

A HYBRID MODEL FOR FAKE NEWS DETECTION USING CLICKBAIT

*Report submitted in partial fulfillment of the requirements for
the
B. Tech. degree in Computer Science & Engineering*

By

NAME OF THE STUDENT

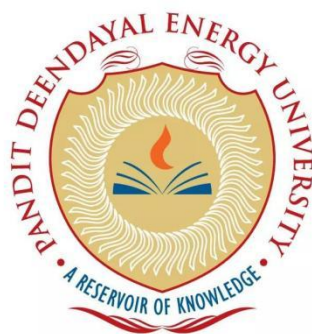
Roll No.

- | | | |
|----|-----------------------|-----------------|
| 1. | <i>Gandhi Deep</i> | <i>18BCP017</i> |
| 2. | <i>Devam Zanzmera</i> | <i>18BCP022</i> |
| 3. | <i>Ronak Makwana</i> | <i>18BCP093</i> |

Under the supervision

Of

Dr. Shakti Mishra



**SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA
May 2022**

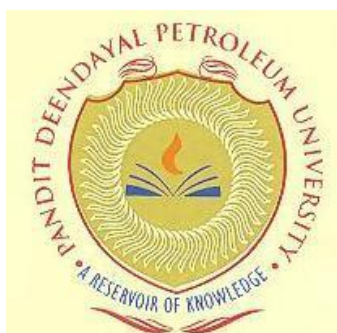
A HYBRID MODEL FOR FAKE NEWS DETECTION USING CLICKBAIT

*Report submitted in partial fulfillment of the requirements for
the B. Tech. degree in Computer Science & Engineering*

By

	NAME OF THE STUDENT	Roll No.
1.	<i>Gandhi Deep</i>	<i>18BCP017</i>
2.	<i>Devam Zanzmera</i>	<i>18BCP022</i>
3.	<i>Ronak Makwana</i>	<i>18BCP093</i>

Under the supervision
Of
Dr. Shakti Mishra



**SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL PETROLEUM UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA
May/June 20**

Approval Sheet

This Project Report entitled “**A Hybrid model for fake news detection using Clickbait**” by **Gandhi Deep (18BCP017)**, **Devam Zanzmera (18BCP022)** and **Ronak Makwana (18BCP093)** is recommended for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiners

Supervisors

Date :

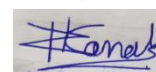
Place :

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the **PANDIT DEENDAYAL ENERGY UNIVERSITY** and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

D. R. Gwella





Gandhi Deep

Address: B3, Alap Flats, Opp caravan school, Kalol, Gandhinagar-382721

Email ID: deep.gce18@sot.pdpu.ac.in

Contact No: +917984841734

Devam Zanzmera

Address : c-701, Shreepad Panorama, Palanpur, Surat, Gujarat 395009

Email ID : devam.zce18@sot.pdpu.ac.in

Contact No : +917265837425

Ronak Makwana

Address : 32, Royal Arcade Society, National Highway, Porbandar, Gujarat 360575

Email ID : ronak.mce18@sot.pdpu.ac.in

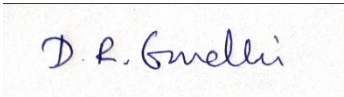

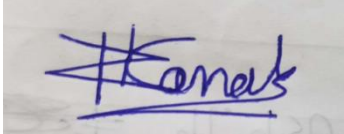
Contact No : 9737867066

Date :

Place :

CERTIFICATE

This is to certify that the report on “ **Fake News Detection**” submitted by the students, as a requirement for the degree in Bachelor of Technology (B. Tech) in Computer Science & Engineering, under my guidance and supervision for the session 2019-2020.

Name of the student	Roll No	Signature
<i>1 Gandhi Deep</i>	<i>18BCP017</i>	
<i>2 Devam Zanzmera</i>	<i>18BCP022</i>	
<i>3 Ronak Makwana</i>	<i>18BCP093</i>	

Date:

Place:

Dr.Shakti Mishra

CONTENTS

ACKNOWLEDGEMENT.....	I
ABSTRACT.....	II
LIST OF FIGURES.....	III
LIST OF TABLES.....	V
ABBREVIATION.....	VI
CHAPTER I - INTRODUCTION.....	1
➤ 1.1 Brief of the problem.....	2
➤ 1.2 Study of the existing solution.....	3
➤ 1.3 Research gaps/problems found with existing solutions.....	4
CHAPTER II - LITERATURE REVIEW.....	5
CHAPTER III - PROPOSED SYSTEM AND ITS REQUIREMENTS.....	8
➤ 3.1 Proposed System.....	8
➤ 3.2 Assumption and dependencies.....	9
➤ 3.3 Specific requirements.....	10
○ 3.3.1 Software.....	10
○ 3.3.2 Hardware.....	11
➤ 3.4 Tools and technologies used.....	11
➤ 3.5 Advantage offered by the proposed system.....	11
CHAPTER IV - DESIGN.....	12
➤ 4.1 System Architecture.....	12
➤ 4.2 Data dictionaries.....	16
○ 4.2.1 Clickbait dataset.....	16
○ 4.2.2 Fake news dataset.....	17
○ 4.2.3 Fake news and clickbait combined dataset.....	18
CHAPTER V - IMPLEMENTATION.....	19
➤ 5.1 Clickbait detector.....	19

○ 5.1.1 Downloading dependencies.....	19
○ 5.1.2 Cleaning and splitting data.....	20
○ 5.1.3 Preprocessing the data.....	21
○ 5.1.4 Making and training the model.....	24
➤ 5.2 Fake news detector.....	25
○ 5.2.1 Downloading dependencies.....	25
○ 5.2.2 Preprocessing data.....	26
○ 5.2.3 One hot representation and word embedding.....	26
○ 5.2.4 Building model.....	27
○ 5.2.5 Splitting data.....	28
○ 5.2.6 Training the model.....	28
➤ 5.3 Combined model of fake news detection with clickbait detection.....	29
○ 5.3.1 Finding correlation between them.....	29
○ 5.3.2 Downloading dependencies.....	31
○ 5.3.3 Cleaning the data.....	32
○ 5.3.4 Splitting the data.....	32
○ 5.3.5 Fake news prediction.....	32
■ 5.3.5.1 Preparing the data according to the model.....	32
■ 5.3.5.2 Preprocessing.....	33
■ 5.3.5.3 One hot encoding and word embedding.....	34
■ 5.3.5.4 Building model.....	34
■ 5.3.5.5 Training the model.....	35
○ 5.3.6 Clickbait detection.....	35
■ 5.3.6.1 Cleaning the data.....	35
■ 5.3.6.2 Prepossessing the data.....	35
■ 5.3.6.3 Generating frequency dictionary.....	37
■ 5.3.6.4 Tf-idf Vectorizer.....	38
■ 5.3.6.5 Training Naive Bayes model.....	39
○ 5.3.7 Generating the hybrid model.....	39

CHAPTER VI - RESULTS.....41

- 6.1 Clickbait detector accuracy.....41
- 6.2 Fake news detector accuracy.....41
- 6.3 Accuracy of hybrid model.....42
- 6.4 Comparison of Models.....43

CHAPTER VII - CONCLUSION.....44

CHAPTER VIII - REFERENCES.....45

LAST PAGE.....47

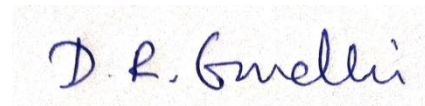
ACKNOWLEDGEMENT

We would like to express our special thanks to our mentor Dr. Shakti Mishra who helped us out throughout the project. The successful completion of the project is the result of the guidance, constant support provided by Dr. Shakti Mishra and we are truly grateful for that as it was an important aspect of our project completion.

Name of the Student

Student Signature

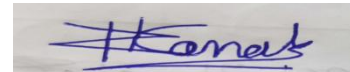
Gandhi Deep

A handwritten signature in blue ink that reads "D. R. Gwelli".

Devam Zanzmera

A handwritten signature in blue ink, appearing to be "Devam Zanzmera", with a circular flourish at the end.

Ronak Makwana

A handwritten signature in blue ink that reads "Ronak" with a horizontal line underneath.

ABSTRACT

We have developed a model to predict fake news which also includes clickbait detection as a parameter. After developing two different models for fake news detection and clickbait detection, we used them to generate a dataset containing both clickbait labels and fake news labels. Now the first step was to find the correlation between two labels. So we did chi-square test to find that out. After establishing that they are correlated we have run model of clickbait on the heading of the dataset and model of the fake news detection on the content of the dataset. Then we have combined results of both the models to generate a hybrid model through a regex equation. Through this process our accuracy was increase by 1.3% i.e. from 82.8% to 84.1%. So we can safely conclude that adding more parameters such clickbait detection can increase the model's accuracy a easily.

