**FAKE NEWS DETECTION USING ML**

(A multi-source validation of news)

Synopsis Report Minor Project

**Bachelor of Technology**

in

**Computer Science and Engineering**

Efforts by:

Prachi Chauhan

Abhijeet Sanjiv Bonde

Abhishek Pundir

Under the Esteemed Guidance of

**Mr. Ankit Vidyarthi**



**Jaypee Institute of Information Technology**

**Noida, Uttar Pradesh**

**Students’ Self Declaration for Open Source libraries and other source code usage in Minor Project**

We Prachi Chauhan, Abhijeet Sanjiv Bonde, and Abhishek Pundir hereby declare the following usage of the open-source code and prebuilt libraries in our minor project in 5th Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with best of our knowledge and abilities.

1. List of pre build libraries
2. List of pre build features in libraries or in source code.
3. Percentage of pre written source code and source written by us.

|  |  |  |
| --- | --- | --- |
| Student ID | Student Name | Student signature |
| 19103002 | Abhijeet Bonde |  |
| 19103309 | Prachi Chauhan |  |
| 19103016 | Abhishek Pundir |  |

**Declaration by Supervisor (To be filled by Supervisor only)**

I, ........................................(Name of Supervisor) declares that I above submitted project with Titled **Fake news detection using ML** was conducted in my supervision. The project is original and neither the project was copied from External sources not it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

**Introduction and Problem statement**

Fake new or “Fictious articles fabricated intentionally to deceive readers”. Even trusted media houses are known to spread fake news in order to get more readers. They acquire a major portion of cyber space. The advancement of technology has also contributed to its menace. Studies show that over 43 to 51 % of the news that an average reader reads are fake (online resources). This can be done to create a different user perspective in the readers mind towards something that they read.

Fortunately, there are some systems that can help in reducing the amount of fake news from the cyber space.

In this project, we propose to implement a system that fuses into the world of news and helps us detect if a news is fake or real based on source of the news and the Headline of the news.

**Related Work**

Recently, there have been work that addresses this issue by using different Machine Learning and Deep Learning algorithms.

**Spam detection**

Spam detection uses statistical machine learning techniques to classify text (i.e. Tweets or emails as spam or not. This type of analysis requires pre-processing of text, feature extraction (i.e., Bag of words) and feature selection based on which features lead to the best performance on a test dataset. Once these features are obtained, they can be classified using Nave Bayes, Support Vector Machines, TF-IDF, or K-nearest neighbors’ classifiers. All of these classifiers are characteristic of supervised machine learning, meaning that they require some labeled data in order to learn the function. The task of detecting fake news is similar to spam detection. in that both aim to separate examples of real text from examples of fake, ill-intended texts.

