PROJECT REPORT

A report submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ELECTRONICS AND COMMUNICATION ENGINEERING

The Institute Of Electronics And Telecommunication Engineers Mysuru

A Project report on

FAKE NEWS DETECTION

Sl.No	Student Name	College Name
1.	Sourabh Singh Thakur	Shri Shankaracharya Engineering College, Bhilai
2.	Pradnya Torase	Government Engineering College, Haveri
3.	Shweta .C. Majagi	S.G.Balekundri Institute of Technology Belagavi

Contents:-

Introduction

With people spending more time on the social media platforms, they are more prone to consume information from social media. Social media is free of cost, easy to access and help one to express opinions and hence it acts an excellent source for an individual to consume information from social media. But the quality of news on social media is generally lower than the traditional news organizations. It is because anyone can spread information they want in the social media and there is no regulating authority to control the information.

Fake news, as a specific type of disinformation, means the false information that is spread deliberately to deceive people. Some individuals and organization use social media as a tool to spread disinformation for financial and political gains. It was approximated that over million tweets are related to fake news.

Fake news detection is an important and technically challenging problem. In an attempt to tackle the growing misinformation, several fact-checking websites have been deployed to expose the fake news. These websites play a crucial role in clarifying fake news, but they require expert analysis which is time-consuming. Numerous articles and blogs are written in order to distinguish fake news from the true news. However, they are not from authority sources and may be biased, which make the labels not fully reliable and convincing. Due to the volume and diversity of the social media, it is almost impossible to manually label the fake news and true news.

The major functionalities of Fake News Detection are as follows:

1. *Fake News Collection:* collecting news contents and social context automatically, which provides valuable datasets for the study of fake news; 2. Fake News Detection: extracting useful features and build various machine learning models to detect fake news; 3. Fake News Visualization: presenting the characteristics of fake news dissemination through effective visualization technique

2. *Fake News Detection:* extracting useful features and build various machine learning models to detect fake news;

3. *Fake News Visualization*: presenting the characteristics of fake news dissemination through effective visualization techniques.

1. Collecting Fake News Data :

Fake news is widely spread across various online platforms. We use some of the factchecking websites as a source for collecting fake news information. In these fact-checking sites, fake news information is provided by the trusted authors and relevant claims are made by the authors on why the mentioned news is not true.

We propose a strategy for collecting fake news in a periodic manner to update the repository. First, we collect the verified fake news and true news from fact-checking websites on daily basis.

Then,we gather the tweets related to the fake/real news that is spread. Moreover, we gather social engagements of users such as replies of tweet, retweet, and favorites through Twitter APIs. Users who interact with these tweets that are related to fake news are more vulnerable to them. If the user likes the tweet related to fake news they are prone to be affected by the fake news.

Based on the comments on the retweets we could infer whether the user is able to differentiate fake news or not.

2. Detecting Fake News :

Detection of fake news is a difficult task as it is intentionally written to falsify information. We can use linguistic features likes news content to find the clues between fake news and real news. Although fake news is intentionally written to appear similar to fake news studies have shown that the language style used to falsify information and the topic content could be a factor for determining fake news. For using the news content in our classification, we use auto encoders to learn the news content in low dimensional latent feature space.

3. Fake News Visualisation:

We have developed various interfaces for visualizing the data from our dataset. For identifying the differences in the news content of the true news and the fake news we have used word cloud representation of the words for the textual data. we can search for fake news within a time frame and identify the relevant data. Also, we have provided the comparison of feature significance and model performance as part of this dashboard.

We have fake news visualization interface. Word cloud of fake news content and true news content.

We had devised this problem into 3 different phases: pre-processing, text-to-numeric representation conversion using pre-trained algorithms, and then evaluate the models using state-of-the-art machine learning algorithms. We had analysed the data set and in particular the text part of the data explaining how it is distributed and then we converted each text into numeric representation using pre-training models such as TFIDF and CV for vector representation.

Finally, we evaluated our numeric conversion data using significant machine learning algorithms such as Random forest, classification algorithms etc to perform the classification.

Methods

➢ The Dataset:

The training data set has four features: Unnamed, title, text, label. The ID uniquely identifies the news article. The title and text are the title and text of the news article respectively. The text is the content of the article, and may be incomplete. The label indicates whether the article is reliable (real) or not (fake):

$$\text{label} = \begin{cases} 0, \text{ if reliable news} \\ 1, \text{ if fake news} \end{cases}$$

The training data set contains 6335 odd number of samples. The test data set does not have labels, so we do not use it.

The test data set will be selected from the training data set randomly when we are evaluating our models.

In our project, since we hypothesized that the text and the words used within the text are key to distinguish between real and fake news samples, we decided to investigate only the text column.