LIST OF FIGURES

Figure No.	Title	Page No
1	Architecture of Fake News Detector	12
2	Architecture of Clickbait Detector	13
3	Architecture to make combined dataset	14
4	Architecture of Hybrid dataset	15
5	Clickbait dataset	16
6	Fake news dataset	17
7	Combined dataset	18
8	Importing dependencies for Clickbait detector	19
9	Loading dataset	20
10	Filtering dataset	20
11	Splitting the dataset	21
12	Converting text into lowercase	21
13	Removing spaces from text	22
14	Removing punctuation	22
15	Removing stopwords	23
16	Removing Numbers	23
17	Lemmetization of data	24
18	Tokenizing the text	24

19	Training TF-IDF	25
20	Importing Naive Bayes model	25
21	Training Naive Bayes with TF-IDF	25
22	Importing dependencies for fake news detection	26
23	Stemming and removing stopwords	26
24	Doing one hot coding and word embedding	27
25	Building Bidirectional LSTM	27
26	Splitting the dataset	28
27	Training the model	28
28	Finding correlation	29
29	Chart showing the relation between fake news and clickbait	30
30	Importing dependencies for combined model	31
31	Cleaning data for combined model	32
32	Splitting dataset	32
33	Filtering dataset for model training	33
34	Stemming and removing stopwords	33
35	Word embedding and one hot encoding	34
36	Building Bi-LSTM model	34
37	Training the model	35
38	Cleaning data for clickbait detection	35
39	Tokenizing and lowercasing words	36
40	Removing numbers and punctuation	36

41	Removing stop words and spaces	37
42	Lemmatizing words	37
43	Making frequency dictionary	38
44	Converting words into vectors using TF-IDF	39
45	Training Naive Bayes	39
46	Generating Hybrid model	40
47	Clickbait detector accuracy	41
48	Fake news detector accuracy	41
49	Accuracy of Hybrid model	42

LIST OF TABLE

Table No	Title	Page No
1	Clickbait dataset attributes	16
2	Fake news dataset attributes	17
3	Combined dataset attributes	18
4	Model Comparison Table	43

ABBREVIATION

Abbreviation	Full Form
NLP	Natural Language Processing
TF-IDF	Term Frequency- Inverse Document Frequency
CFG	Context Free Grammar
LIWC	Linguistic Inquiry and Word Count
SVM	Support Vector Machine
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short - term Memory
bi-LSTM	Bidirectional long short - term Memory
POS tagging	Part of Speech tagging
DNN	Deep Neural Network

1. INTRODUCTION

Over the past few years the consumption of the internet has increased drastically. This change has affected us and the ways we do our day to day life. One such change is how we gather the information. We rely more on the articles and blogs present on the internet rather than the newspaper. This habit has caused concern among the people as the news on the internet might not be authentic and could lead to misinformation or misunderstanding and might lead to some unfortunate consequences.

Also with the rapid spread of the internet everyone is connected through social media platforms and sharing information has never been much easier, we can send information with just one click to an ample number of people around us and they can share with the people they know and this can go further on[1]. If misinformation is spread in such a way then it can be drastic. Due to their busy lives nowadays people generally do not have time to verify the news and believe it to be correct. This is how misinformation is spread nowadays like wildfire. This misinformation is generally known as fake news, and it is a major problem.

Fake news articles are intentionally crafted and verified articles to mislead the readers. It can also be defined as a type of yellow journalism or propaganda. Fake news has been used by traditional news agencies since 1439[2][3]. The main reason behind this is the unethical reporters who are paid to publish these articles. Fake news often fabricates headlines to increase its readership and mislead people. These headlines are called clickbaits.

Every news agency has to compete with a lot of agencies for reader's attention, also they generate money from the clicks of the users. So to grab user's attention they employ various techniques to make the headline catchy[4]. Clickbaits exploit the cognitive phenomenon known as the Curiosity Gap[5]. They generally provide forward references in the heading to generate enough information gaps in order to increase user's curiosity. This will in the long term affect users' attention abilities.

As clickbaits are highly associated with fake news, detection of clickbait will also give us an idea about whether the news is fake or real. Due to these reasons, we have also made an attempt to predict clickbait and use them for the detection of fake news.

1.1 BRIEF OF THE PROBLEM:

We have tried to implement a hybrid model for fake news detection which will also take into consideration clickbait as a parameter, as we have seen in one research paper that it will increase the accuracy by 1-2% so have tried to implement a hybrid model on the basis of that. In this project, our main goal is to predict whether the given news is fake news or not. To achieve this we have first predicted if the heading is a clickbait or not and then we have predicted if the content is fake news or not. Then we concatenated two results through a regex equation to predict the output of the model.

1.2 STUDY OF THE EXISTING SOLUTIONS:

For detection of clickbait various methods are used ranging from detection on the basis of similarities between content and heading and on the basis of heading only. For the first method, NLP models are used. TF-IDF, CountVectorizer, Word2Vec, Doc2Vec, and N-grams are used for the classification of the dataset and cosine similarity, etc are used to predict the similarity between the heading and the content. For the second approach, various NLP models like n-grams+TF-IDF, CFG+TF-IDF, LIWC, word2vec, Doc2Vec, N-gram, CountVectorizer for classification of the heading and machine learning models and deep learning models like Naive Bayes, SVM, Linear regression, CNN, RNN, LSTM, etc are used for comparison of the classification and prediction.

For fake news detection NLP, Machine learning models and deep learning models are used in various Research papers, they have also used TF-IDF, CountVectorizer, Word2Vec, Doc2Vec, and N-grams, porter stemmer for classification and used hybrid models for comparison of the classification and prediction. Some research papers have used different hybrid models where they predict fake news on the basis of one parameter from one model and in the second model they use another parameter for fake news prediction and then they will concatenate the result using a Deep learning model or using regex equations. These are the various approaches that we have seen in different research papers.

1.3 RESEARCH GAP/PROBLEMS FOUND WITH EXISTING SOLUTIONS:

There are a lot of parameter from which we can predict fake news, we have seen in a review paper that display all such parameter, it includes sentence structures, stop words, hyperbolic, Common phrases, subject, determiners, possessive cases, N-grams, POS tagging, systemic N-grams, word patterns, clickbait, etc. These are the parameters that can be used for prediction, if we try to include 2-3 of them then also we will gain a lot more accuracy then the models without that. We have gone through a lot of research papers but they have not used these types of parameters. So we have tried to implement the model with clickbait detection.

2. LITERATURE REVIEW

Various models and methodology has been used to find fake news, which varies from NLP models, Machine learning models to Multiple deep learning models. One of the important aspects that we find while going through Research papers is that the researchers found the presence of numerous clickbait titles detrimental to the perceived credibility of a source[6].

With a significant amount of clickbait present in a news article, they have also tested how much clickbait headlines can affect the model and the accuracy of the model was increased by 2-3%. They have applied clickbait detection on each source of the news and then combined the detection of each source. They have used linear SVM for clickbait detection and feed-forward deep averaging network (FFDAN) for fake news detection. One important aspect that we thought is missing here is the credibility score of the source, as nowadays there are many propaganda-driven news websites.

Another approach for clickbait detection that we read was to classify the content into related or unrelated content using n-gram lemmatization and then classify it to agree or disagree based on linear regression[7].

As clickbait detection is one of the important aspects of fake news detection to increase its accuracy we have gone through the various research papers and articles for clickbait detection. There are various models and methodologies used for this also, if we classify them into two parts then it would be as follows. The first approach, detecting clickbait

on the similarity between the title and the content within. If the clickbait headings are used there is most likely it was only used to grab our attention to it and the content within will be different. Various NLP models are used for this approach like Tf-Idf, Bag of Words, n-gram etc for classification and then Machine learning models like SVM, Naive Bayes, etc for comparison of the classification[8]. The second approach is to predict the clickbait based on the headlines only, this approach is fast and effective with various NLP and machine learning models. There are many parameters that we can extract from a headline for its classification like “Starting with Number”, “punctuation patterns”, “hyperbolic words”, “N-grams”, “POS tagging for both”, etc[4].

For classification of the clickbait NLP models like ”n-grams”, “TF-IDF”, “n-grams+TF-IDF”, “word2vec”, “Doc2Vec”, “Bag of words”, “count vectorizer”, etc are used for the classification and Machine learning models and deep learning models like naive Bayes, SVM, Linear Regression, CNN, RNN, LSTM, etc are used for comparison of the classification and prediction. The approach that we have used is the classification of headlines based on the TF-IDF vectorizer and the N-gram approach. For prediction and comparison of the classification, we have used the Naive Bayes model.

Once we have calculated clickbait successfully, we have to predict fake news now. Mainly fake news detection studies can be generally grouped into content-based and propagation-based methods, Depending on whether the approaches detect fake news by exploring its content or by exploring how it propagates on social networks[9].

In content-based, various machine learning models, deep learning models and NLP models are used. Different NLP methods are used for vectorization like word2vec,

fasttext, pos tagging, etc. We have used porter stemmer for dividing sentences into words and have used One_hot for vectorization of the words list. With more computational capabilities and to handle massive datasets, deep learning models present a finer performance over traditional text mining techniques and machine learning techniques. Convolutional Neural Network (CNN) and Recurrent Neural networks (RNN), are widely explored Deep Neural Network (DNN) architectures to handle various NLP tasks[10]. They used word2vec for vectorization of the sentence and then used CNN, RNN, Unidirectional LSTM and Bidirectional LSTM in their model, and the most accurate was the Bi-LSTM model.

A research paper has compared machine learning models and deep learning models and based on it concluded that data with a huge number of articles is beneficial for LSTM and it will perform very well in it also there is a most likely chance that it will overcome overfitting. Also, they have concluded that a naive Bayes model with n-gram will also perform equally well as deep learning models[3].

Hybrid models with one deep learning model predicting based on one parameter and another model predicting on the basis of another parameter/s are also widely used now, a same model is used in a model that we have referred to, they have used a CNN for the author profile validation and Bi-LSTM prediction on the basis of content then author have concatenated both to a single vector to generate one output based on that[2].

3. PROPOSED SYSTEM AND ITS REQUIREMENTS

3.1 PROPOSED SYSTEM:

Our objective is to detect fake news on the basis of a given dataset and also include clickbait as a parameter for the prediction of the fake news. For this we have to develop a model that can predict clickbait headings on the basis of the data given and also we have to develop a different model for fake news detection on the basis of the content of a given dataset. Then we have to combine the model's output on the basis of a Regex equation.

Also we were not able to find a proper dataset that will contain article headings, content, clickbait label and fake news label, so we take two separate datasets for the models of fake news and clickbait detection and then we have to save those models.

