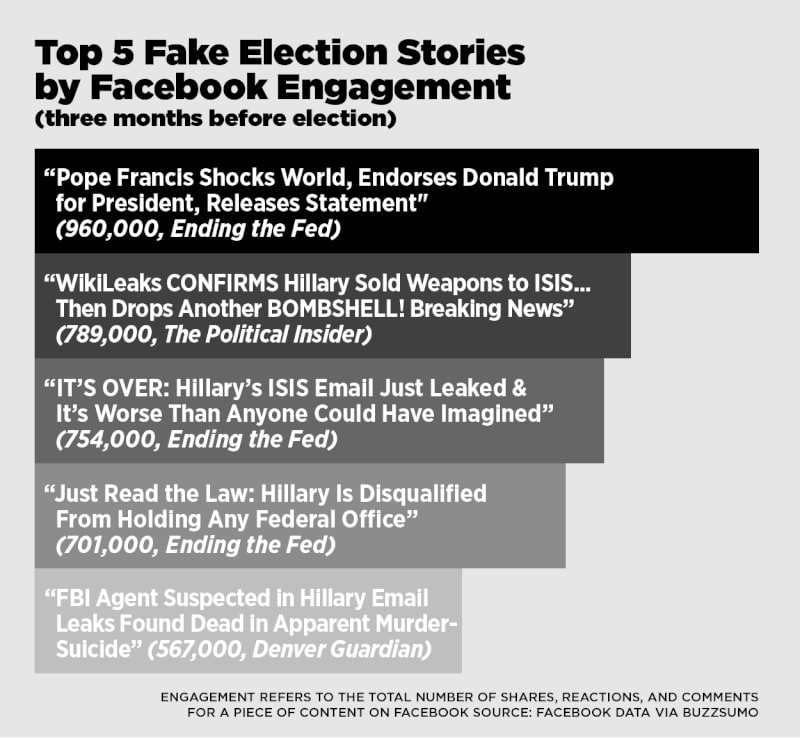
|  |  |
| --- | --- |
| Declaration | ii |
| Certificate | iii |
| Acknowledgments | iv |
| Abstract | v |
| Table of Contents | vi |
| 1. **Introduction**(This chapter must describe introduction about your project) | **1** |
| 1.1 Motivation and Overview | 1 |
| 1.3 Objective | 2 |
| 1. **Software Requirement Analysis** | **7** |
| 2.1Define the problem | 7 |
| 2.2 Define the modules and their functionalities (SRS) ……………..………………………………… | 7 |
| 1. **Software Design** | **16** |
| * 1. Use Case Diagram | 16 |
| * 1. Class\ Object Diagram | 17 |
| * 1. Sequence Diagram |  |
| 1. **Testing** | **22** |
| 4.1 Black- Box Testing | 22 |
| 4.2 White- Box Testing | 22 |
| 1. **Implementation and User Interface** | **27** |
| 5.1 Output | 27 |
| **References/Bibliography** | **28** |
| 1. **Appendices** |  |

**1. Introduction**

Fake news, one of the biggest new-age problems has the potential to mold opinions and influence decisions However, the dawn of the social media age which can be approximated by the start of the 20th century has developed the generation and circulation of fake news many folds. Fake news can be simply explained as a piece of article which is usually written for personal, economic or political gains. Detection of such false news articles is possible by using various NLP techniques, Machine learning, and Artificial intelligence.

**VIRAL FAKE NEWS**

* In 2016, the Prime Minister of India, Mr. Narendra Modi declared that most of the cash that people possessed had become worthless and in the span of one month all this old currency had to be deposited in the banks. This led to a chain reaction of a series of fake news being published used mainly for click bait and political gains. News about the new paper bills having a GPS tracker or the daily limit of the amount that can be deposited in banks has increased, were spreading like wildfire.
* The image of two men on a motorcycle pulling up to a group of children with a message that people were kidnapping children in the area - this single message-fuel led rumor has led to the killing of at least 25 people in other parts of India since June 2018.
* In winter, parents of more than 240,000 children in Kannur began refusing the combined measles, mumps and rubella (MMR) vaccine for their children after a fake message saying that the vaccine harmed children went viral. An immunization drive stalled for nearly two months.
* This is not limited to indian Subcontinent. In the final months of the US presidential campaign 2016, the fake election news stories on Facebook. Facebook skyrocketed and surpassed that of the content from major news outlets.



\* Source –Buzz Feed News (2016)

This is just a small instance of how the spread of false news can impact a much greater audience than it may seem. This PROJECT provides an insight into the procedure of detecting fake news and its results. In order to reach a conclusion on the authenticity of the news article we first take the text article, analyze the data and then use various classification algorithms to classify the news as legitimate or fake.

**Some Existing systems**

**BS Detector,** works by searching through web page references of links which have already been flag get unreliable in their database.

