CS-351 MACHINE LEARNING

FAKE NEWS DETECTION

National Institute of Technology Karnataka Surathkal

By:

Aditya Chirania (181CO104)

Anusha P. Das (181CO108)

Table of Contents

INTRODUCTION

Fake news can cause considerable misunderstandings in society, and the consequences can be catastrophic. It is imperative to differentiate which news is true and which is not. This project leverages the help of machine learning algorithms to perform the classification/prediction of news as true and false.

We will depict how good cleaning techniques of data can impact the performance of the fake news classifier in this project. We use text-preprocessing techniques like removing stop-words, lemmatization, tokenization, and vectorization before we feed the data to models. We have fed data to various models and compared and contrasted their performance. These data cleaning techniques fall under Natural Language Processing.

Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

DATA PRE-PROCESSING

MISSING DATA IMPUTATION

Datasets may have missing values, and this can cause problems for many machine learning algorithms. As such, it is good practice to identify and replace missing values for each column in your input data prior to modeling your prediction task. This is called missing data imputation, or imputing for short.

If a particular attribute has been marked with null, we replace it with a space (‘ ‘). This is a rather better alternative than deleting tuples with null values.

USING A REGEX TO REMOVE SPECIAL CHARACTERS

The special characters in a sentence do not contribute to indicating whether a news is true or false. Hence, we remove all the punctuation marks from the dataset. We remove all punctuation marks with the help of regular expressions.

```
Example String : "! NLP is $$ ^sh!!!o%%rt &&%$fo@@@r^^^&&!&
*Natural@# Language&&\”
Resultant String : NLP is  short for Natural Language Processing
```

`Regex used: [^\w\s]` , This would match only words or spaces.

TOKENIZATION OF DATA

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. In this context breaking sentences up into its constituent words makes the most sense instead of breaking it up into letters, since words have meaning.

```
Example: "Computers are not as great at understanding words as they are
numbers."
Resultant tokens: ['Computers', 'are', 'not', 'as', 'great', 'at',
'understanding', 'words', 'as', 'they', 'are', 'numbers', '.']
```

REMOVING ALL STOP WORDS

Stop words like “if”, “the”, “is”, “a”, “an” etc. should not be given importance by a machine learning model as they are common english words which don't contribute to the authenticity or originality of any news. Since they are used frequently, including them in the dataset can influence the prediction of the model, which is undesirable. Hence, we remove all the stop words from the dataset.

```
Example: "Does this thing really work? Lets see."
Result: ['thing', 'really', 'work', 'lets', 'see']
```

LEMMATIZATION

Words like ‘playing’ and ‘plays’ originate from the same root word ‘play’. Replacing words in different tenses and participles with the root word can help analyse the frequency of a word better. Hence, we convert all words originating from the same word into that word.

```
Example:
Kites-> kite
Babies-> baby
languages-> language
cities-> city
mice-> mouse
```

COMBINING ATTRIBUTES INTO ONE

We believe all attributes relating to the news are important to help determine if the news is fake or real. Hence, we combine the various attributes about the news into one aggregated attribute and use that for vectorization.

COUNT VECTORIZATION

The pre-processed text need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called **feature extraction (or vectorization)**.

We will be creating vectors that have a dimensionality equal to the size of our vocabulary, and if the text data features that vocab word, we will put a one in that dimension. Every time we encounter that word again, we will increase the count, leaving 0s everywhere we did not find the word even once.

```
Example:
Sentence 1: "the sky is blue sky"
Sentence 2: "the sun is bright sun"
Feature set: ['blue', 'is', 'the','sun','bright', 'sky']

Resulting Matrix:
     [1 1 1 0 0 2]
     [0 1 1 2 1 0]
```

TF-IDF TRANSFORMATION

With the help of the count vectorized matrix we transform the count vectorized matrix to a matrix with TF-IDF Values for each feature.

TF - Term Frequency (the same as what we had seen in the CountVectorizer) IDF - Inverse Document Frequency

Merely considering frequencies of words may be misleading as some words may turn out to be very trivial. Hence to maintain a balance between the importance of a word in the document and its frequency we use TF-IDF. TF-IDF is the product of Term frequency and Inverse document frequency.

TERM FREQUENCY

Term frequency, tf(t,d), is the frequency of term t,

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

where $f_{t,d}$ is the *raw count* of a term in a document, i.e., the number of times that term t occurs in document d. There are various other ways to define term frequency:

- the raw count itself: tf(t,d) = $f_{t,d}$
- Boolean "frequencies": tf(t,d) = 1 if t occurs in d and 0 otherwise;
- term frequency adjusted for document length: tf(t,d) = $f_{t,d}$ ÷ (number of words in d)
- logarithmically scaled frequency: tf(t,d) = log (1 + $f_{t,d}$);
- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the raw frequency of the most occurring term in the document:

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

INVERSE DOCUMENT FREQUENCY

The **inverse document frequency** is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N : total number of documents in the corpus , N = |D|
- {d∈D : t∈d} : number of documents where the term t appears (i.e.,
- tf(t,d)≠ 0). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to 1 + |{d∈D : t∈d}|.

TF-IDF VALUE

TF-IDF = Term Frequency x Inverse Document frequency

We shall be using these TF-IDF values to create the tf-idf matrix and feed to various machine learning models.

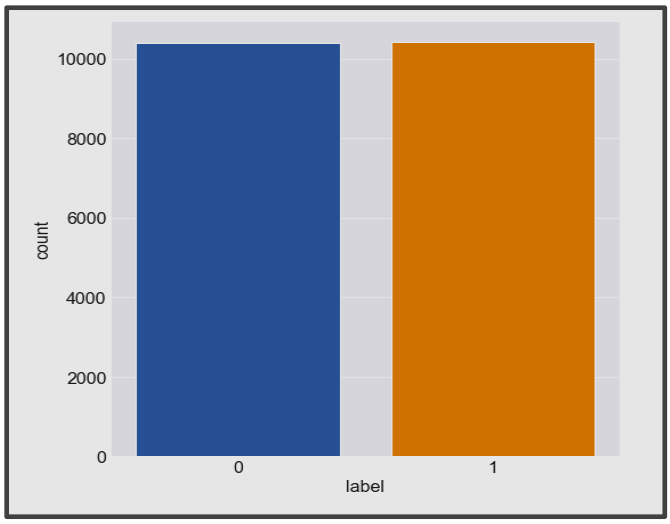
DATASET AND ITS PRIMARY EXAMINATION

The dataset is a collection of about 20800 news articles. This dataset has been compiled and created by the University of Tennessee's Machine Learning Club, USA. This dataset is freely available here : https://www.kaggle.com/c/fake-news/data.

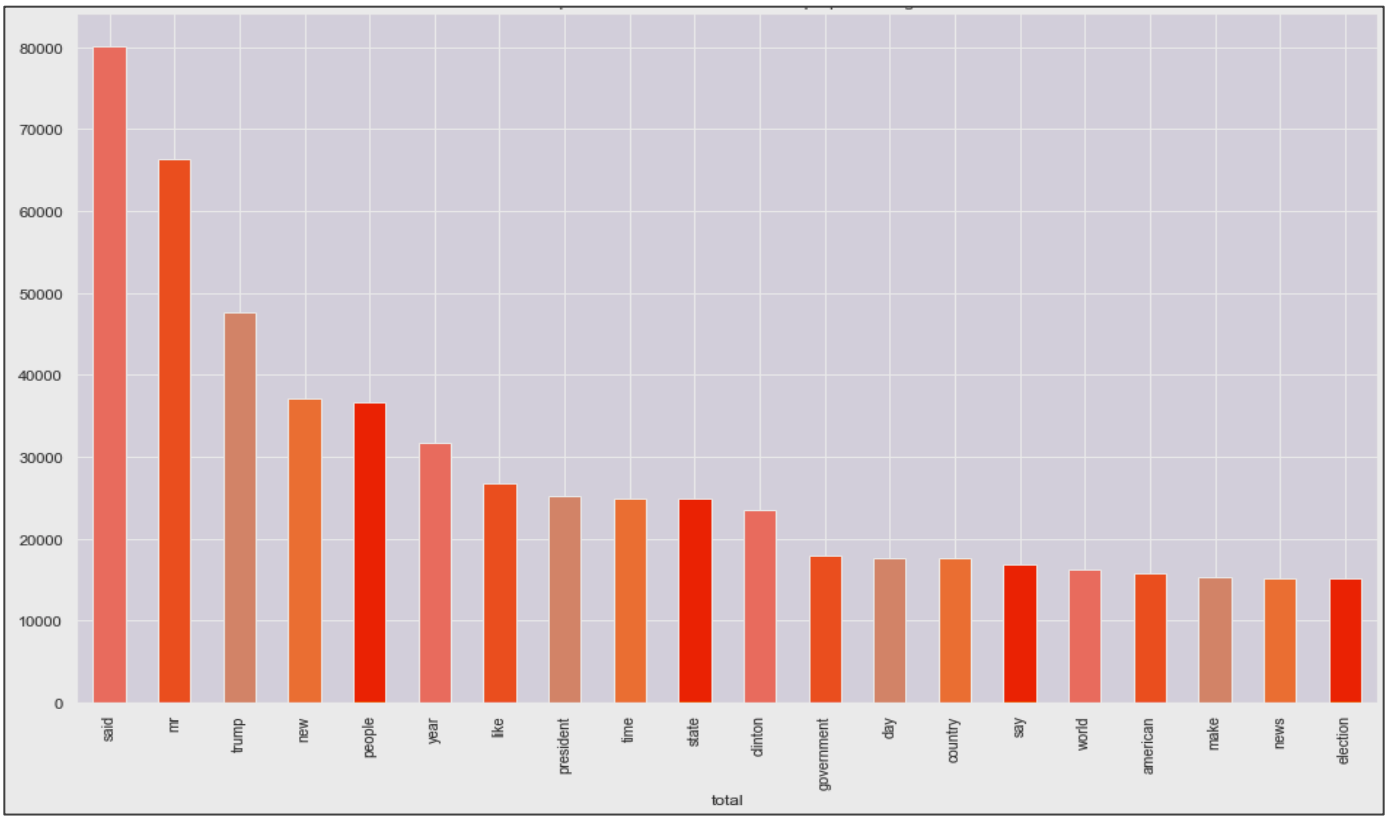
The dataset consists of the following attributes:

- id: unique id for a news article
- title: the title of a news article
- author: author of the news article
- text: the text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable
    - 1: False News or Unreliable
    - 0: True News or reliable

Firstly, we have seen that the dataset is a rather balanced dataset. It has 10387 false news articles and 10413 true news articles. This is a good characteristic of the dataset. It shall help models give unbiased decisions.