To generate the required dataset we have saved the model of clickbait detection and applied it to the dataset containing fake news labels and generated a predictive column clickbait. Now we have applied the Chi_square test as both label columns contain binary. After applying the test we got p value equal to 0.0000000018603149 which is far lower than the determining score of 0.5, in the chi-square test if the value is less than 0.5 the variables are dependent on each other. Hence we have established that they both are dependent on each other.

After having the required dataset we have to develop a code containing two models for fake news and clickbait detection respectively, which have detected the output on the

basis of the same dataset, after having the prediction for the both we have seen the relation between them. After carefully analyzing the outputs we have constructed a Regex equation to concatenate the results of two predictions. The final array of prediction of fake news that we get is being compared with the fake news column in the dataset.

This is how we have designed our model, due to this we have seen an increased accuracy of 1-2 % in the prediction depending on the size of the input data.

3.2 ASSUMPTION AND DEPENDENCIES:

We do not have as such much dependencies although there are few things that we have to take care of:

1. To increase the efficiency of clickbait detection we have used the max feature in the Tf-IdfVectorizer, when we are classifying the data into vectors. The max features that we have considered is equal to the number of words that are coming in the dataset at least 2 times. So we have to calibrate the max feature according to that dataset.
2. There is no proper data set available so in order to generate the required dataset we need to build and save the model of clickbait first and then run on the fake news label containing the dataset.
3. Inorder to generate the the hybrid model and do regex equation we need to properly:

- Remove any row that contain errors or unwanted data prior to applying two models for fake news and clickbait detection respectively
- Divide it to train and test dataset before applying both models.

The main reason behind this is that we need an equal number of features in the output array from both the models and also in the same order, to apply the regex equation properly.

3.3 SPECIFIC REQUIREMENTS:

3.3.1 Software:

If you are running code in google colab the latest version of the browser id preferable and if running code locally in an IDE then the latest version of the IDE is preferable. In order to run the code/program locally in an IDE, programming language python must be downloaded on the computer. Furthermore, the computer must have the following libraries downloaded to run the programme.

- Numpy
- Pandas
- Keras
- Tensorflow
- Seaborn
- NLTK
- Sklearn
- String

- matplotlib
- Re
- os.

3.3.2 Hardware:

There is no need for the specific hardware to run the programme however, high-end Graphical processing unit and processor will execute the code in less time.

3.4 TOOLS AND TECHNOLOGIES USED:

We have used Google Colab to , which is an online IDE to write and run the code.

3.5 Advantages offered by proposed system:

- We see that fake news and clickbait are related to each other and clickbait can also be used as a parameter in predicting fake news.
- Able to verify that including more parameters that are relevant to fake news can affect the accuracy of the model, so including them will definitely increase the accuracy of the model.
- You will be able to detect fake news with 84% accuracy only on the basis of content and the title.

4. DESIGN:

4.1 SYSTEM ARCHITECTURE:

1. Our first step is to design a system to predict fake news solely on the basis of the content of the articles so we have developed a code for that purpose, the architecture for the same is:

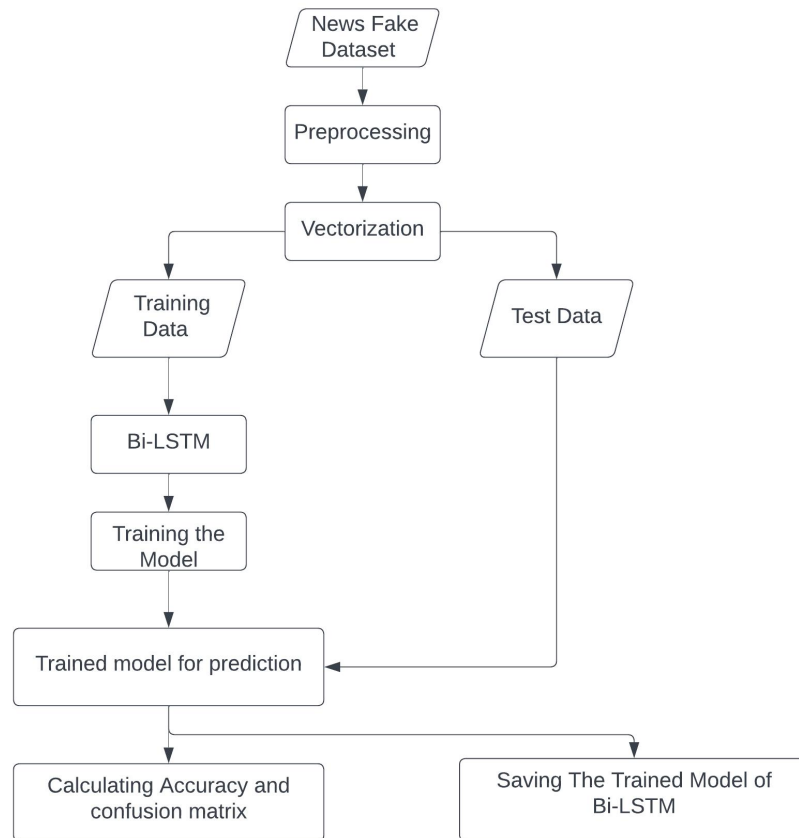


Figure 1: Architecture of Fake News detector

2. After constructing these architectures we got an accuracy of 82%. We have now developed an architecture to predict clickbait on the basis of article headings from a different dataset and saved the trained model for further usage. The architecture for the same is:

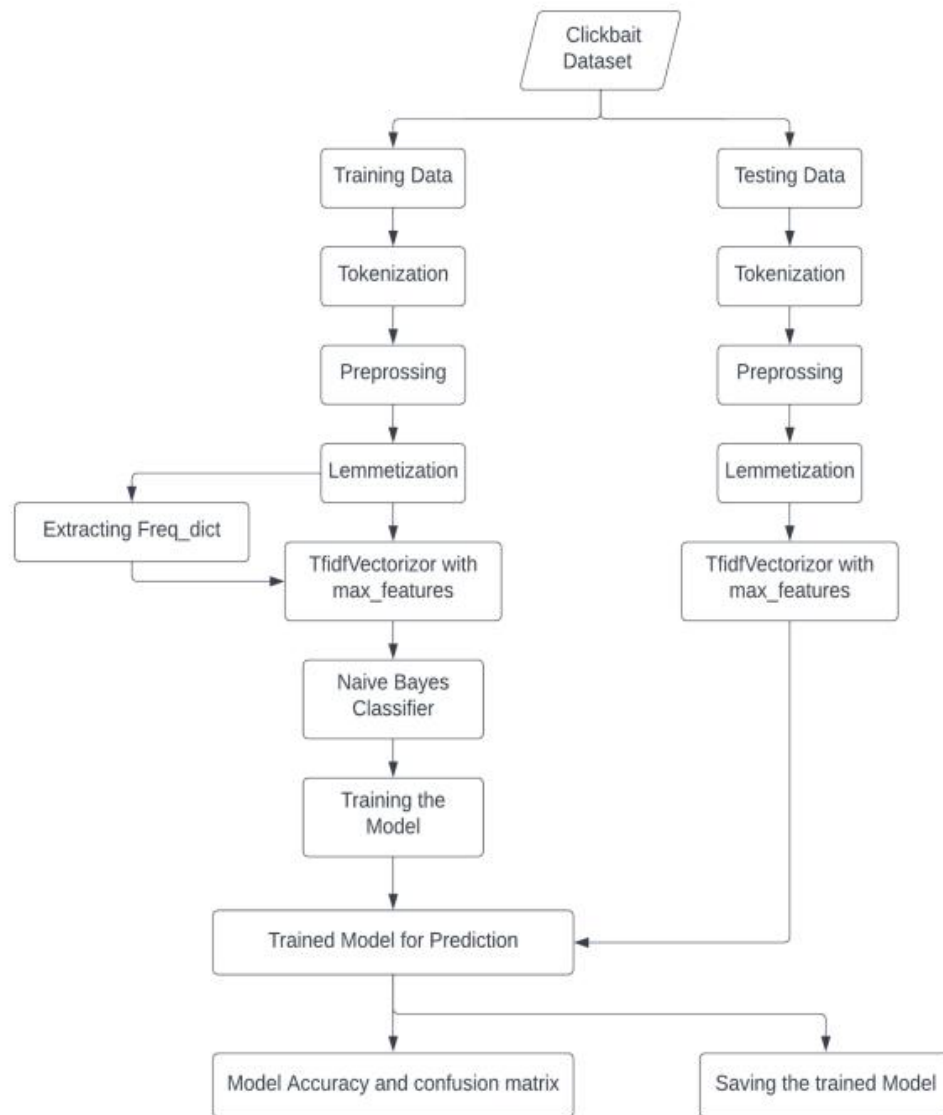


Figure 2: Architecture of Clickbait detector

3. After we have to model saved for clickbait detection, we will use the fake news dataset to predict clickbait values for each article and we have embedded this as a new column in the dataset as click bait label. The architecture for the same is:

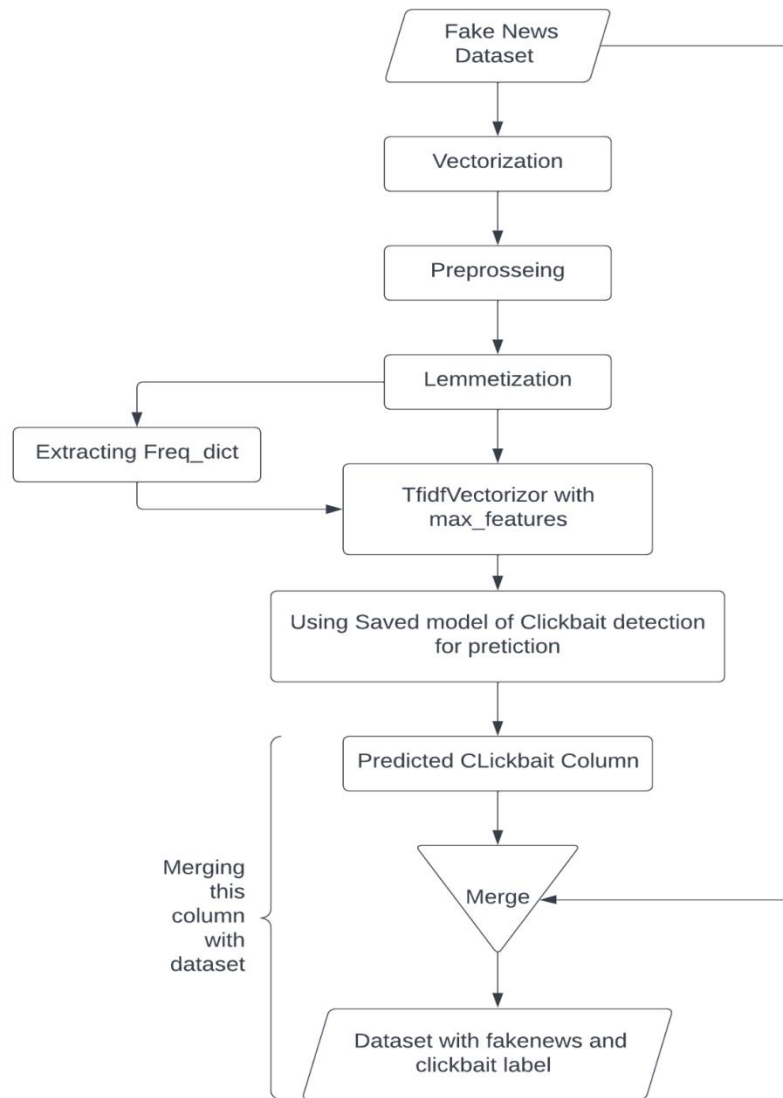


Figure 3: Architecture to make combined dataset

- After getting the required dataset we have trained both the models of fake news detection and clickbait detection on the same dataset and find the correlation between them based on the predicted values and have generated Regex equation according to that to merge output of both models to generate final predicted array of fake news labels. The architecture for the same is as follows:

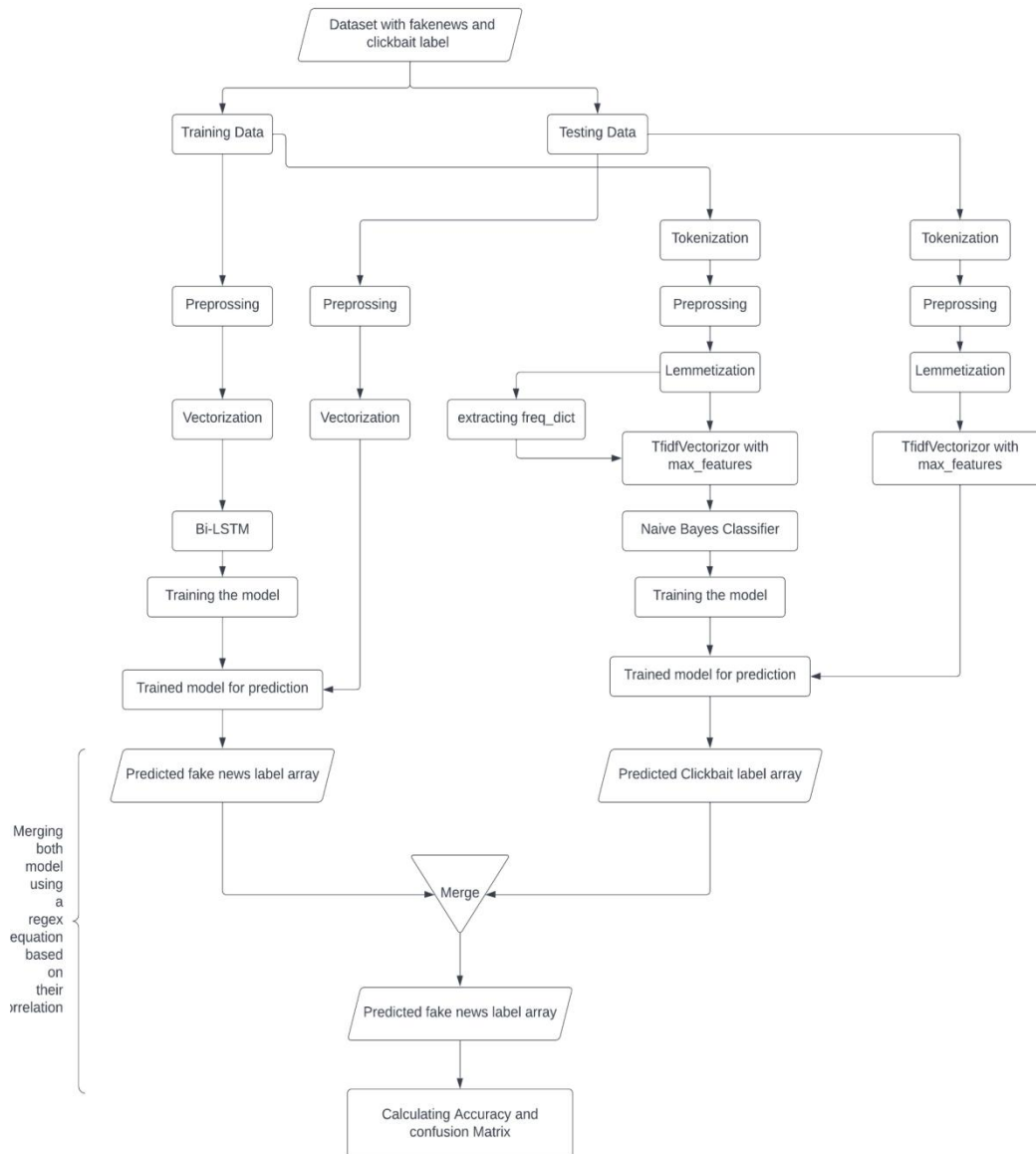


Figure 4: Architecture of Hybrid model

4.2 DATA DICTIONARIES:

4.2.1 Clickbait Dataset:

The Clickbait dataset consists of 4 different columns ID, Title, Text and Clickbait. The first column ID provides each entry a different number so that we can identify. The next column title has the title of all the articles which are present in the database. Text column has the content of the title and the last column has values 0s indicating it is not a clickbait and 1 indicating it is a clickbait. Format of the dataset is as follows:

Sr. no.	Column Name	Type	Description
1	Id	Integer data	Id number of each article
2	Title	Text data	Article Heading
3	Text	Text data	Article Content
4	Clickbait	Binary data	Label whether the heading is clickbait or not

Tabel 1 : Clickbait dataset attributes

```
#1 means it is a clickbait and 0 means it is not a clickbait
cb_data= pd.read_csv('train.csv')
cb_data.head(10)
```

	id	title	text	label
0	0	China and Economic Reform: Xi Jinping's Track ...	Economists generally agree: China must overhau...	news
1	1	Trade to Be a Big Topic in Theresa May's U.S. ...	LONDON—British Prime Minister Theresa May said...	news
2	2	The Top Beaches In The World, According To Nat...	Beaches come in all sorts of shapes and sizes ...	clickbait
3	3	Sheriff's Report Provides New Details on Tamir...	A timeline of what happened after Tamir Rice, ...	clickbait
4	4	Surgeon claiming he will transplant volunteer'...	An Italian neurosurgeon who has claimed for mo...	news
5	5	This Is How Differently Priced Spanx Can Actua...	Spanx does not do the thing I thought it did. ...	news
6	6	Samantha Bee and Jane Pauley Are Breaking the ...	"Jane Pauley! You're all in leather," said Sam...	news
7	7	Krauthammer: Syria Strike 'Total Contradiction...	Charles Krauthammer called President Trump's 5...	news
8	8	Rust Belt voters made Trump president. Now the...	The United States is making more things than e...	news
9	9	As Illegal Outpost Cleared, Israeli PM Netanya...	As Jewish settlers and protesters were removed...	news

Figure 5: Clickbait dataset

4.2.2 FAKE NEWS DATASET:

The fake news detection dataset has five columns ID, Title, Author and Text and Label. The column ID provides each row a number to identify them individually. The column title has the title of the article and the text column consists of the content of the article. The name of the author is provided in the author column and the last column Label has values 0s and 1s indicating whether the article is fake or not, where 0 suggests that the article is real and 1 the opposite. Format of the dataset:

Sr. no.	Column Name	Type	Description
1	Id	Integer data	Id of each article
2	Title	Text data	Article Titles
3	Author	Text data	Article's authors name
4	Text	Text data	Article content
5	Label	Binary data	Label whether its fake news or not

Tabel 2: Fake news dataset attributes

```
#1 means the news is fake and 0 means that the news is real
df.head()
```

	id		title	author	text	label
0	0		House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucas	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1		FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2		Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3		15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1
4	4		Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Figure 6: Fake news dataset

4.2.3 Clickbait and Fake news combined dataset:

The combined dataset has a total of six columns. Different Identification numbers are written in the ID column. Title and the inside text of the article is provided in the title and text columns respectively. Name of the author is in the author column. The details of whether the text is fake or not and the title is clickbait or not is given in the Label and Clickbait column respectively, where 0 indicates it is not clickbait and real news and 1 dictates that the title is clickbait and the news is fake. Format of the dataset is:

Sr. no.	Column Name	Type	Description
1	Id	Integer data	Id of each article
2	Title	Text data	Article Titles
3	Author	Text data	Article's authors name
4	Text	Text data	Article content
5	Label	Binary data	Label whether its fake news or not
6	Clickbait	Binary data	Label whether the heading is clickbait or not.

