FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING (NLP)

A MAJOR PROJECT REPORT

Submitted by

RACHNA KOLA

17RH1A05A9

Under the Esteemed Guidance of

Ms. B. Sravani

Assistant Professor, Department of CSE

in partial fulfillment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade, UGC, Govt. of India

NIRF Indian Ranking, Accepted by MHRD, Govt. of India Band A (6th to 25th) National Ranking by ARIIA, MHRD, Govt. of India Approved by AICTE, ISO 9001:2015 Certified Institution AAAA+ Rated by Digital Learning Magazine, AAA+ Rated by Careers 360 Magazine 3rd Rank CSR, Platinum Rated by AICTE-CII Survey, 141 National Ranking by India Today Magazine National Ranking-Top 100 Rank band by Outlook Magazine, National Ranking-Top 100 Rank band by Times News Magazine

2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project work entitled “**FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING (NLP)**” is carried by **Rachna Kola (17RH1A05A9)** in partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad during the academic year 2020-2021.

Supervisor’s Signature	**Head of the Department, CSE**
Ms. B. Sravani	**Dr. C.V.P.R. Prasad**
Assistant Professor, Dept. of CSE	**Professor**

External Examiner

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NBA & NAAC with 'A' Grade, UGC, Govt. of India

NIRF Indian Ranking, Accepted by MHRD, Govt. of India
Band A (6th to 25th) National Ranking by ARIIA, MHRD, Govt. of India
Approved by AICTE, ISO 9001:2015 Certified Institution
AAAA+ Rated by Digital Learning Magazine, AAA+ Rated by Careers 360 Magazine
3rd Rank CSR, Platinum Rated by AICTE-CII Survey, 141 National Ranking by India Today Magazine
National Ranking-Top 100 Rank band by Outlook Magazine, National Ranking-Top 100 Rank band by Times News Magazine

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the Major Project entitled "**FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSOR (NLP)**" submitted to **Malla Reddy Engineering College for Women, affiliated to Jawaharlal Nehru Technological University, Hyderabad** (JNTUH) for the award of the **Degree of Bachelor of Technology in Computer Science and Engineering** is a result of original research work done by us.

It is declared that the technical Major Project report has not been previously submitted to any University or Institute for the award of Degree.

Submitted By

RACHNA KOLA (17RH1A05A9)

ACKNOWLEDGEMENT

I feel myself honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women** and Department of **Computer science and Engineering** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

I would like to deeply thank our Honorable Minister of Telangana State **Sri.Ch. Malla Reddy Garu,** founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

I wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha,** for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

I express my sincere gratitude to **Dr. C.V.P.R. Prasad, Head of the Department** of Computer science and Engineering for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

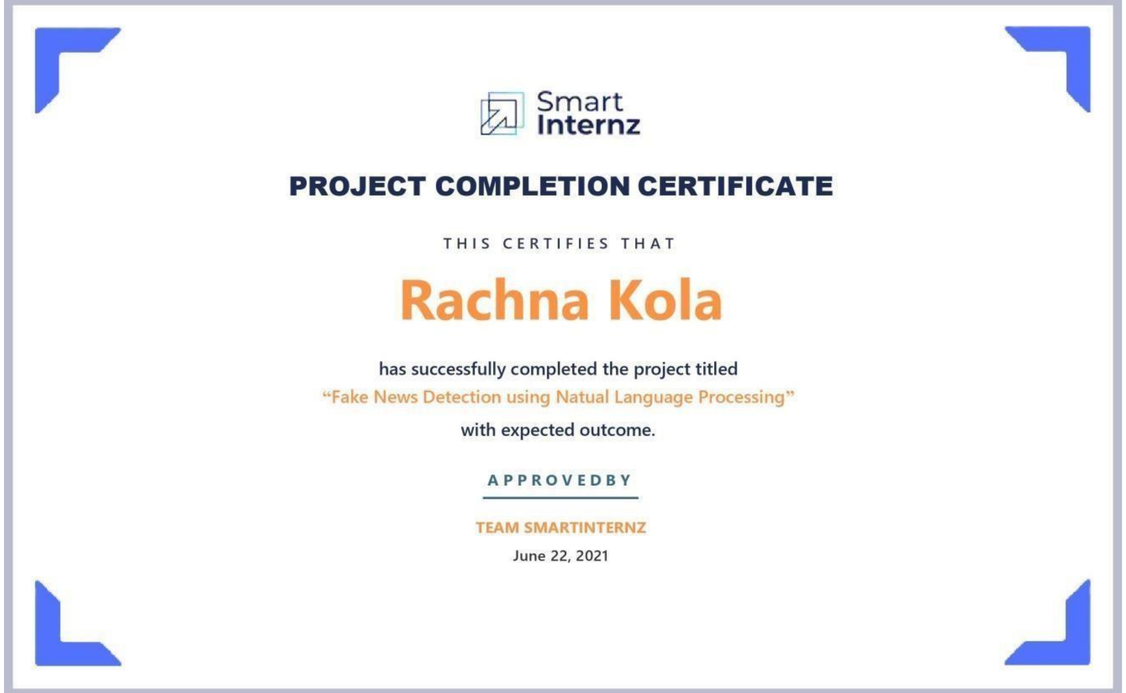
I would also like to thank our Project coordinator **Mrs. V.Naramada,** for her kind encouragement and overall guidance in viewing this program a good asset with profound gratitude.

I would like to thank our internal guide **Ms. B.Sravani,** and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

RACHNA KOLA (17RH1A05A9)

ABSTRACT

Information preciseness on Internet, especially on social media, is an increasingly important concern, but web-scale data hampers, ability to identify, evaluate and correct such data, or so called "fake news," present in these platforms. In this project, we propose a method for "fake news" detection and ways to apply it on Facebook, one of the most popular online social media platforms. This method uses Naive Bayes classification model to predict whether a post on Facebook will be labeled as REAL or FAKE. The results may be improved by applying several techniques that are discussed in the paper. Received results suggest, that fake news detection problem can be addressed with natural language processing methods. Recently, fake news has been incurring many problems to our society. As a result, many researchers have been working on identifying fake news. Most of the fake news detection systems utilize the linguistic feature of the news. However, they have difficulty in sensing highly ambiguous fake news which can be detected only after identifying meaning and latest related information. Our system receives a proposition, and searches the semantically related articles from Fact DB in order to verify whether the given proposition is true or not by comparing the proposition with the related articles in fact DB. To achieve this, we utilize a deep learning model, Bidirectional Multi-Perspective Matching for Natural Language Sentence (BiMPM), which has demonstrated a good performance for the sentence matching task. However, BiMPM has some limitations in that the longer the length of the input sentence is, the lower its performance is, and it has difficulty in making an accurate judgments when an unlearned word or relation between words appears. In order to overcome the limitations, we shall propose a new matching technique which exploits article abstraction as well as entity matching set in addition to BiMPM. In our experiment, we shall show that our system improves the whole performance for fake news detection.



Smart Internz

PROJECT COMPLETION CERTIFICATE

THIS CERTIFIES THAT

Rachna Kola

has successfully completed the project titled
"Fake News Detection using Natual Language Processing"
with expected outcome.

APPROVEDBY

TEAM SMARTINTERNZ
June 22, 2021

INDEX

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

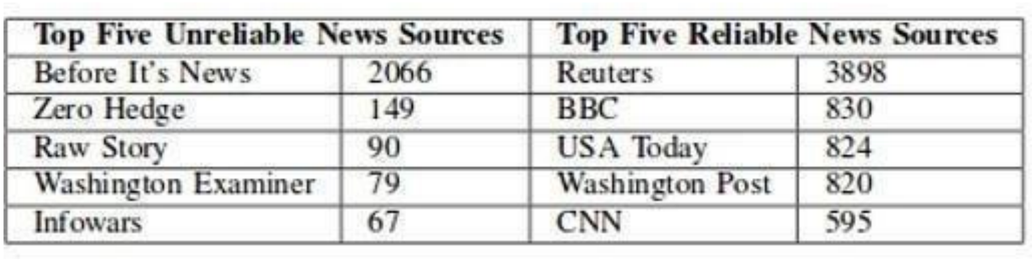
1.1 SCOPE

Nowadays, fake news has become a common trend. Even trusted media houses are known to spread fake news and are losing their credibility. So, how can we trust any news to be real or fake? In this project, I have built a classifier model that can identify news as real or fake. For this purpose, I have used data from drive, but you can use any data to build this model following the same methods. With the help of this project you can create an NLP classifier to detect whether the news is real or fake. Fake news detection topic has gained a great deal of interest from researchers around the world. When some event has occurred, many people discuss it on the web through the social networking. They search or retrieve and discuss the news events as the routine of daily life. Some types of news such as various bad events from natural phenomenal or climate are unpredictable. When the unexpected events happen there are also fake news that are broadcasted that creates confusion due to the nature of the events. There is increasing evidence that consumers have reacted absurdly to news that later proved to be fake. One recent case is the spread of novel corona virus, where fake reports spread over the Internet about the origin, nature, and behavior of the virus.The situation worsened as more people read about the fake contents online. Identifying such news online is a daunting task.Fortunately, there are a number of computational techniques that can be used to mark certain articles as fake on the basis of their textual content. Majority of these techniques use fact checking websites such as “PolitiFact” and “Snopes.” There are a number of repositories maintained by researchers that contain lists of websites that are identified as ambiguous and fake . However, the problem with these resources is that human expertise is required to identify articles/websites as fake. More importantly, the fact checking websites contain articles from particular domains such as politics and are not generalized to identify fake news articles from multiple domains such as entertainment, sports, and technology.The World Wide Web contains data in diverse formats such as documents, videos, and audios. News published online in an unstructured format (such as news, articles, videos, and audios) is relatively difficult to detect and classify as this strictly requires human expertise. However, computational techniques such as natural language processing (NLP) can be used to detect anomalies that separate a text article that is deceptive in nature from articles that are based on facts . Other techniques involve the analysis of propagation of fake news in contrast with real news.More specifically, the approach analyzes how a fake news article propagates differently on a network relative to a true article. The response that an article gets can be differentiated at a theoretical level to classify the article as real or fake.

A number of studies have primarily focused on detection and classification of fake news on social media platforms such as Facebook and Twitter. At conceptual level, fake news has been classified into different types; the knowledge is then expanded to generalize machine learning (ML) models for multiple domains.The study by Ahmed et al. included extracting linguistic features such as n-grams from textual articles and training multiple ML models including K-nearest neighbor (KNN), support vector machine (SVM), logistic regression (LR), linear support vector machine (LSVM), decision tree (DT), and stochastic gradient descent (SGD), achieving the highest accuracy (92%) with SVM and logistic regression. According to the research, as the number of increased in -grams calculated for a particular article, the overall accuracy decreased. The phenomenon has been observed for learning models that are used for classification. Shu et al. achieved better accuracies with different models by combining textual features with auxiliary information such as user social engagements on social media. The authors also discussed the social and psychological theories and how they can be used to detect false information online. Further, the authors discussed different data mining algorithms for model constructions and techniques shared for features extraction. These models are based on knowledge such as writing style, and social context such as stance and propagation.A different approach is followed by Wang .The author used textual features and metadata for training various ML models.

1.2 EXISTING SYSTEM

There exists a large body of research on the topic of machine learning methods for deception detection, most of it has been focusing on classifying online reviews and publicly available social media posts. Particularly since late 2016 during the American Presidential election, the question of determining 'fake news' has also been the subject of particular attention within the literature. Conroy, Rubin, and Chen outlines several approaches that seem promising towards the aim of perfectly classify the misleading articles. They note that simple content-related n-grams and shallow parts-of-speech (POS) tagging have proven insufficient for the classification task, often failing to account for important context information. Rather, these methods have been shown useful only in tandem with more complex methods of analysis. Deep Syntax analysis using Probabilistic Context Free Grammars (PCFG) have been shown to be particularly valuable in combination with n-gram methods.



Top Five Unreliable News Sources		Top Five Reliable News Sources	
Before It's News	2066	Reuters	3898
Zero Hedge	149	BBC	830
Raw Story	90	USA Today	824
Washington Examiner	79	Washington Post	820
Infowars	67	CNN	595

FIG 1.2: VARIOUS NEWS SOURCES

1.2.1 LIMITATIONS OF EXISTING SYSTEM

- Lack of integrity
- Lack of availability
- Lack of continuity of service
- Lack of accuracy

1.3 PROPOSED SYSTEM

In this paper a model is build based on the count vectorizer or a tfidf matrix (i.e) word tallies relatives to how often they are used in other artices in your dataset) can help . Since this problem is a kind of text classification, Implementing a Naive Bayes classifier will be best as this is standard for text-based processing. The actual goal is in developing a model which was the text transformation (count vectorizer vs tfidf vectorizer) and choosing which type of text to use (headlines vs full text). Now the next step is to extract the most optimal features for countvectorizer or tfidf-vectorizer, this is done by using a n-number of the most used words, and/or phrases, lower casing or not, mainly removing the stop words which are common words such as "the", "when", and "there" and only using those words that appear at least a given number of times in a given text dataset.