**PolitiFact,** is a fact-checking US-based website used by editors and writers which gives the credibility of claims by US officials involved American politics. This system places judgment in the form of Truth-O-Meter which is a measure of the accuracy of a statement. The drawback of this system is that human intervention is required.

**Fake News Detector** was a feature added by Flock-a new generation messaging and collaborative platform. Whenever links are being sent to each while chatting, FND algorithm gets activated. It checks the content of links to their databases of websites computed according to rankings. The drawback of this system is that their database for fact checking is less in number and chances of hoaxes still not being determined are high.

By the help of the above mentioned systems, by some extent we were able identify misleading news. But they require human intervention and domain dependent which makes it limited in some sense.

The purpose of this project is to propose a new model for fake news detection which is using Stance (Attitude) Detection and TF-IDF/BOW method for analyzing the data which is taken from various datasets of fake and genuine news. Various Machine Learning Techniques have in used for classifying the output Real and Fake.

**2. Software Requirement Analysis**

Different modules in our project are :-

**Natural Language Processing:**

NLP is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data in a smart and efficient manner. By utilizing NLP and its components, one can organize the massive chunks of text data, perform numerous automated tasks and solve a wide range of problems such as – automatic summarization, machine translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation etc.

**Machine Learning**

Machine learning (ML) is the study of algorithms and mathematical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

**Web Scraping :**

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list.

**Data Analysis**

Data Analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data Analysis can easily be performed with the help of different libraries like pandas, matplotlib, seaborn etc.

**Pandas** is a **Python** package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in **Python**. **Pandas** is a software library written for the **Python** programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. **pandas** is free software released under the three-clause BSD license.

**Matplotlib** is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, or GTK+.

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

**Dash**

Open-source – Dash lets you wrap a GUI around that analytical code, without leaving the familiarity of Python. It is a data visualization library for creating interactive, beautiful, embeddable charts in Python — without JavaScript?

Dash lets you wrap a GUI around that analytical code, without leaving the familiarity of Python. Explore your data with rich, interactive drop-down menus, sliders, and other components, all in the web browser. Update one component, and you’ll see the changes reflected in your other data views. Dash handles all the JavaScript and React for you, so can transform your analytical Python code into a powerful, reactive, web-based application.

**3. Software Design**

**3.1 USE CASE Diagram**

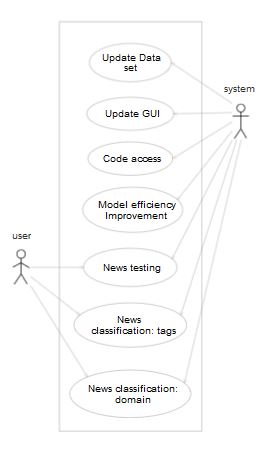
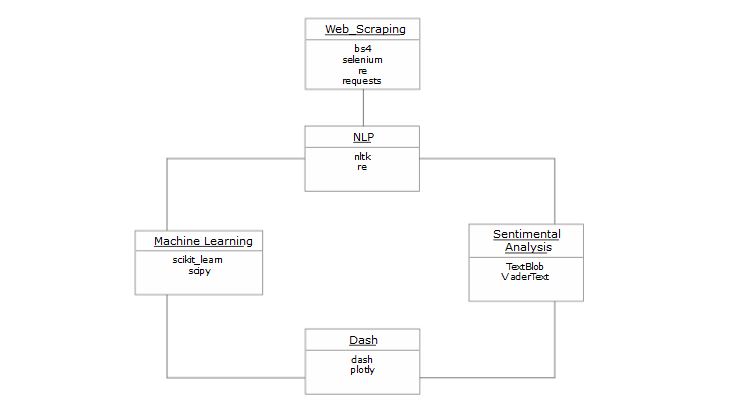
****

Fig1: Use case diagram

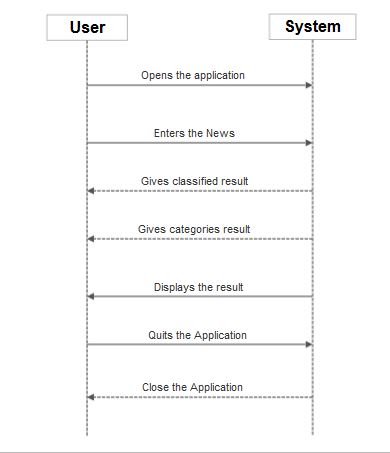
|  |  |
| --- | --- |
| **USE CASE** | **DESCRIPTION** |
| Update dataset | Admin can update dataset. |
| Update GUI | Admin can change the output patterns. |
| Code access | Admin can update/modify the code as per the requirements. |
| Model Efficiency Improvement | Admin can increase the efficiency by updating the models |
| News Testing | News provided by the user is tested by the model and gives the result as fake/real. |
| News Classification | User gets the classified news as provided. |

**3.2Class / Object Diagram**

****

|  |  |
| --- | --- |
| **Class** | **Description** |
| Web Scrapping | Data extraction from different sources to prepare dataset |
| NLP | Manipulating text data. |
| Machine learning | Data is preprocessed and model is trained on it to classify the news. |
| Sentimental Analysis | Identifying and categorizing opinions expressed in a piece of text. |

**3.3 Sequence Diagram**

****

**4. Testing**

“Testing is the process of executing a program with the intent of finding errors.”