Tabel 3: Combined dataset attributes

```
df = pd.read_csv('FakeNews_and_Clickbait_Detection.csv')
df.head(5)
```

Unnamed: 0	id	title	author	text	label	clickbait
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucas	House Dem Aide: We Didn't Even See Comey's Let...	1	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1	0
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	0

Figure 7: Combined dataset

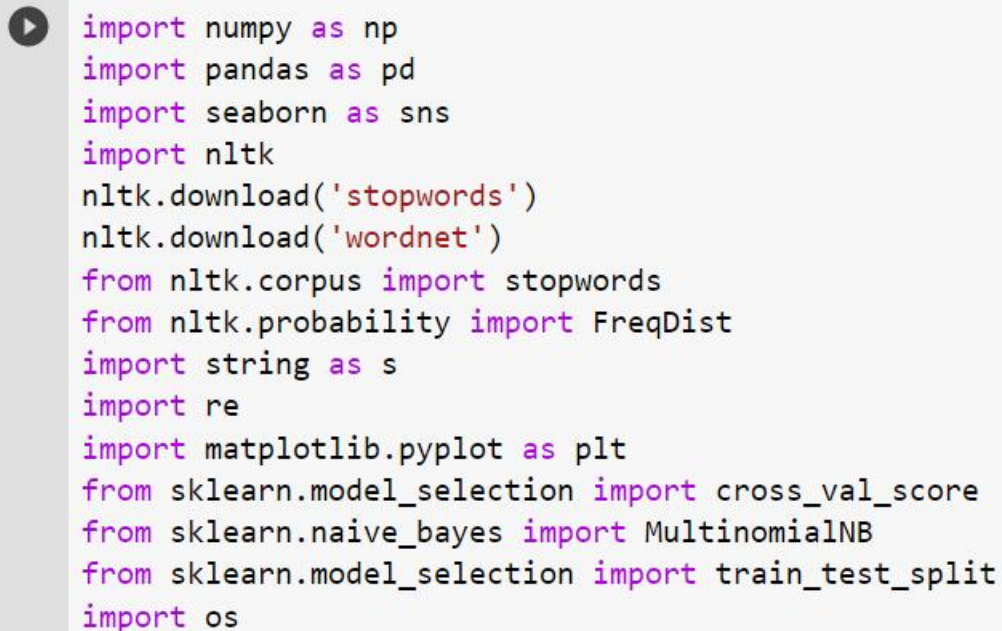
5. IMPLEMENTATION

5.1 CLICKBAIT DETECTOR:

As discussed in the proposed system our first step is to create the clickbait detector to detect clickbait. For that we have used Naive Bayes Classifier and N-gram model for better outcome.

5.1.1 Downloading Dependencies:

To make the clickbait detector first of all we have to download the dependencies described in chapter 3.

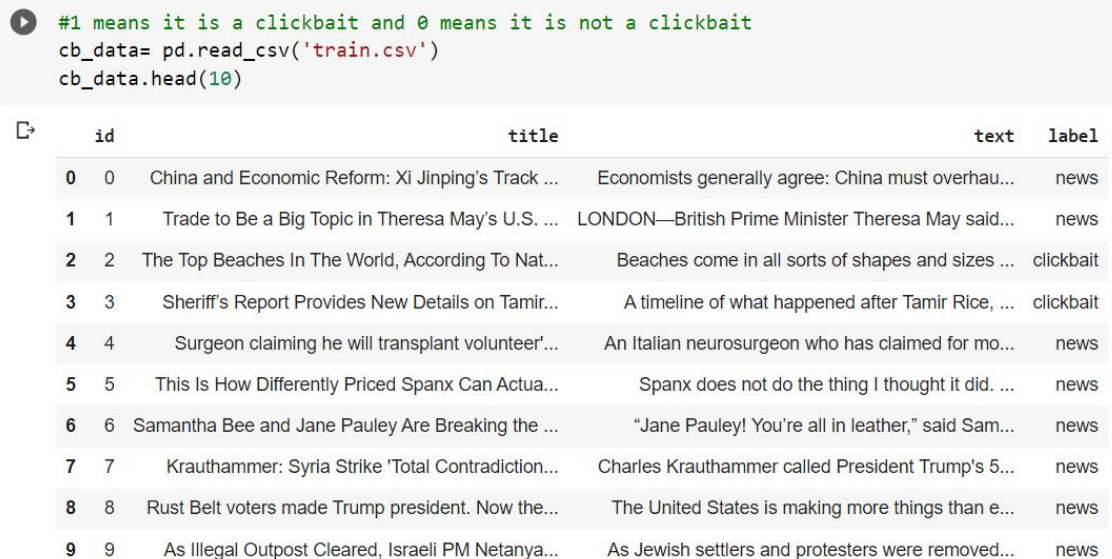


```
import numpy as np
import pandas as pd
import seaborn as sns
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.probability import FreqDist
import string as s
import re
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
import os
```

Figure 8: Importing Dependencies for Clickbait detector

5.1.2 Cleaning and splitting the data:

Then we will load our dataset into dataframe and remove the rows which have null values and will also remove the columns like 'Author', 'Text' and 'label' as we do not require that data to make the clickbait detector. Then we will split the data for training and testing, here we have used 75% of the data for testing and the remaining 25% will be used for testing.



```
#1 means it is a clickbait and 0 means it is not a clickbait
cb_data= pd.read_csv('train.csv')
cb_data.head(10)
```

	id	title	text	label
0	0	China and Economic Reform: Xi Jinping's Track ...	Economists generally agree: China must overhau...	news
1	1	Trade to Be a Big Topic in Theresa May's U.S. ...	LONDON—British Prime Minister Theresa May said...	news
2	2	The Top Beaches In The World, According To Nat...	Beaches come in all sorts of shapes and sizes ...	clickbait
3	3	Sheriff's Report Provides New Details on Tamir...	A timeline of what happened after Tamir Rice, ...	clickbait
4	4	Surgeon claiming he will transplant volunteer'...	An Italian neurosurgeon who has claimed for mo...	news
5	5	This Is How Differently Priced Spanx Can Actua...	Spanx does not do the thing I thought it did. ...	news
6	6	Samantha Bee and Jane Pauley Are Breaking the ...	"Jane Pauley! You're all in leather," said Sam...	news
7	7	Krauthammer: Syria Strike 'Total Contradiction...	Charles Krauthammer called President Trump's 5...	news
8	8	Rust Belt voters made Trump president. Now the...	The United States is making more things than e...	news
9	9	As Illegal Outpost Cleared, Israeli PM Netanya...	As Jewish settlers and protesters were removed...	news

Figure 9: Loading dataset

```
[ ] x=cb_data.title
    y=cb_data.clickbait
    train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.25,random_state=2)
```

Figure 10: Splitting dataset

```
[ ] print("No. of elements in training set")
    print(train_x.size)
    print("No. of elements in testing set")
    print(test_x.size)
```

```
No. of elements in training set
15181
No. of elements in testing set
5061
```

Figure 11: Training and testing dataset

5.1.3 Preprocessing the data:

In the next step we are preprocessing the training and the testing dataset. With the help of different methods provided by the different libraries we are tokenizing the text, converting it into the lowercase, removing punctuation marks, stopwords, numbers and spaces and lemmatizing the text using wordnet lemmatizer present in the NLTK library.

```
[ ] def clickbait_tokenization(text):
    temp=text.split()
    return temp
train_x=train_x.apply(clickbait_tokenization)
test_x=test_x.apply(clickbait_tokenization)
```

Figure 12: Tokenizing the text

```
[ ] def clickbait_lower casing(temp):  
    new_lst=[]  
    for i in temp:  
        i=i.lower()  
        new_lst.append(i)  
    return new_lst  
train_x=train_x.apply(clickbait_lower casing)  
test_x=test_x.apply(clickbait_lower casing)
```

Figure 13: Converting text into lowercase

```
▶ def clickbait_remove_punctuations(temp):  
    new_lst=[]  
    for i in temp:  
        for j in s.punctuation:  
            i=i.replace(j, '')  
        new_lst.append(i)  
    return new_lst  
train_x=train_x.apply(clickbait_remove_punctuations)  
test_x=test_x.apply(clickbait_remove_punctuations)
```

Figure 14: Removing punctuations

```

▶ def clickbait_remove_numbers(temp):
    nodig_lst=[]
    new_lst=[]
    for i in temp:
        for j in s.digits:
            i=i.replace(j,'')
        nodig_lst.append(i)
    for i in nodig_lst:
        if i!='':
            new_lst.append(i)
    return new_lst
train_x=train_x.apply(clickbait_remove_numbers)
test_x=test_x.apply(clickbait_remove_numbers)

```

Figure 15: Removing numbers

```

▶ def clickbait_remove_stopwords(temp):
    stop=stopwords.words('english')
    new_lst=[]
    for i in temp:
        if i not in stop:
            new_lst.append(i)
    return new_lst

train_x=train_x.apply(clickbait_remove_stopwords)
test_x=test_x.apply(clickbait_remove_stopwords)

```

Figure 16: Removing stopwords

```
[ ] def clickbait_remove_spaces(temp):
    new_lst=[]
    for i in temp:
        i=i.strip()
        new_lst.append(i)
    return new_lst
train_x=train_x.apply(clickbait_remove_spaces)
test_x=test_x.apply(clickbait_remove_spaces)
```

Figure 17: Removing spaces

```
▶ lemmatizer=nltk.stem.WordNetLemmatizer()
def clickbait_lemmatization(temp):
    new_lst=[]
    for i in temp:
        i=lemmatizer.lemmatize(i)
        new_lst.append(i)
    return new_lst
train_x=train_x.apply(clickbait_lemmatization)
test_x=test_x.apply(clickbait_lemmatization)
```

Figure 18: Lemmetization of data

5.1.4 Making and Training the model:

As we have pre processed the data now we will vectorize the text using tf-idf and then train the Naive Bayes model. After the training we will further use the uni-gram model to enhance the accuracy we are getting.


```
[ ] #analyzer="char_wb", ngram_range=(2,4)
    from sklearn.feature_extraction.text import TfidfVectorizer
    tfidf=TfidfVectorizer(max_features=6296)
    train_1=tfidf.fit_transform(train_x)
    test_1=tfidf.transform(test_x)
```

Figure 19: Training TF-IDF

```
[ ] NB_MN=MultinomialNB()
```

Figure 20: Importing Naive Bayes model

```
▶ NB_MN.fit(train_arr,train_y)
  pred=NB_MN.predict(test_arr)
  print('first 20 actual labels: ',test_y.tolist()[:20])
  print('first 20 predicted labels: ',pred.tolist()[:20])
```

first 20 actual labels: [1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]
 first 20 predicted labels: [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Figure 21: Training Naive Bayes with TF-IDF

5.2 FAKE NEWS DETECTOR:

As discussed in the proposed system our second step is to create the Fake news detector to detect fake news. For that we have used Bidirectional lstm for better outcome.