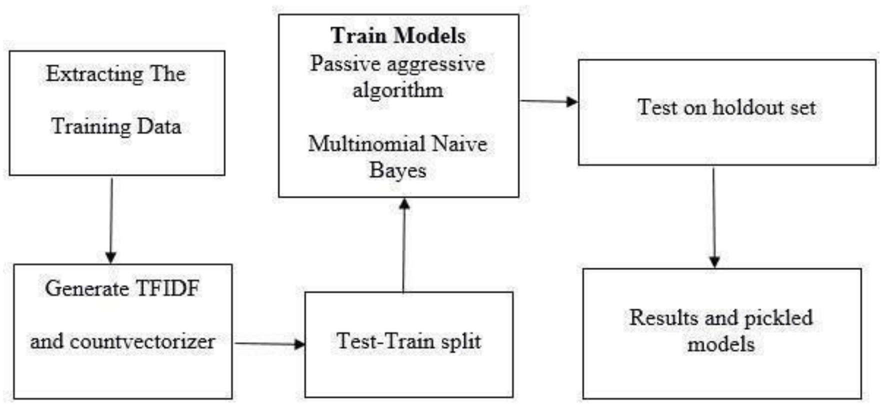

Extracting The Training Data
Generate TFIDF and countvectorizer
Test-Train split
Train Models
Passive aggressive algorithm
Multinomial Naive Bayes
Test on holdout set
Results and pickled models

FIG 1.3: TRAIN MODELS

1.3.1 ADVANTAGES OF PROPOSED SYSTEM

Apart from reduction in storage costs data outsourcing to the cloud also helps in reducing the maintenance.

- Avoiding local storage of data.
- By reducing the costs of storage, maintenance and personnel.
- It reduces the chance of losing data by hardware failures.

CHAPTER 2
LITERATURE SURVEY

2.1 LITERATURE REVIEW

Fake news is the deliberate spread of misinformation via traditional news media or via social media. False information spreads extraordinarily fast. This is demonstrated by the fact that, when one fake news site is taken down, another will promptly take its place. In addition, fake news can become indistinguishable from accurate reporting since it spreads so fast. People can download articles from sites, share the information, re-share from others and by the end of the day the false information has gone so far from its original site that it becomes indistinguishable from real news. They have used time period frequency-inverse record frequency (TFIDF) of bi- grams and probabilistic context free grammar (PCFG) detection. They have examined their dataset over more than one class algorithms to find out the great model. They locate that TF- IDF of bi-grams fed right into a Stochastic Gradient Descent model identifies non-credible resources with an accuracy of seventy seven.2%. Mykhailo Granik proposed simple technique for fake news detection the usage of naive Bayes classifier. They used BuzzFeed news for getting to know and trying out the Naïve Bayes classifier. The dataset is taken from facebook news publish and completed accuracy upto seventy four% on test set. In , Cody Buntain advanced a method for automating fake news detection on Twitter. They applied this method to Twitter content sourced from BuzzFeed's fake news dataset. Furthermore, leveraging non- professional, crowdsourced people instead of journalists presents a beneficial and much less costly way to classify proper and fake memories on Twitter rapidly. Marco L. Della offered a paper which allows us to recognize how social networks and gadget studying (ML) strategies may be used for faux news detection .They have used novel ML fake news detection method and carried out this approach inside a Facebook Messenger chatbot and established it with a actual-world application, acquiring a fake information detection accuracy of eighty one.7%. In , Rishabh Kaushal carried out 3 getting to know algorithms specifically Naive Bayes, Clustering and Decisionbushes on some of features such astweet-degree and consumer-level like Followers/Followees, URLs, SpamWords, Replies and HashTags. Improvement of unsolicited mail detection is measured on the premise of general Accuracy, Spammers Detection Accuracy and Non Spammers Detection Accuracy. In, Saranya Krishnan used superior framework to indentify faux information contents. Initially, they've extracted content material capabilities and consumer functions via Twitter API. Then functions together with statistical analysis of twitter user accounts, reverse picture searching, verification of fake news assets are used by facts mining algorithms for class and analysis. Distorted news and “alternate facts” were not a problem in society two years ago, despite the long-term deep changes in the news market 1 . The social concern about these kinds of news has been rather deeply accelerated by the term “fake news”, coined by the US elected President, Donald Trump, conveying its origins in the political arena. For example, among other fake news that emerged during the Trump campaign one of the most popular ones consisted on the Pope Francis reported endorsement of Donald Trump for president of the US. The news piece was advanced by the website “Ending The Fed”, managed by a Romanian youngster. BBC 4, 6 also refers to the advancement of particular (often extreme) political causes as one of the main sources of fake news, defining them as false information deliberately circulated by those who have scant regard for the truth and act under the motivation of fostering political causes or obtaining revenue out of the online traffic. In this domain, Facebook has faced an increasing criticism over its role in the 2016 US presidential election because it allowed the propagation of fake news disguised as news stories coming from unchecked websites. Automatic fake news detection is the task of assessing the truthfulness of claims in news. This is a new, but critical NLP problem because both traditional news media and social media have huge social-political impacts on every individual in the society. For example, exposure to fake news can cause attitudes of inefficacy, alienation, and cynicism toward certain political candidates (Balmas, 2014). The worst part of the spread of fake news is that sometimes it does link to offline violent events that threaten the public safety (e.g., the PizzaGate Kang and Goldman (2016)). Detecting fake news is of crucial importance to the NLP community, as it also creates broader impacts on how technologies can facilitate the verification of the veracity of claims while educating the general public.The conventional solution to this task is to ask professionals such as journalists to check claims against evidence based on previously spoken or written facts. However, it is time- consuming and costs a lot of human resources, For example, PolitiFact takes three editors to judge whether a piece of news is real or not.As the Internet community and the speed of the spread of information are growing rapidly, automated fact checking on internet content has gained plenty of interests in the Artificial Intelligence research community. The goal of automatic fake news detection is to reduce the human time and effort to detect fake news and help us to stop spreading them. The task of fake news detection has been studied from various perspectives with the development in subareas of Computer Science, such as Machine_Learning (ML), Data Mining (DM), and NLP. As you can imagine, with so much data and so many information sources, the challenge of detecting intent and emotion in free text is no easy feat. Fortunately, natural language processing (NLP) is built for this. Sentiment analysis—or opinion mining—uses NLP and machine learning to interpret and classify emotions in subjective data. It’s often used in business to detect sentiment in social media, gauge brand reputation, and understand customers, from polarity—positive, negative, neutral—to emotion detection—angry, happy, sad, or afraid.NLP has the power to help businesses sort through an otherwise impossible amount of data, and in most cases identify the motive behind the words. But this is especially important when you consider the complexities of human language. With context, formal and informal language, misspellings—sometimes intentional to mean something different than the origin word—accuracy becomes important, especially if there are consequences to these actions. Social media has been used to incite violence, discriminate, bully, shame, and harass.Even for less extreme cases, many NLP tools available today can be used to automatically detect the tone of text and more difficult projects, like analyzing the sentiment of whole-text with specific aspects in it. For example, what someone thought about a restaurant’s food vs. its service or price—and whether text seems sarcastic, fake, or includes toxic language such as threats, insults, obscenity, or hate speech.Take cyberbullying, for example. More than 40% of adults have personally experienced some form of online harassment, and 75% have seen cyberbullying occurring. That number jumps to 85% when taking into account young people, making it clear why it’s important to keep tabs on this. John Snow Labs’ Spark NLP library includes a trainable multi-class text classification model that uses state-of-the-art universal sentence embeddings as an input for text classifications. The document classifier uses a deep learning model and supports up to 100 classes. Pre-trained models that are freely available with the open source library include detectors for cyberbullying, racism, sexism, or threatening tweets.Let’s look at how this works in real life. Take the following tweet:

“@AMohedin Okay, we have women being physically inferior and either emotionally or mentally inferior in some way.”

The emotions detector model classified this as 100% a sexist tweet.

On the other hand, take this example: “@LynnMagic people think that implying association via follow is a bad thing. but it’s shockingly accurate.” This was classified as a neutral tweet.As you can imagine, the levels of sexism and racism, or even implications of neutral tweets vary, but this is a good jumping off point to keep an eye on those who cross the line regularly. It’s also vital to keep in mind that the better your training data, the better your results will be, so NLP needs to be constantly improved and fine-tuned as you go.Similar to cyberbullying, toxic content is another area where NLP can help uncover harmful discussions. Toxic content can be classified as language that infers hate, insult, obscenity, or threats. In this case, we’ll look at social media comments using our Spark NLP’s Multiclassifier DL, another pretrained model. This is a real comment that’s been analyzed as toxic, and more specifically, a threat, using NLP: “I’m also a sock puppet of this account…SUPRISE!! -sincerely, The man that will track you down from the Internet and kill you.” While not every disturbing social media post has credibility, this type of comment can hurt your business, frighten people, or in some cases, cause real harm, and it’s worth taking note of The same goes for ‘fake news,’ which has been an especially popular topic of late. While it’s arguably not as detrimental as cyber bullying or toxic content, it does have the power to incite unhealthy debate, which can lead to these other behaviors. NLP can also help comb through article contents, and social media posts promoting them to identify what’s real and what’s not. While a headline like, “White House Makes Trade Pitch, With Focus on Moderates,” would be classified as real news, “Morning Joe Destroys Corrupt Clinton Foundation (Laughable) ‘Total Corruption’,” would be classified as fake.From cyberbullying and toxic content to fake news, it’s clear why we need to stay ahead of potentially dangerous dialogue online. In fact, another survey by Pew Research drives home why being able to detect this is so important to how we conduct business and ourselves on the internet. Nearly 40% of respondents said they expect the online future will be “more shaped” by negative activities, when asked if they believed public discourse online will become more or less shaped by bad actors, harassment, trolls, and an overall tone of griping, distrust, and disgust. This is a grim outlook, but probably not far from reality, especially taking into consideration current events over the last year.For businesses, it’s important to keep your websites, platforms, services, and social spaces safe to attract and retain customers. But the implications go far beyond the enterprise. results will be, so NLP needs to be constantly improved and fine-tuned as you go.Similar to cyberbullying, toxic content is another area where NLP can help uncover harmful discussions. Toxic content can be classified as language that infers hate, insult, obscenity, or threats. In this case, we’ll look at social media comments using our Spark NLP’s Multiclassifier DL, another pretrained model. This is a real comment that’s been analyzed as toxic, and more specifically, a threat, using NLP: “I’m also a sock puppet of this account…SUPRISE!! -sincerely, The man that will track you down from the Internet and kill you.” While not every disturbing social media post has credibility, this type of comment can hurt your business, frighten people, or in some cases, cause real harm, and it's worth taking note of The same goes for 'fake news,' which has been an especially popular topic of late. While it's arguably not as detrimental as cyber bullying or toxic content, it does have the power to incite unhealthy debate, which can lead to these other behaviors. NLP can also help comb through article contents, and social media posts promoting them to identify what's real and what's not. While a headline like, "White House Makes Trade Pitch, With Focus on Moderates," would be classified as real news, "Morning Joe Destroys Corrupt Clinton Foundation (Laughable) 'Total Corruption'," would be classified as fake.From cyberbullying and toxic content to fake news, it's clear why we need to stay ahead of potentially dangerous dialogue online. In fact, another survey by Pew Research drives home why being able to detect this is so important to how we conduct business and ourselves on the internet. Nearly 40% of respondents said they expect the online future will be "more shaped" by negative activities, when asked if they believed public discourse online will become more or less shaped by bad actors, harassment, trolls, and an overall tone of griping, distrust, and disgust. This is a grim outlook, but probably not far from reality, especially taking into consideration current events over the last year.For businesses, it's important to keep your websites, platforms, services, and social spaces safe to attract and retain customers. But the implications go far beyond the enterprise. The Internet should be a venue where people can voice their opinions freely, but also one where people aren't victimized, abused, misled, or slandered by others hiding behind a screen. While there's rightful concern that regulations may hurt the open exchange of ideas, opinions, disparate views, and conversations – NLP is one step in the right direction to help monitor what's going on online without infringing on individual rights or enabling toxic content to persevere.

CHAPTER 3
SOFTWARE REQUIREMENT ANALYSIS

3.1 DEFINITION OF PROBLEM INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

3.1.1 INPUT STAGES

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

3.1.2 INPUT TYPES

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to thesystem?
- Interactive, which are inputs entered during a dialogue.

3.1.3 INPUT MEDIA

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to:

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

3.2 ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

3.3 ERROR DETECTION

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

3.4 DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.
The system is designed to be a user friendly one. In other words, the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

3.5 ERROR MESSAGE DESIGN

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error, he/she has committed.

3.6 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User’s main interface with the computer.
- Operational outputs whose use is purely within the computer department
- Interface outputs, which involve the user in communicating directly

3.7 OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

3.8 PROBLEM DEFINATION

The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate or false information acquires a tremendous potential to cause real world impacts, within minutes, for millions of users. presenting the proposals into categories: content based, source based and diffusion based. We describe two opposite approaches and propose an algorithmic solution that synthesizes the main concerns. We conclude the paper by raising awareness about concerns and opportunities for businesses that are currently on the quest to help automatically detecting fake news by providing web services, but who will most certainly, on the long term, profit from their massive usage.