Although software testing is itself an expensive activity, yet launching of software without testing may lead to cost potentially much higher than that of testing, specially in systems where human safety is involved.

In the software life cycle the earlier the errors are discovered and removed, the lower is the cost of their removal.

Test and Test case terms are used interchangeably. In practice, both are same and are treated as synonyms. Test case describes an input description and an expected output description. The set of test cases is called a test suite. Hence any combination of test cases may generate a test suite.

**Verification** is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

**Validation** is the process of evaluating a system or component during or at the end of development process to determine whether it satisfies the specified requirements.

**Testing= Verification + Validation**

The term Acceptance Testingis used when the software is developed for a specific customer. A series of tests are conducted to enable the customer to validate all requirements. These tests are conducted by the end user / customer and may range from adhoc tests to well planned systematic series of tests.

The terms alpha and beta testing are used when the software is developed as a product for anonymous customers.

Alpha Testsare conducted at the developer’s site by some potential customers. These tests are conducted in a controlled environment. Alpha testing may be started when formal testing process is near completion.

Beta Testsare conducted by the customers / end users at their sites. Unlike alpha testing, developer is not present here. Beta testing is conducted in a real environment that cannot be controlled by the developer.

**Functional Testing**

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These **tests** can be functional or non-functional, though usually functional.

Fig: Black box testing

**Structural Testing**

A complementary approach to functional testing is called structural / white box testing. It permits us to examine the internal structure of the program.

**Path Testing**

Path testing is the name given to a group of test techniques based on judiciously selecting a set of test paths through the program. If the set of paths is properly chosen, then it means that we have achieved some measure of test thoroughness.

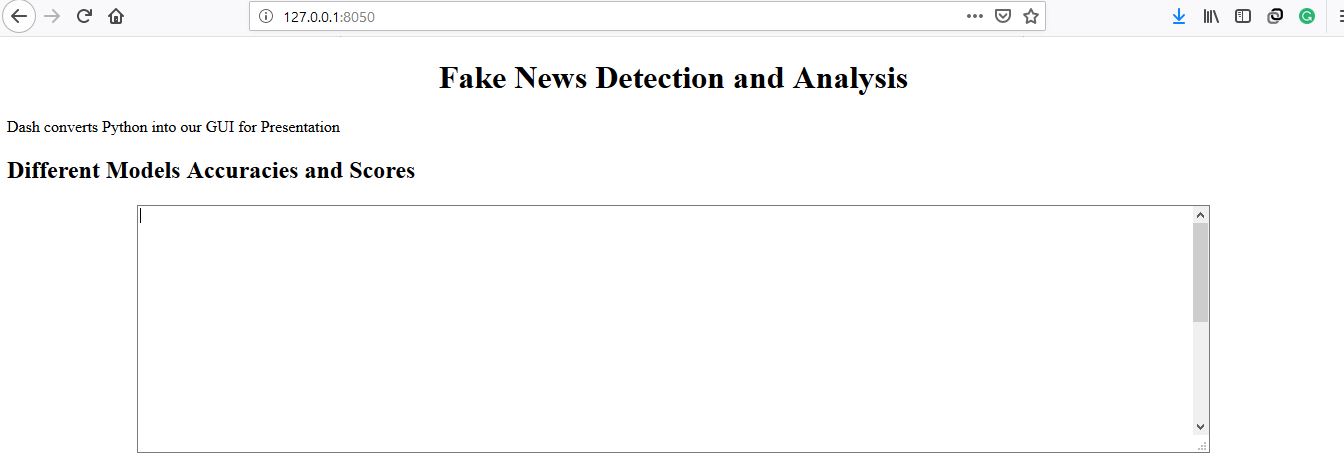
This type of testing involves:

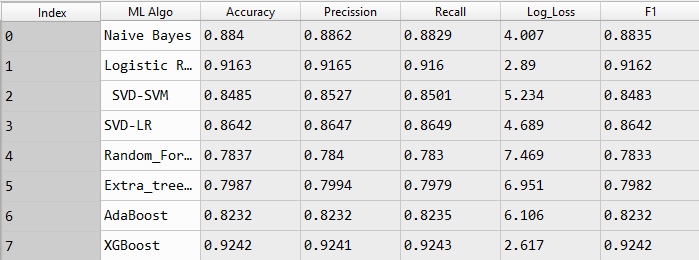
1. generating a set of paths that will cover every branch in the program.
2. finding a set of test cases that will execute every path in the set of program paths.