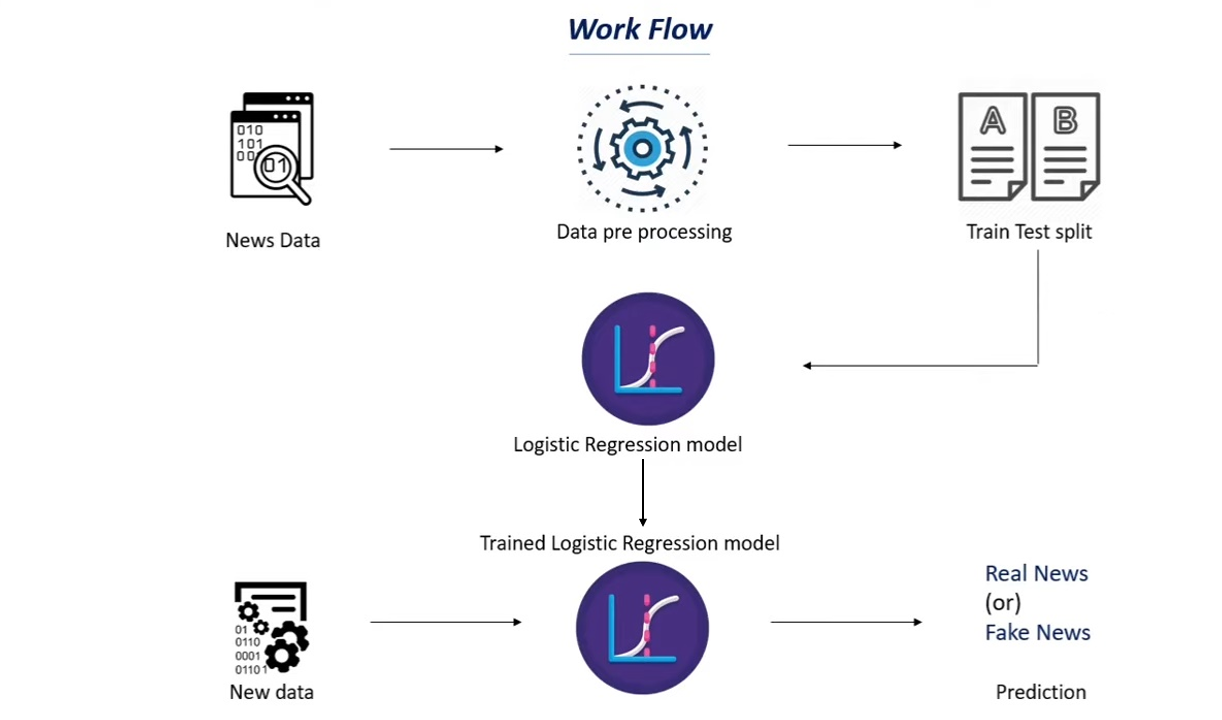
**High-level Approach**

The methodology that we are going to use is the build classifiers to predict the validity of the news based on the content (text). This type of method falls under “Clickbait news classification”. This may be achieved using LogisticRegression and K-nearest Neighbor which fall under Machine Learning.

The initial stage is to get the dataset to work on, the dataset that we have used is a Labeled dataset this dataset consists of 20800 news articles. The dataset will be pre-processed as computer doesn’t understand text it just understands numbers. After this process the data will be converted into Meaningful Numbers. Then the dataset will be slitted into 2 categories i.e., training data and testing data. As we will require the training dataset to train the Machine Learning model and the testing dataset will help us evaluate the model and get an approximate accuracy.

The Pre-processed training data is fed to the Logistic Regression and K-Nearest Neighbor model as this is binary classification i.e., either Real of Fake. Once these models are trained, they will be ready to be evaluated. Here the test data will come handy. The models will be fed with testing data and based on the results of this testing we can calculate the accuracy of these models.  
Once all this is completed, the model will be ready to process new data that we want to predict (whether it is real or fake).

WORK FLOW:



METHODOLOGY

1. **Dataset:**

**Id:** A unique ID that has been assigned to the news articles

Title: Short summary of the news content that try to attract the reader.

Author: Name of the person who wrote the article.

Text: The actual text content of the news.

Label: A simple flag to indicate whether the news is Real or Fake.

1: Fake news

0: Real news

1. **Libraries:**

**NLTK:** For filtering out stopwords and for stemming.

**RE:** For searching works in text or paragraph.

**Numpy:** Useful for making NumPy arrays.

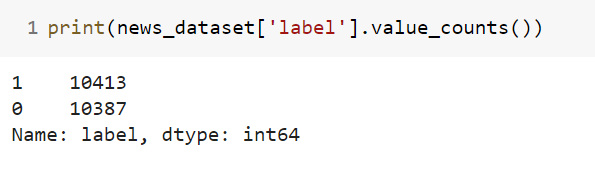
**Python-telegram-bot:** Useful for taking the input for a real time news.

**Pandas:** For creating and storing data frame.

**Scikit-learn:** To convert text into feature vectors, to split the dataset into training and testing data. And to import models, This will also be helpful for calculating accuracy score.

1. **Reading dataset and Pre-Processing:**

**Exploratory data analysis:**

****

As the above numbers says, the difference between number of real and fake news is quite small. Therefore, there is a very slight deviation in pie chart as well as bar graph which is not possible to visualize with naked eye.

news\_dataset['label'].value\_counts().plot(kind='bar')

plt.title('Count of fake/real News')

plt.xlabel('Label')

plt.ylabel('Count')

#addlabels('Label', 'Count')

plt.show()

****

The following script will read and store the dataset in pandas

Import pandas as pd

DataFrame news\_dataset = pd.read\_csv('BotModules/train.csv')

The metrics of dataset:

The code segment:

news\_dataset.shape

generates the dataset metrics as given below (rows, columns):

(20800, 5)

**The dataset looks like:**

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

**Stats of the dataset:**

id 0

title 558

author 1957

text 39

label 0

dtype: int64

As there are missing values in the dataset, we need to fill them up with null/empty string.

**New stats of the dataset:**

id 0

title 0

author 0

text 0

label 0

dtype: int64

for the prediction we will be using **title** and **author** as using this combination it yields a really good accuracy score.