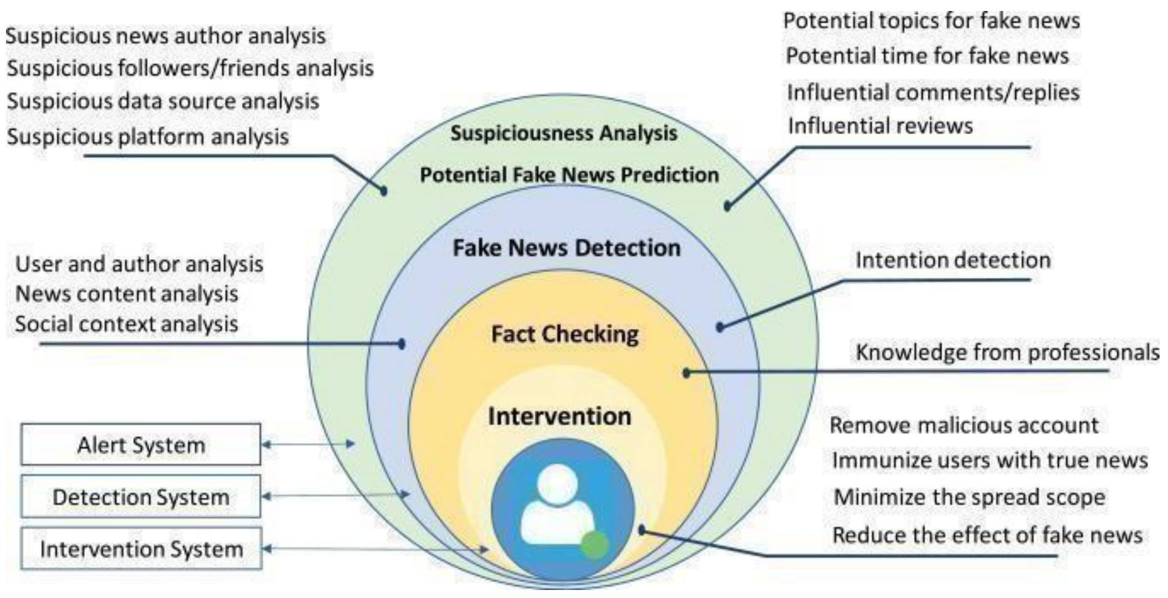

Suspicious news author analysis
Suspicious followers/friends analysis
Suspicious data source analysis
Suspicious platform analysis
Potential topics for fake news
Potential time for fake news
Influential comments/replies
Influential reviews
Suspiciousness Analysis
Potential Fake News Prediction
Fake News Detection
User and author analysis
News content analysis
Social context analysis
Intention detection
Fact Checking
Knowledge from professionals
Intervention
Remove malicious account
Immunize users with true news
Minimize the spread scope
Reduce the effect of fake news
Alert System
Detection System
Intervention System

FIG 3.8 : OVERVIEW OF FAKE NEWS

3.9 MODULES AND FUNCTIONALITIES

3.9.1 MODULES USED

- Speech recognition
- Pyaudio
- Pyttsx3
- Os
- Time
- Threading
- Random

3.9.2 TECHNOLOGIES USED

- Machine Learning
- CNN
- NLP

3.10 HARDWARE AND SOFTWARE REQUIREMENTS

3.10.1 HARDWARE REQUIREMENTS

Intel i3 or higher Ram 4gb or more

3.10.2 SOFTWARE REQUIREMENTS

Python 3.6.8

3.11 ALGORITHMS

We used the following learning algorithms in conjunction with our proposed methodology to evaluate the performance of fake news detection classifiers.

LOGISTIC REGRESSION

As we are classifying text on the basis of a wide feature set, with a binary output (true/false or true article/fake article), a logistic regression (LR) model is used, since it provides the intuitive equation to classify problems into binary or multiple classes. We performed hyperparameters tuning to get the best result for all individual datasets, while multiple parameters are tested before acquiring the maximum accuracies from LR model. Mathematically, the logistic regression hypothesis function can be defined as follows
Logistic regression uses a sigmoid function to transform the output to a probability value; the objective is to minimize the cost function to achieve an optimal probability.

SUPPORT VECTOR MACHINE

Support vector machine (SVM) is another model for binary classification problem and is available in various kernels functions. The objective of an SVM model is to estimate a hyperplane (or decision boundary) on the basis of feature set to classify data points.The dimension of hyperplane varies according to the number of features. As there could be multiple possibilities for a hyperplane to exist in an *N*-dimensional space, the task is to identify the plane that separates the data points of two classes with maximum margin.

The function above uses a linear kernel. Kernels are usually used to fit data points that cannot be easily separable or data points that are multidimensional. In our case, we have used sigmoid SVM, kernel SVM (polynomial SVM), Gaussian SVM, and basic linear SVM models.

MULTILAYER PERCEPTRON

A multilayer perceptron (MLP) is an artificial neural network, with an input layer, one or more hidden layers, and an output layer. MLP can be as simple as having each of the three layers; however, in our experiments we have fine-tuned the model with various parameters and number of layers to generate an optimum predicting model. A basic multilayered perceptron model with one hidden layer can be represented as a function as shown below Here, are the bias vectors, are the weight matrices, and and are the activation functions. In our case, the activation function is ReLU and the Adam solver, with 3 hidden layers.

K-NEAREST NEIGHBORS (KNN)

KNN is an unsupervised machine learning model where a dependent variable is not required to predict the outcome on a specific data. We provide enough training data to the model and let it decide to which particular neighborhood a data point belongs. KNN model estimates the distance of a new data point to its nearest neighbors, and the value of *K* estimates the majority of its neighbors' votes; if the value of *K* is 1, then the new data point is assigned to a class which has the nearest distance.

3.12 ENSEMBLE LEARNERS

We proposed using existing ensemble techniques along with textual characteristics as feature input to improve the overall accuracy for the purpose of classification between a truthful and a false article. Ensemble learners tend to have higher accuracies, as more than one model is trained using a particular technique to reduce the overall error rate and improve the performance of the model. The intuition behind the ensemble modeling is synonymous to the one we are already used to in our daily life such as requesting opinions of multiple experts before taking a particular decision in order to minimize the chance of a bad decision or an undesirable outcome. For example, a classification algorithm can be trained on a particular dataset with a unique set of parameters that can produce a decision boundary which fits the data to some extent. The outcome of that particular algorithm depends not only on the parameters that were provided to train the model, but also on the type of training data. If the training data contains less variance or uniform data, then the model might over fit and produce biased results over unseen data. Therefore, approaches like cross validation are used to minimize the risk of over fitting. A number of models can be trained on different set of parameters to create multiple decision boundaries on randomly chosen data points as training data. Hence, using ensemble learning techniques, these problems can be addressed and mitigated by training multiple algorithms, and their results can be combined for near optimum outcome. One such technique is using voting classifiers where the final classification depends on the major votes provided by all algorithms. However, there are other ensemble techniques as well that can be used in different scenarios such as the following.

RANDOM FOREST (RF)

Random forest (RF) is an advanced form of decision trees (DT) which is also a supervised learning model. RF consists of large number of decision trees working individually to predict an outcome of a class where the final prediction is based on a class that received majority votes. The error rate is low in random forest as compared to other models, due to low correlation among trees. Our random forest model was trained using different parameters; i.e., different numbers of estimators were used in a grid search to produce the best model that can predict the outcome with high accuracy. There are multiple algorithms to decide a split in a decision tree based on the problem of regression or classification. For the classification problem, we have used the Gini index as a cost function to estimate a split in the dataset. The Gini index is calculated by subtracting the sum of the squared probabilities of each class from one.

BAGGING ENSEMBLE CLASSIFIERS

Bootstrap aggregating, or in short bagging classifier, is an early ensemble method mainly used to reduce the variance (overfitting) over a training set. Random forest model is one of the most frequently used as a variant of bagging classifier. Intuitively, for a classification problem, the bagging model selects the class on the basis of major votes estimated by number of trees to reduce the overall variance, while the data for each tree is selected using random sampling with replacement from overall dataset. For regression problems, however, the bagging model averages over multiple estimates.

BOOSTING ENSEMBLE CLASSIFIERS

Boosting is another widely used ensemble method to train weak models to become strong learners. For that purpose, a forest of randomized trees is trained, and the final prediction is based on the majority vote outcome from each tree. This method allows weak learners to correctly classify data points in an incremental approach that are usually misclassified. Initially equal weighted coefficients are used for all data points to classify a given problem. In the successive rounds, the weighted coefficients are decreased for data points that are correctly classified and are increased for data points that are misclassified. Each subsequent tree formed in each round learns to reduce the errors from the preceding round and to increase the overall accuracy by correctly classifying data points that were misclassified in previous rounds. One major problem with boosting ensemble is that it might overfit to the training data which may lead to incorrect predictions for unseen instances. There are multiple boosting algorithms available that can be used for both the purposes of classification and regression. In our experiments we used XGBoost and AdaBoost algorithms for classification purpose.

VOTING ENSEMBLE CLASSIFIERS

Voting ensemble is generally used for classification problems as it allows the combination of two or more learning models trained on the whole dataset. Each model predicts an outcome for a sample data point which is considered a “vote” in favor of the class that the model has predicted. Once each model predicts the outcome, the final prediction is based on the majority vote for a specific class. Voting ensemble, as compared to bagging and boosting algorithms, is simpler in terms of implementation. As discussed, bagging algorithms create multiple subsets of data by random sampling and replacement from the whole dataset, thus creating a number of datasets. Each dataset is then used to train a model, while the final result is an aggregation of outcome from each model. In case of boosting, multiple models are trained in a sequential manner where each model learns from the previous by increasing weights for the misclassified points, thus creating a generic model that is able to correctly classify the problem. However, voting ensemble on the other hand is a combination of multiple independent models that produces classification results that contribute to the overall prediction by majority voting.

3.13 BENCHMARK ALGORITHMS

In this section, we discuss the benchmark algorithms with which we compare the performance of our methodology.

LINEAR SVM

We use linear SVM approach proposed in to ensure a meaningful comparison, we trained the linear SVM on the feature set as discussed in with 5-fold cross validation.

CONVOLUTIONAL NEURAL NETWORK

Wang used convolutional neural network (CNN) for automatic detection of fake news. We employed the same approach using our dataset. However, we could not use the feature set of Wang as the dataset contains only short statements. The approach is referred to as Wang-CNN in the text.

BIDIRECTIONAL LONG SHORT-TERM MEMORY NETWORKS

Wang also used bidirectional long short-term memory networks (Bi-LSTM), and we used the same approach with different feature sets. The approach is referred to as Wang-Bi-LSTM in the text.

3.14 PERFORMANCE METRICS

To evaluate the performance of algorithms, we used different metrics. Most of them are based on the confusion matrix. Confusion matrix is a tabular representation of a classification model performance on the test set, which consists of four parameters: true positive, false positive, true negative, and false negative.

ACCURACY

Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used, In most cases, high accuracy value represents a good model, but considering the fact that we are training a classification model in our case, an article that was predicted as true while it was actually false (false positive) can have negative consequences; similarly, if an article was predicted as false while it contained factual data, this can create trust issues. Therefore, we have used three other metrics that take into account the incorrectly classified observation, i.e., precision, recall, and F1-score.

RECALL

Recall represents the total number of positive classifications out of true class. In our case, it represents the number of articles predicted as true out of the total number of true articles.

PRECISION

Conversely, precision score represents the ratio of true positives to all events predicted as true.

3.15 CONVOLUTION NEURAL NETWORK (CNN)

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. The patterns they recognize are numerical, contained in vectors, into which all real world data, be it images, sound, text or time series must be translated. Neural networks help us cluster and classify. You can think of them as a cluster and classify. It consists of layers which are made of nodes.In our project the wave files get converted into spectrum. In a graph of X-axis and Y-axis, the X-axis represents time and Y-axis represents frequency.CNN is used for data analysis. CNN detects patterns and makes use of them. The most important part of the convolution neural network are the hidden layers, detects a pattern. The pattern in our project is detected based on attributes like tone, voice, slang, modulation and distortion in voice. The main role of the Convolution neural network is to make sense of the unnecessary attributes too.

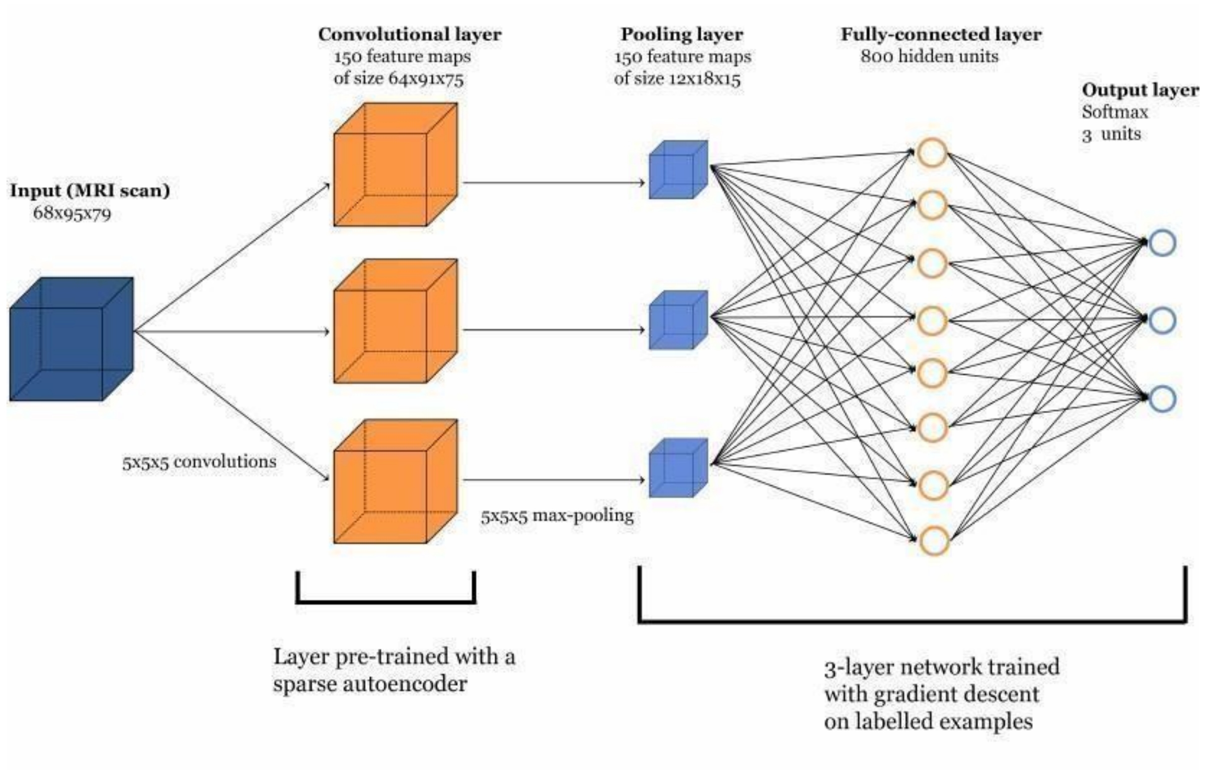


FIG 3.15 : CNN

3.16 NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing(NLP) is the technology used to aid computers to understand the human’s natural language.It’s not an easy task teaching machines to understand how we communicate.Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language.The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.Most NLP techniques rely on machine learning to derive meaning from human languages.