➢ Data Pre-processing:

Removed numbers:

Within the context of a news article title or text, numbers simply quantify claims and do not change the meaning of the text. Therefore it is best to remove all numbers to minimize noise in our data. We use the string.digits string constant in Python as well as the translate and maketrans methods from Python's string module to convert all numerical digits to an empty string, effectively removing all digits.

Removed punctuation and and special characters:

In addition of pre-processing the textual data, we removed all characters that are not textual (not alphabets such as punctuation, extra delimiters etc.). We used the string. punctuation module in Python to find all punctuation characters. We remove all those punctuation characters from every word in the texts, with the exception of the symbols '#' and '@'.

Next, we removed an assortment of special characters that don't appear on traditional American keyboards and don't contribute to the meaning of the tweets. The long dash ("–"), single and double Asian quotations, ellipse characters (...), and bullet points (•) all were removed for this reason. After removing all special characters, there are still a couple of pre-processing cases we account for. For these cases, we used regular expressions to detect certain patterns we wish to remove.

Proceeding further, we make all of our texts lowercase and then remove all rows that have foreign language characters in their text, since we are only interested in identifying fake news in English. This finally ensures the text we preserve is only with English words with no non-alpha character.

Removed stopwords:

Stop words are a list of the most common words in a language, such as "a", "be", "quite", "should"...etc. They are often void of meaning, and does not add anything to the content. They are also most frequently present in every text. Hence, we presumed removal of stop words can have multiple advantages.

For once, it decreases memory overhead, since we cut down a huge amount of text (and hence narrows down the number of features to train our models on).

Second, it reduces noise, since by eliminating stop words, we are able to focus on more meaningful contents (the more distinct features between these two classes). Although it is not often the case that removing stop words are the most optimal, sometimes the information that we are looking for may be included in the stop words that we removed.

Lemmatisation(or **lemmatization**):

Lemmatisation in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.

In computational linguistics, lemmatisation is the algorithmic process of determining the lemma of a word based on its intended meaning. Unlike stemming, lemmatisation depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.

As a result,developing efficient **lemmatisation** algorithms is an open area of research.

In many languages, words appear in several *inflected* forms.

For example, in English, the verb 'to walk' may appear as 'walk', 'walked', 'walks' or 'walking'. The base form, 'walk', that one might look up in a dictionary, is called the *lemma* for the word.

Tokenisation:

Tokenization is the process by which big quantity of text is divided into smaller parts called **tokens**.

Natural language processing is used for building applications such as Text classification, intelligent chatbot, sentimental analysis, language translation, etc. It becomes vital to understand the pattern in the text to achieve the above-stated purpose**.** These tokens are very useful for finding such patterns as well as is considered as a base step for stemming and lemmatization.

We use the method **word_tokenize()** to split a sentence into words. The output of word tokenization can be converted to Data Frame for better text understanding in machine learning applications.

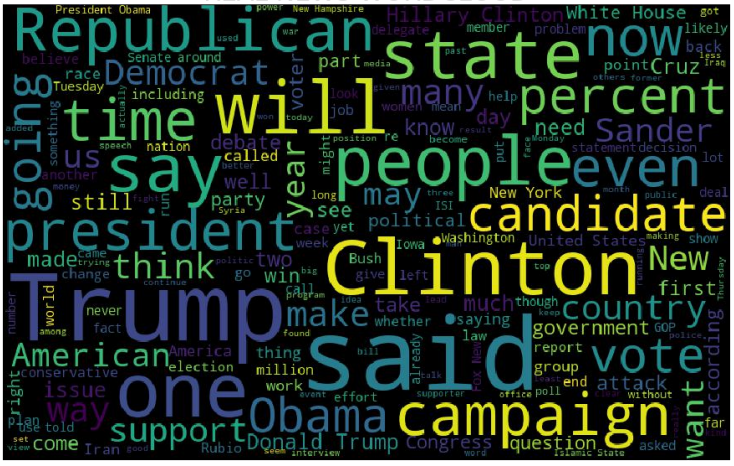
It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning models need numeric data to be trained and make a prediction. Word tokenization becomes a crucial part of the text (string) to numeric data conversion.

➢ Data Distribution:

We performed some data analysis on the text and wanted to understand how the text is distributed. We had analysed and represented our data (text) distribution in a few different perspectives. We first analysed the data through graphing.

