SB3001- PROJECT BASED

EXPERIENTIAL LEARNING PROGRAM

NAA-MUDHALVANREPORT:

Date	01/11/2023
…NM ID	AU9506
Team ID	Au_212171
Project Name	Fake news detection using nlp
Maximum Marks	

INDEX

Acknowledgment

I would like to express my gratitude to the many individuals and institutions that have contributed to the successful completion of this project on fake news detection using Natural Language Processing (NLP). Without their support, guidance, and resources, this research would not have been possible. First and foremost, I would like to thank my supervisor [Supervisor's Name], for their invaluable guidance and mentorship throughout the project. Their expertise in the field of NLP and their unwavering support played a crucial role in shaping the direction and methodology of this research. I am also thankful to [Name of Funding Institution or Organization] for providing financial support for this research. Their funding made it possible to access the necessary tools, datasets, and computing resources required for the experiments and analysis. I am deeply appreciative of the participants who volunteered their time and feedback for the data collection and evaluation phases of this project. Their contributions were instrumental in fine-tuning the fake news detection models and ensuring the quality of the research. I would like to acknowledge the research community and organizations that have shared valuable datasets and research findings related to fake news detection. Their contributions provided a solid foundation for our work.

FAKE NEWS DETECTION USING NLP

ABSTRACT:

Fake news is a peculiarity which is fundamentally effecting our public activities through social media platforms. The data acqurracy on internet particularly via online media. The fake news detection is a subtask of text classification and is often defined as task of classifying news as real or fake. There are different social media platforms that area cessible to these users. A human being is unable to detect all these fake news. So there is a need for a machine learning classifiers that can detect these fake news automatically. This strategy utilizes NLP Classification model(logistic Regression) to anticipate whether the news from the social media is real or fake . With this undertaking we are attempting to get high exactness and furthermore decrease an opportunity to distinguish the Fake News.

Key Words: Conterfeit news, Real, Fake, Logical Regression.

INTRODUCTION:

Humans are ureliable detector of fake news. This is because people are susceptible to bias. People tend to believe those that does not contradict their preconceived ideas. The fake news is a wonder which is altogether influencing our public movement. Fake news area is a rising investigation district which is getting interest with the resourcesavailable. In this paper we have displayed anacknowledgement model for fake news using logical regression metholodologies.In the last decade, Fake News phenomenon has experienced a very significant spread, favored by social networks. This fake newscan be broadcasted for different purposes. Hence there is a requirement of a technology where the human being can understand and react to such fake news. Hence the fake news detection is a bang for all of it .

LITERATURE SURVERY:

The differentiating between actual and fake information propagation via online social networks is an important issue in lots of programs. And the time when the news release is very closer to broadcasting the actual records .the proposed method makes use of recurrent neural networks with a unique loss feature, and a new preventing rule.Experiments on real datasets demonstrate the effectiveness of our model both in phrases of early labeling and accuracy. We introduced a new real time early news labeling approach.The news articals that are written with an cause to deliberately mislead or manage readers are inherently elaborate. These so-called „false information‟ articles are believed to have contributed to election manipulation or even resulted in intense injury and demise, through actions that they have caused. The data set su[[;ed on this paper consists of manually recognized and categorized information that can be used for the education and checking out of type systems that discover valid as opposed to fake and manipulative news testimonies. This "fake news detection" helps in clearing the usage of false information channels of india and how they are the use of social media and fake information to fuel nationalism and create department among communities to avoid essential problems of the people and mostly girls and children, financial system and many others.

Problem Statement:

Given a vast and ever-expanding volume of textual data sourced from news articles, social media posts, websites, and other online platforms, the objective is to develop NLP-based algorithms and models that can accurately identify and classify instances of fake news, misinformation, or disinformation within this data.

Key Components:

Textual Data: The input consists of unstructured textual data, typically in the form of news articles, social media posts, or other written content.

Classification: The problem involves classifying each piece of textual data into one of two or more categories, such as "fake" or "legitimate," "misinformation" or "factual," or a multi-class categorization based on the severity or type of fake news.

NLP Techniques: The solution leverages NLP techniques to process, analyze, and extract features from the textual data, including linguistic, contextual, and semantic information.