So, in our data frame we create a new column called as **content** and store the merged result of title and author column.

**After merging the new column looks like:**

|  |  |
| --- | --- |
| 0 | Darrell Lucus House Dem Aide: We Didn’t Even S... |
| 1 | Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo... |
| 2 | Consortiumnews.com Why the Truth Might Get You... |
| 3 | Jessica Purkiss 15 Civilians Killed In Single ... |
| 4 | Howard Portnoy Iranian woman jailed for fictio... |
| .  .  . | .  .  . |
| 20795 | Jerome Hudson Rapper T.I.: Trump a ’Poster Chi... |
| 20796 | Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma... |
| 20797 | Michael J. de la Merced and Rachel Abrams Macy... |
| 20798 | Alex Ansary NATO, Russia To Hold Parallel Exer... |
| 20799 | David Swanson What Keeps the F-35 Alive |
| Name: content, Length: 20800, dtype: object | |

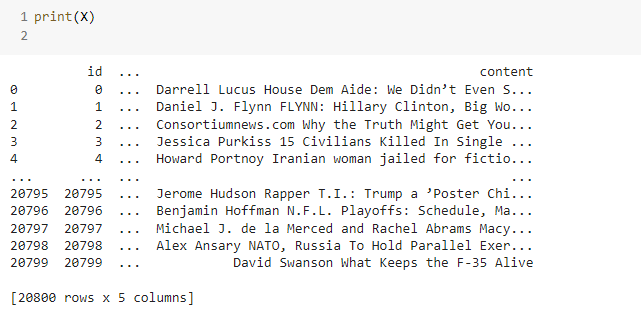
Now we need to separate the data and label for further processes:

The above suggested process can be done by using 2 variables

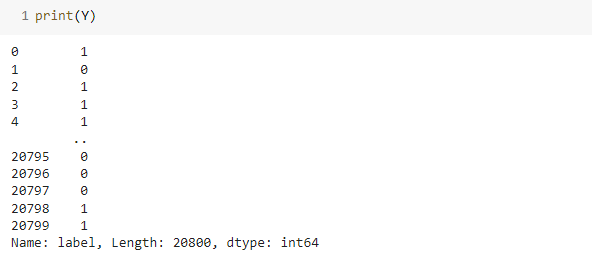
X: will contain all the columns other than label

Y: Will contain labels only

After performing the mentioned operation X looks like:



And Y looks like:



**Stemming and stop-words removal:**

**Stemming:**

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words also known as lemma, or reducing the word to its root word.

This process is important here as the it makes the training data denser**.**

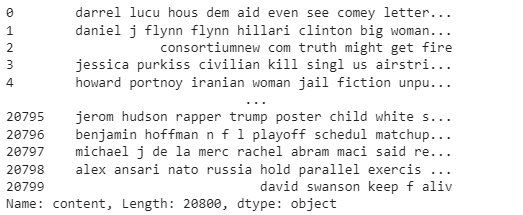
**Stop-words:**

Stop-words usually refer to the most common words in a language. These words do not add much meaning to the sentence. They can be safely removed without sacrificing the original meaning of the sentence. In other words, ‘**keeping out unwanted words out of our corpus**’.

**Filtering (re.sub):**

During this process we will also replace the special characters which contain numbers, symbols, other unwanted language letters except a-z and A-Z, with a space.

**After performing the above mentioned 3 steps the content looks like:**



**TF-IDF vectorizer:**

TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set).

**TF: Term Frequency**

Term Frequency: In document d, the frequency represents the number of instances of a given word ‘t’. Therefore, we can see that it becomes more relevant when a word appears in the text, which is rational. Since the ordering of terms is not significant, we can use a vector to describe the text in the bag of term models. For each specific term in the document, there is an entry with the value being the term frequency.

**Term frequency(TF) = count of particular word present in a sentence / total number of words in sentence**

**IDF: Inverse Document Frequency**

**Document Frequency:**

This tests the meaning of the text, which is very similar to TF, in the whole corpus collection. The only difference is that in document d, TF is the frequency counter for a term ‘t’, while df is the number of occurrences in the document set N of the term ‘t’. In other words, the number of papers in which the word is present is DF.