Information Retrieval
Doc A
Doc 1
Doc 2
Doc 3
Sentiment Analysis
Information Extraction
Machine Translation
Natural Language Processing
QuestionAnswering

FIG 3.16 : NLP

CHAPTER 4
SOFTWARE DESIGN

4.1 UML DIAGRAMS

The Unified Modeling Language is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. In this article, we will give you detailed ideas about what is UML, the history of UML and a description of each UML diagram type, along with UML examples. UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. It was initially started to capture the behavior of complex software and non-software system and now it has become an OMG standard. This tutorial gives a complete understanding on UML.

GOALS OF UML

- Since it is a general-purpose modeling language, it can be utilized by all the modelers.
- UML came into existence after the introduction of object-oriented concepts to systemize and consolidate the object-oriented development, due to the absence of standard methods at that time.
- The UML diagrams are made for business users, developers, ordinary people, or anyone who is looking forward to understand the system, such that the system can be software or non-software.
- Thus it can be concluded that the UML is a simple modeling approach that is used to model all the practical systems.

4.2 RESPONSE ENGINE CYCLE

Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

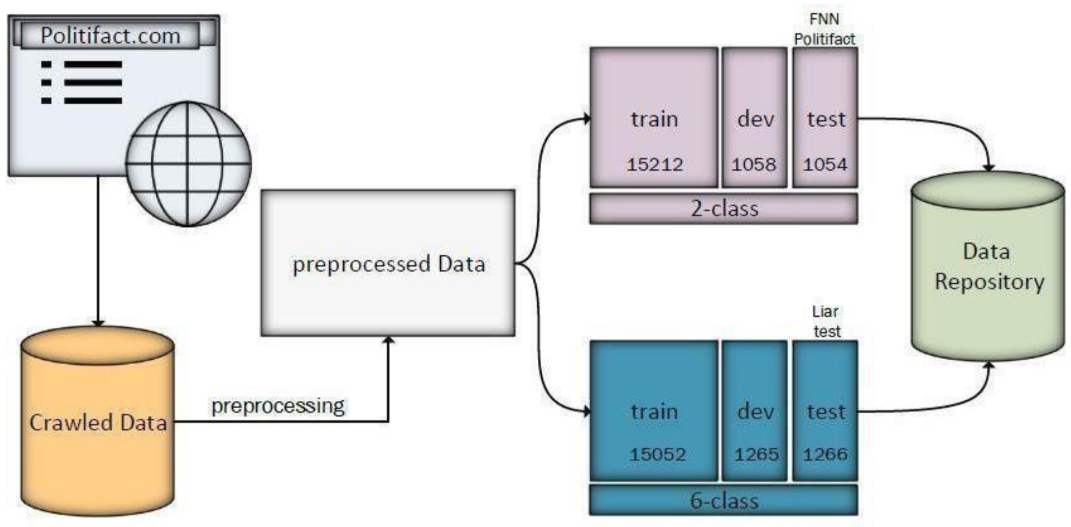


FIG NO. 4.2: RESPONSE ENGINE'S PROCESS CYCLE

4.3 SOFTWARE DESIGN PROCESS

In software engineering, a software development process is the process of dividing software development work into smaller, parallel or sequential steps or subprocesses to improve design, product management, and project management. It is also known as a software development life cycle. Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. The output of this process can directly be used into implementation in programming languages.

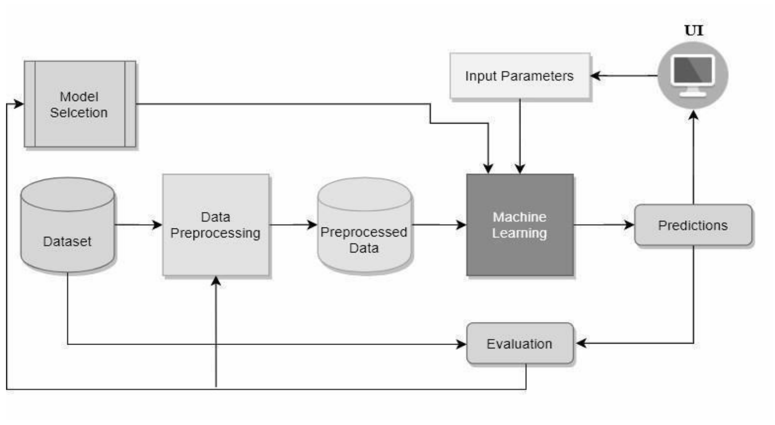


FIG NO. 4.3: SOFTWARE PROCESS DESIGN

4.4 ACTIVITY DIAGRAM

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

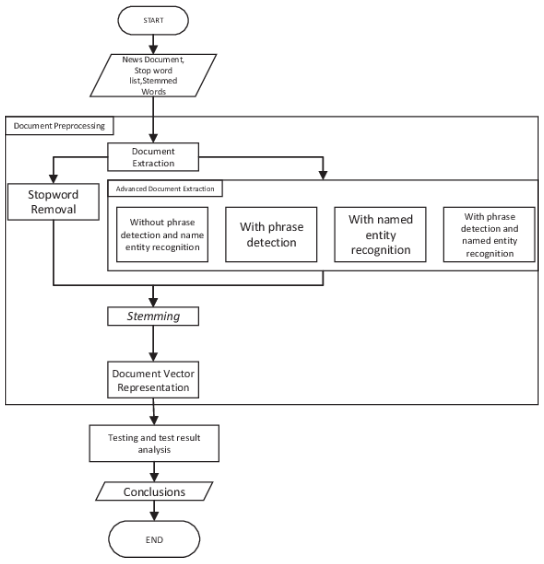

START
News Document, Stop word list,Stemmed Words
Document Preprocessing
Document Extraction
Stopword Removal
Advanced Document Extraction
Without phrase detection and name entity recognition
With phrase detection
With named entity recognition
With phrase detection and named entity recognition
Stemming
Document Vector Representation
Testing and test result analysis
Conclusions
END

FIG NO. 4.4: ACTIVITY DIAGRAM

4.5 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects. Class diagram is a static diagram. It represents the static view of an application.

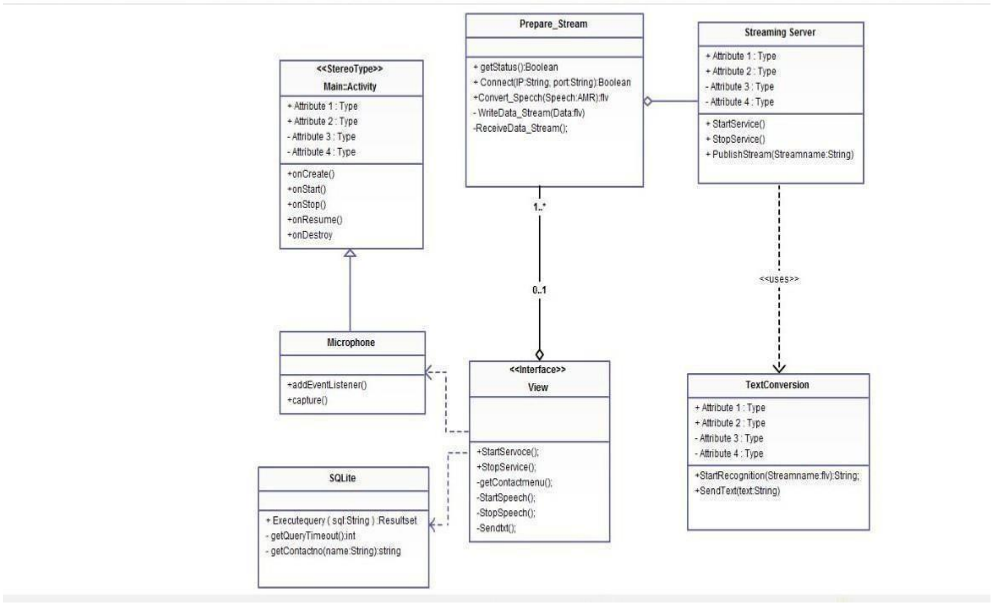


FIG NO 4.5: CLASS DIAGRAM

CHAPTER 5
SOFTWARE REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use. The requirement specification for any system can be broadly stated as given below: The system should be able to interface with the existing system.The system should be accurate.The system should be better than the existing system.The existing system is completely dependent on the user to perform all the duties.

5.2 FUNCTIONAL REQUIREMENTS

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are: External Outputs, whose destination is outside the organization Internal Outputs whose destination is within organization and they are the User’s main interface with the computer. Operational outputs whose use is purely within the computer department. Interface outputs, which involve the user in communicating directly.

5.3 NON FUNCTIONAL REQUIREMENTS

- Secure access of confidential data (user's details). SSL can be used.
- 24 X 7 availability.
- Better component design to get better performance at peak time

Flexible service-based architecture will be high.

CHAPTER 6
CODING

6.1 HTML:

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
<meta charset="UTF-8">
<title>Fake News Detection</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
    type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
    type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
    type='text/css'>
<link      href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
    rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body bgcolor="white">
<div class="login">
<h1 style="color:grey">Predict Fake News</h1>
<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}"method="post">
<input type="text" name="news" placeholder="Enter the news url" required="required"/>
<button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
<br>
<br>
</div>
</body>
</html>
```

6.2 APP.py:

```
#Importing the Libraries
#flask is use for run the web application. import flask
#request is use for accessing file which was uploaded by the user on our application.
    from flask import Flask, request,render_template
from flask_cors import CORS
#Python pickle module is used for serializing # and de-serializing a Python object
    structure. import pickle
#OS module in python provides functions for interacting with the operating system
    import os
#Newspaper is used for extracting and parsing newspaper articles.
 #For extracting all the useful text from a website.
from newspaper import Article
#URLlib is use for the urlopen function and is able to fetch URLs.
 #This module helps to define functions and classes to open URLs import urllib
import nltk nltk.download('punkt')
#Loading Flask and assigning the model variable app = Flask(_name_)
CORS(app)
app=flask.Flask(_name_,template_folder='templates')
with open('model.pkl', 'rb') as handle:
model = pickle.load(handle)
@app.route('/') #default route def main():
return render_template('main.html')
#Receiving the input url from th user and using Web Scrapping to extract the news
    content
#Route for prediction
@app.route('/predict',methods=['GET','POST']
def predict():
#Contains the incoming request data as string in case.
url =request.get_data(as_text=True)[5:]
```

```
#The URL parsing functions focus on splitting a URL string into its components,
#or on combining URL components into a URL string.
url = urllib.parse.unquote(url)
#A new article come from Url and convert onto string article = Article(str(url))

#To download the article article.download()
#To parse the article article.parse()
#To perform natural language processing ie..nlp article.nlp()
#To extract summary news = article.summary
print(type(news))
#Passing the news article to the model and returing whether it is Fake or Real
pred = model.predict([news])
print(pred)
return render_template('main.html', prediction_text='The news is "{}"'.format(pred[0]))
if name ==" main ": port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```