Challenges:

The problem of fake news detection using NLP is fraught with numerous challenges, including:

Data Variability:

Textual data is diverse, and fake news can manifest in various forms, from subtle misinformation to blatant disinformation, making it challenging to create a one-size-fits-all solution.

Data Quality:

Ensuring the quality and reliability of training data is crucial, as well-labeled datasets for fake news detection are often scarce and can be subject to bias.

Feature Extraction:

Extracting relevant linguistic features, such as sentiment, tone, authorship, and source credibility, is a complex task, as fake news detection often requires an understanding of nuanced language and context.

Contextual Understanding: NLP models need to capture the nuances of language and context, as fake news can be context-dependent and rely on subtleties that may not be immediately obvious.

Scalability:

Handling the vast and continually growing volume of textual data available on the internet poses scalability challenges for real-time or large-scale fake news detection.

Evaluation Metrics:

The performance of fake news detection models is typically assessed using metrics such as accuracy, precision, recall, F1 score, and area under the ROC curve (AUC). These metrics measure the model's ability to correctly classify fake and legitimate news, minimizing false positives and false negatives.

Applications:

The successful detection of fake news using NLP has numerous practical applications, including safeguarding public trust in information sources, preventing the spread of harmful misinformation, and assisting social media platforms, news organizations, and fact-checking agencies in identifying and countering fake news.

In summary, fake news detection using NLP is a critical problem that aims to leverage natural language processing techniques to sift through vast volumes of textual data, distinguishing between accurate and deceptive information. It is an interdisciplinary challenge that combines linguistic analysis, machine learning, and data science to combat the spread of misinformation in the digital age.

Data Preprocessing:

There are some exploratory data analyses is performed on training data to prepare the data for modelling of system like null or missing values, removing social media slangs, removing stop-words, correcting contraction. Also, Part of Speech (PoS) Tagging has been performed in the data to meet the accuracy of prediction model. Data has been also lemmatized to get root form of the words so that prediction algorithm gets trained on the quality data. Before model training the data was tokenize so that each word in the sentence can treated as element for model training.

Lemmatization:

Lemmatization is one of the most common text pre-processing techniques used in Natural Language Processing (NLP) and machine learning in general. lemmatization involves deriving the meaning of a word from something like a dictionary. Lemmatization gives the root form of the word i.e., studying is studies. This makes sure that the root word is not just achieved by removing the suffix from a given word that is done in stemming. This makes the lemmatization algorithm slow but for the NLP technique where meaning each word is equally important by its root word. Thus, we have used lemmatization to get the root word.

Tokenization:

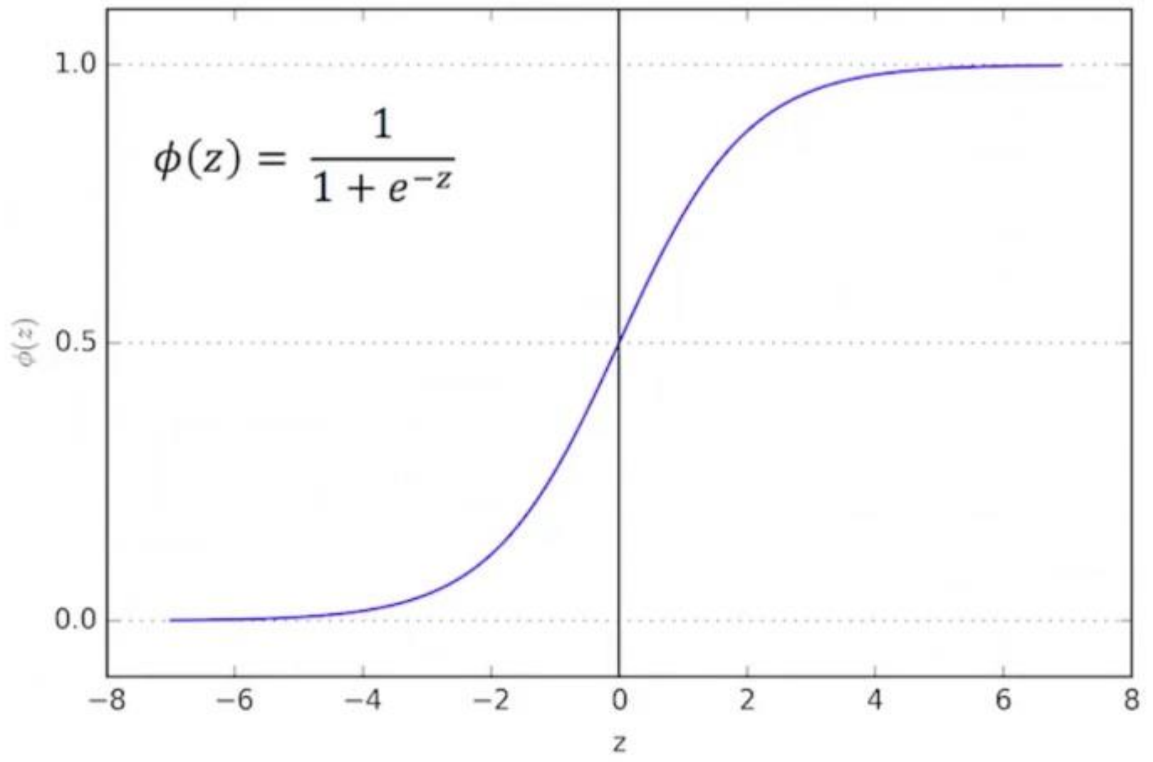
Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. Here, tokens can be either words, characters, or sub-words as tokens are the building blocks of Natural Language, the most common way of processing the raw text happens at the token level. Hence, Tokenization is the foremost step while modelling text data. Tokenization is performed on the corpus to obtain tokens. The following tokens are then used to prepare a vocabulary. Vocabulary refers to the set of unique tokens in the corpus. Remember that vocabulary can be constructed by considering each unique token in the corpus or by considering the top K Frequently Occurring Words.

DATA VISUALIZATION:

Logistic Regression:

Before diving into the code let us revise the Logistic Regression concept. Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no(binary classification), based on prior observations of a data set. It is a Supervised statistical technique to find the probability of dependent variable.

The graph shown below is a **Sigmoid Function**, which we also call as a **Logit**. This function converts the probabilities into binary values which could be further used for predictions.



Sigmoid Function from rasbt

According to this graph, if we obtain the probability value to be **less than 0.5**, then it is considered to be of the **Class 0** and if the value is **more than 0.5**, then it would be a part of **Class 1**.

If you want to learn this concept in depth, you can check **this link** out or even watch **this video**!

WHY WE TAKE LOGISTIC REGRESSION ALGORITHM:

Logistic regression is a statistical model in which the response variable takes a discrete value and the explanatory variables can either be continuous or discrete. If the outcome variable takes only two values, then the model is called binary logistic regression model.Here the outcomes can be real news ($Y = 1$) and false news ($Y = 0$).Then the probability that a record belongs to a positive class, $P(Y = 1)$, using the binary logistic regression model is given by:

P(Y=1) = e^z/(1 + (e^z))

Logistic Function-The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It is an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Here y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x).Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.
Similarly, we fit the model to the training set, predict test set results and calculate accuracy, precision and recall as well as tune hyperparameters for optimal results and accuracy.

We obtain an Accuracy score of 93.52%

Precision score of 0.89 and

Recall score of 0.97

The best accuracy is 93.63% with C value as 0.8.

We use confusion matrix to evaluate the performance of the model. It compares predicted values and the actual values. We use 4 measures to evaluate the performance:

True positive: The cases in which the predicted values and the actual values are the same, and the value is positive.

- True Negative: The cases in which the predicted values and the actual values are the same, and the value is negative.
- False Positive: The cases in which the prediction is 'YES' ,but the actual value is 'NO'.

- False Negative: The cases in which the prediction is 'NO', but the actual value is 'YES'.

Code:

Now finally starting off with our code, you can either write it in your Jupyter Notebook or Google Colab or any other platform you like.
Also download your dataset from **here**. (I only used the training dataset so you can also download that itself.)

Importing libraries/dependencies:

```
import numpy as np
```

```
import pandas as pd
```

```
import re
import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import sklearn

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
```

Covering the importance of each library/module/function that we imported:

- **NumPy** : It is a general-purpose array and matrices processing package.