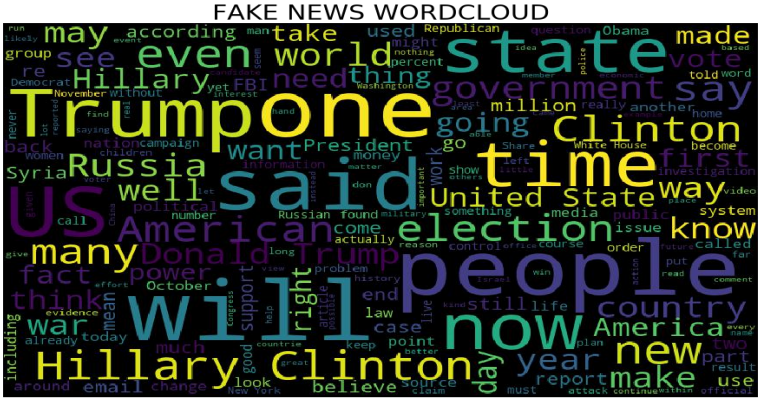
The most frequently occurring words in the REAL news wordcloud:

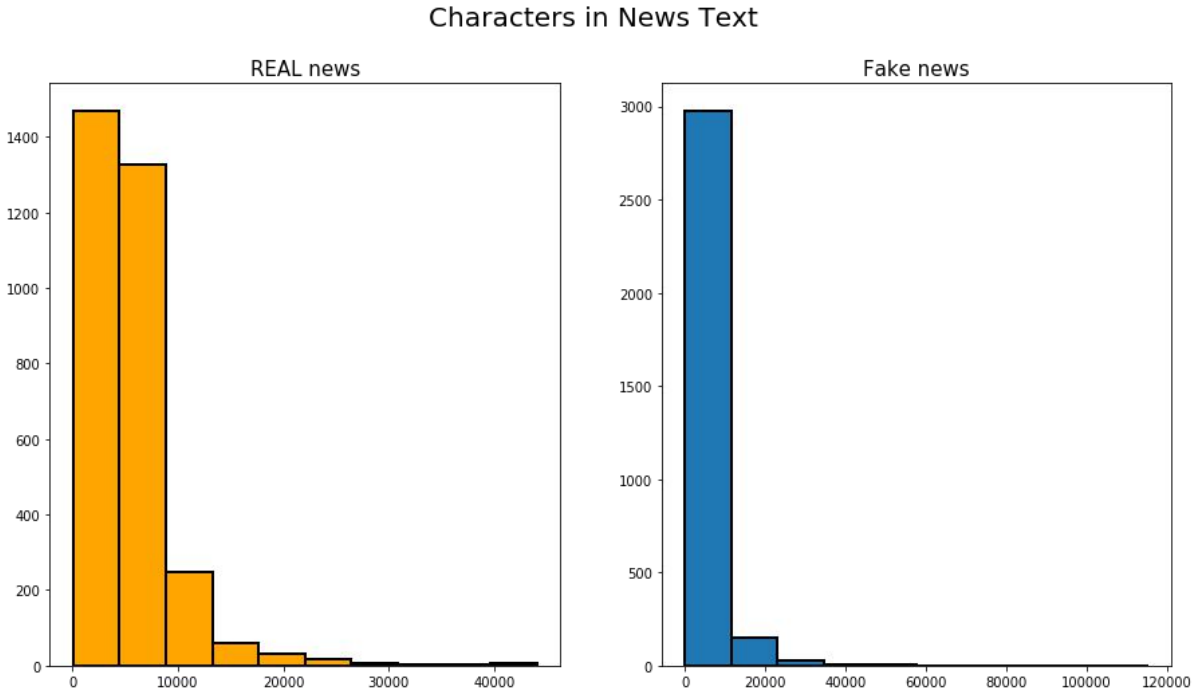
REAL NEWS WORDCLOUD

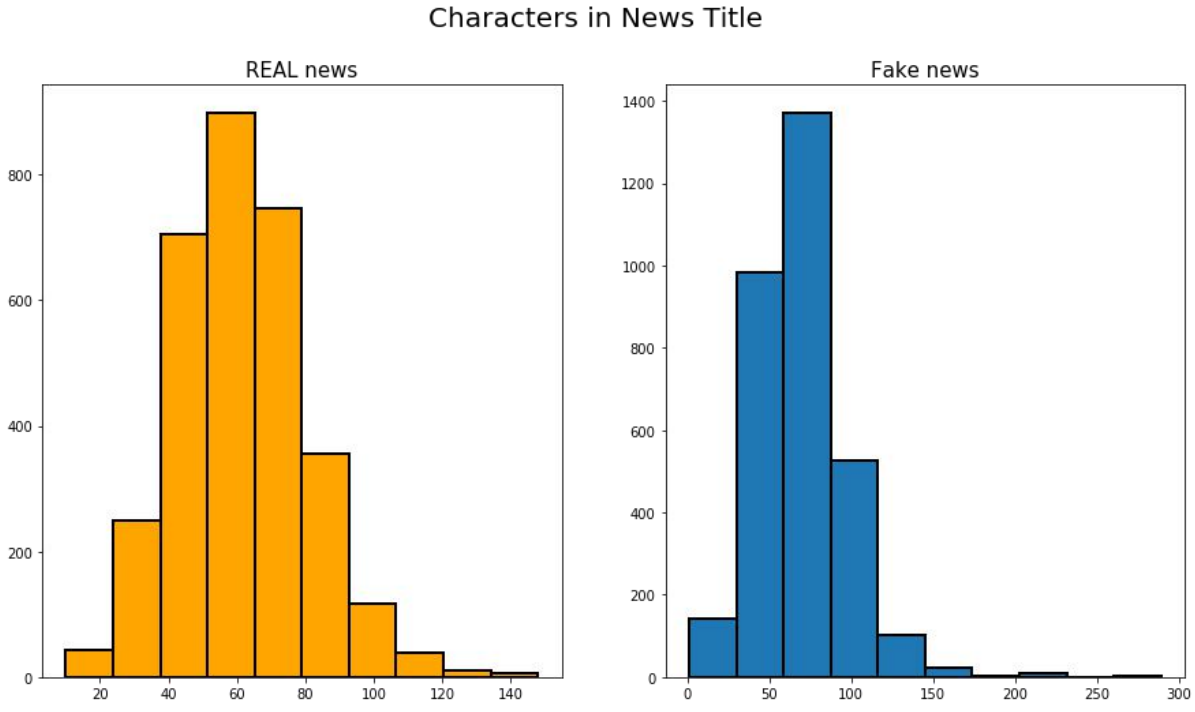


The most frequently occurring words in the FAKE news wordcloud:

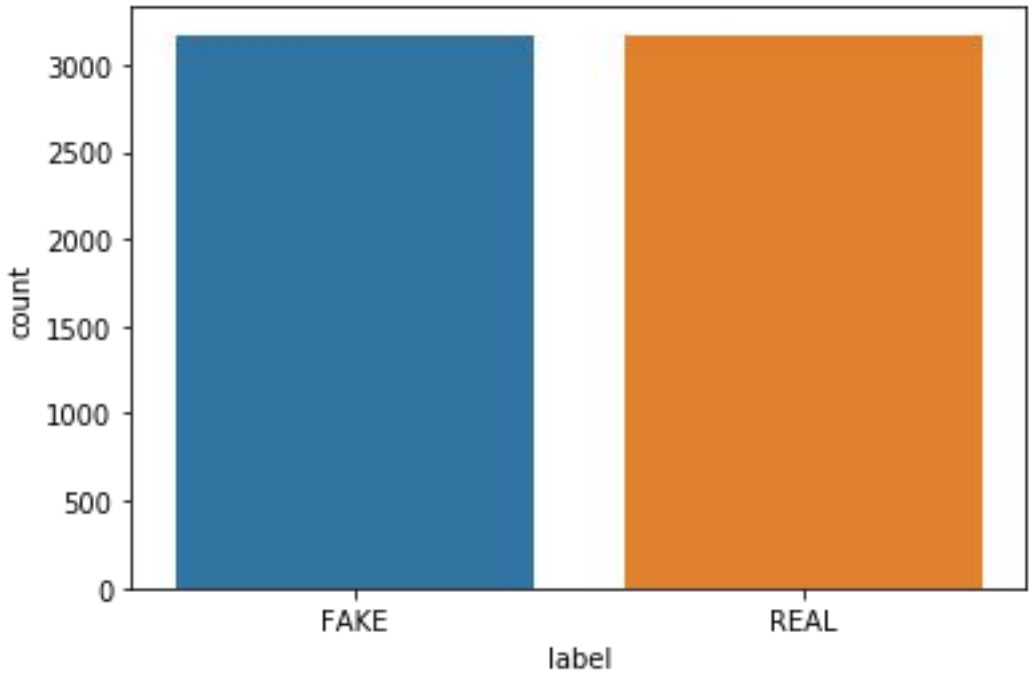
FAKE NEWS WORDCLOUD




Characters in News Text
REAL news
Fake news
0
200
400
600
800
1000
1200
1400
0
10000
20000
30000
40000
0
500
1000
1500
2000
2500
3000
0
20000
40000
60000
80000
100000
120000