5.2.1 Downloading Dependencies:

To make the fake news detector first of all we have to download the dependencies as mentioned in chapter 3.

```
[ ] import numpy as np
import pandas as pd
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt
```

Figure 22: Importing dependencies for fake news detection

5.2.2 Preprocessing data:

Then we will load our dataset into dataframe and remove the rows which have null values and will also remove the columns like ‘Author’, and ‘clickbait’ as we do not require that data to make the fake news detector. We also remove ‘stop words’ and stemming words and split words.

```
corpus = []
for i in range(0, len(msg)):

    review = re.sub('[^a-zA-Z]', ' ', msg['text'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

Figure 23: Stemming and removing stopwords

5.2.3 One Hot representation and word embedding:

In this step ,we convert our categorical data into numeric data with help of one hot encoding as we are going to use Bidirectional lstm. For that, first of all we create

vocabulary whose size is 5000 as we created. After that, we simply use ontot representation. Then, we convert all sequences in the same length which is 20.

```
[ ] onehot_representation = [one_hot(words, voc_size) for words in corpus]

[ ] sent_length = 20
    embeded_docs = pad_sequences(onehot_representation, padding = "pre", maxlen = sent_length)

[ ] embeded_docs[0]

array([ 653, 2509, 3023, 2108, 1765,  815, 4039, 1860, 2859, 3542, 4545,
        4941, 3362, 4257, 2509, 1869, 4236,  328, 1448, 3362], dtype=int32)
```

Figure 24: Doing one hot coding and word embedding

5.2.4 Building Model:

As we have preprocessed data and converted it into vector form and then we build Bidirectional LSTM model. For that, we add three layers : 'Embedding Layer', 'Bidirectional layer' and 'Dense layer'.

```
#Building a model
dimension = 40
model = Sequential()
model.add(Embedding(voc_size, dimension, input_length = sent_length))
model.add(Bidirectional(LSTM(100)))
model.add(Dense(1, activation = "sigmoid"))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 20, 40)	200000
bidirectional (BidirectionalLSTM)	(None, 200)	112800
dense (Dense)	(None, 1)	201

=====

Total params: 313,001
Trainable params: 313,001
Non-trainable params: 0

None

Figure 25: Building Bi-LSTM model

5.2.5 Splitting the data:

we will split the data for training and testing, here we have used 80% of the data for testing and the remaining 20% will be used for testing.

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0.20, random_state=42)
```

Figure 26: Splitting the dataset

5.2.6 Training the model:

We divide our dataset into batches. The size of the batches are 64. We train our dataset in 20 epochs.

```
#Training a model
model.fit(X_final_train, y_final_train, validation_data = (X_final_test, y_final_test), epochs = 20, batch_size = 64)

Epoch 1/20
229/229 [=====] - 16s 53ms/step - loss: 0.3915 - accuracy: 0.8048 - val_loss: 0.3273 - val_accuracy: 0.8567
Epoch 2/20
229/229 [=====] - 11s 46ms/step - loss: 0.2341 - accuracy: 0.9036 - val_loss: 0.3470 - val_accuracy: 0.8534
Epoch 3/20
229/229 [=====] - 11s 47ms/step - loss: 0.1676 - accuracy: 0.9361 - val_loss: 0.3681 - val_accuracy: 0.8482
Epoch 4/20
229/229 [=====] - 11s 47ms/step - loss: 0.1186 - accuracy: 0.9575 - val_loss: 0.5024 - val_accuracy: 0.8409
Epoch 5/20
229/229 [=====] - 11s 48ms/step - loss: 0.0902 - accuracy: 0.9676 - val_loss: 0.5931 - val_accuracy: 0.8376
Epoch 6/20
229/229 [=====] - 12s 50ms/step - loss: 0.0640 - accuracy: 0.9772 - val_loss: 0.6690 - val_accuracy: 0.8269
Epoch 7/20
229/229 [=====] - 11s 47ms/step - loss: 0.0499 - accuracy: 0.9835 - val_loss: 0.6989 - val_accuracy: 0.8359
Epoch 8/20
229/229 [=====] - 11s 47ms/step - loss: 0.0363 - accuracy: 0.9881 - val_loss: 0.8752 - val_accuracy: 0.8332
Epoch 9/20
229/229 [=====] - 11s 49ms/step - loss: 0.0268 - accuracy: 0.9911 - val_loss: 0.9177 - val_accuracy: 0.8305
Epoch 10/20
229/229 [=====] - 11s 47ms/step - loss: 0.0249 - accuracy: 0.9910 - val_loss: 1.0222 - val_accuracy: 0.8357
Epoch 11/20
229/229 [=====] - 11s 47ms/step - loss: 0.0168 - accuracy: 0.9949 - val_loss: 1.0192 - val_accuracy: 0.8362
Epoch 12/20
229/229 [=====] - 11s 47ms/step - loss: 0.0139 - accuracy: 0.9954 - val_loss: 1.2054 - val_accuracy: 0.8332
Epoch 13/20
229/229 [=====] - 11s 47ms/step - loss: 0.0170 - accuracy: 0.9939 - val_loss: 1.1749 - val_accuracy: 0.8422
Epoch 14/20
229/229 [=====] - 11s 47ms/step - loss: 0.0172 - accuracy: 0.9947 - val_loss: 1.0948 - val_accuracy: 0.8340
Epoch 15/20
229/229 [=====] - 11s 47ms/step - loss: 0.0092 - accuracy: 0.9973 - val_loss: 1.2794 - val_accuracy: 0.8329
Epoch 16/20
229/229 [=====] - 11s 47ms/step - loss: 0.0059 - accuracy: 0.9986 - val_loss: 1.5165 - val_accuracy: 0.8250
Epoch 17/20
229/229 [=====] - 11s 47ms/step - loss: 0.0095 - accuracy: 0.9971 - val_loss: 1.2789 - val_accuracy: 0.8305
Epoch 18/20
229/229 [=====] - 11s 47ms/step - loss: 0.0169 - accuracy: 0.9943 - val_loss: 1.1557 - val_accuracy: 0.8288
Epoch 19/20
229/229 [=====] - 11s 47ms/step - loss: 0.0091 - accuracy: 0.9973 - val_loss: 1.2884 - val_accuracy: 0.8337
Epoch 20/20
229/229 [=====] - 11s 47ms/step - loss: 0.0031 - accuracy: 0.9992 - val_loss: 1.5639 - val_accuracy: 0.8387
<keras.callbacks.History at 0x7f6996bee550>
```

Figure 27: Training the model

5.3 COMBINED MODEL OF FAKE NEWS DETECTION WITH CLICKBAIT DETECTION:

5.3.1 Finding correlation between them:

As we have merged the dataset it's very much important for us to find the correlation between them both the labels so we have first applied the chi_square method to find out if they are correlated or not. These variables are binary so we can't use other normal methods, so we have to check correlations using the chi-square method only.

In the chi_square method if the value of p-value is less than 0.5 then the variables are correlated. So we have applied the same in our dataset and this is the result that we got:

```
contingency= pd.crosstab(df['label'], df['clickbait'])
print(contingency)
```

clickbait	0	1
label		
0	7745	2616
1	5607	2317

```
[ ] contingency_pct = pd.crosstab(df['label'], df['clickbait'], normalize='index')
contingency_pct
```

clickbait	0	1
label		
0	0.747515	0.252485
1	0.707597	0.292403