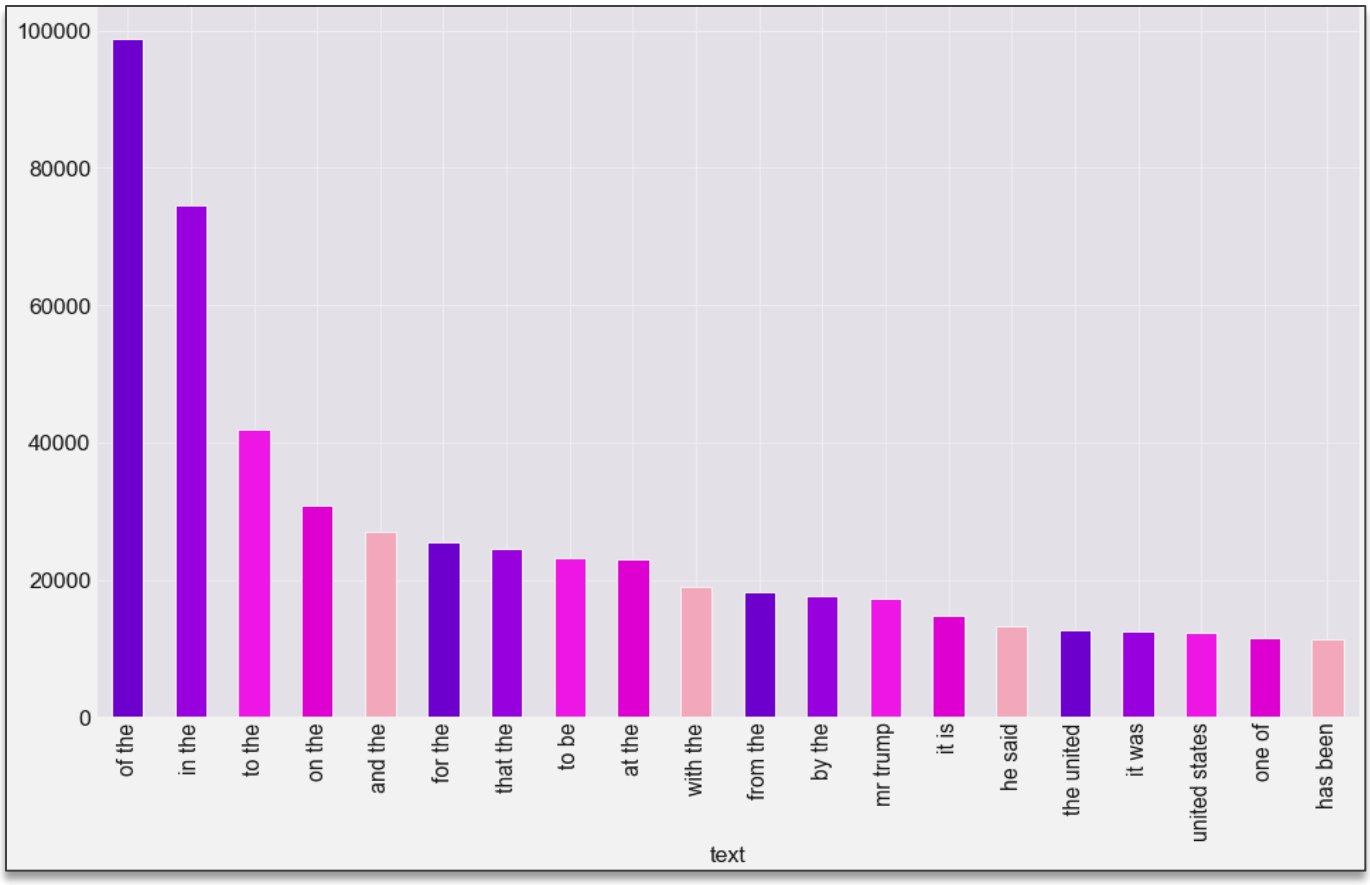
Some analysis has been done on the data and as mentioned earlier, stop words like “the”,”to”,”of”,”and”, etc are amongst the most frequent words present in the dataset. The top 20 words that occur in the dataset before removing any stop words are as follows.



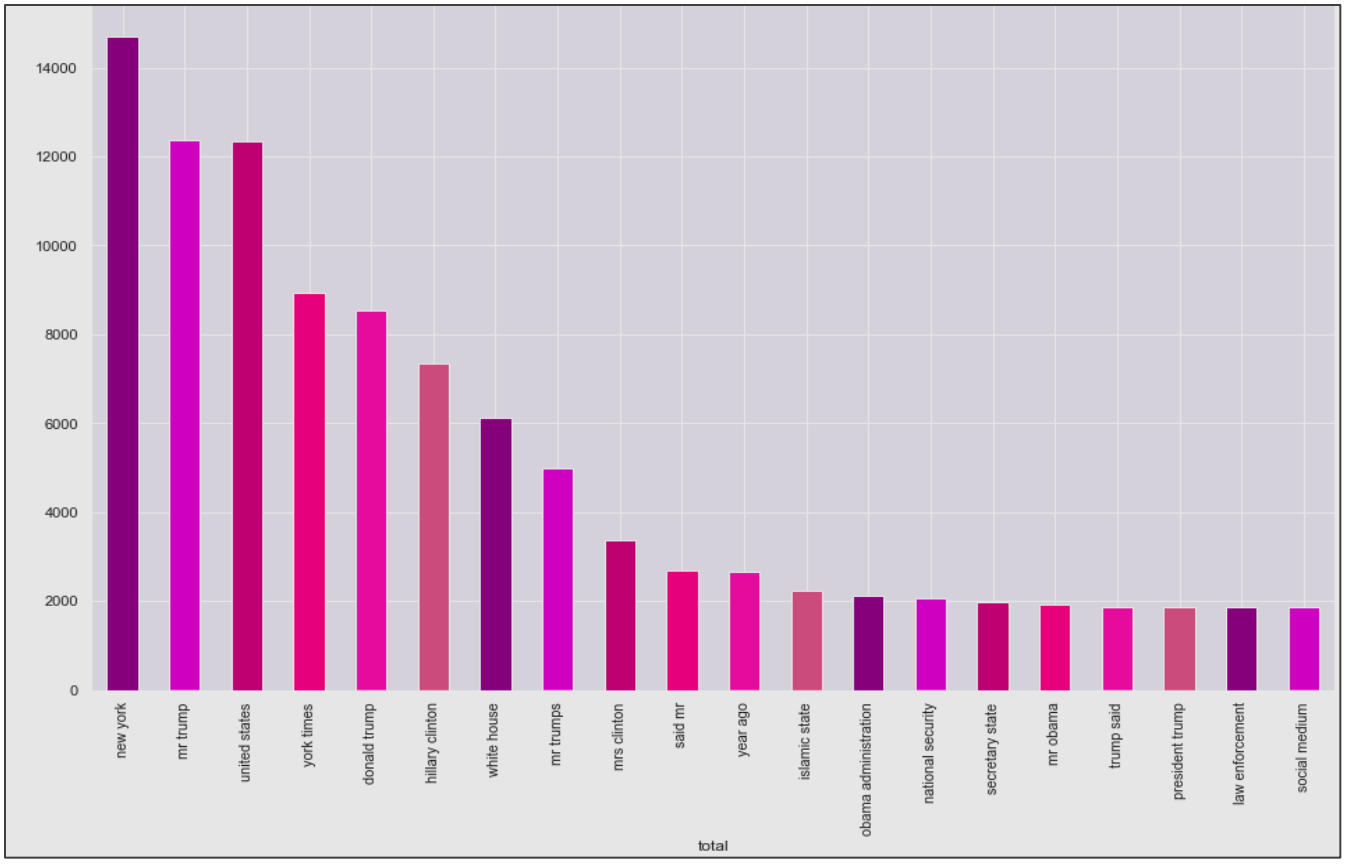
Now words like “said”, “mr”, “trump”, “new”, “people”, ”year” etc. are most frequent, which can serve as useful information for the models.

We also have explored the bigrams in the dataset as well to gain more familiarity with the subject matter of the news articles. Before removal of stop words there is very little clarity of the subject matter of the news articles. Hence removal of stop words aids in understanding the subject matters in the news articles as well.

The following graph represents the 20 most frequent bigrams which occur in the dataset before removing any stop words. As one can see constructs like “of the”, “in the”, “to the” are occurring most frequently but are not helping one gain an idea of the subject matter of the news.