6.3 FAKE_NEWS.py :

```
#Importing libraries import pandas as pd import numpy as np
from sklearn.feature_extraction.text
import TfidfVectorizer from sklearn.feature_extraction.text
import CountVectorizer from sklearn.model_selection
import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix import pickle
from sklearn.pipeline import Pipeline
df = pd.read_csv(r'C:\Users\ADMIN\Desktop\Info\MRECW\MAJOR PROJECT
    RK07\Major_Project_RK\news.csv')
df.head()
# Create a series to store the labels: y # y = df.label
#cleaned file containing the text and label X = df['text'] # independent variable
y = df['label'] #dependent variable
```

```
# Create training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=53)
# Initialize a CountVectorizer object: count_vectorizer count_vectorizer =
    CountVectorizer(stop_words='english')
# Transform the training data using only the 'text' column values: count_train count_train
    = count_vectorizer.fit_transform(X_train)
# Transform the test data using only the 'text' column values: count_test count_test =
    count_vectorizer.transform(X_test)
# Print the first 10 features of the count_vectorizer
    print(count_vectorizer.get_feature_names()[:10])
# Initialize a TfidfVectorizer object: tfidf_vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
# Transform the training data: tfidf_train tfidf_train =
    tfidf_vectorizer.fit_transform(X_train)
# transform the test data: tfidf_test
tfidf_test = tfidf_vectorizer.transform(X_test)
# Print the first 10 features print(tfidf_vectorizer.get_feature_names()[:10])
# Print the first 5 vectors of the tfidf training data print(tfidf_train.A[:5])
count_df = pd.DataFrame(count_train.A,
    columns=count_vectorizer.get_feature_names())
# Create the TfidfVectorizer DataFrame: tfidf_df
tfidf_df = pd.DataFrame(tfidf_train.A, columns=tfidf_vectorizer.get_feature_names())
# Print the head of count_df print(count_df.head())
# Print the head of tfidf_df print(tfidf_df.head())
# Calculate the difference in columns: difference difference = set(count_df.columns) -
    set(tfidf_df.columns) print(difference)
# Check whether the DataFrame are equal
print(count_df.equals(tfidf_df))
# Instantiate a Multinomial Naive Bayes classifier: nb_classifier nb_classifier =
    MultinomialNB()
# Fit the classifier to the training data nb_classifier.fit(count_train, y_train)
# Create the predicted tags: pred
pred = nb_classifier.predict(count_test)
# Calculate the accuracy score: score score = accuracy_score(y_test, pred) print(score)
# Calculate the confusion matrix: cm
cm =confusion_matrix(y_test, pred, labels=['FAKE', 'REAL']) print(cm)
#Training and testing the "fake news" model with TfidfVectorizer nb_classifier =
    MultinomialNB()
# Fit the classifier to the training data nb_classifier.fit(tfidf_train, y_train)
# Create the predicted tags: pred
pred = nb_classifier.predict(tfidf_test)
# Calculate the accuracy score: score score = accuracy_score(y_test, pred) print(score)
# Calculate the confusion matrix: cm
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL']) print(cm)
#Improving the model to test a few different alpha levels using the Tfidf vectors, #to
    determine.if there is a better performing combination
alphas = np.arange(0, 1, 0.1)
# Define train_and_predict() def train_and_predict(alpha):
# Instantiate the classifier: nb_classifier nb_classifier = MultinomialNB(alpha=alpha)
# Fit to the training data nb_classifier.fit(tfidf_train, y_train)
# Predict the labels: pred
pred = nb_classifier.predict(tfidf_test)
# Compute accuracy: score
score = accuracy_score(y_test, pred) return score
# Iterate over the alphas and print the corresponding score for alpha in alphas:
print('Alpha: ', alpha)
print('Score: ', train_and_predict(alpha))
print()
class_labels = nb_classifier.classes_
# Extract the features: feature_names
feature_names = tfidf_vectorizer.get_feature_names()
# Zip the feature names together with the coefficient array # and sort by weights:
    feat_with_weights
feat_with_weights = sorted(zip(nb_classifier.coef_[0], feature_names))
# Print the first class label and the top 20 feat_with_weights entries
print(class_labels[0], feat_with_weights[:20])
# Print the second class label and the bottom 20 feat_with_weights entries
    print(class_labels[1], feat_with_weights[-20:])
```

```
#Creating a pipeline that first creates bag of words(after applying stopwords) & then
    applies Multinomial Naive Bayes model
pipeline = Pipeline([('tfidf', TfidfVectorizer(stop_words='english')), ('nbmodel',
    MultinomialNB())])
#Training our data pipeline.fit(X_train, y_train)
#Predicting the label for the test data pred = pipeline.predict(X_test)
print(confusion_matrix(y_test, pred))
#Serialising the file #pickle.dump(nb_classifier,open('fake_news.pkl','wb'))
 #Serialising the file
with open('model.pkl', 'wb') as handle:
pickle.dump(pipeline, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

CHAPTER 7 TESTING

UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

WHITE BOX TESTING

This type of testing ensures that

- All independent paths have been exercised at least once.
- All logical decisions have been exercised on their true and false sides.
- All loops are executed at their boundaries and within their operational bounds.
- All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

BASIC PATH TESTING

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions.

CONDITIONAL TESTING

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

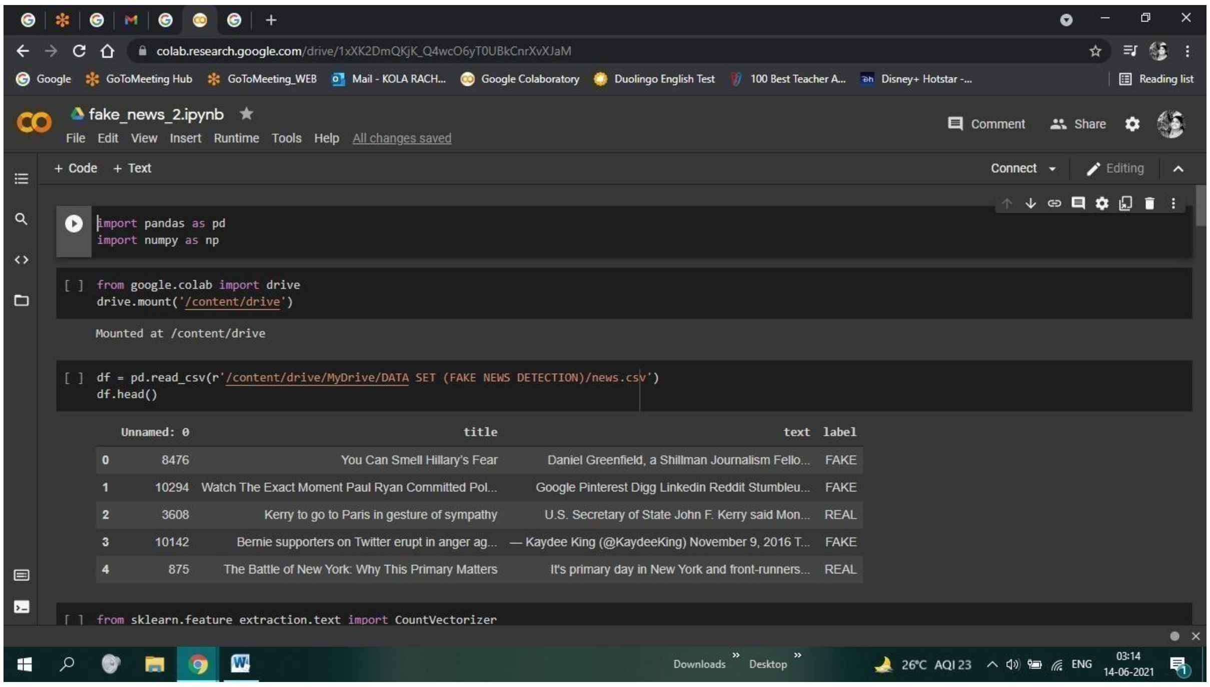
7.1 TESTING SCENARIOS

S.no	Input	Expected output	Actual output	Remarks
1.	Speech (Hi how are you question mark)	Text(Hi how are you?)	No internet connection	Internet connection is mandatory to get desired output.
2.	Speech(Good morning exclamation mark)	Text(Good morning!)	Audio cannot be understood	The speaker should be clear and there should not be any disturbances
3.	Speech(Hello full stop)	Text(Hello.)	Text(Hello full stop)	The punctuations should be in a symbol format.This can be done by adding a list of punctuations in code and replacing them.
4.	Speech(What are you doing question mark)	Text(What are you doing?)	Text(What are you doing?)	Expected output and Actual output match.

TABLE NO. 7.5 : TESTING SCENARIOS

TESTING SCREENS

TESTCASE 1



```
import pandas as pd
import numpy as np
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

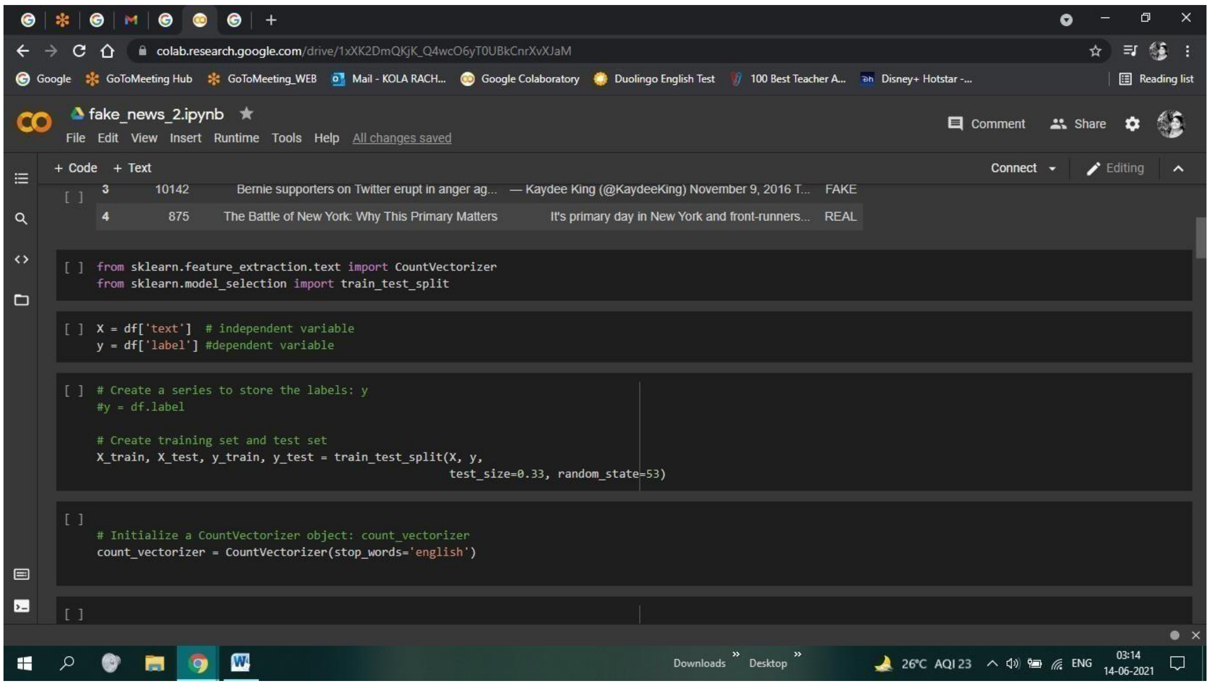
```
[ ] df = pd.read_csv(r'/content/drive/MyDrive/DATA SET (FAKE NEWS DETECTION)/news.csv')
    df.head()
```

```
   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
```

```
[ ] from sklearn.feature extraction.text import CountVectorizer
```

FIG 7.1 : MOUNTED THE DRIVE

TEST CASE 2



```
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
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

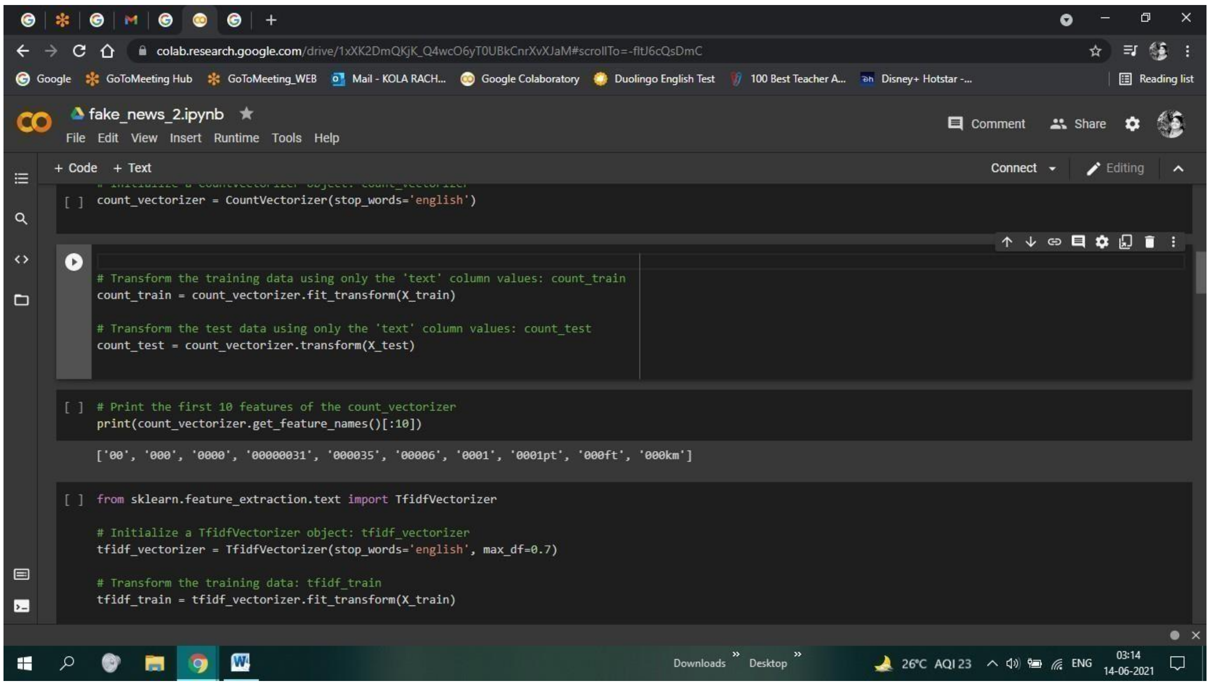
```
X = df['text']  # independent variable
y = df['label'] #dependent variable
```

```
# Create a series to store the labels: y
#y = df.label

# Create training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33, random_state=53)
```

```
# Initialize a CountVectorizer object: count_vectorizer
count_vectorizer = CountVectorizer(stop_words='english')
```

FIG 7.2: CREATE A SERIES TO STORE LABEL

TEST CASE 3


colab.research.google.com/drive/1xXK2DmQKjK_Q4wcO6yT0UBkCnrXvXJaM#scrollTo=-fltJ6cQsDmC
fake_news_2.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Comment Share
Connect
Editing

```
count_vectorizer = CountVectorizer(stop_words='english')
```

```
# Transform the training data using only the 'text' column values: count_train
count_train = count_vectorizer.fit_transform(X_train)