- **Pandas** : It allows us to perform various operations on datasets.
- **re** : It is a built-in RegEx package, which can be used to work with Regular Expressions.
- **NLTK** : It is a suite of libraries and programs for symbolic and statistical natural language processing (NLP).
- **nltk.corpus** : This package defines a collection of corpus reader classes, which can be used to access the contents of a diverse set of corpora.
- **stopwords** : The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any

  language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. (Example-and, of, are etc.)

- **PorterStemmer** : A package to help us with stemming of words. (More about stemming in the Data Preprocessing section)
- **Sci-kit Learn (sklearn)** : It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- **feature_extraction.text** : It is used to extract features in a format supported by machine learning algorithms from datasets consisting of text.
- **TfidfVectorizer** : It transforms text to feature vectors that can be used as input to estimator. (More about TfidfVectorizer in the Data Preprocessing section)

- **train_test_split** : It is a function in Sklearn model selection for splitting data arrays into two subsets - for training data and for testing data.
- **LogisticRegression** : A pretty self explanatory part of the code, used to import the Logistic Regression Classifier.
- **metrics** and **accuracy_score** : To import Accuracy classification score from the metrics module.

Loading the Dataset

I hope you have downloaded the dataset by now. Now you can load the dataset as,

```
data = pd.read_csv('fakenews.csv')
```

```
data.head()
```

Here I have renamed my csv file as **fakenews.csv** and saved it in

the same folder as my jupyter notebook. If you saved your dataset

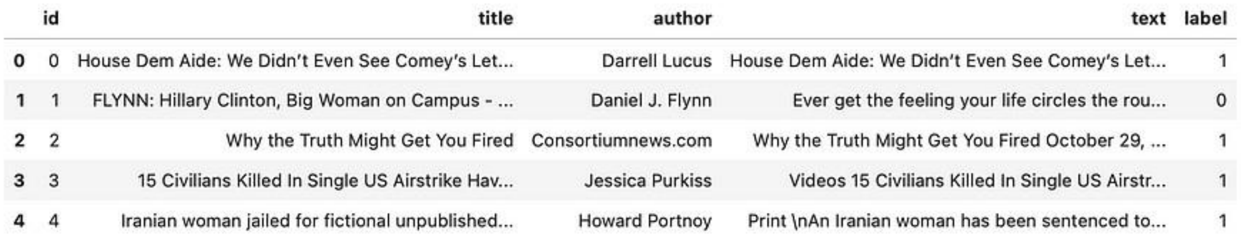
and the jupyter notebook in 2 different folders, you can add the path

of the dataset file as the prefix in the code as, `data =`

`pd.read_csv('/Users/chandana/Documents/fakenews.csv')`

(This is a macbook path, so if you are using Windows or any other

OS, the path might look different.)
The dataframe wi



	id	title	author	text	label
0	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	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 Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

ll look like this,

Here Label indicates whether a news article is fake or not, 0

denotes that it is Real and 1 denotes that it is Fake.

Data Preprocessing:

After importing our libraries and the dataset, it is important to preprocess the data before we train our ML model since there might be some anomalies and missing datapoints which might make our predictions a bit skewed from the actual values.
Now, we can check the size of the dataframe/table as it

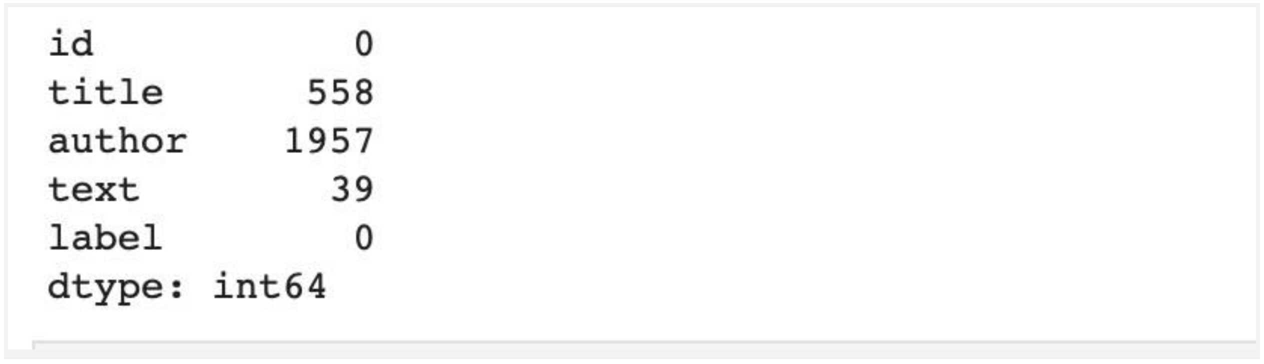
would decide whether we can drop the rows with null values