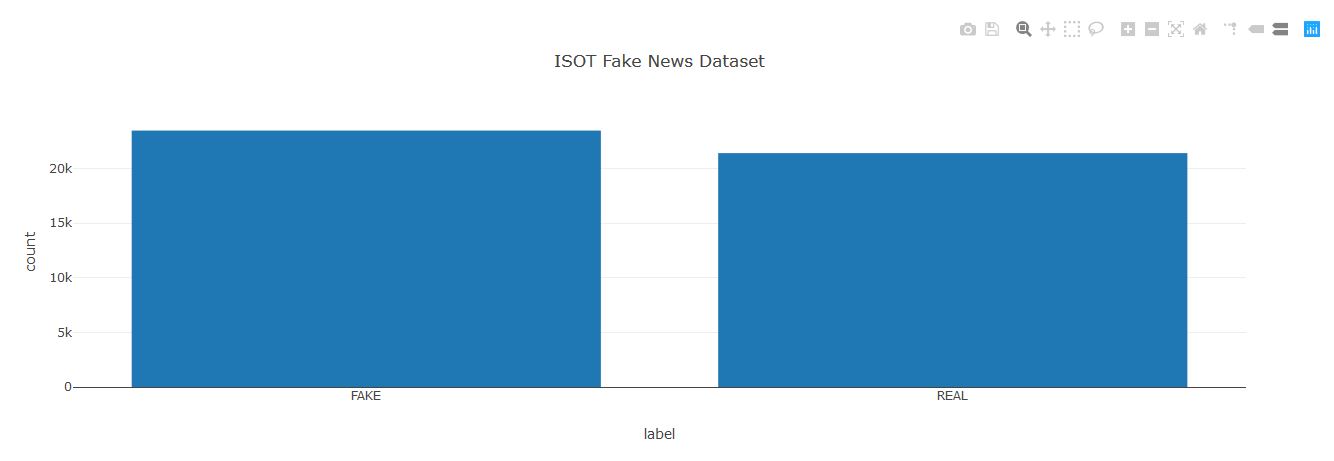
**WHITE BOX TESTING**(also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure / design / implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

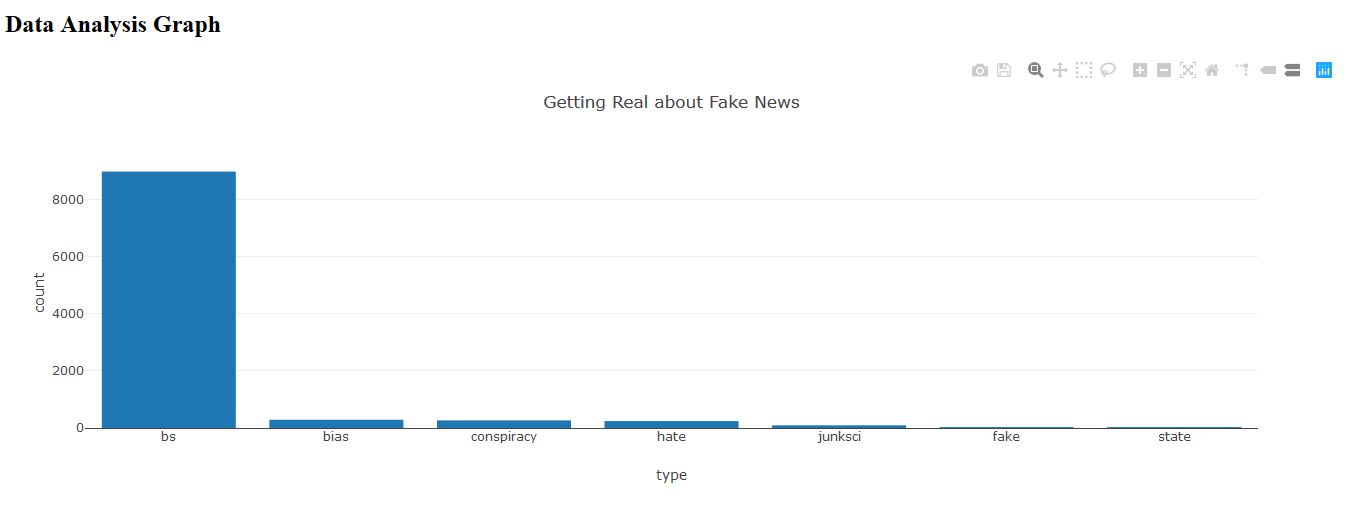
**5 . Implementation and User Interface**