Top 20 bigrams after text-preprocessing is shown below:



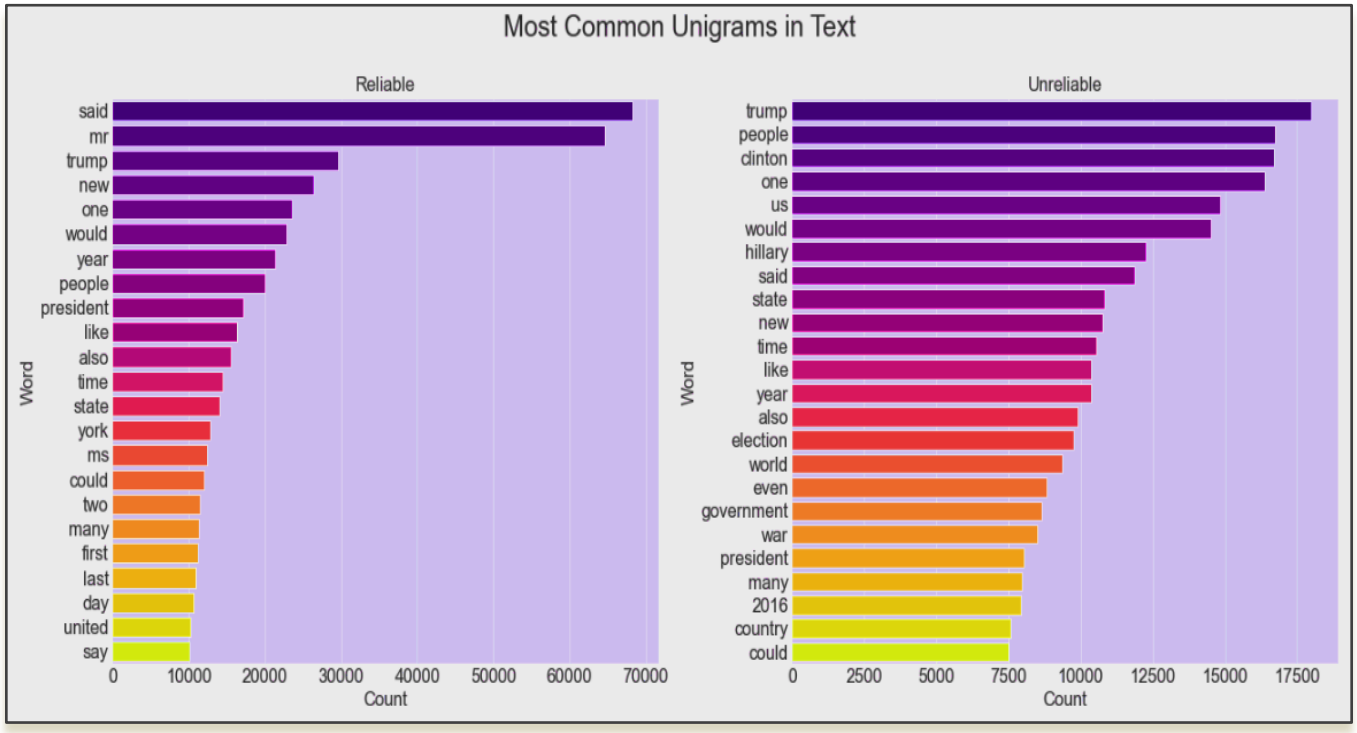
It can be observed from the above figure that most news articles in the dataset revolve around the subject matter “new york”, “mr trump”, and “united states”.

The above observations concreted our approach of removing stop words from the dataset.

We have also inferred that the dataset is well balanced which helps us use models in an unbiased manner.

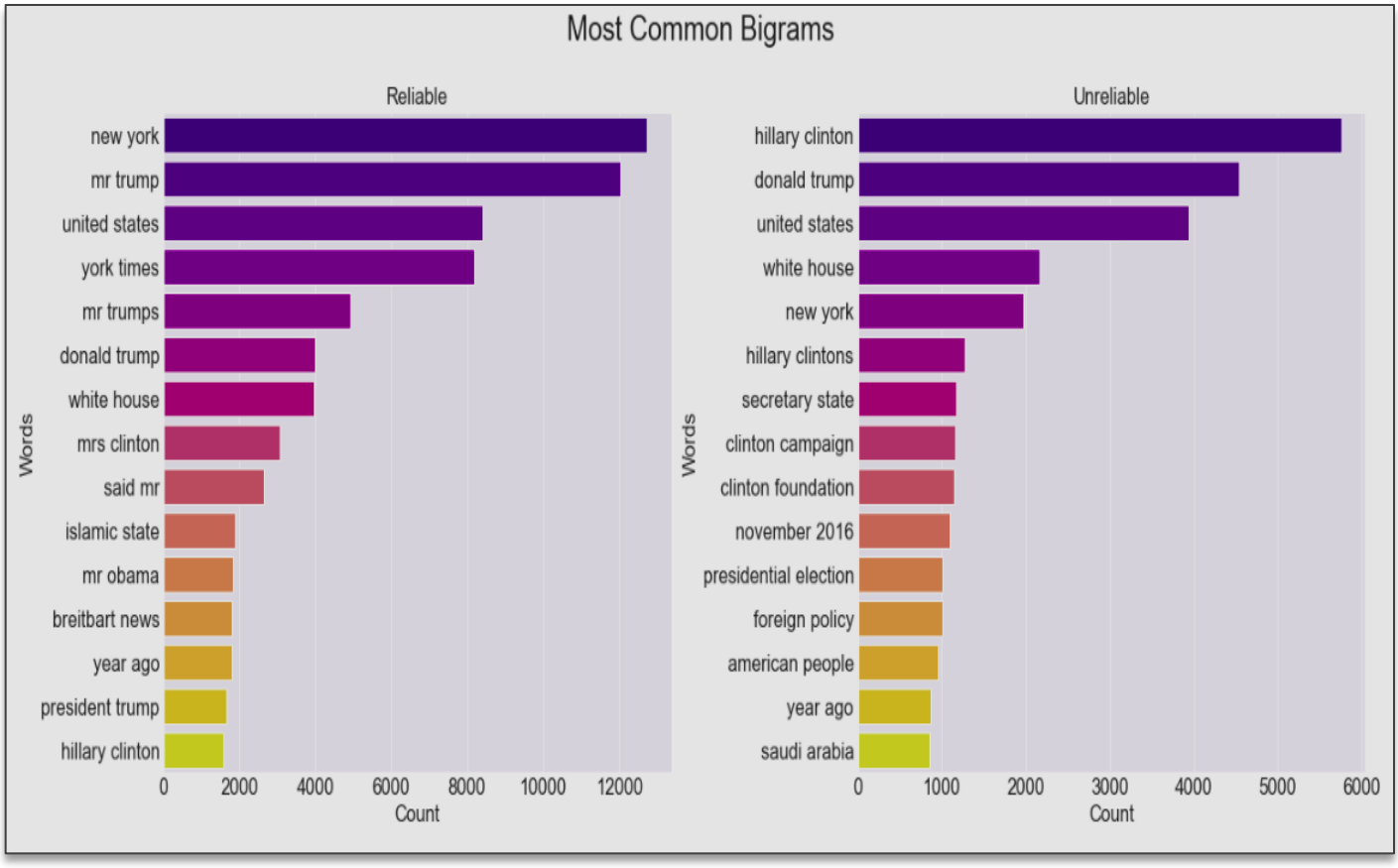
Now we shall move on to analyse the data label wise post preprocessing of data. This is being done to attempt to understand certain patterns and gain some intuitive ideas about the model that can be a perfect fit on this dataset

.



This helps us attempt to find certain trends that could be present in only unreliable news or only in reliable news. As you can see here, trump is the most frequent word in unreliable news and said is the most frequent word in reliable data.

The most frequent bigrams may give us a better idea of the subject matter of reliable news and unreliable news.



It is observed that "new york" has been most spoken of in reliable data compared to in unreliable data. Comparing the presence of a certain important bigram in reliable and unreliable news can serve to be a good way to guess the classification for the news.

Now in the next section, we shall be leveraging the TF-IDF transformed vectors to classify news as fake or real.

MODELS APPLIED

We have applied models with 2 approaches to data cleaning:

- **Approach 1:** In the first approach, we have selected only one feature i.e. the attribute-news text and have directly applied the Feature Extraction tools like TF-IDF vectorization after eliminating rows with null values from the text.
- **Approach 2:** In the second approach, we have followed the steps of combining all attributes including “author”, “text” and “title” into one column. , replacing null values with spaces(missing data imputation), removing stop-words and special characters, applied lemmatization, and finally converting the preprocessed textual data into numerical values, for use as input in Machine Learning models.

The difference of results from these approaches shall be indicating how important good NLP techniques are and how cleaning techniques like lemmatization and removal of stop words can impact the performance of models.

PASSIVE AGGRESSIVE CLASSIFIER:

The results obtained on applying Passive Aggressive Classifier on our dataset are given below.