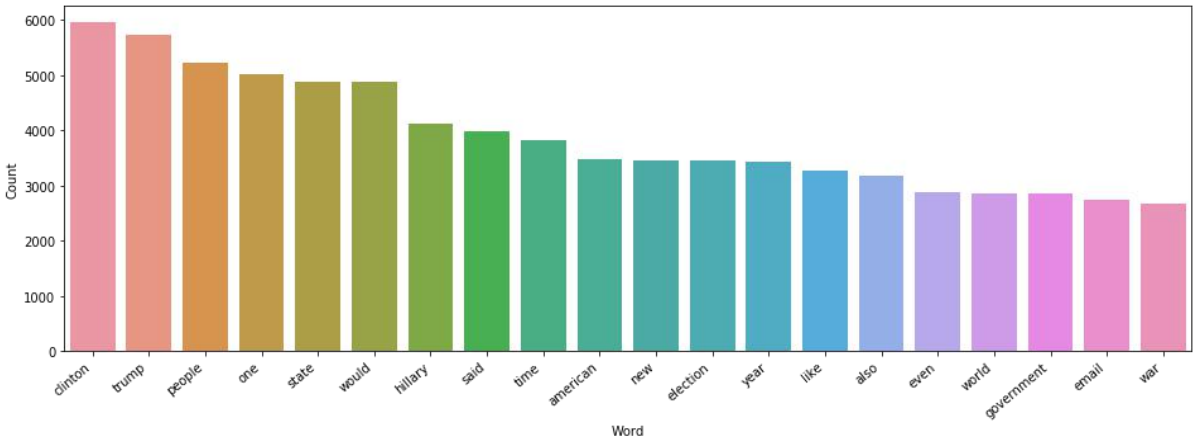

Characters in News Title
REAL news
Fake news
0
200
400
600
800
20
40
60
80
100
120
140
0
200
400
600
800
1000
1200
1400
0
50
100
150
200
250
300

The Fake and Real news representation in a bar graph:

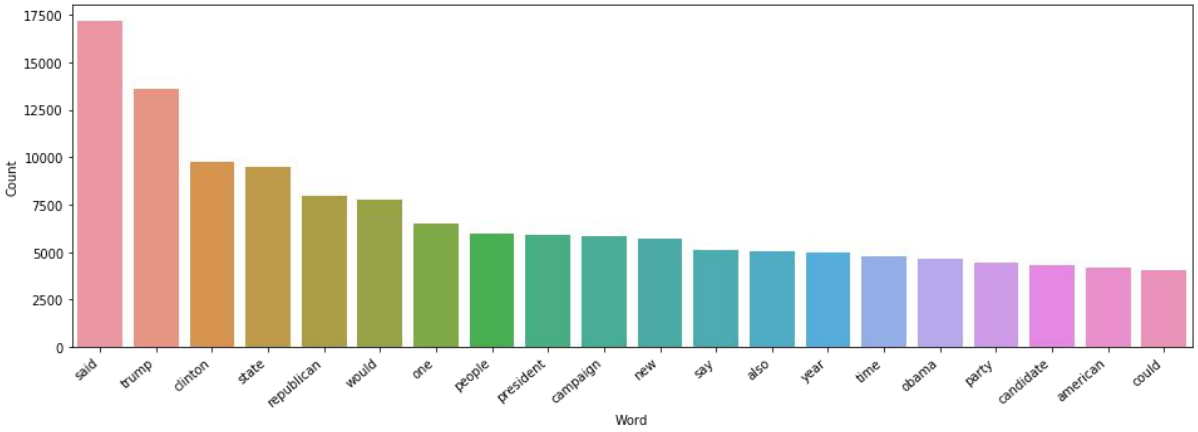


Frequency of words that occur frequency in data set shown in plots:

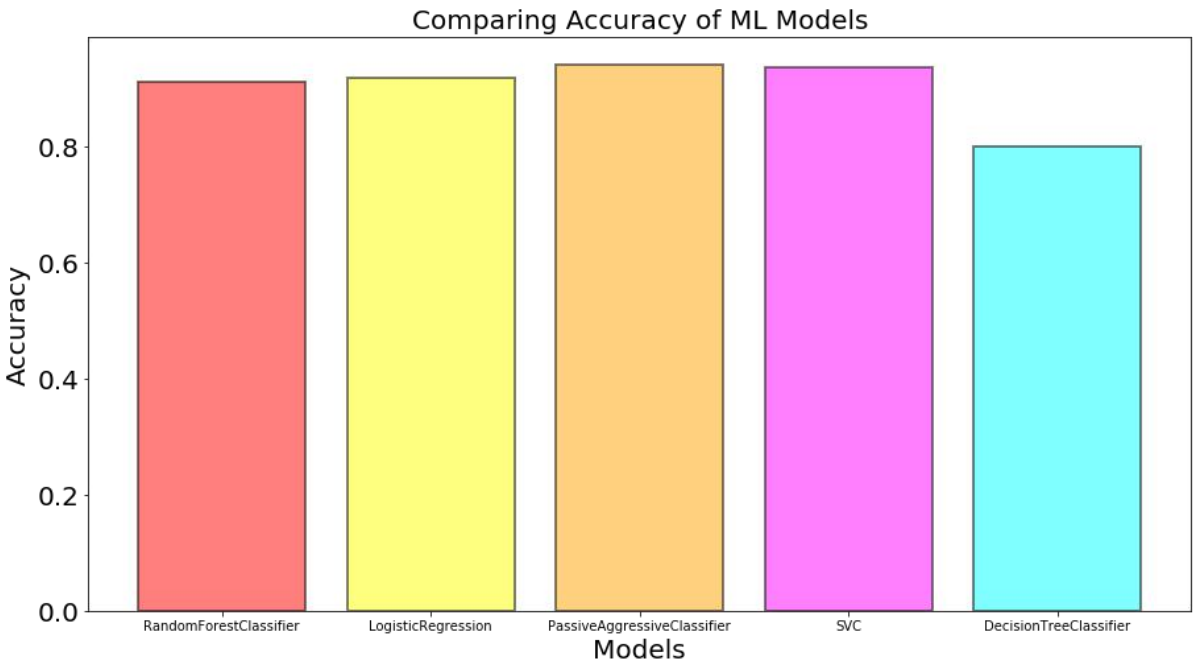
1. Fake news



2.Real news



Accuracy of different ML models:



➢ Natural Language Processing Models:

After text have been cleaned, they are mapped into numeric representations in form of vectors of the textual data using pre-training algorithm (i.e.TF-IDFVectorizer). Each sample, originally consisting of all text, is converted into a vector of features. Since only the text is passed into these pre-training algorithm, this stage is unsupervised. In the cases of Tf-idf Vectorizer, the number of features is clipped at 10000 to avoid memory overrun and overfitting (because of the large number of features (the vocabulary)).

TF-IDF Vectorizer:

Although TF-IDF is an old algorithm, it is simple and effective to be used in the phase of pretraining. The computation of Tf-idf Vectorizer involves computing the product of term frequency and inverse document frequency. As the term implies, TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears.

The term frequency tf(t,d) calculates the proportion of times that the term $t \in V$ (d) appears in the document d.

The vocabulary V (d) =summation of [n(t,d)] is constructed by the document d. Thus, if a word w0 does not appear in a document d0, the term frequency tf(t0,d0) in this case would be zero. The idea of the term frequency is essentially the same as CountVectorizer.

$$tf(t,d) = n(t,d) / V(d)$$

n(t,d) = occurrence of the word t in the document d

Given a document collection D, the inverse document frequency idf(t,D) is the log of the number of documents N divided by df(t,D), the number of documents $d \in D$ containing the term t. As a result, common words in D will have a low term frequency score, while infrequent words will have a high term frequency. Thus, the term frequency will be very likely to separate fake news that often have less common words (even ungrammatical) from real news that usually consist of common words.

$$idf(t,D) = \log(N / df(t,D))$$

As a summary, TF-IDF score w(t,d) for a word increases with its count, but will be counteracted if the word appears in too many documents.

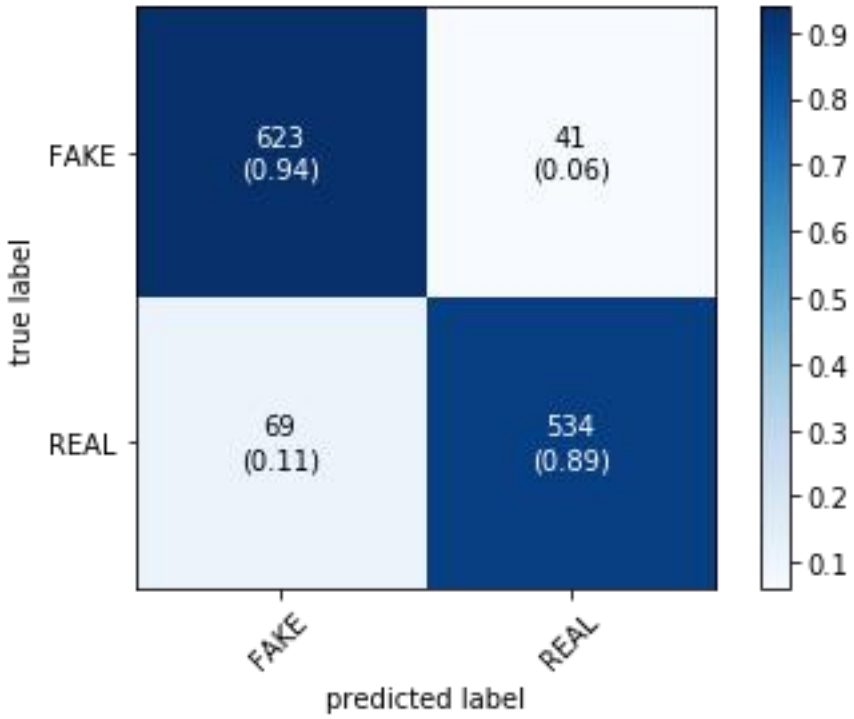
$$w(t,d) = tf(t,d) \times idf(t,D)$$

➢ Different models for implementation of the data:

1. RandomForestClassifier:

A random forest is an ensemble classifier that estimates based on the combination of different decision trees. So random forest will fit a number of decision tree classifiers on various subsamples of the dataset. A random best subsets are built by each tree in the forest. In the end, it gives the best subset of features among all the random subsets of features.