# Transform the test data using only the 'text' column values: count_test
count_test = count_vectorizer.transform(X_test)
```

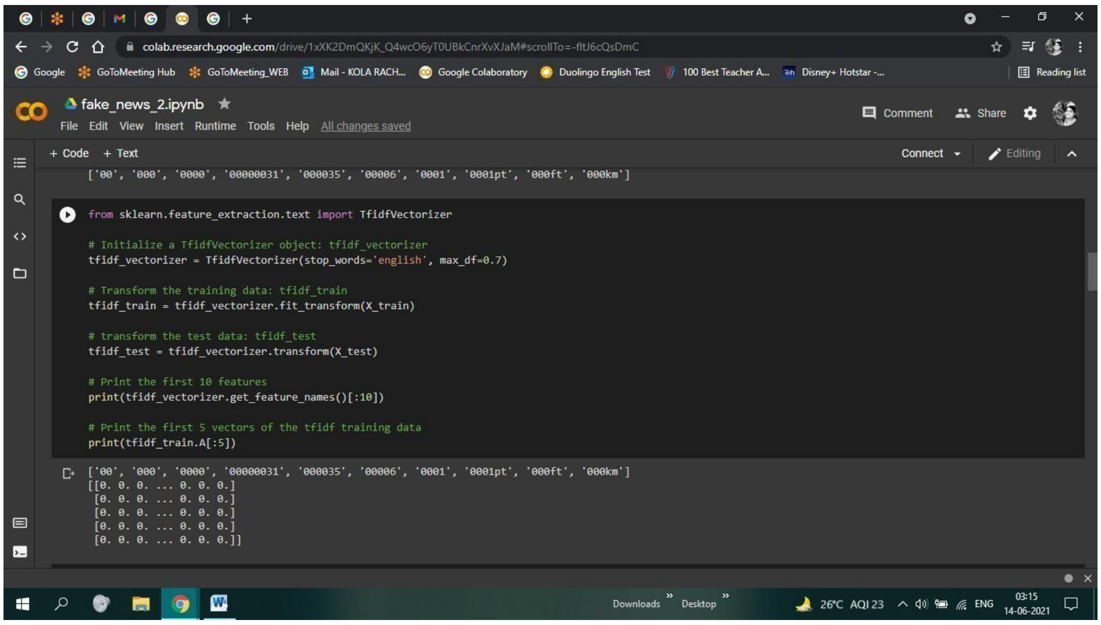
```
# Print the first 10 features of the count_vectorizer
print(count_vectorizer.get_feature_names()[:10])
```

```
['00', '000', '0000', '00000031', '000035', '00006', '0001', '0001pt', '000ft', '000km']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize a TfidfVectorizer object: tfidf_vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Transform the training data: tfidf_train
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
```

FIG 7.3: PRINT 10 FEATURES

TEST CASE 4



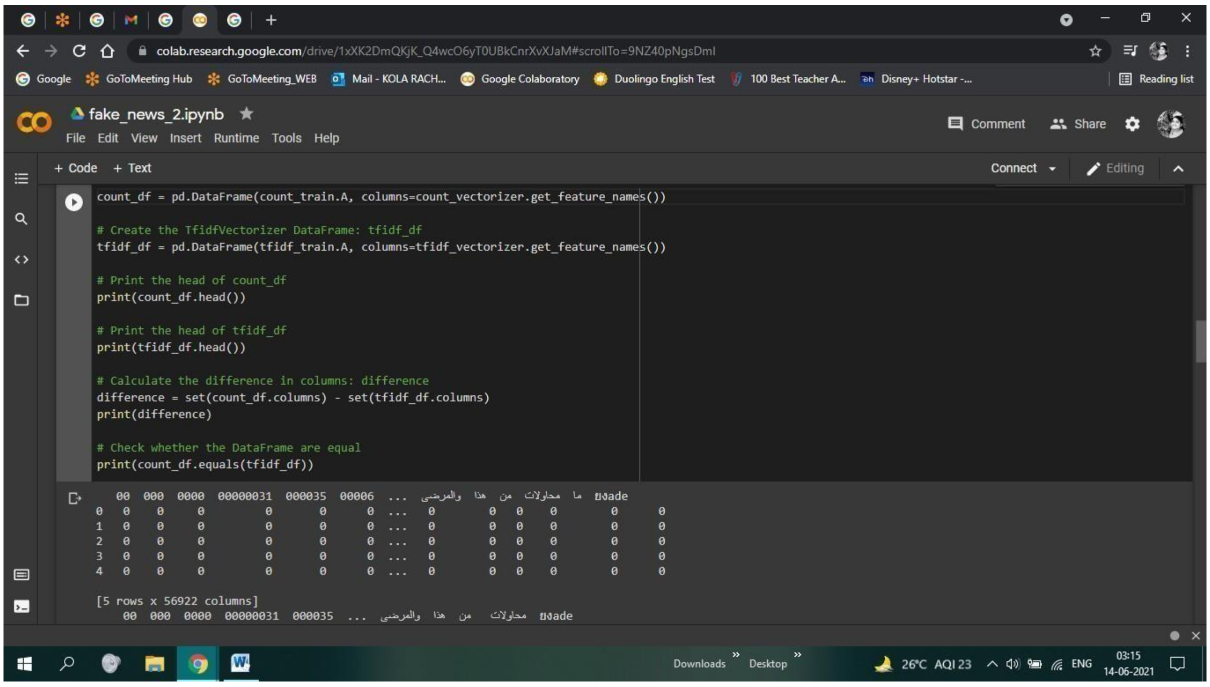
```
['00', '000', '0000', '00000031', '000035', '00006', '0001', '0001pt', '000ft', '000km']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize a TfidfVectorizer object: tfidf_vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Transform the training data: tfidf_train
tfidf_train = tfidf_vectorizer.fit_transform(X_train)

# transform the test data: tfidf_test
tfidf_test = tfidf_vectorizer.transform(X_test)

# Print the first 10 features
print(tfidf_vectorizer.get_feature_names()[:10])

# Print the first 5 vectors of the tfidf training data
print(tfidf_train.A[:5])
```

```
['00', '000', '0000', '00000031', '000035', '00006', '0001', '0001pt', '000ft', '000km']
[[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.]]
```

FIG 7.4: PRINT FIRST 5 VECTORS

TEST CASE 5

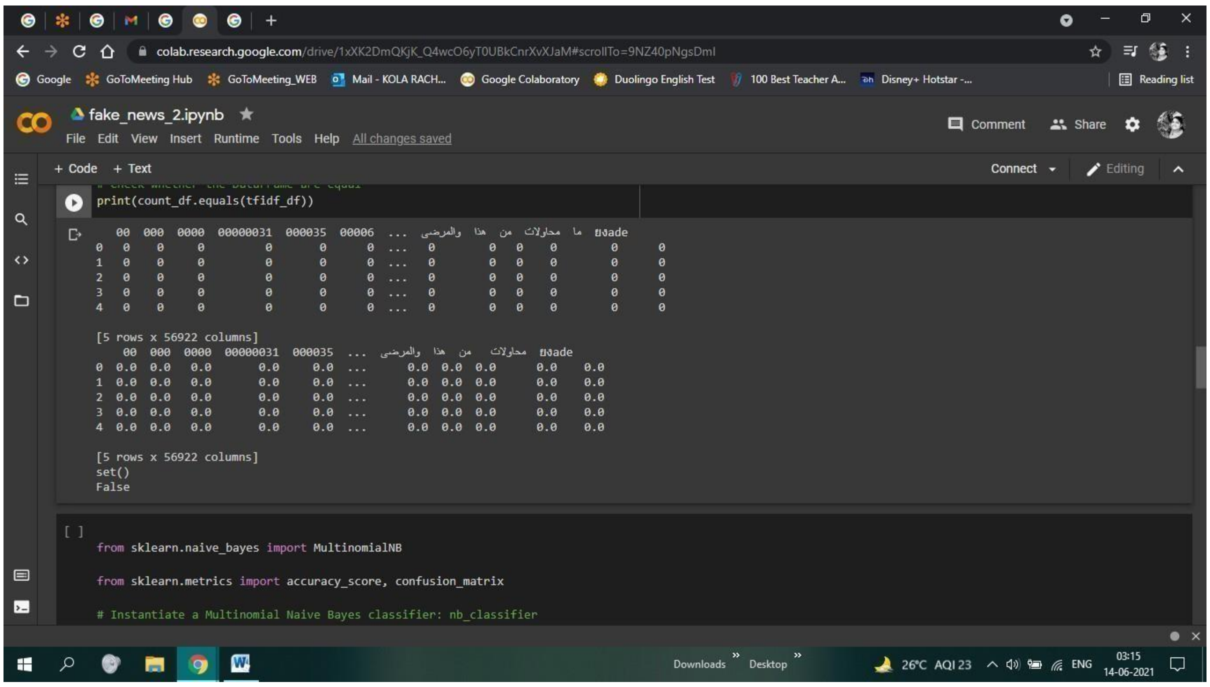


```
count_df = pd.DataFrame(count_train.A, columns=count_vectorizer.get_feature_names())

# Create the TfidfVectorizer DataFrame: tfidf_df
tfidf_df = pd.DataFrame(tfidf_train.A, columns=tfidf_vectorizer.get_feature_names())

# Print the head of count_df
print(count_df.head())

# Print the head of tfidf_df
print(tfidf_df.head())

# Calculate the difference in columns: difference
difference = set(count_df.columns) - set(tfidf_df.columns)
print(difference)

# Check whether the DataFrame are equal
print(count_df.equals(tfidf_df))
```

```
   00  000  0000  00000031  000035  00006  ...  والمرضى  هذا  من  محاولات  ما  ยงade
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
2   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
4   0    0     0         0       0      0  ...        0    0   0        0   0      0

[5 rows x 56922 columns]
    00  000  0000  00000031  000035  ...  والمرضى  هذا  من  محاولات  ยงade
```

FIG 7.5: PRINT HEAD OF COUNT_DF

TEST CASE 6



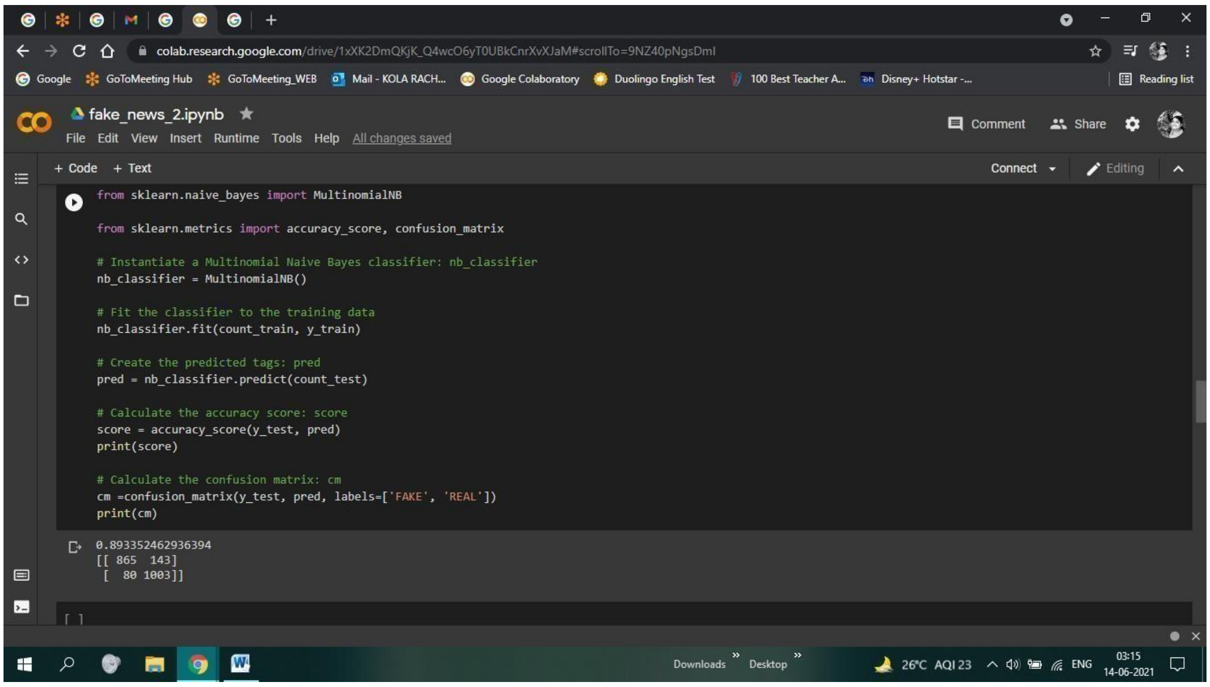
```
print(count_df.equals(tfidf_df))
```

```
   00  000  0000  00000031  000035  00006  ...  والمرضى  هذا  من  محاولات  ما  ยงade
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
2   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
4   0    0     0         0       0      0  ...        0    0   0        0   0      0

[5 rows x 56922 columns]
    00  000  0000  00000031  000035  ...  والمرضى  هذا  من  محاولات  ยงade
0  0.0  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