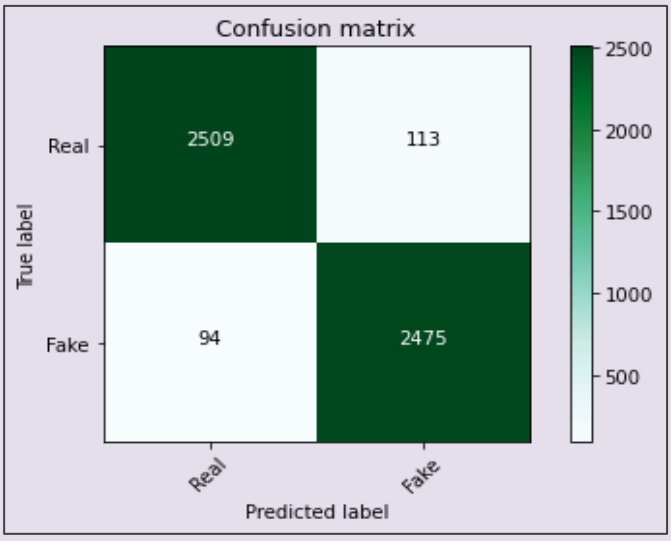
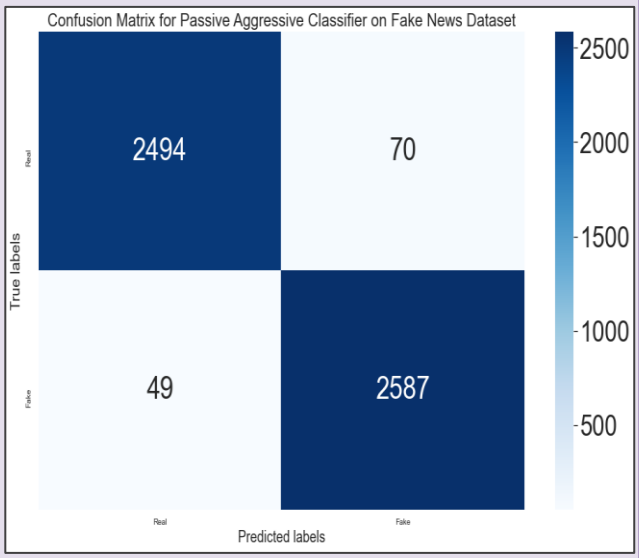
Approach 1:

	precision	recall	f1-score	support
0	0.96	0.97	0.97	2630
1	0.96	0.96	0.96	2561
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

Approach 2:

	precision	recall	f1-score	support
0	0.98	0.97	0.98	2564
1	0.97	0.98	0.98	2636
accuracy			0.98	5200
macro avg	0.98	0.98	0.98	5200
weighted avg	0.98	0.98	0.98	5200

Classification Report for Approach 1 & Approach 2

Approach 1	Approach 2
	

MULTI LAYER PERCEPTRON

The results obtained on applying Multi-Layer Perceptron model on our dataset are given below.

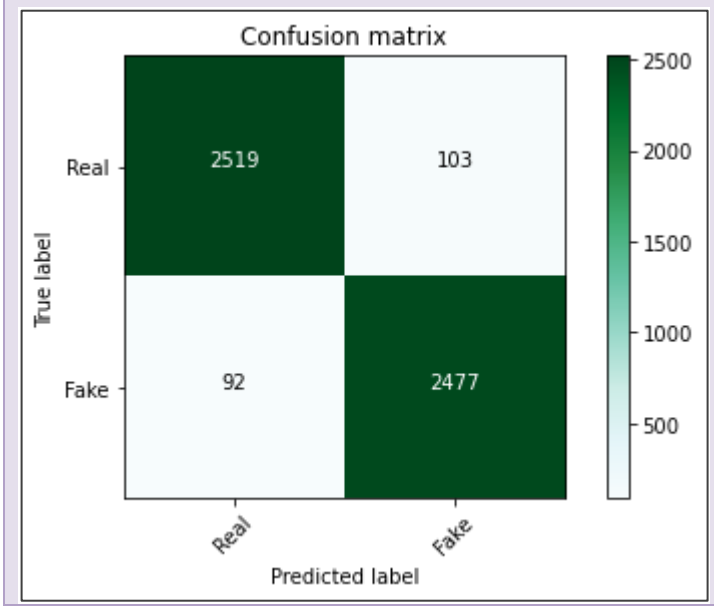
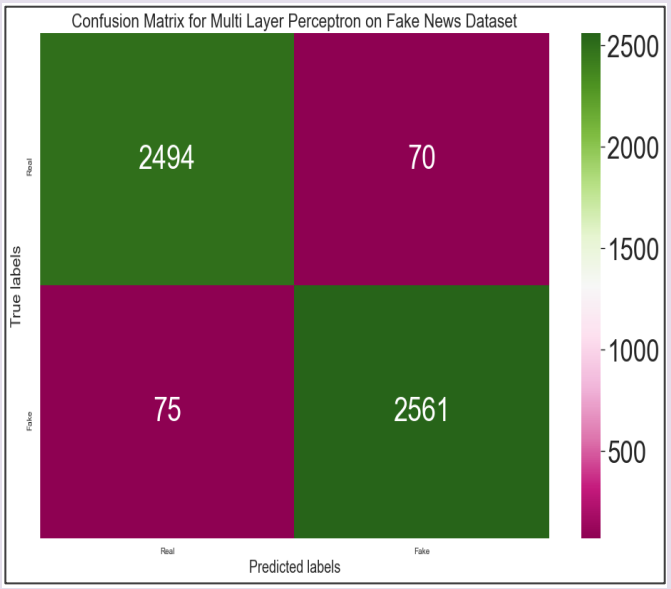
Approach 1:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	2622
1	0.96	0.96	0.96	2569
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

Approach 2:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	2564
1	0.97	0.97	0.97	2636
accuracy			0.97	5200
macro avg	0.97	0.97	0.97	5200
weighted avg	0.97	0.97	0.97	5200

Classification Report for Approach 1 and Approach 2

Approach 1	***Approach 2***
	

LOGISTIC REGRESSION

The results obtained on using Logistic Regression model on our dataset are given below.

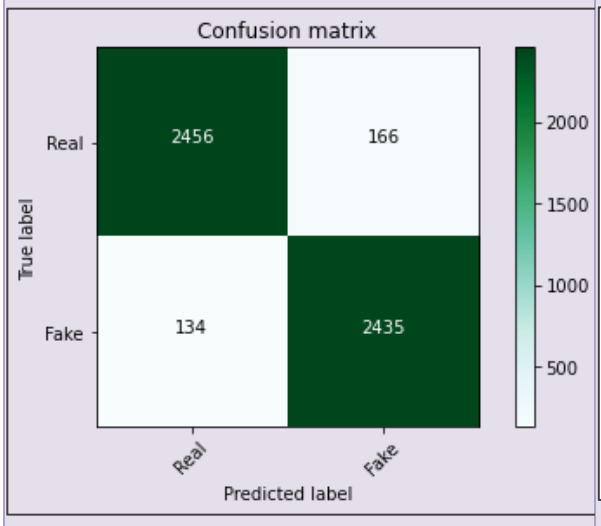
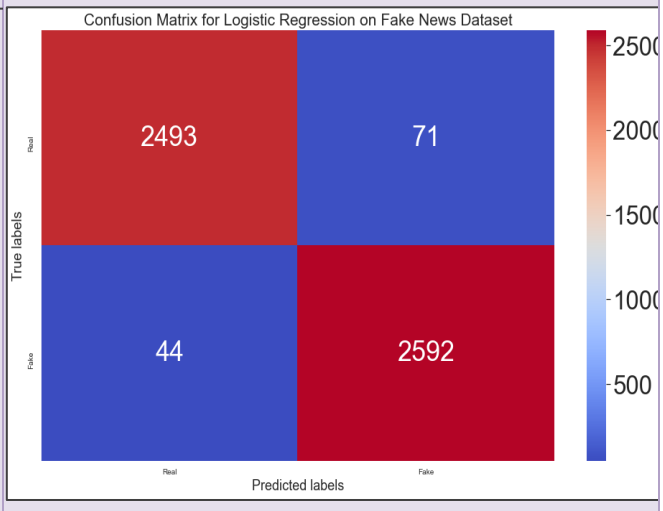
Approach 1:

	precision	recall	f1-score	support
0	0.95	0.94	0.94	2622
1	0.94	0.95	0.94	2569
accuracy			0.94	5191
macro avg	0.94	0.94	0.94	5191
weighted avg	0.94	0.94	0.94	5191

Approach 2:

	precision	recall	f1-score	support
0	0.98	0.97	0.98	2564
1	0.97	0.98	0.98	2636
accuracy			0.98	5200
macro avg	0.98	0.98	0.98	5200
weighted avg	0.98	0.98	0.98	5200

Classification Report for Approach 1 & Approach 2

Approach 1	***Approach 2***
	

MULTINOMIAL NAÏVE BAYES

The results obtained on using Multinomial Naïve Bayes model on our dataset are given below.

Approach 1:

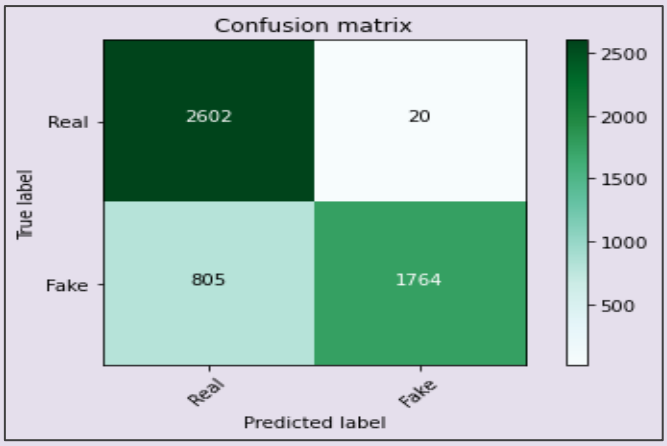
	precision	recall	f1-score	support
0	0.76	0.99	0.86	2622
1	0.99	0.69	0.81	2569
accuracy			0.84	5191
macro avg	0.88	0.84	0.84	5191
weighted avg	0.88	0.84	0.84	5191

Approach 2:

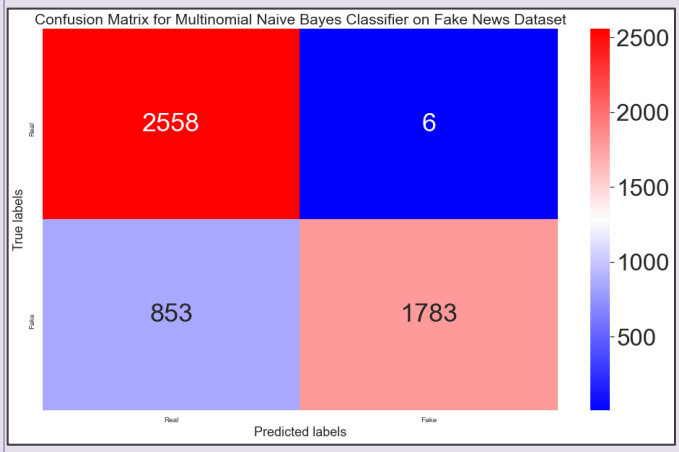
	precision	recall	f1-score	support
0	0.75	1.00	0.86	2564
1	1.00	0.68	0.81	2636
accuracy			0.83	5200
macro avg	0.87	0.84	0.83	5200
weighted avg	0.87	0.83	0.83	5200

Classification Report for Approach 1 & Approach 2

Approach 1



Approach 2



DECISION TREE

The results obtained on using Decision Tree model on our dataset are given below.