```
[ ] # Chi-square test of independence.
from scipy.stats import chi2_contingency
chi_sq, p_value, dof, expected = chi2_contingency(contingency)
# Print the p-value
print(format(p_value, '.16f'))
```

0.0000000018603149

Figure 28: Finding correlation

So the p value is 0.0000000018603149, therefore both clickbait labels and fake news are correlated. Now we have also generated a graph which contains the representation of data in a simpler way. We have designed it in such a way that we will give us the number of fake news or not if clickbait is present and number of fake news or true news if clickbait is not present. On the basis of this only we have designed our regex equation. The graph is given below:

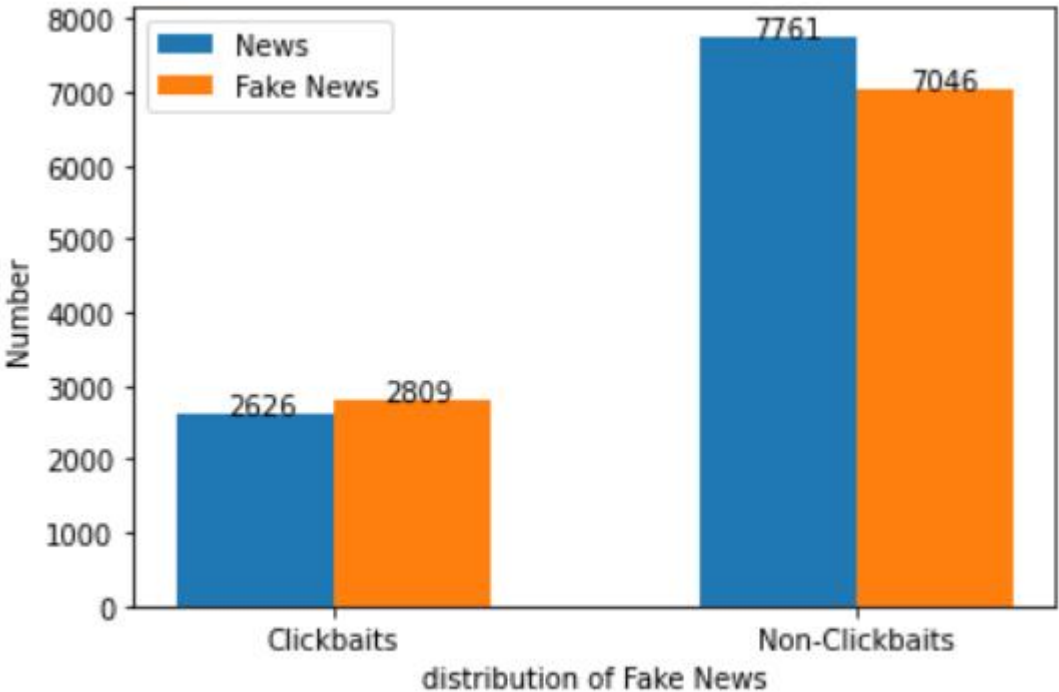


Figure 29: Chart showing the relation between fake news and clickbait

So if the clickbait is present in the database then there is 52 percent chance that the news is fake and if it is not present 47% chance that it's fake news, although due to the non availability of proper dataset the amount of clickbait in this dataset are far less then the non-clickbait headings so if we have gotten a proper dataset then we would get more

difference. From this graph we can say that chances of being fake news increases by 5% if clickbait is present.

So now we have the proper understanding of how to develop our hybrid model. So we will proceed with that.

5.3.2 Downloading Dependencies:

To make the fake news detector first of all we have to download the dependencies as mentioned in chapter 3.

```
import numpy as np
import pandas as pd
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import seaborn as sns
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.probability import FreqDist
import string as s
import re
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
import os
nltk.download('punkt')
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

Figure 30: Importing dependencies for combined model

5.3.3 Cleaning the data:

We have to clean the rows containing null data and also separate them into two dataframes one containing title, content and author's name and another containing labels of fake news and clickbait.

```
[ ] df = df.dropna()

[ ] x = df.drop(df.columns[[0,5,6]], axis = 1)
    y = pd.DataFrame()
    y['label'] = df['label'].values
    y['clickbait'] = df['clickbait'].values
```

Figure 31: Cleaning data for combined model

5.3.4 Splitting the data:

We will split the data for training and testing, here we have used 80% of the data for training and the remaining 20% will be used for testing.

```
[ ] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=42)
```

Figure 32: Splitting dataset

5.3.5 Fake news detection:

5.3.5.1: Preparing data according to the model:

We have selected the dolumns important to us from the given test and train sets and prepare it for preprocessing:


```
[ ] y_train_fakenews = y_train['label']
    y_test_fakenews = y_test['label']

[ ] voc_size = 9500
    message_train = X_train.copy()
    message_test = X_test.copy()
    message_train.reset_index(inplace = True)
    message_test.reset_index(inplace = True)
```

Figure 33: Filtering data for model training

5.3.5.2: Preprocessing:

Now we will generate the corpus from the content of the article, where we will remove stopwords, stemming and split the sentences into lists. We will do this for both the test and train dataset.

```
▶ corpus_train = []
  for i in range(0, len(message_train)):

    review = re.sub('[^a-zA-Z]', ' ', message_train['text'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus_train.append(review)

[ ] corpus_test = []
  for i in range(0, len(message_test)):

    review = re.sub('[^a-zA-Z]', ' ', message_test['text'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus_test.append(review)
```

Figure 34: Stemming and removing stop words

5.3.5.3 One Hot encoding and word embedding:

In this step ,we convert our categorical data into numeric data with help of one hot encoding as we are going to use Bidirectional lstm. For that, first of all we create vocabulary whose size is 5000 as we created. After that,we simply use ontot representation.Then, we convert all sequences in the same length which is 20.

```
▶ voc_size = 5000
from tensorflow.keras.layers import Bidirectional
onehot_repr_train = [one_hot(words, voc_size) for words in corpus_train]
onehot_repr_test = [one_hot(words, voc_size) for words in corpus_test]
sent_length = 20
embedded_docs_train = pad_sequences(onehot_repr_train, padding = "pre", maxlen = sent_length)
embedded_docs_test = pad_sequences(onehot_repr_test, padding = "pre", maxlen = sent_length)
```

Figure 35: Word embedding and one hot encoding

5.3.5.4 Building Model:

As we have preprocessed data and converted it into vector form and then we build Bidirectional LSTM model. For that, we add three layers : ‘Embedding Layer’, ‘Bidirectional layer’ and ‘Dense layer’.

```
▶ #Building a model
dims = 40
model = Sequential()
model.add(Embedding(voc_size, dims, input_length = sent_length))
model.add(Bidirectional(LSTM(100)))
model.add(Dense(1, activation = "sigmoid"))
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 20, 40)	380000
bidirectional_2 (Bidirectional)	(None, 200)	112800
dense_2 (Dense)	(None, 1)	201

=====
Total params: 493,001
Trainable params: 493,001
Non-trainable params: 0

Figure 36: Building bidirectional LSTM- model

5.3.5.5 Training the model:

We divide our dataset into batches. The size of the batches are 64. We train our dataset in 20 epochs.

```
#Training a model
model.fit(X_final_train, y_final_train, validation_data = (X_final_test, y_final_test), epochs = 20, batch_size = 64)
```

Figure 37: Training the model

5.3.6 Clickbait Detection:

5.3.6.1: Cleaning of the data:

Now we will take the required columns for our model for both the training set and testing set.

```
[ ] x_train_clickbait=X_train.title
    x_test_clickbait=X_test.title
    y_train_clickbait=y_train.clickbait
    y_test_clickbait=y_test.clickbait
```

Figure 38: Cleaning data for clickbait detection

5.3.6.2 Preprocessing the data:

In the next step we are preprocessing the training and the testing dataset. With the help of different methods provided by the different libraries we are tokenizing the text, converting it into the lowercase, removing punctuation marks, stopwords, numbers and spaces and lemmatizing the text using wordnet lemmatizer present in the NLTK library.

```
[ ] def clickbait_tokenization(text):
    temp_lst=text.split()
    return temp_lst
x_train_clickbait=x_train_clickbait.apply(clickbait_tokenization)
x_test_clickbait=x_test_clickbait.apply(clickbait_tokenization)

[ ] def clickbait_lower casing(temp_lst):
    new_lst=[]
    for i in temp_lst:
        i=i.lower()
        new_lst.append(i)
    return new_lst
x_train_clickbait=x_train_clickbait.apply(clickbait_lower casing)
x_test_clickbait=x_test_clickbait.apply(clickbait_lower casing)
```

Figure 39: Tokenizing and lowercasing text

```
[ ] def clickbait_remove_punctuations(temp_lst):
    new_lst=[]
    for i in temp_lst:
        for j in s.punctuation:
            i=i.replace(j,'')
        new_lst.append(i)
    return new_lst
x_train_clickbait=x_train_clickbait.apply(clickbait_remove_punctuations)
x_test_clickbait=x_test_clickbait.apply(clickbait_remove_punctuations)

[ ] def clickbait_remove_numbers(temp_lst):
    nodig_lst=[]
    new_lst=[]
    for i in temp_lst:
        for j in s.digits:
            i=i.replace(j,'')
        nodig_lst.append(i)
    for i in nodig_lst:
        if i!='':
            new_lst.append(i)
    return new_lst
x_train_clickbait=x_train_clickbait.apply(clickbait_remove_numbers)
x_test_clickbait=x_test_clickbait.apply(clickbait_remove_numbers)
```

Figure 40: Removing numbers and punctuation

```
[ ] def clickbait_remove_stopwords(temp_lst):
    stop=stopwords.words('english')
    new_lst=[]
    for i in temp_lst:
        if i not in stop:
            new_lst.append(i)
    return new_lst

x_train_clickbait=x_train_clickbait.apply(clickbait_remove_stopwords)
x_test_clickbait=x_test_clickbait.apply(clickbait_remove_stopwords)

[ ] def clickbait_remove_spaces(temp_lst):
    new_lst=[]
    for i in temp_lst:
        i=i.strip()
        new_lst.append(i)
    return new_lst

x_train_clickbait=x_train_clickbait.apply(clickbait_remove_spaces)
x_test_clickbait=x_test_clickbait.apply(clickbait_remove_spaces)
```

Figure 41: Removing stop words and spaces

```
[ ] lemmatizer=nltk.stem.WordNetLemmatizer()
def clickbait_lemmatization(temp_lst):
    new_lst=[]
    for i in temp_lst:
        i=lemmatizer.lemmatize(i)
        new_lst.append(i)
    return new_lst

x_train_clickbait=x_train_clickbait.apply(clickbait_lemmatization)
x_test_clickbait=x_test_clickbait.apply(clickbait_lemmatization)

[ ] x_train_clickbait=x_train_clickbait.apply(lambda x: ''.join(i+' ' for i in x))
x_test_clickbait=x_test_clickbait.apply(lambda x: ''.join(i+' ' for i in x))
```

Figure 42: Lemmatizing words

5.3.6.3 Generating frequency dictionary:

After doing lemmatization now we will generate a frequency dictionary for each word.

After generating this dictionary we will count the number of words that are occurring

minimum 2 times in the corpus. We will use this while tokenization in the “max_features” parameter.