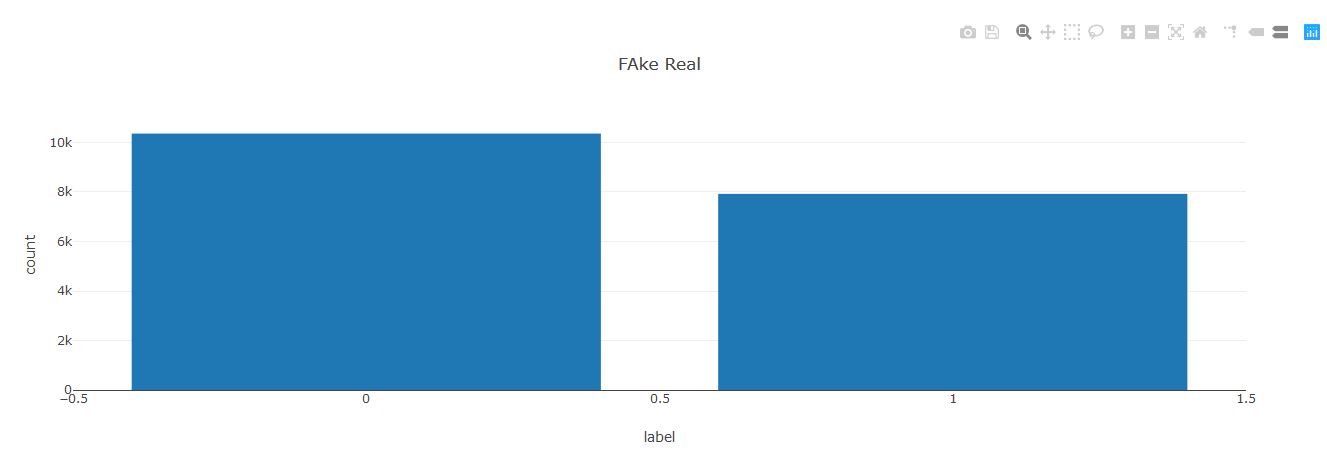
* 1. **OUTPUT :-**

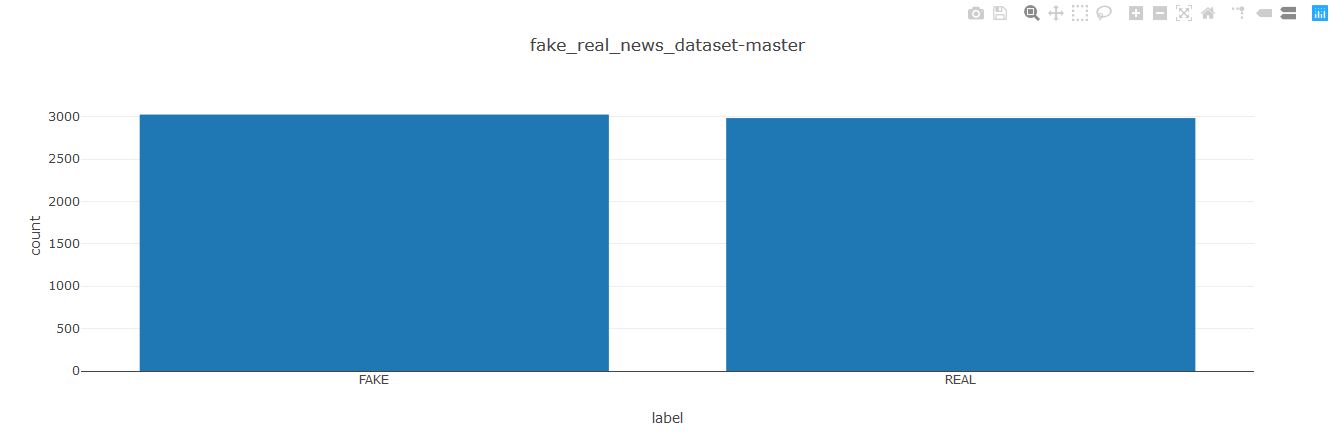
****

****

****

****

****

****

1. **References/Bibliography**

* SuperDataScience Team : - https://www.superdatascience.com/
* NLKT : - <https://www.nltk.org/>
* Scipy Lecture Notes : - <http://www.scipy-lectures.org/>
* Dataset links :

<https://github.com/GeorgeMcIntire/fake_real_news_dataset>

<https://www.kaggle.com/c/fake-news/data>

<https://www.kaggle.com/mrisdal/fake-news#fake.csv>

* Naïve Bayes :-<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
* Bag of words Model :-<https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428>
* TF-IDF Model :-

<https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

* Ensemble Methods :-<https://scikit-learn.org/stable/modules/ensemble.html>
* Scraping :-<https://medium.freecodecamp.org/how-to-scrape-websites-with-python-and-beautifulsoup-5946935d93fe>
* Dash :-<https://plot.ly/products/dash/>
* TextBlob :-<https://textblob.readthedocs.io/en/dev/>
* Kaggle :-<https://www.kaggle.com/>
* ResearchGate :- https://www.researchgate.net/

1. **Appendices**

**Code Template 1.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Importing the libraries

Import numpy as np

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('fake.csv')

# A quick look at the data

dataset.info()

# finding the dimensions

print(dataset.shape)

# Combining Both title and text

dataset['total']=dataset['author']+' '+dataset['title']+' '+dataset['text']

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

In this template we are importing our dataset using pandas library and seeing its dimensions and giving a quick look to it. After that we are combining our ‘title’ and ‘text’ column so that we can work on combined features.