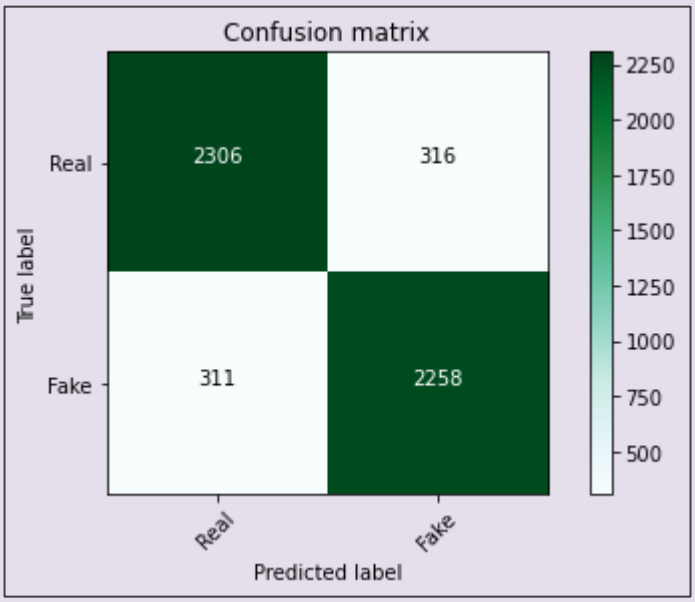
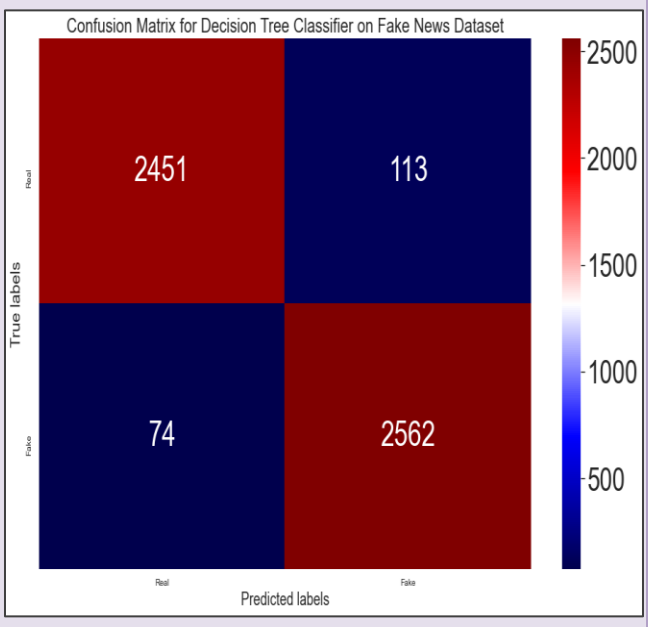
Approach 1:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	2622
1	0.88	0.88	0.88	2569
accuracy			0.88	5191
macro avg	0.88	0.88	0.88	5191
weighted avg	0.88	0.88	0.88	5191

Approach 2:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	2564
1	0.96	0.97	0.96	2636
accuracy			0.96	5200
macro avg	0.96	0.96	0.96	5200
weighted avg	0.96	0.96	0.96	5200

Classification Report for Approach 1 & Approach 2

Approach 1	***Approach 2***
	

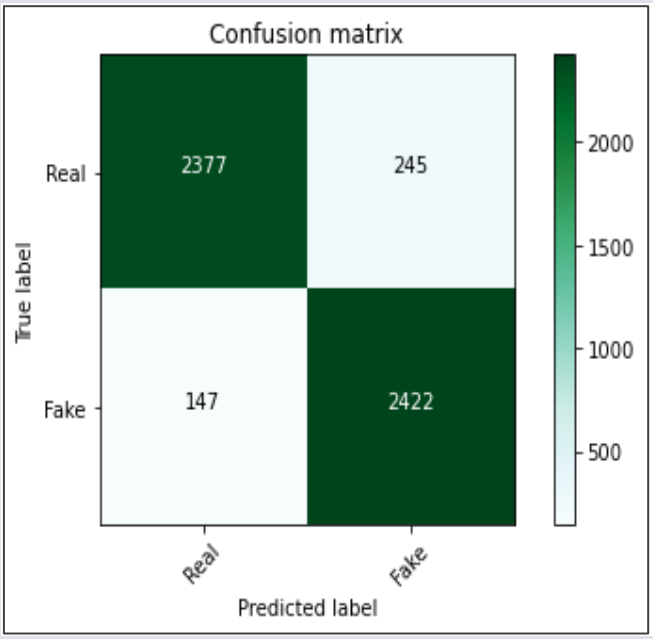
GRADIENTBOOSTINGCLASSIFIER

The results obtained on applying Gradient Boosting Classifier on our dataset are given below.

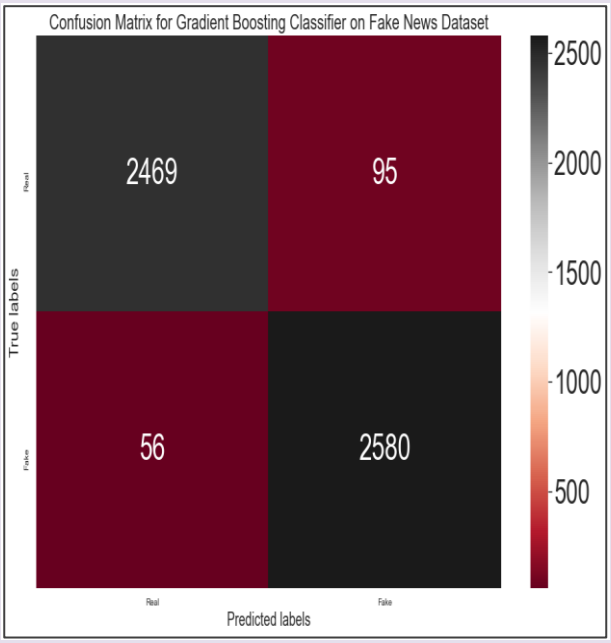
```
Approach 1:
              precision    recall  f1-score   support

           0       0.94      0.91      0.92      2622
           1       0.91      0.94      0.93      2569

    accuracy                           0.92      5191
   macro avg       0.92      0.92      0.92      5191
weighted avg       0.93      0.92      0.92      5191

Approach 2:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      2564
           1       0.96      0.98      0.97      2636

    accuracy                           0.97      5200
   macro avg       0.97      0.97      0.97      5200
weighted avg       0.97      0.97      0.97      5200
```

Classification Report for Approach 1 & Approach 2

Approach 1



Approach 2

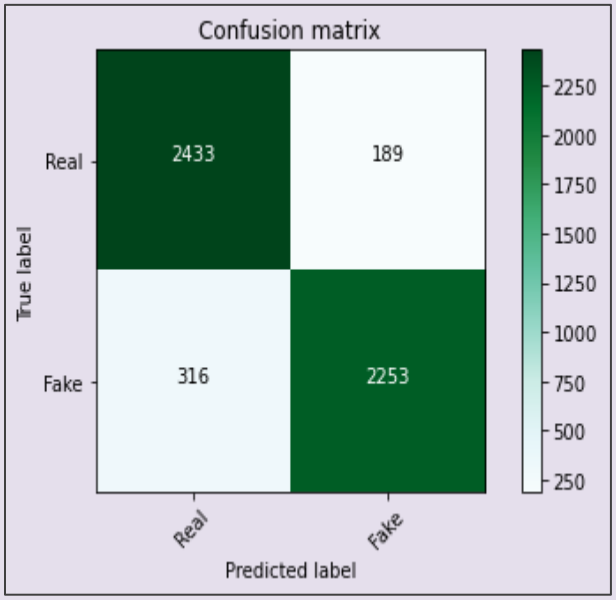
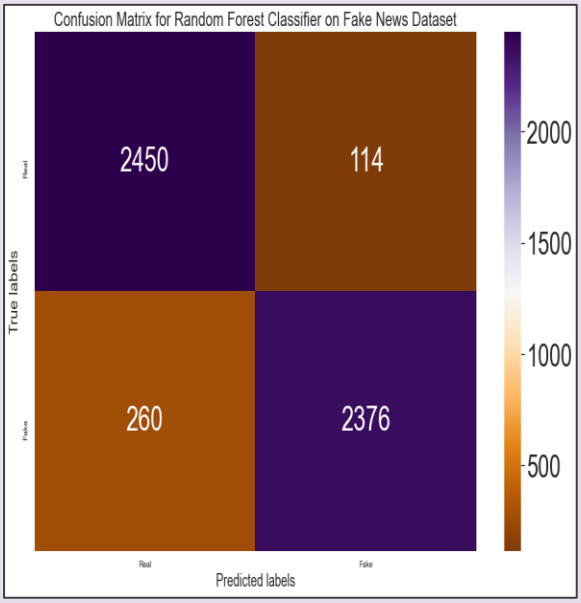


RANDOM FOREST CLASSIFIER

The results obtained on applying Random Forest Classifier on our dataset are given below.

```
Approach 1:
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      2622
           1       0.92      0.88      0.90      2569

    accuracy                           0.90      5191
   macro avg       0.90      0.90      0.90      5191
weighted avg       0.90      0.90      0.90      5191

Approach 2:
              precision    recall  f1-score   support

           0       0.90      0.96      0.93      2564
           1       0.95      0.90      0.93      2636

    accuracy                           0.93      5200
   macro avg       0.93      0.93      0.93      5200
weighted avg       0.93      0.93      0.93      5200
```

Classification Report for Approach 1 & Approach 2

Approach 1	***Approach 2***
	

K-NEAREST NEIGHBOURS

The results obtained on applying K-Nearest Neighbours on our dataset are given below.