[5 rows x 56922 columns]
set()
False
```

```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix

# Instantiate a Multinomial Naive Bayes classifier: nb_classifier
```

FIG 7.6: PRINT HEAD OF TFIDF_DF

TEST CASE 7

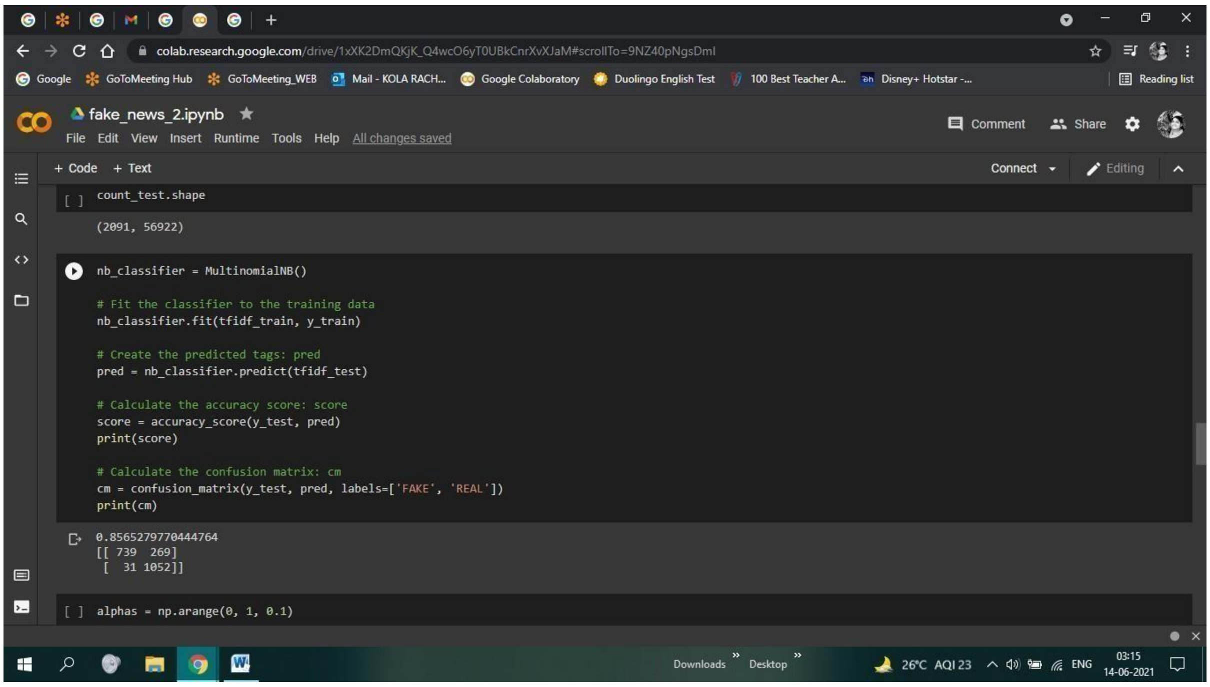


```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix

# Instantiate a Multinomial Naive Bayes classifier: nb_classifier
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(count_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(count_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm =confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)
```

```
0.893352462936394
[[ 865  143]
 [  80 1003]]
```

FIG 7.7: CREATE PREDICTED TAG

TEST CASE 8


colab.research.google.com/drive/1xXK2DmQKjK_Q4wcO6yT0UBkCnrXvXJaM#scrollTo=9NZ40pNgsDml
fake_news_2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect
Editing

```
count_test.shape
```

```
(2091, 56922)
```

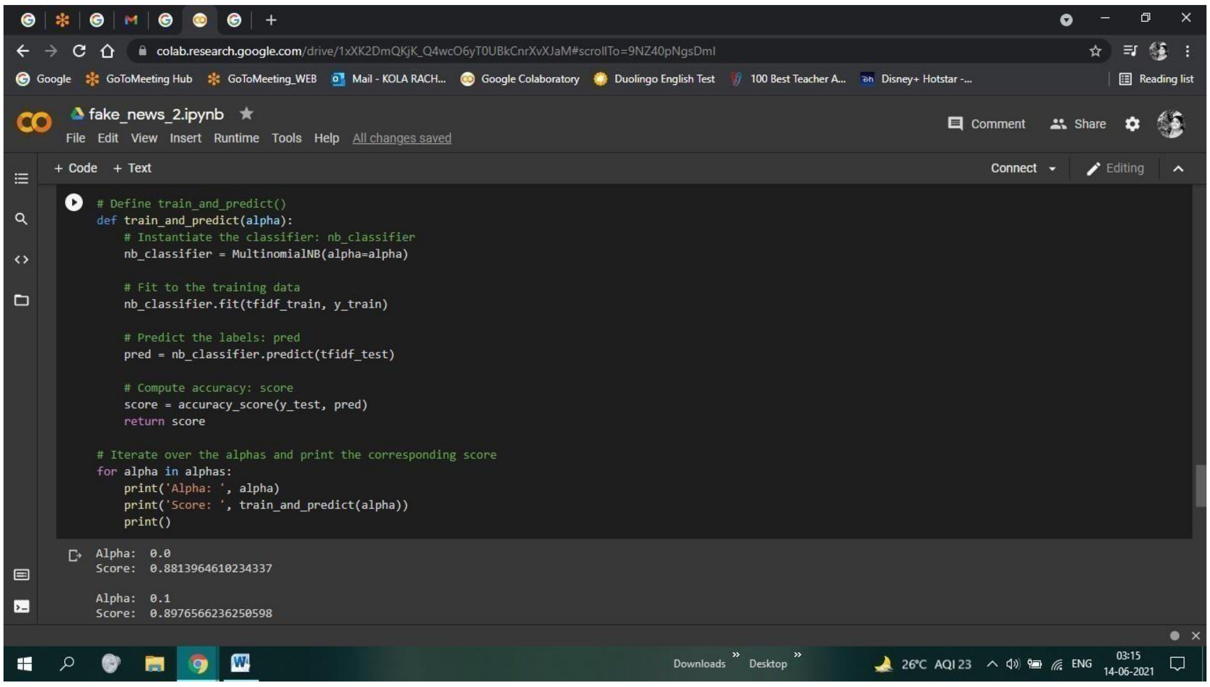
```
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(tfidf_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(tfidf_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)
```

```
0.8565279770444764
[[ 739  269]
 [  31 1052]]
```

```
alphas = np.arange(0, 1, 0.1)
```

FIG 7.8 : CALCULATE THE ACCURACY SCORE

TEST CASE 9

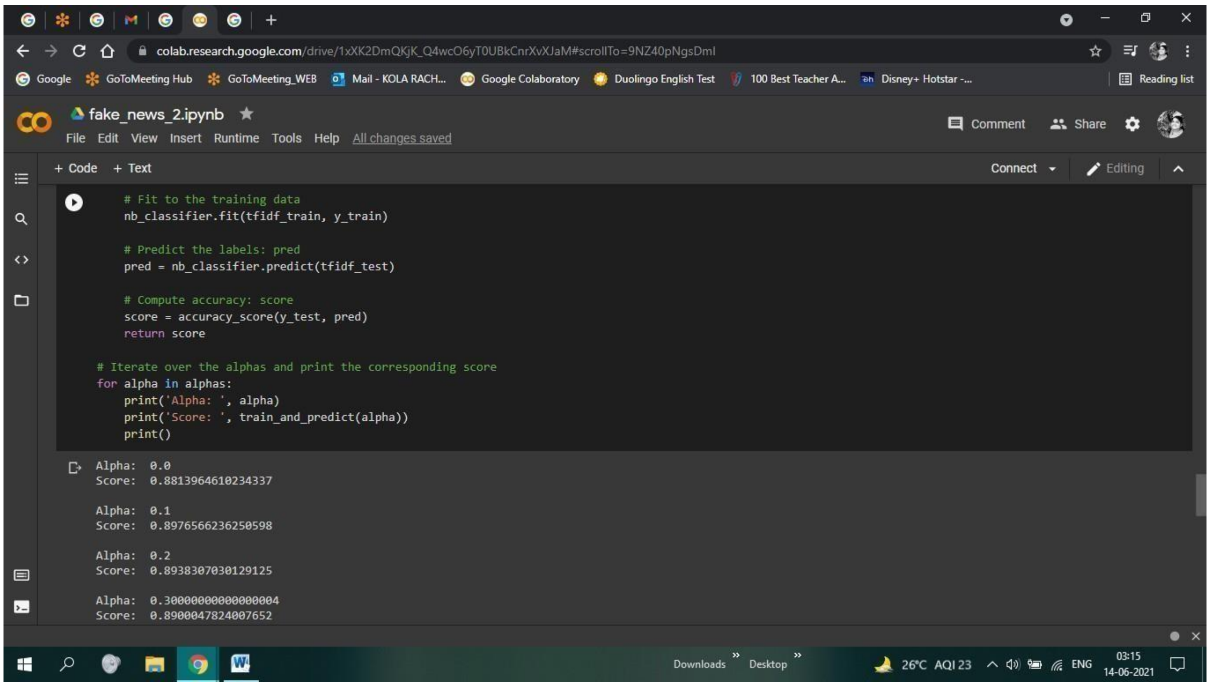

colab.research.google.com/drive/1xXK2DmQKjK_Q4wcO6yT0UBkCnrXvXJaM#scrollTo=9NZ40pNgsDml
fake_news_2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing

```python
# Define train_and_predict()
def train_and_predict(alpha):
    # Instantiate the classifier: nb_classifier
    nb_classifier = MultinomialNB(alpha=alpha)

    # Fit to the training data
    nb_classifier.fit(tfidf_train, y_train)

    # Predict the labels: pred
    pred = nb_classifier.predict(tfidf_test)

    # Compute accuracy: score
    score = accuracy_score(y_test, pred)
    return score

# Iterate over the alphas and print the corresponding score
for alpha in alphas:
    print('Alpha: ', alpha)
    print('Score: ', train_and_predict(alpha))
    print()
```

```
Alpha:  0.0
Score:  0.8813964610234337

Alpha:  0.1
Score:  0.8976566236250598
```

FIG 7.9 : ITERATE OVER THE ALPHAS

TEST CASE 10

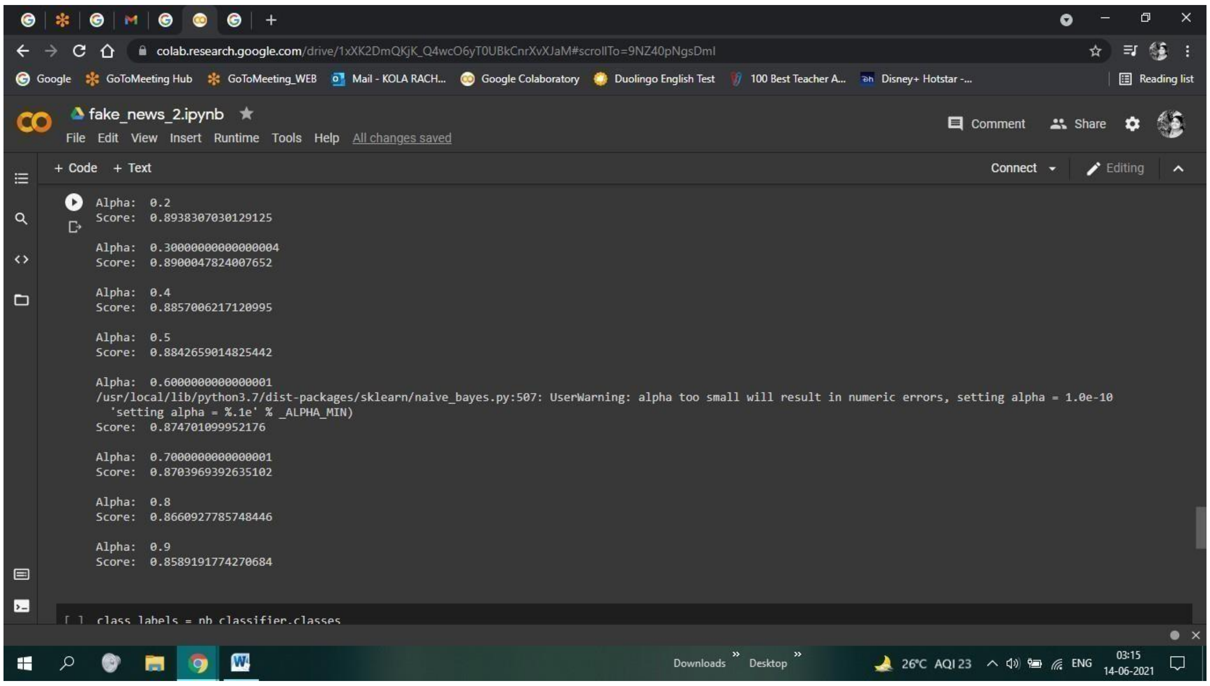


```
        # Fit to the training data
        nb_classifier.fit(tfidf_train, y_train)

        # Predict the labels: pred
        pred = nb_classifier.predict(tfidf_test)

        # Compute accuracy: score
        score = accuracy_score(y_test, pred)
        return score

    # Iterate over the alphas and print the corresponding score
    for alpha in alphas:
        print('Alpha: ', alpha)
        print('Score: ', train_and_predict(alpha))
        print()
```