without affecting the size of our dataset or not. `data.shape`

This gives us **(20800, 5)** which means that we have 20800 number of entries and 5 columns (features).

Checking the total number of missing values in each of the columns.

```
data.isnull().sum()
```



```
id            0
title       558
author     1957
text         39
label         0
dtype: int64
```

From this we can see that we will have to delete a minimum of 1957 lines to remove all the null values so it would be better to fill these null values with an empty string. For that we can use **fillna.**

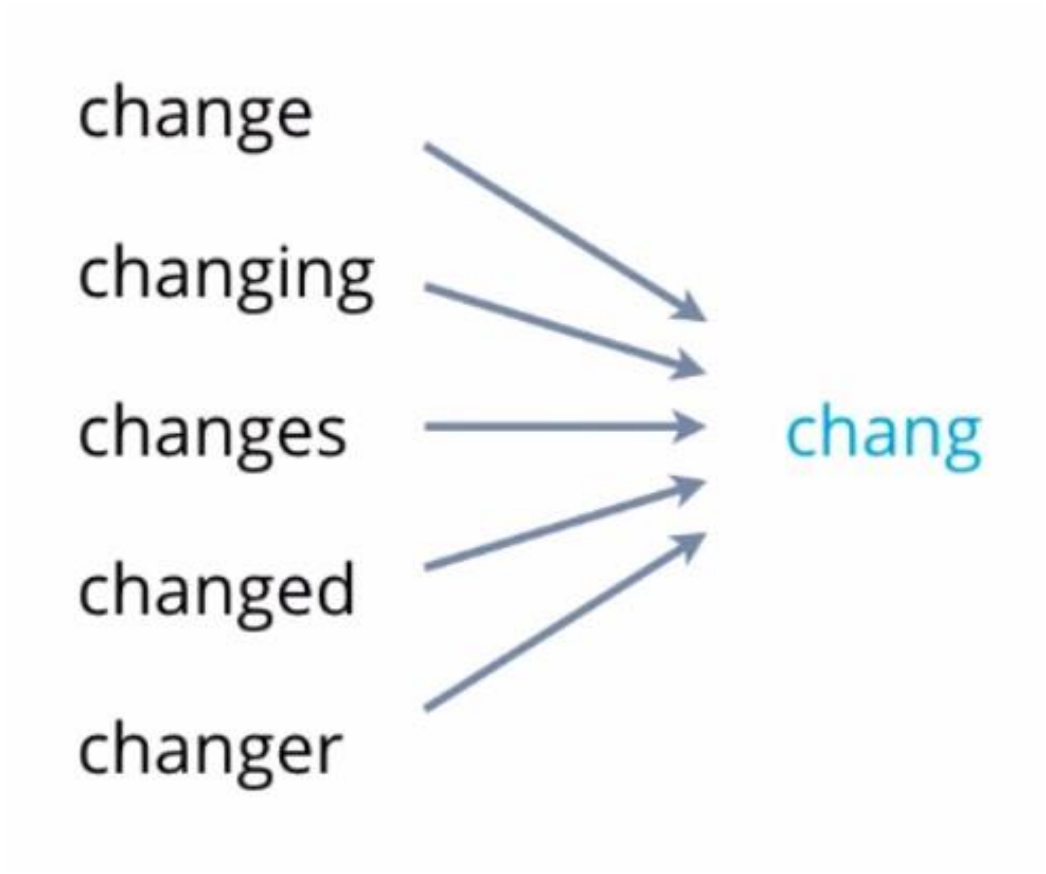
```
df1 = data.fillna('')
```

After this step we no longer have any missing datapoints, you can check that using the isnull().sum()

Now, we'll try to reduce those 5 columns to only 2 columns since it will be easier for us to train the model. For that we'll combine the **title** and the **author** columns into one, naming it as **content**. We can drop the other columns as they don’t have much effect on determining whether the article is fake or not. This step will leave us with 2 columns - **content** and **label**. `df1['content'] = df1['author'] + ' ' + df1['title']`

Stemming:

Now coming to the stemming part, it basically is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words.