Approach 1:

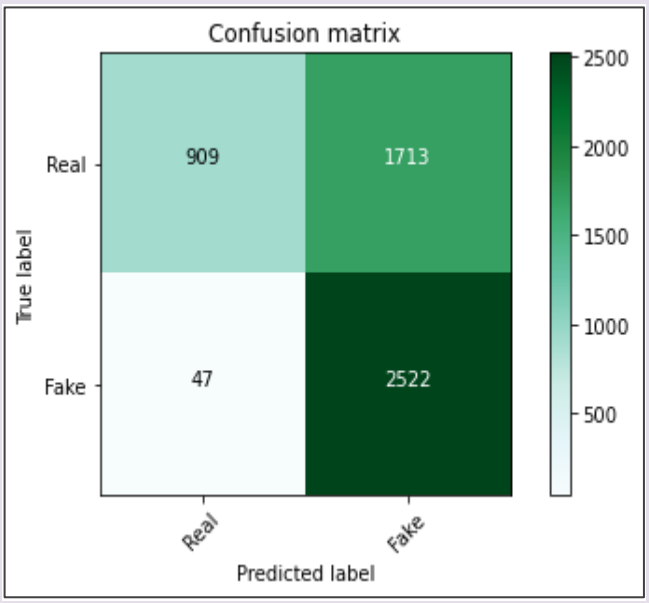
	precision	recall	f1-score	support
0	0.95	0.35	0.51	2622
1	0.60	0.98	0.74	2569
accuracy			0.66	5191
macro avg	0.77	0.66	0.62	5191
weighted avg	0.77	0.66	0.62	5191

Approach 2:

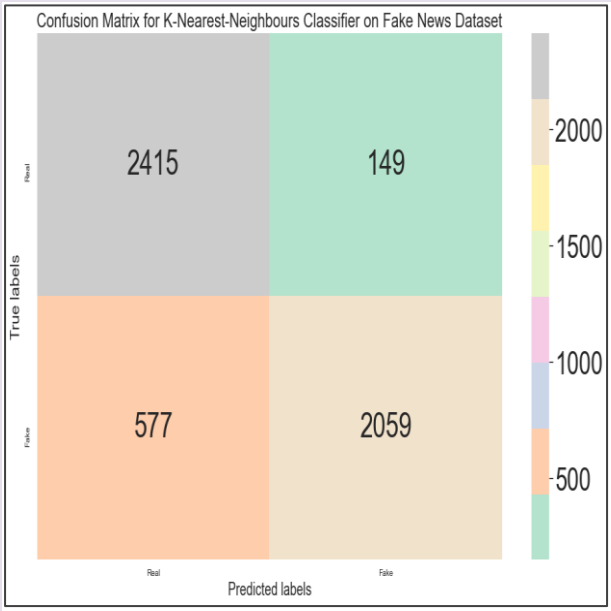
	precision	recall	f1-score	support
0	0.81	0.92	0.86	2564
1	0.91	0.79	0.85	2636
accuracy			0.85	5200
macro avg	0.86	0.86	0.85	5200
weighted avg	0.86	0.85	0.85	5200

Classification Report for Approach 1 & Approach 2

Approach 1



Approach 2



SUPPORT VECTOR MACHINE -LINEAR KERNEL

The results obtained on applying Support Vector Machine- Linear Kernel on our dataset are given below.

Approach 1:

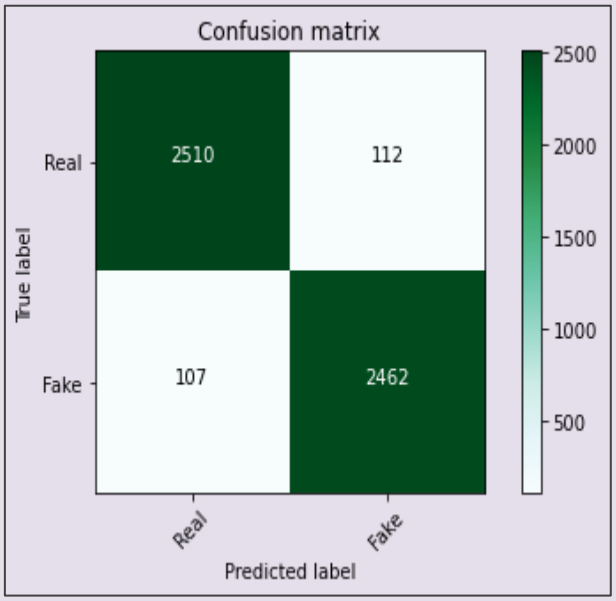
	precision	recall	f1-score	support
0	0.96	0.96	0.96	2622
1	0.96	0.96	0.96	2569
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

Approach 2:

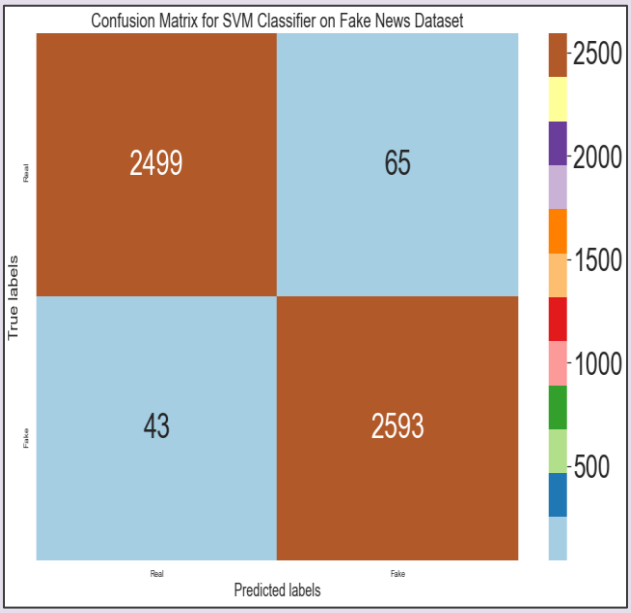
	precision	recall	f1-score	support
0	0.98	0.97	0.98	2564
1	0.98	0.98	0.98	2636
accuracy			0.98	5200
macro avg	0.98	0.98	0.98	5200
weighted avg	0.98	0.98	0.98	5200

Classification Report for Approach 1 & Approach 2

Approach 1



Approach 2

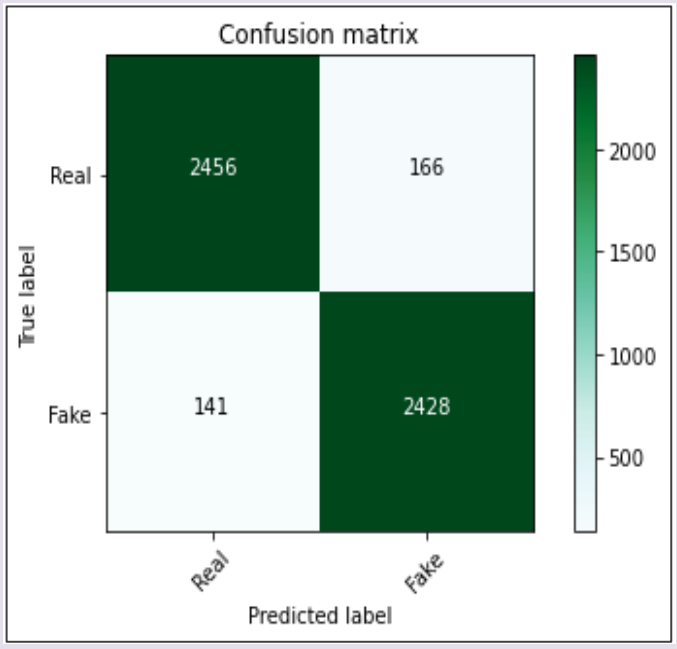
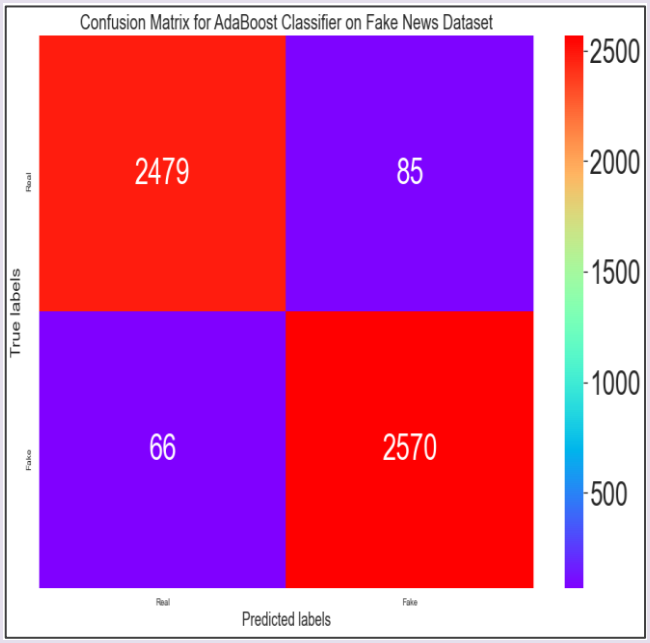


ADABOOST

The results obtained on applying AdaBoost on our dataset are given below.

```
Approach 1:
                precision    recall  f1-score   support

             0       0.95      0.94      0.94      2622
             1       0.94      0.95      0.94      2569

    accuracy                           0.94      5191
   macro avg       0.94      0.94      0.94      5191
weighted avg       0.94      0.94      0.94      5191

Approach 2:
                precision    recall  f1-score   support

             0       0.97      0.97      0.97      2564
             1       0.97      0.97      0.97      2636

    accuracy                           0.97      5200
   macro avg       0.97      0.97      0.97      5200
weighted avg       0.97      0.97      0.97      5200
```

Classification Report for Approach 1 & Approach 2

Approach 1	***Approach 2***
	

XGBOOST

The results obtained on using XGBoost on our dataset are given below.