**Code Template 2.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# TRAINING SET

# Cleaning the texts

import re

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

corpus = []

for i in range(12999):

# REMOVE PUNTUATIONS AND ANY CHARACTER OTHER THAN ALPHABET

review = re.sub('[^a-zA-Z]', ' ', str(dataset['total'][i]))

review = review.lower()

review = review.split()

# Stemming object

ps = PorterStemmer()

# Stemming + removing stopwords

review = [ps.stem(word) for word in review if not word in \

set(stopwords.words('english'))]

review = ' '.join(review)

corpus.append(review)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

In this template we are doing task from cleaning to creating corpus of the given text with the help of various libraries.

Firstly we are downloading the stop words. Then we are removing any other character other than alphabets, then we are transforming them in to lower case words and then splitting them all.

Then we are taking only the root words and removing the stop words. After doing this task iteratively we are storing them all in the list called ‘corpus’.

**Code Template 2.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Creating BAG OF WORDS MODEL:-

From sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features = 11000)

X = cv.fit\_transform(corpus).toarray()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Creating TF-IDF MODEL :

fromsklearn.feature\_extraction.text import TfidfVectorizer

tf = TfidfVectorizer(analyzer='word', ngram\_range=(1,3), min\_df = 2)

X = tf.fit\_transform(corpus)

feature\_names = tf.get\_feature\_names()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Y = dataset.iloc[:,19].values

# Encoding categorical data

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

Y = labelencoder.fit\_transform(Y)

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.20,random\_state = 0)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation:-**

Here, Firstly we are creating two models

**BAG OF WORDS MODEL:-**

**Bag of words** model is one of a series of techniques from a field of computer science known as **Natural Language Processing** or **NLP** to extract features from text. The way it does this is by counting the frequency of words in a document. A document can be defined as you need, it can be a single sentence or all Wikipedia. The output of the bag of words model is a **frequency vector**.

**TF-IDF MODEL :-**

TF-IDF, which stands for **term frequency — inverse document frequency,** is a scoring measure widely used in information retrieval (IR) or summarization. TF-IDF is intended to reflect **how relevant a term is in a given document.**

The intuition behind it is that if a word occurs *multiple times in a document*, we should boost its relevance as it should be more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a document but also *along many other documents*, maybe it is because this word is just a frequent word; not because it was relevant or meaningful (IDF).

**Secondly,**

We are labelencoding our categorical data and the splitting our dataset into test and train

Sets with the help of test\_train\_split module present in sklearn package.

**Code Template 3.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.metrics import accuracy\_score

from sklearn.metrics import log\_loss

# PREDICTION : Accuracy Score

defacc\_score(y\_test, y\_pred ) :

acc = accuracy\_score(y\_test, y\_pred)

return print("Final Accuracy: %0.04f" %(acc))

# PREDICTION : Log-Loss

def log\_loss(y\_test, y\_pred ) :

acc = log\_loss(y\_test, y\_pred)

return print("Final Accuracy: %0.04f" %(acc))

# acc\_score(y\_test, y\_pred )

# log\_loss(y\_test, y\_pred )

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

The metric used to evaluate our Machine Learning Algorithms in this project is multi class logarithmic loss and accuracy score, which is implemented by the log\_loss and accuracy\_score function present in the scikit-learn library.

Logarithmic loss (or logloss) is a performance metric for evaluating the predictions of probabilities of membership to a given class. The scalar probability between 0 and 1 can be seen as a measure of confidence for a prediction by an algorithm. Predictions that are correct or incorrect are rewarded or punished proportionally to the confidence of the prediction. Smaller logloss is better with 0 representing a perfect logloss.

**Code Template 4.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Fitting NAIVE BAYES to the Training set

from sklearn.naive\_bayes import MultinomialNB

classifier\_1 = MultinomialNB()

classifier\_1.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier\_1.predict(X\_test)

# Accuracy of the model

acc\_score(y\_test, y\_pred )

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

In this template we are applying Naïve Bayes algorithm.