Stemming example

Stemming of words might or might not end up with a root word with meaning, like in this example chang doesn't mean change or anything as a matter of fact. For the root word to have meaning we use **lemmatization**. But for this project stemming works just fine.

```
stemmer = PorterStemmer()
```

We create a new Porter stemmer for us so that we can use the function without explicitly typing PorterStemmer() every time.

```
def stemming(content):

    stemmed_content = re.sub('[^a-zA-Z]',' ', content) #1

    stemmed_content = stemmed_content.lower() #2

    stemmed_content = stemmed_content.split() #3

    stemmed_content = [stemmer.stem(word) for word in
stemmed_content if not word in stopwords.words('english')] #4

    stemmed_content = ' '.join(stemmed_content) #5

    return stemmed_content #6
```

Okay, so let’s go in depth and see what this function actually does. I have numbered each line from 1 to 6 so that you can easily distinguish between different lines of code and understand each line’s use.

#1 First we use the **re** package and remove everything that is not a letter (lower or uppercase letters).

#2 We then convert every uppercase letter to a lower one.

#3 We then split the each sentence into a list of words.
#4 Then we use the stemmer and stem each word which exists in the column and remove every english stopword present in the list.

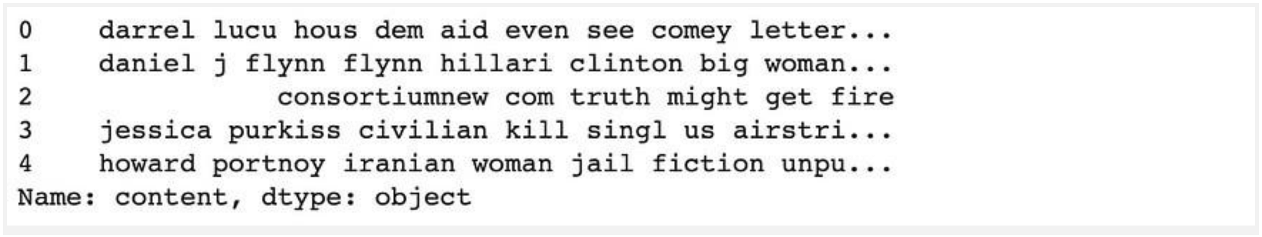
#5 We then join all these words which were present in the form of a list and convert them back into a sentence.

#6 Finally we return the stemmed_content which has been preprocessed.

Applying this function to our dataset,

```
df1['content'] = df1['content'].apply(stemming)
```

```
df1['content'].head()
```