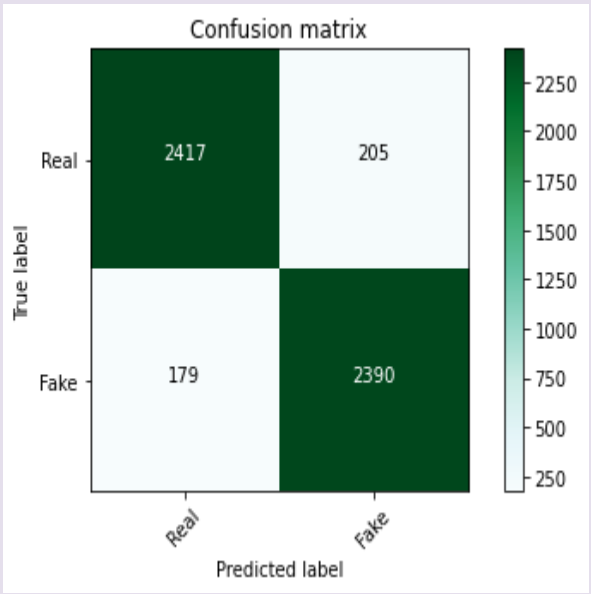
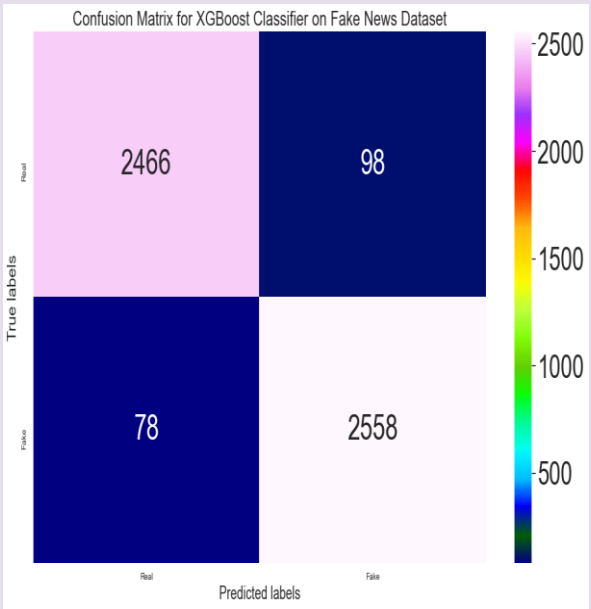
Approach 1:

	precision	recall	f1-score	support
0	0.93	0.92	0.93	2622
1	0.92	0.93	0.93	2569
accuracy			0.93	5191
macro avg	0.93	0.93	0.93	5191
weighted avg	0.93	0.93	0.93	5191

Approach 2:

	precision	recall	f1-score	support
0	0.97	0.96	0.97	2564
1	0.96	0.97	0.97	2636
accuracy			0.97	5200
macro avg	0.97	0.97	0.97	5200
weighted avg	0.97	0.97	0.97	5200

Classification Report for Approach 1 & Approach 2

Approach 1	***Approach 2***
	

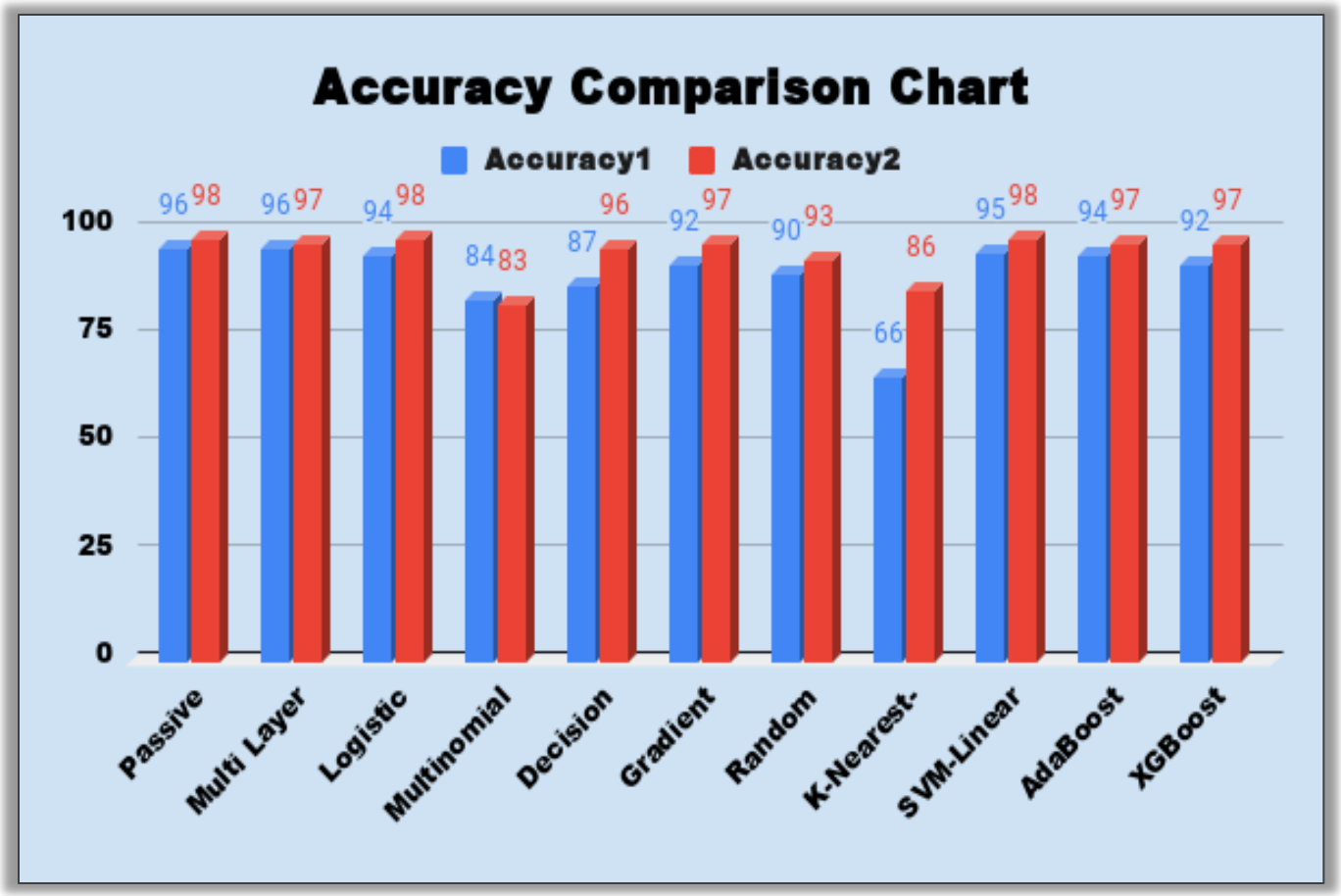
RESULTS AND ANALYSIS

In this section we will be visualizing the results and comparing and contrasting their performance metrics of the 11 models across the two notebooks.

1. ACCURACY COMPARISON ACROSS MODELS

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

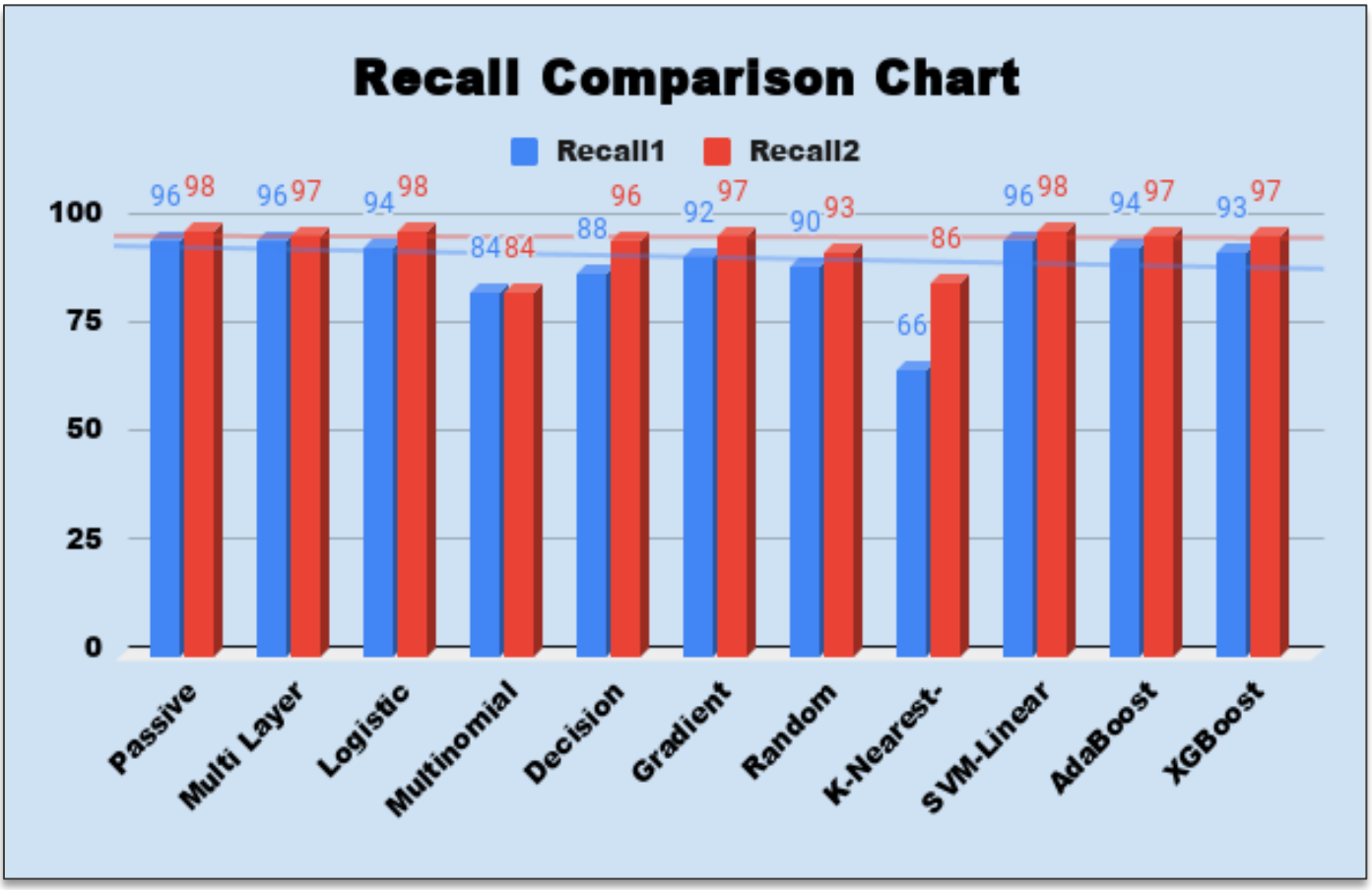
$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$