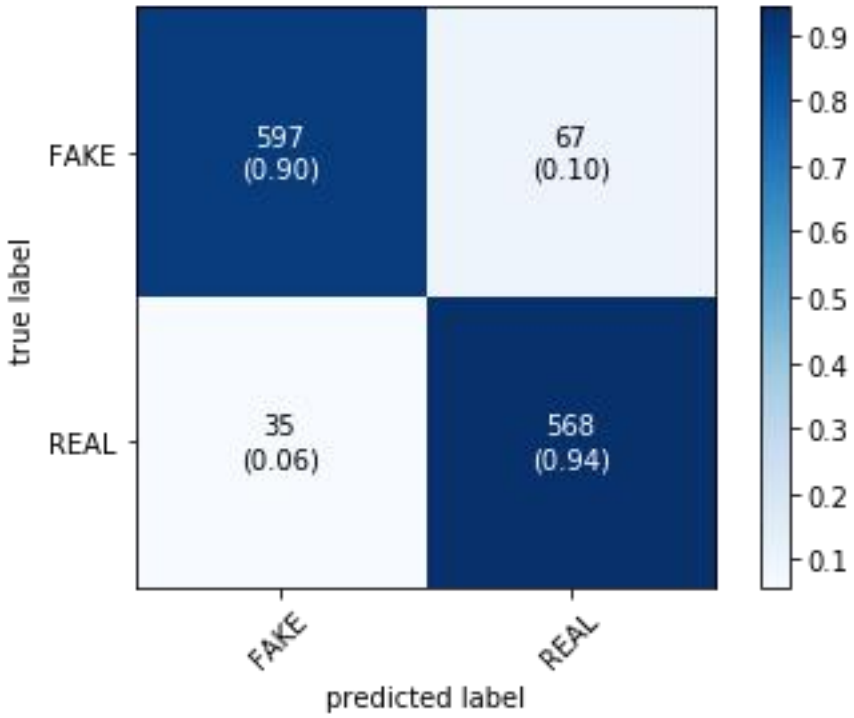
Confusion matrix for RandomForestClassifier:



2. LogisticRegression:

Logistic regression is a statistical machine learning algorithm that classifies the data by considering outcome variables on extreme ends and this algorithm is providing a discriminatory line between classes. Compared to another simple model, linear regression, which requires hard threshold in classification,logistic regression can overcome threshold values for a large dataset. Logisticregression produces a logistic curve, which is limited to values between 0 to 1, by adding sigmoid function in the end.

Confusion matrix for LogisticRegression:

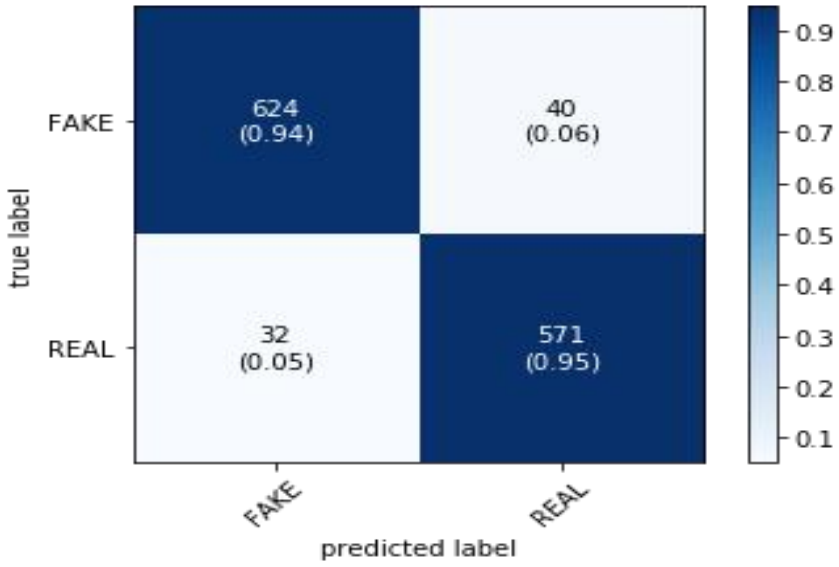


3. The Passive-AggressiveClassifier:

These algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications.

Passive-Aggressive algorithms are generally used for large-scale learning. It is one of the few '**online-learning algorithms**'. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated step-by-step, as opposed to batch learning, where the entire training dataset is used at once.