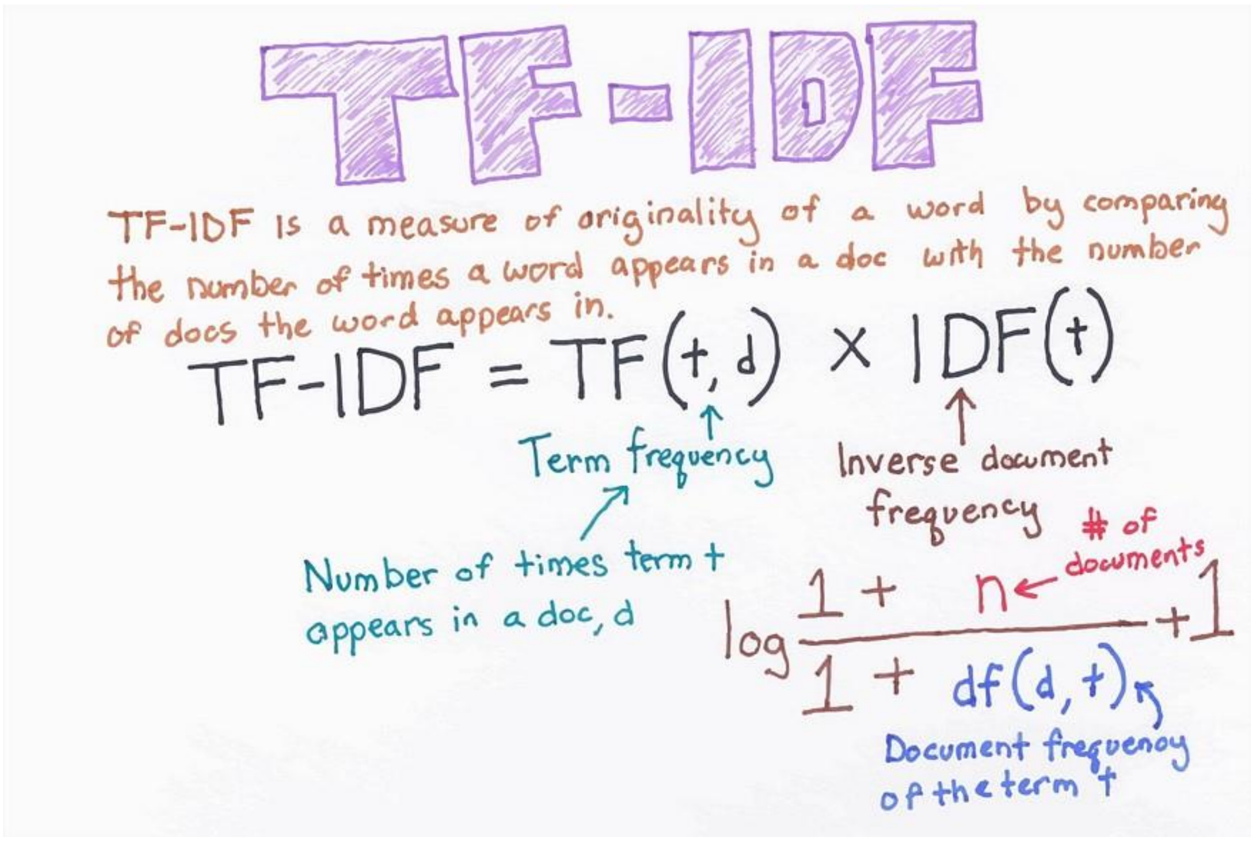
```
0    darrel lucu hous dem aid even see comey letter...
1    daniel j flynn flynn hillari clinton big woman...
2                consortiumnew com truth might get fire
3    jessica purkiss civilian kill singl us airstri...
4    howard portnoy iranian woman jail fiction unpu...
Name: content, dtype: object
```

Next step is to name our input and output features

```
X = df1.content.values
```

```
y = df1.label.values
```

Our last preprocessing step would be to transform our textual X to numerical so that our ML model can understand it and can work with it. This is where **TfidfVectorizer** comes into play. Here is a picture explaining it in brief,


TF-IDF
TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.
TF-IDF = TF(t, d) × IDF(t)
Term frequency
Inverse document frequency
Number of times term t appears in a doc, d
# of documents
log (1 + n) / (1 + df(d, t)) + 1
Document frequency of the term t

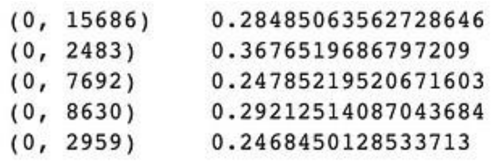
Source: https://becominghuman.ai/word-vectorizing-and-statistical-meaning-of-tf-idf-d45f3142be63

To understand it in depth, visit **this link**.

```
X = TfidfVectorizer().fit_transform(X)
```

```
print(X)
```

The output of this code should like this,



```
(0, 15686)    0.28485063562728646
(0, 2483)     0.3676519686797209
(0, 7692)     0.24785219520671603
(0, 8630)     0.29212514087043684
(0, 2959)     0.2468450128533713
```

Now that we have the X in our desired form, we can move onto the next step.