```
[ ] freq_dist={}
    for i in x_train_clickbait:
        x=i.split()
        for j in x:
            if j not in freq_dist.keys():
                freq_dist[j]=1
            else:
                freq_dist[j]+=1

#freq_dist
max_v=0
for i in freq_dist.values():
    if(i > max_v):
        max_v = i

ct=0
for i in freq_dist.values():
    if(i > 2):
        ct+=1
ct
```

6105

Figure 43: Making Frequency dictionary

5.3.6.4 Tfidf Vectorization:

We will use TfidfVectorizer for converting the data into vectors, we take a parameter called “max_features” which will only take in consideration that number of features and generate vectors based on that number of features only. This will help us in increasing the accuracy of the model. We will take max_feature equal to 6105.

```
[ ] #analyzer="char_wb", ngram_range=(2,4)
    from sklearn.feature_extraction.text import TfidfVectorizer
    tfidf=TfidfVectorizer(max_features=6105)
    clickbait_train_1=tfidf.fit_transform(x_train_clickbait)
    clickbait_test_1=tfidf.transform(x_test_clickbait)

[ ] clickbait_train_arr=clickbait_train_1.toarray()
    clickbait_test_arr=clickbait_test_1.toarray()
```

Figure 44: Converting words into vectors using TF-IDF

5.3.6.5 Training Naive Bayes Model:

Now we will train the naive bayes model according to our data for the prediction of clickbait.

```
▶ NB_MN=MultinomialNB()

[ ] NB_MN.fit(clickbait_train_arr,y_train_clickbait)
    clickbait_pred=NB_MN.predict(clickbait_test_arr)
    print('first 20 actual labels: ',y_test_clickbait.tolist()[0:20])
    print('first 20 predicted labels: ',clickbait_pred.tolist()[0:20])

first 20 actual labels: [0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]
first 20 predicted labels: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1]
```

Figure 45: Training Naive Bayes

5.3.7 Generating a hybrid model :

After running both the models we got two arrays of equal length containing prediction of clickbait and fake news. Now we know that if clickbait is not present that there is 47% chance of the news being fake so in the regex equation we have taken a weighted

avg both to generate the final array, where we have given fake news prediction accuracy of 55% and 45% to clickbait.

```
[ ] for i in range(len(Fake_news_Binary_pred)):
    if(clickbait_news_pred[i] == 1):
        #Fake_news_pred[i] = 0.60*Fake_news_pred[i] + 0.40*clickbait_news_pred[i]
        pass
    else:
        Fake_news_pred[i] = 0.55*Fake_news_pred[i] + 0.45*clickbait_news_pred[i]

Final_binary = []
for i in range(len(Fake_news_pred)):
    if(Fake_news_pred[i]>=0.5):
        Final_binary.append(1)
    else:
        Final_binary.append(0)
```

Figure 46: Generating Hybrid model

6. RESULTS

6.1 CLICKBAIT DETECTOR ACCURACY:

According to our proposed method we have to find if the title is clickbait or not. The clickbait detector which we have made using the Naive Bayes and Tf-Idf while using parameter such as max_features and unigram model that got us an accuracy of 83%.

```
Accuracy of the model in percentage
83.08634657182375 %
Confusion Matrix
[[3653  36]
 [ 820 552]]
Classification Report
```

	precision	recall	f1-score	support
0	0.82	0.99	0.90	3689
1	0.94	0.40	0.56	1372
accuracy			0.83	5061
macro avg	0.88	0.70	0.73	5061
weighted avg	0.85	0.83	0.81	5061

Figure 47: Clickbait detector accuracy

6.2 FAKE NEWS DETECTOR ACCURACY:

As our second step is to detect the fake news based on the context of the article. We have used Bidirectional-lstm to detect the fake news and we have got around 82.8%.

```
[ ] print(accuracy_score(y_test, y_pred))

0.8282745419742958
```

Figure 48: Fake news detector accuracy

6.3 ACCURACY OF HYBRID MODEL:

The accuracy of this model is increased by 1.2% and the accuracy is near to 84.1%, so embedding clickbait as a parameter is beneficial for the models accuracy.

```
[83] from sklearn.metrics import classification_report
      print(classification_report(y_final_test, Nfinal))
```

	precision	recall	f1-score	support
0	0.86	0.84	0.85	2082
1	0.80	0.82	0.81	1575
accuracy			0.83	3657
macro avg	0.83	0.83	0.83	3657
weighted avg	0.83	0.83	0.83	3657

```
[ ] Nfinal = np.where(final > 0.5, 1, 0)
     cf_matrix=confusion_matrix(y_final_test, Nfinal)
     print(accuracy_score(y_final_test, Nfinal))
     print(cf_matrix)
```

```
0.8408531583264971
[[1837  245]
 [ 337 1238]]
```

Figure 49: Hybrid model accuracy

6.4 COMPARISON OF MODELS:

From the hybrid model we got an accuracy of 84.1% which is approx 1.3% higher then the previous model so comparing all the classification tables values will give us a fair idea about the final model.

Sr. No.	Models	Precision		Recall		F1 score		Accuracy
		1	0	1	0	1	0	
1.	Bi-LSTM Model	0.85	0.80	0.84	0.81	0.85	0.80	0.8282
2.	Hybrid Model	0.86	0.80	0.84	0.82	0.85	0.81	0.8408

Tabel 4: Model Comparison Table

From the above table can conclude that in precision, recall and F1-score are increasing by 1% in all of them, whether its in prediction of fake news(1) or prediction of true news(0).

7. CONCLUSION

Our approach to add clickbait as the parameter has helped us to increase the accuracy of the fake news detection. Increasing the accuracy of the proposed system even more would be the next step in line and for that we suggest that one need to make a proper dataset which has clickbait as the parameter, as we faced many difficulties in making this particular database in order to train both the models.

Also we can use other parameters for fake news detection as discussed in on of the review paper that we read[4]. Including these parameters will surely increase the accuracy of the model. We can think to implement this as our next step.

8. REFERENCES:

1. Manoj Kumar Balwant(2019). A Closer Look at Fake News Detection: A Deep Learning Perspective. Doi: [10.1145/3369114.3369149](https://doi.org/10.1145/3369114.3369149)
2. Ayat Abedalla, Aisha Al-Sadi, Malak Abdullah(2019). Bidirectional LSTM Based on POS tags and CNN Architecture for Fake News Detection. Doi: [10.1109/ICCCNT45670.2019.8944460](https://doi.org/10.1109/ICCCNT45670.2019.8944460)
3. Junaed Younus Khan, Md. Tawkat Islam Khondaker, Anindya Iqbal and Sadia Afroz(2019). A Benchmark Study on Machine Learning Methods for Fake News Detection. Project link: <https://www.researchgate.net/project/A-Benchmark-Study-on-Machine-Learning-Methods-for-Fake-News-Detection>
4. Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, Niloy Gangulyh(2016). Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media. Doi: [10.1109/ASONAM.2016.7752207](https://doi.org/10.1109/ASONAM.2016.7752207)
5. G. Loewenstein(1994). The psychology of curiosity: A review and reinterpretation. Psychological bulletin, vol. 116, 1994. Doi:[10.1037/0033-2909.116.1.75](https://doi.org/10.1037/0033-2909.116.1.75)
6. Peter Bourgonje, Julian Moreno Schneider, Georg Rehm DFKI GmbH(2017). From Clickbait to Fake News Detection: An Approach based on Detecting the Stance of Headlines to Articles. Doi: [10.18653/v1/W17-4215](https://doi.org/10.18653/v1/W17-4215)
7. Qiwei Li(2019). Clickbait and Emotional Language in Fake News. Retrived from: www.ischool.utexas.edu/~ml/papers/li2019-thesis.pdf

8. Abinash Pujahari¹ and Dilip Singh Sisodia(2019). Clickbait Detection using Multiple Categorization Techniques. Doi: [10.1177/0165551519871822](https://doi.org/10.1177/0165551519871822)
9. XINYI ZHOU, ATISHAY JAIN, VIR V. PHOHA, and REZA ZAFARANI(2019). Fake News Early Detection: A Theory-driven Model. Retrived from:
https://www.researchgate.net/publication/332726212_Fake_News_Early_Detection_A_Theory-driven_Model
10. Pritika Bahada, Preeti Saxenaa ,Raj Kamalb(2020). Fake News Detection using Bi-directional LSTM-Recurrent Neural Network. Doi:
[10.1016/j.procs.2020.01.072](https://doi.org/10.1016/j.procs.2020.01.072)
11. Kaggle Community Prediction Competition(2018), Fake News. Link:
<https://www.kaggle.com/competitions/fake-news/data>
12. Kaggle Community Prediction Competition(2020), Clickbait news detection. Link: <https://www.kaggle.com/competitions/clickbait-news-detection/data>

LAST PAGE

Project Group Personal Details:

1 Name of the Student: Gandhi Deep

Permanent Address: B3, Alap Flats, opp Caravan
School, near Sardar Garden,
Kalol - 382721, Gandhinagar,
Gujarat

Photograph



Email: deep.gce18@sot.pdpu.ac.in

Mobile No: +917984841734

2 Name of the Student: Devam Zanzmera

Permanent Address: c-701, Shreepad
Panorama, near L P Savani
school, Palanpur, Surat,
Gujarat 395009

Photograph



Email: devam.zce18@sot.pdpu.ac.in

Mobile No: +917265837425

3 Name of the Student: Ronak Makwana **Photograph**

Permanent Address: 32, Royal Arcade Society,
Opp. Madhvani Hostel,
National Highway, Porbandar,
Gujarat, India-360575



Email: ronak.mce18@sot.pdpu.ac.in

Mobile No: +919737867066