**Inverse Document Frequency:**

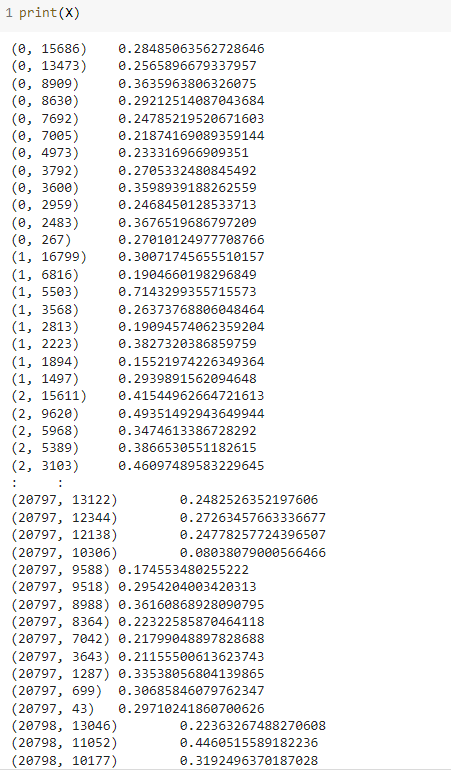
Mainly, it tests how relevant the word is the key aim of the search is to locate the appropriate records that fit the demand. Since tf considers all terms equally significant, it is

therefore, not only possible to use the term frequencies to measure the weight of the term in the paper. First, find the document frequency of a term t by counting the number of documents containing the term.

**Inverse Document Frequency (IDF) = log(total no of sentences /no of sentences containing that words)**

After performing TfidfVectorizer function all the values of X will be converted to their respective features.

So, the feature vectors look like:



**Splitting the Dataset into training and testing data:**

We are splitting the data into 2 datas as X\_train (used for training), X\_test (used for testing) and with them parallelly Y\_train (labels of X\_train), Y\_test (labels of Y\_test).

Here the data of real and fake news will be segregated in a equal proportion. The proportion will be same as that of the original dataset of fake is to real news.

The splitting can be done as

Training: 80 % data

Testing: 20 % data

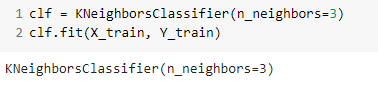
**K-Nearest Neighbor:**

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.



The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle).

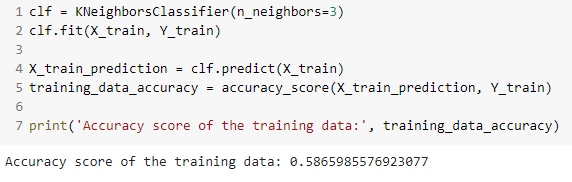
**Training:**



**Evaluating:**

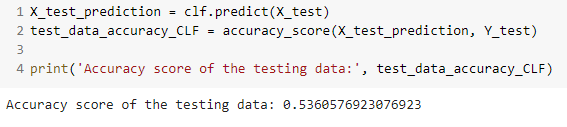
**Accuracy score on training data (X\_train, Y\_train):**

Here we will be predicting the vale based on X\_train data and the outcome labels of the process will be stored in X\_train\_prediction. Further we will compare X\_train\_prediction values with the Y\_train (Original labels) to find the accuracy.

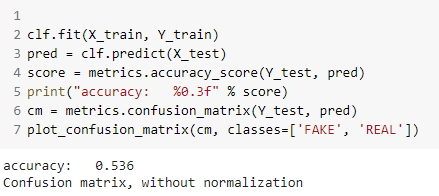


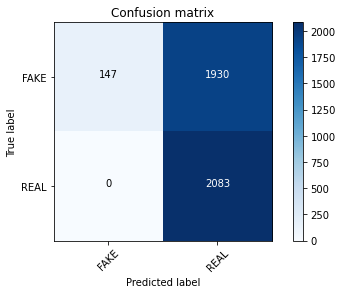
**Accuracy score on testing data (X\_test, Y\_test):**

Here we will be predicting the vale based on X\_test data and the outcome labels of the process will be stored in X\_test\_prediction. Further we will compare X\_test\_prediction values with the Y\_test (Original labels) to find the accuracy.