```
Alpha:  0.0
Score:  0.8813964610234337

Alpha:  0.1
Score:  0.8976566236250598

Alpha:  0.2
Score:  0.8938307030129125

Alpha:  0.30000000000000004
Score:  0.8900047824007652
```

FIG 7.10: PRINT CORRESPONDING SCORE

TEST CASE 11

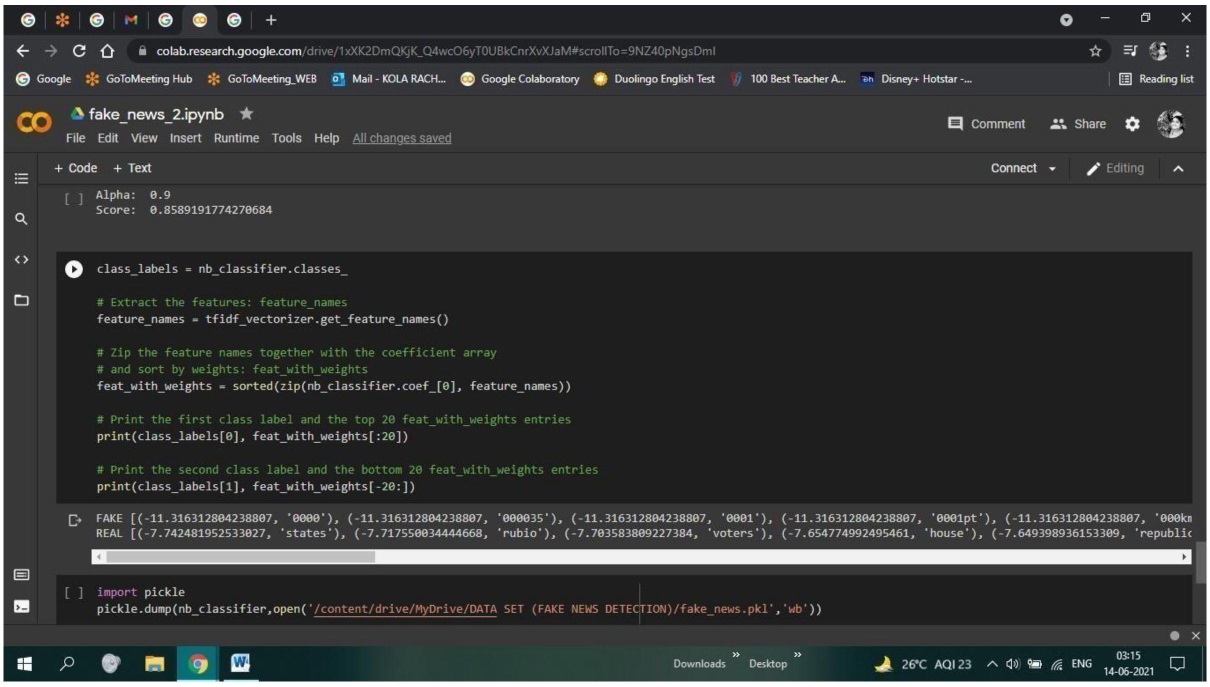


```
Alpha:  0.2
Score:  0.8938307030129125

Alpha:  0.30000000000000004
Score:  0.8900047824007652

Alpha:  0.4
Score:  0.8857006217120995

Alpha:  0.5
Score:  0.8842659014825442

Alpha:  0.6000000000000001
/usr/local/lib/python3.7/dist-packages/sklearn/naive_bayes.py:507: UserWarning: alpha too small will result in numeric errors, setting alpha = 1.0e-10
  'setting alpha = %.1e' % _ALPHA_MIN)
Score:  0.874701099952176

Alpha:  0.7000000000000001
Score:  0.8703969392635102

Alpha:  0.8
Score:  0.8660927785748446

Alpha:  0.9
Score:  0.8589191774270684
```

```
[ ] class_labels = nb_classifier.classes_
```

FIG 7.11: COMPUTE ACCURACY

TEST CASE 12



```
Alpha:  0.9
Score:  0.8589191774270684
```

```
class_labels = nb_classifier.classes_

# Extract the features: feature_names
feature_names = tfidf_vectorizer.get_feature_names()

# Zip the feature names together with the coefficient array
# and sort by weights: feat_with_weights
feat_with_weights = sorted(zip(nb_classifier.coef_[0], feature_names))

# Print the first class label and the top 20 feat_with_weights entries
print(class_labels[0], feat_with_weights[:20])

# Print the second class label and the bottom 20 feat_with_weights entries
print(class_labels[1], feat_with_weights[-20:])
```

```
FAKE [(-11.316312804238807, '0000'), (-11.316312804238807, '000035'), (-11.316312804238807, '0001'), (-11.316312804238807, '0001pt'), (-11.316312804238807, '000km
REAL [(-7.742481952533027, 'states'), (-7.717550034444668, 'rubio'), (-7.703583809227384, 'voters'), (-7.654774992495461, 'house'), (-7.649398936153309, 'republic
```

```
import pickle
pickle.dump(nb_classifier,open('/content/drive/MyDrive/DATA SET (FAKE NEWS DETECTION)/fake_news.pkl','wb'))
```

FIG 7.12: GENERATE PICKLE FILE

CHAPTER 8
OUTPUT SCREENS

OUTPUT 1

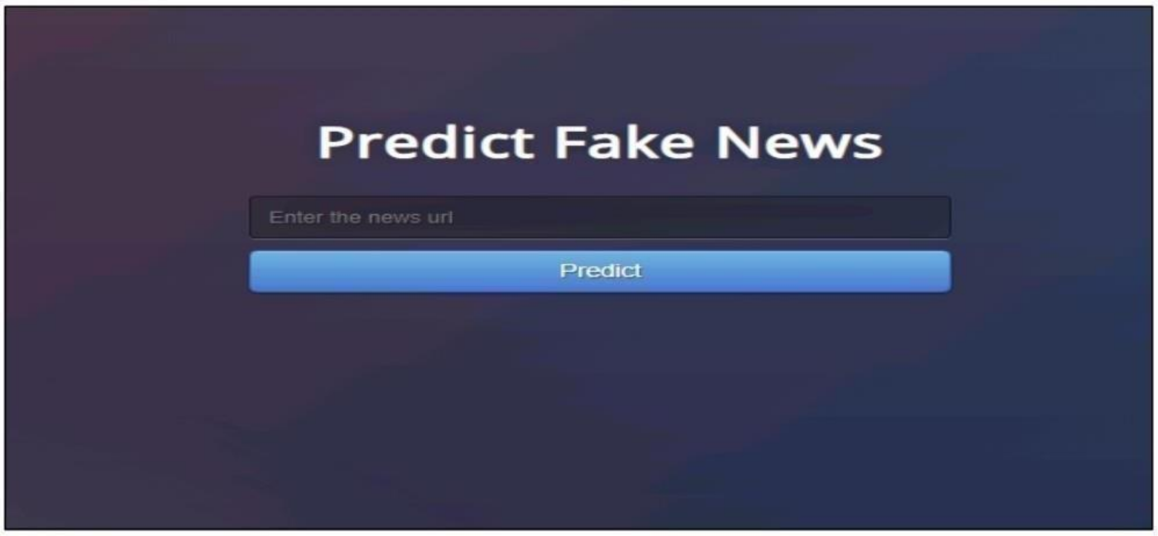
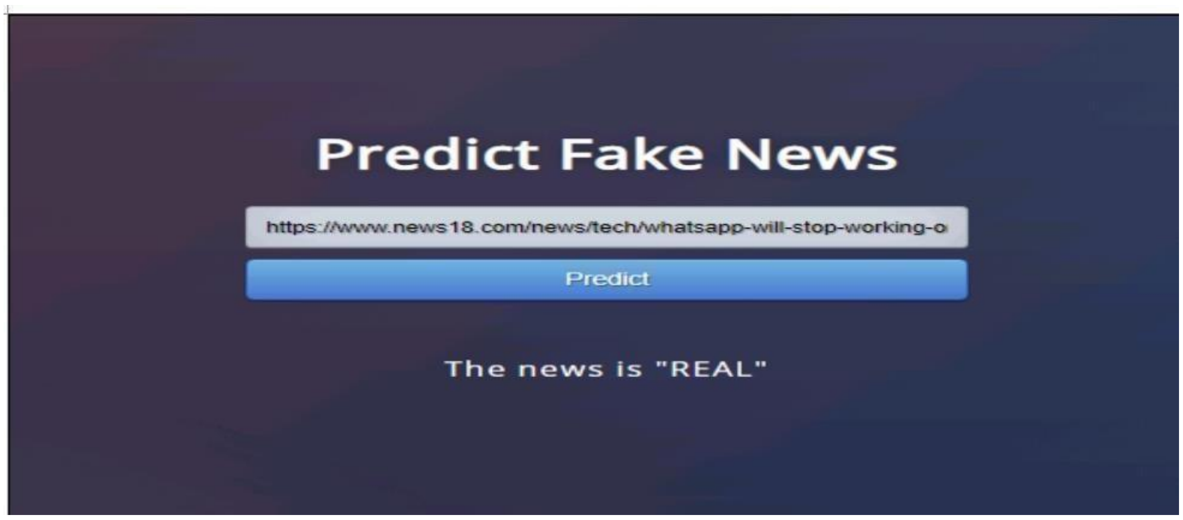


FIG 8.1:HOMEPAGE

OUTPUT 2

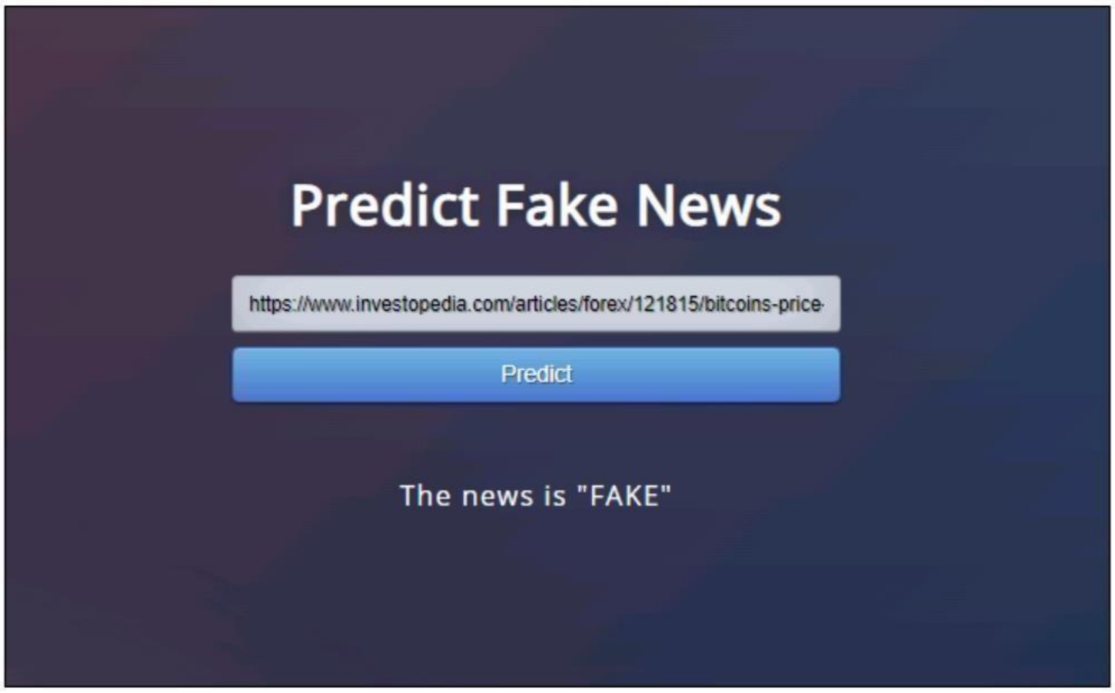

Predict Fake News

https://www.news18.com/news/tech/whatsapp-will-stop-working-o

The news is "REAL"

FIG 8.2 : URL TO PREDICT THE NEWS

OUTPUT 3



Predict Fake News

https://www.investopedia.com/articles/forex/121815/bitcoins-price-

The news is "FAKE"

FIG 8.3: WEBPAGE AFTER PREDICTION THE URL

CHAPTER 9
CONCLUSION

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques. The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news. The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models. The learning models were trained and parameter-tuned to obtain optimal accuracy. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners.

CHAPTER 10
FURTHER ENHANCEMENT

Massive amounts of data gave birth to AI systems that are already producing human-like synthetic texts, powering a new scale of disinformation operation. Based on Natural Language Processing (NLP) techniques, several lifelike text-generating systems have proliferated and they are becoming smarter every day. This year, *OpenAI* announced the launch of GPT-3, a tool to produce text that is so real, that in some cases it’s nearly impossible to distinguish from human writing. GPT-3 can also figure out how concepts relate to each other, and discern context. Tools like this one can be used to generate misinformation, spam, phishing, abuse of legal and governmental processes, and even fake academic essays. We have made a systematic review of the research trends of misinformation detection. Having given a brief review of the literature of MID, we present several new research challenges and techniques of it, including early detection, detection by multimodal data fusion, and explanatory detection. We further investigate the usage of crowd intelligence in MID, including crowd intelligence-based MID models and hybrid human-machine MID models.

CHAPTER 11
BIBLIOGRAPHY

[1] Shao, C., Ciampaglia, G. L., Varol, O., Flammini, A., &Menczer, F. (2017). The spread of fake news by social bots. arXiv preprint arXiv:1707.07592, 96-104.
[2] Hunt, E. (2016). What is fake news? How to spot it and what you can do to stop it. The Guardian, 17.
[3] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22-36.
[4] Ruchansky, N., Seo, S., & Liu, Y. (2017, November). Csi: A hybrid deep model for fake news detection. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 797-806). ACM. [5] Volkova, S., Shaffer, K., Jang, J. Y., &Hodas, N. (2017, July). Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 647-653).
[6] Wang, W. Y. (2017). " liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648.
[7] Reis, J. C., Correia, A., Murai, F., Veloso, A., Benevenuto, F., & Cambria, E. (2019). Supervised Learning for Fake News Detection. IEEE Intelligent Systems, 34(2), 76-81.
[8] Pal, S., Kumar, T. S., & Pal, S. (2019). Applying Machine Learning to Detect Fake News Indian Journal of Computer Science, 4(1), 7- 12.