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

**Code Template 5.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Fitting LOGISTIC REGRESSION to the Training set

from sklearn.linear\_model import LogisticRegression

classifier\_2 = LogisticRegression(random\_state = 0)

classifier\_2.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier\_2.predict(X\_test)

# Accuracy of the model

acc\_score(y\_test, y\_pred )

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

In this template we are applying Logistic Regressing algorithm.

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, etc.) or 0 (FALSE, failure, etc.).

**Code Template 6.**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.ensemble import RandomForestClassifier

Rando= RandomForestClassifier(n\_estimators=5)

Rando.fit(X\_train, y\_train)

print('Accuracy of Random Forest classifier on test set: %0.04f'

%(Rando.score(X\_test, y\_test)))

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.ensemble import ExtraTreesClassifier

Extr = ExtraTreesClassifier(n\_estimators=5,n\_jobs=4)

Extr.fit(X\_train, y\_train)

print('Accuracy of Extratrees classifier on test set: %0.04f'

%(Extr.score(X\_test, y\_test)))

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import AdaBoostClassifier

Adab= AdaBoostClassifier(DecisionTreeClassifier(max\_depth=3),n\_estimators=5)

Adab.fit(X\_train, y\_train)

print('Accuracy of AdaBooost classifier on test set: %0.04f'

%(Adab.score(X\_test, y\_test)))

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation :**

In this template we are using various ENSEMBLE LEARNING METHODS.

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

The [sklearn.ensemble](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble) module includes two averaging algorithms(Bagging Methods) based on randomized [decision trees](https://scikit-learn.org/stable/modules/tree.html#tree): the Random Forest algorithm and the Extra-Trees method.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

In extremely randomized trees, randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

The module [sklearn.ensemble](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble) includes the popular boosting algorithm AdaBoost, introduced in 1995 by Freund and Schapire.

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

**Code Template 7.**

#Importing web scraping libraries

import requests

import html.parser

from bs4 import BeautifulSoup

import re

#get the html code using site URL

url='https://www.thehindu.com/'

page=requests.get(url)

page.status\_code

page.text

#Parsering the html code

soup=BeautifulSoup(page.text,'html.parser')

#print(soup.prettify())

#Fetching all the paragraphs tags text in a list (content)

content=list()

length=len(soup.find\_all('p'))

fori in range(1,length):

content.append(soup.find\_all('p')[i].get\_text())

''.join(content)

#Fetching all the anchor tags text in a list (content1)

content1=list()

length1=len(soup.find\_all('a'))

fori in range(60,length1-168):

content1.append(soup.find\_all('a')[i].get\_text())

''.join(content1)

#Fetching all the 'href' links from (content1) list to (links) list

links = []

content2=[]

for link in soup.findAll('a'):

links.append(link.get('href'))

#Now form (links) list extracting links with .ece extension in (content2) list

length2=len(links)

for j in range(0,length2):

if links[j]==None:

pass

elifre.findall(r'.ece$',links[j]):

content2.append(links[j])

#printing the .ece links list

print('\_\_\_\_\_\_LINKS\_1\_\_\_\_\_\_',content2)

print('',end='\n')

print('\_\_\_\_\_\_CONTS\_\_\_\_\_\_')

#Assigning all links to fetch html code, parsing it and appending the

#content of article in a list (cont)

for i in range(0,len(content2)):

url1=content2[i]

page=requests.get(url1)

page.status\_code

page.text

soup=BeautifulSoup(page.text,'html.parser')

head=soup.find\_all('div',class\_=' ')

para=[]

for h in head:

para=h.find\_all('p')

length=len(para)

cont=[]

for b in range(0,length-5):

cont.append(para[b].get\_text())

#printing articles (cont)

print(cont)

print('',end='\n')

**Explanation:**

In it we are preparing a news dataset by extracting the Hindu newspaper site. So in the code we first importing the supporting libraries in python for this work then fetch the html code and parse it and save it in soup using Beautiful soup library. Then we extracted all the paragraphs text where we got all the anchor tags from which we extract the text of all anchor tags in a list from that anchor tags we extract all the ‘href’ or hyperlinks in another list. Now from those links we only want .ece extension links to get the articles so we extract these links into another list. This list is then used to assign each link one by one to fetch the article content(text) by fetching its webpage html code and then parsing it and finally getting the article text and appending that to another list likewise for all the links with .ece extension. Finally getting that list of articles as dataset.