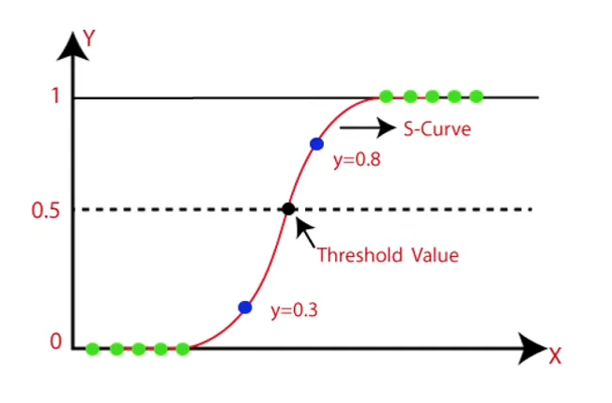
**Confusion Matrix:**





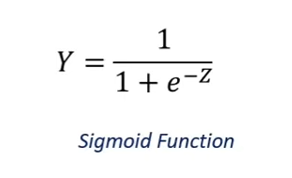
**Training the model: Logistic Regression**

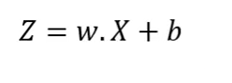
This is how the sigmoid curve in Logistic Regression looks like



If the prediction is greater than 0.5 then the label for that particular instance will be 1. Similarly, if the prediction is lower than 0.5 then the label for that particular instance will be 0.

The above sigmoid function is plotted using the following sigmoid function:





where,

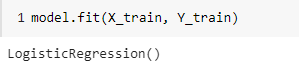
X: Input features

Y: Prediction Probability

w: weights

b: biases

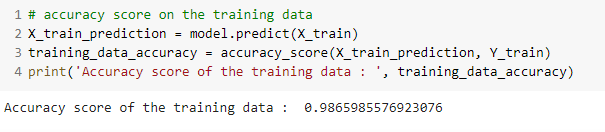
**Training:**



**Evaluating:**

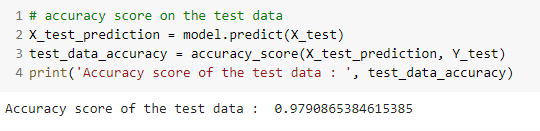
**Accuracy score on training data (X\_train, Y\_train):**

Here we will be predicting the vale based on X\_train data and the outcome labels of the process will be stored in X\_train\_prediction. Further we will compare X\_train\_prediction values with the Y\_train (Original labels) to find the accuracy.

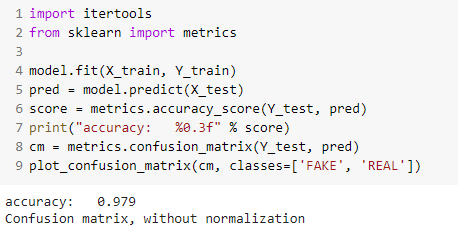


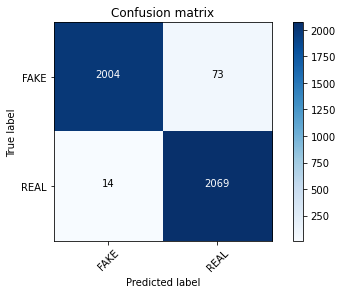
**Accuracy score on testing data (X\_test, Y\_test):**

Here we will be predicting the vale based on X\_test data and the outcome labels of the process will be stored in X\_test\_prediction. Further we will compare X\_test\_prediction values with the Y\_test (Original labels) to find the accuracy.



Confusion Matrix:



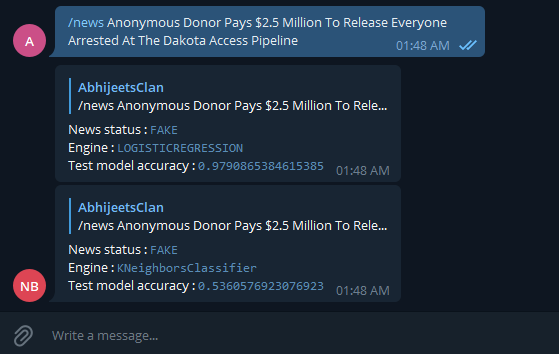


|  |  |  |
| --- | --- | --- |
| Algorithm | Training accuracy | Testing accuracy |
| K-Nearest Neighbor | 0.5865985576923077 | 0.5360576923076923 |
| Logistic Regression | 0.9865985576923076 | 0.9790865384615385 |

**Front-end for the project:**

As, to decide whether a news is real or fake there has to be a data entry point to enter the real-time news. To give the front-end we used Telegram Bot for the ease of access to our predictive system. Using the poll method, we have the facility to get timely updates from bot and by using this method we have the chance to not miss any update from telegram bot.

To give a real-time news to the system the user has to send a command followed by the news that the user wants to predict. An example is as shown below.



**References:**

[1] Nicole O’Brien , ‘Machine Learning for Detection of Fake News’ from Massachusetts Institute of Technology June 2018.

[2] Simon Lorent , ‘Fake News Detection Using Machine Learning’ from University Of Li`ege 2018-19