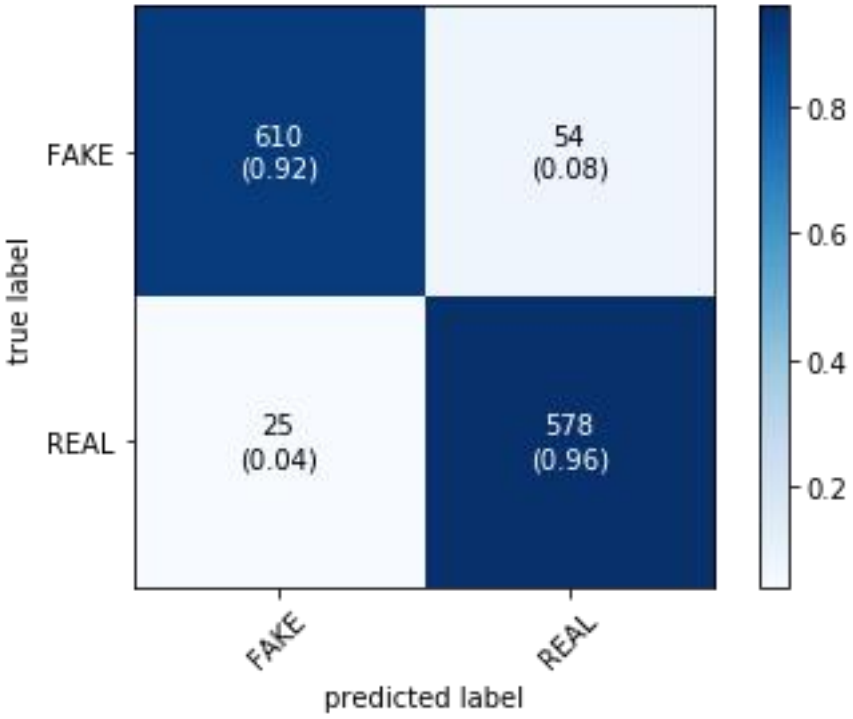
Confusion matrix for Passive Aggressive Classifier:



4. Support Vectors Classifier :

This tries to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane.

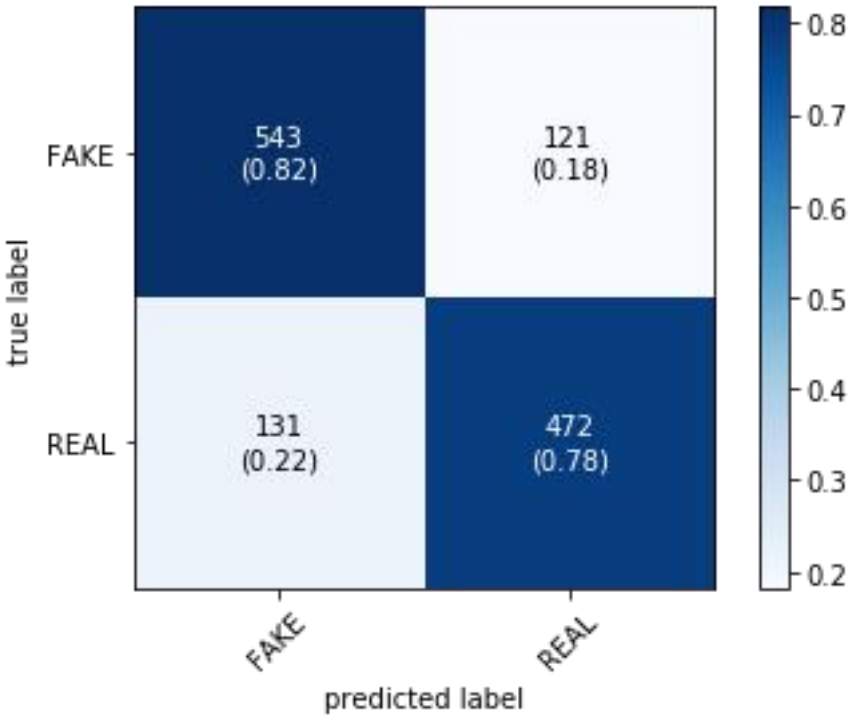
Confusion matrix for Support Vector Classifier:



5. Decision Tree Classifier:

It is a simple and widely used classification technique. It applies a straight forward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receive an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.

Confusion matrix for Decision Tree Classifier:



Conclusion:

This report provides a fairly simple approach to encode texts and how the presence of words in general impacts the classification of texts as real and fake. We achieved high accuracy results in most of our algorithms and in particular neural networks generally do better than the others.

While we achieve great performance in this dataset, the question remains as to whether X (to be replaced by the best model) can still perform well in tasks that classify news into more than two categories, such as the Fake News Challenge.

Lastly, in our model, the pre-training is done on the dataset given (will make the model specific to the task), instead of on the big corpus available online, such as Google's pre-trained Word2Vec model. If the task were a classification of four or eight categories, pre-trained model on large corpus may perform better as the model is pre-trained on more words.