Splitting the Dataset:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, stratify = y, random_state = 2)
```

This means that we have divided our dataset into 80% as training set and 20% as test set. stratify = y implies that we have made sure that the division into train-test sets have around equal distribution of either classes (0 and 1 or Real and Fake). random_state = 2 will guarantee that the split will always be the same.

Training the Model:

Fitting the model to our dataset

```
model = LogisticRegression()
```

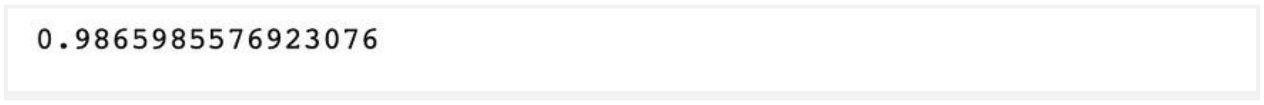
```
model.fit(X_train, y_train)
```

Now that we have trained it, let’s check the accuracy of our training set predictions,

```
X_train_prediction = model.predict(X_train)
```

```
training_accuracy = accuracy_score(X_train_prediction, y_train)
```

```
print(training_accuracy)
```



```
0.9865985576923076
```

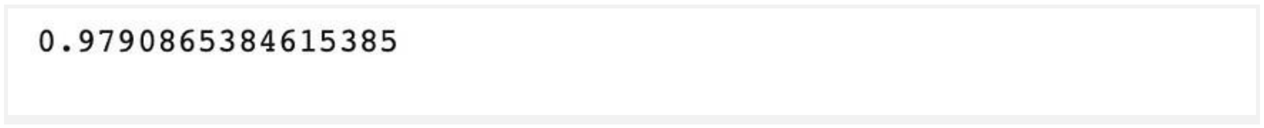
Training accuracy score

So I got about 98.66%, which is pretty good. Similarly for the test dataset.

```
X_test_prediction = model.predict(X_test)
```

```
testing_accuracy = accuracy_score(X_test_prediction, y_test)
```

```
print(testing_accuracy)
```



```
0.9790865384615385
```

Test accuracy score

So test accuracy is also pretty good.

(**Note:** The score may differ for you if you make any changes to this code)

With this we have successfully trained our ML model!

Building a system:

Finally to make this model useful we need to make a system.

Taking a sample out of the test-set (I took the first sample),

```
X_sample = X_test[0]
```

Checking our prediction for this sample,

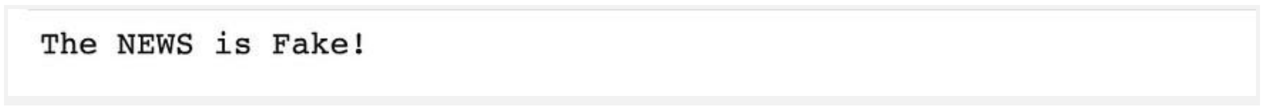
```
prediction = model.predict(X_sample)

if prediction == 0:

    print('The NEWS is Real!')

else:

    print('The NEWS is Fake!')
```



```
The NEWS is Fake!
```

Summary:

We found that the deep learning model and Logistic Regression produced the similar results. Only the training time for Logistic Regression is half of the training time for deep learning.

The obvious difference, from the above process, is that the Deep Neural Network is estimating many more parameters and even more permutations of parameters than the logistic regression. Basically, we can think of logistic regression as a one layer neural network.

To sum it up, I would recommend to solve a classification problem with a simple model first (e.g., logistic regression). In our example, this would’ve already solved our problem sufficiently well. In other cases, when we are not satisfied the simple models’ performance and we have sufficient training data, we’d try to train a deep neural network, which has the advantage to learn more complex, non-linear functions.

CONCLUSION:

The paper presents the stages of creating a model for fake news detection. The pre-processing, training and prediction phases have been described and the performance of the model has been assessed using accuracy, precision, recall and confusion matrix. The model fulfills its task of detecting fake news. However, this is merely a beginning stage in the eradication of fake news. There are more advanced NLP techniques such as BERT,GloVe,ELMo that can be used. Other methods include using stylometry-based provenance, i.e., tracing a text’s writing style back to its producing source and determining whether that source is malicious. Techniques can also be used to evaluate sub-categories of fake news.