**Code Template 8.**

#Data Analysis

# Importing the libraries

import pandas as pd

importseaborn as sns

%matplotlib inline

#Importing the dataset

data=pd.read\_csv("fake.csv")

#Display the first 10 rows

data.head(10)

#Display the dimensions(rows and columns)

data.shape

#Display the datatype of the columns

data.dtypes

#Display the columns of dataset

data.columns

#Displaying how much null values in a column

data.isnull().sum()

#Removing the rows which contain author,text and title as null

data1=data[pd.notnull(data["author"])]

data2=data1[pd.notnull(data1["title"])]

data3=data2[pd.notnull(data1["text"])]

#Dimensions(rows and columns) after removing null values from author,text and title

data3.shape

#Display the count and percentage of each type contain in column type

data3.type.value\_counts()

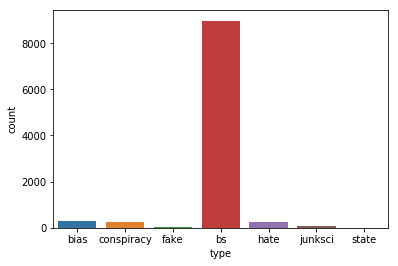
data3.type.value\_counts(normalize=True)\*100

#Description of whole dataset

data2.describe(include="all")

# Show the counts of observations in each categorical bin using bars

s=sns.countplot(x="type",data=data2)



**Explanation:**

In this template we are importing our dataset using pandas library and use pandas library methods to see the top 10 rows , shape , null values , data type , columns, remove null values from title, text and author column, show the percentage of each category in type column and a histogram to show the category in type column.

**Code Template 9.**

#sentimental analysis

# SENTIMENT ANALYSIS

from textblob import TextBlob

l=input()

#l = dataset[‘target’][1]

text=TextBlob(l)

if(text.sentiment.polarity>0 and text.sentiment.subjectivity>0.5):

print("\n General Opinion(Subjective) with positive Sentiment ")

if(text.sentiment.polarity<0 and text.sentiment.subjectivity>0.5):

print("\n General Opinion(Subjective) with Negative Sentiment ")

if(text.sentiment.polarity>0 and text.sentiment.subjectivity<0.5):

print("\n Personal Opinion(Objective) with positive Sentiment ")

if(text.sentiment.polarity<0 and text.sentiment.subjectivity<0.5):

print("\n Personal Opinion(Objective) with Negative Sentiment ")

if(text.sentiment.polarity==0):

print("\n neutral")

**EXPLANATION**

In this section we counted the number of neutral ,positive and negative sentiments with the general and personal subjectivity of the news as per input from the user using textblob.

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

from PIL import Image

file1=open('F:\\result.txt','r')

a=file1.read()

file1.close()

#Dash components

app = dash.Dash()

'''

image\_filename = 'wc\_1.png' # replace with your own image

encoded\_image = base64.b64encode(open(image\_filename, 'rb').read())

'''

app.layout = html.Div(children=[

html.H1(children='Fake News Detection and Analysis',style={'textAlign': 'center'}),

html.Div(children=[

html.P('Dash converts Python into our GUI for Presentation')

]),

html.H2('Different Models Accuracies and Scores'),

#For print of Accuracy

html.Div(children=

dcc.Textarea(

readOnly='True',

#type='text',

value=a,

rows=15,

#value="The Accuracy and score of various algorithms is :\nNAIVE BAYES - %0.4f\nLOGISTIC REGRESSION - %0.4F\nRANDOM FOREST - %0.4\nEXTRA TREE CLASSIFIER - %0.4\nADA BOOST - %0.4\nXGBOOST - %0.4" %(acc\_score\_1,acc\_score\_2,acc\_score\_rfc,acc\_score\_etc,acc\_score\_3),

style={'width': '80%'}

),

style={'textAlign': 'center'},

),

#html.Div(children=

html.H2('Data Analysis Graph'),

dcc.Graph(

id='graph',

figure={

'data': [go.Bar(

x= l1,

y= l2,

)],

'layout': go.Layout (

title='Getting Real about Fake News',

xaxis= {'title': 'type'},

yaxis= {'title': 'count'}

)

}

),

dcc.Graph(

id='graph1',

figure={

'data': [go.Bar(

x= m1,

y= m2,

)],

'layout': go.Layout (

title='FAke Real',

xaxis= {'title': 'label'},

yaxis= {'title': 'count'}

)

}

),

dcc.Graph(

id='graph2',

figure={

'data': [go.Bar(

x= n1,

y= n2,

)],

'layout': go.Layout (

title='ISOT Fake News Dataset',

xaxis= {'title': 'label'},

yaxis= {'title': 'count'}

)

}

),

dcc.Graph(

id='graph3',

figure={

'data': [go.Bar(

x= o1,

y= o2,

)],

'layout': go.Layout (

title='fake\_real\_news\_dataset-master',

xaxis= {'title': 'label'},

yaxis= {'title': 'count'}

)

}

),

html.H2('WordCloud'),

Image.open('F:\\wc\_1.png').show()

])

#For Running the server

app.run\_server()

**Explanation:**

In it first we are importing the libraries the building the layout of the GUI then giving the title of the project. After that representing the different classifications accuracy percentage and score in a table and the displaying the graphs of all four data sets graphs after doing data analysis. Then it show the wordcloud of all four data sets.