2. RECALL COMPARISON ACROSS MODELS

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class. In other words, **recall** (also known as Sensitivity) is the fraction of relevant instances that were retrieved. Precision is therefore based on relevance.

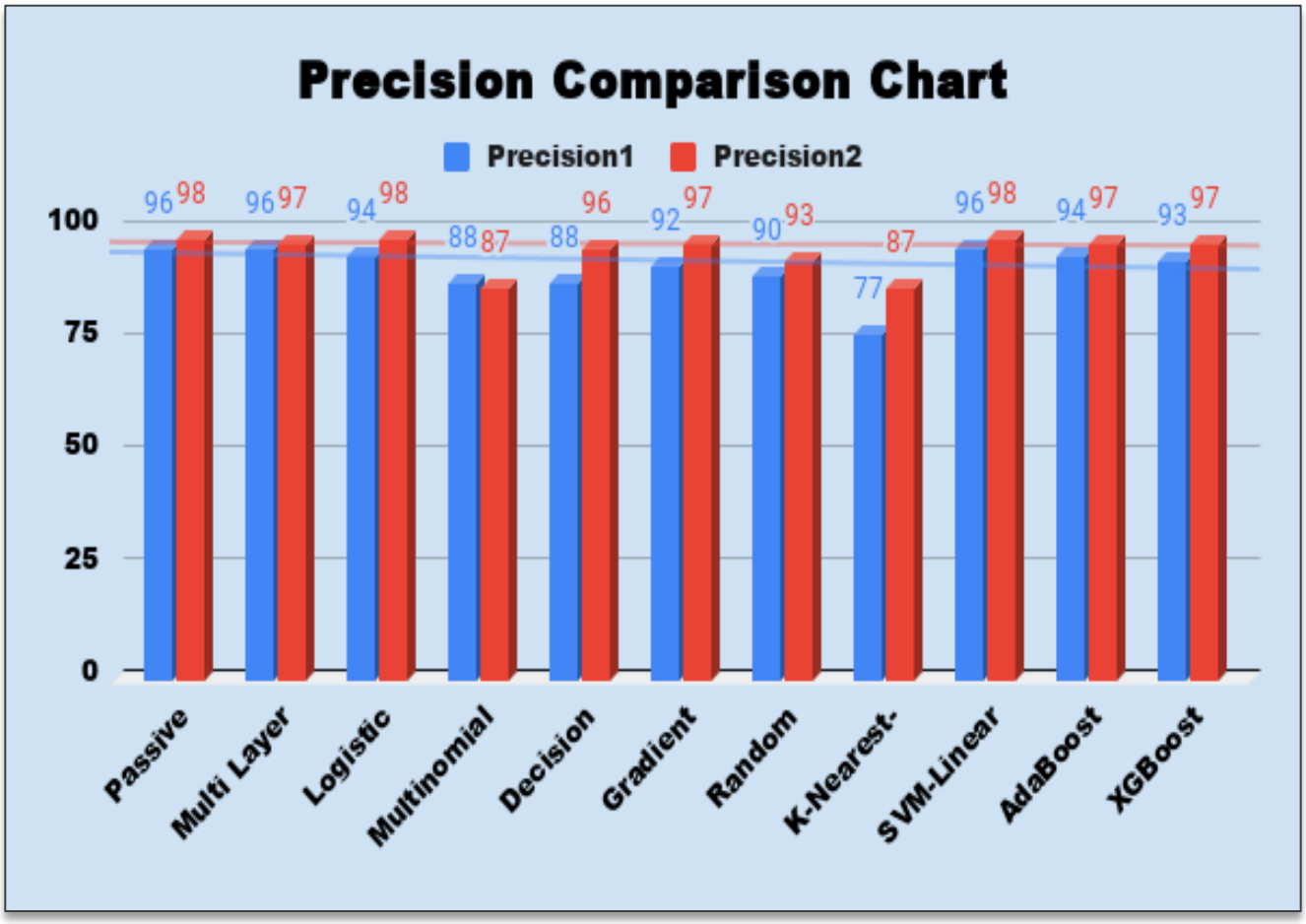
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$



3. PRECISION COMPARISON CHART

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

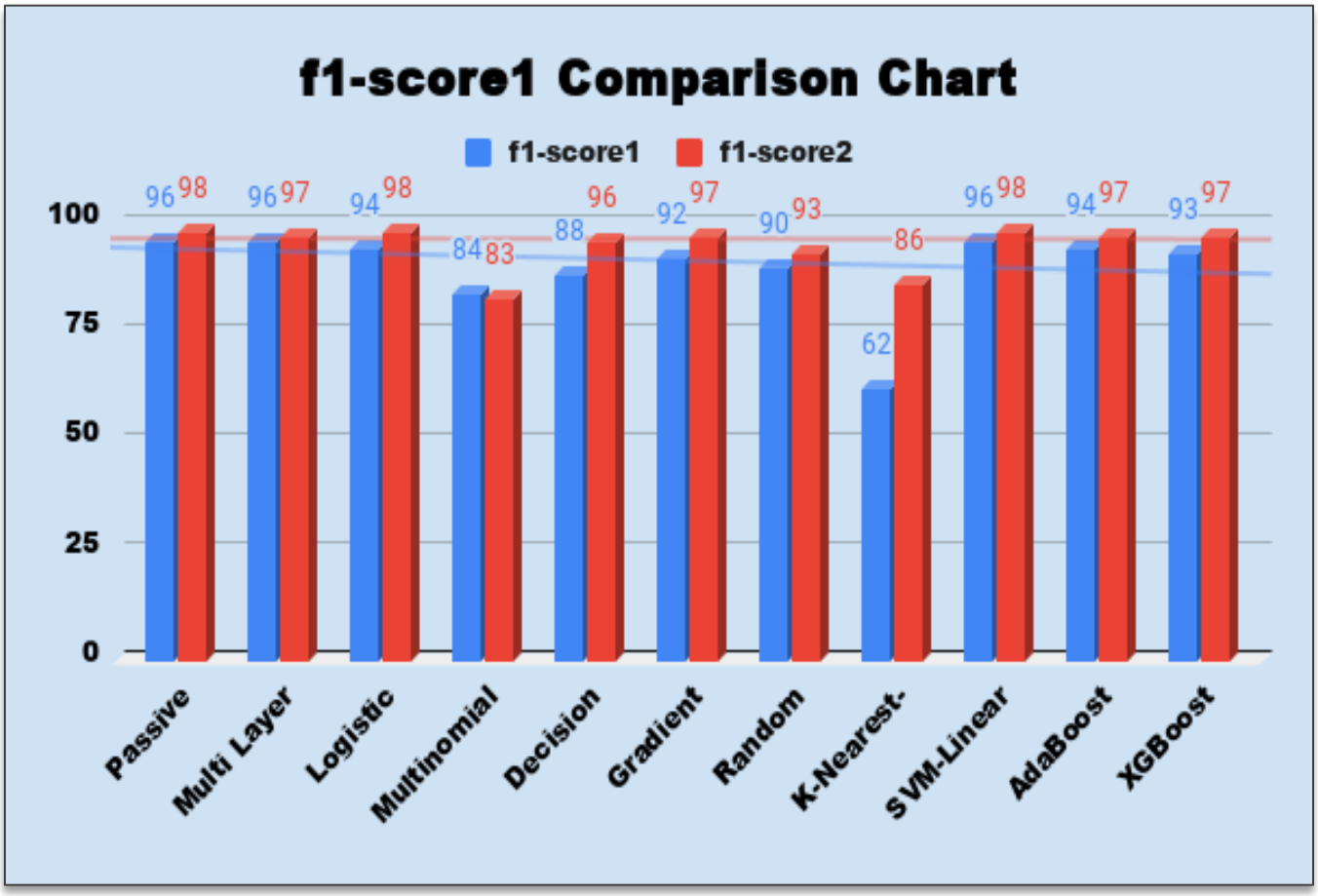
$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$



4. F1 SCORE COMPARISON CHART

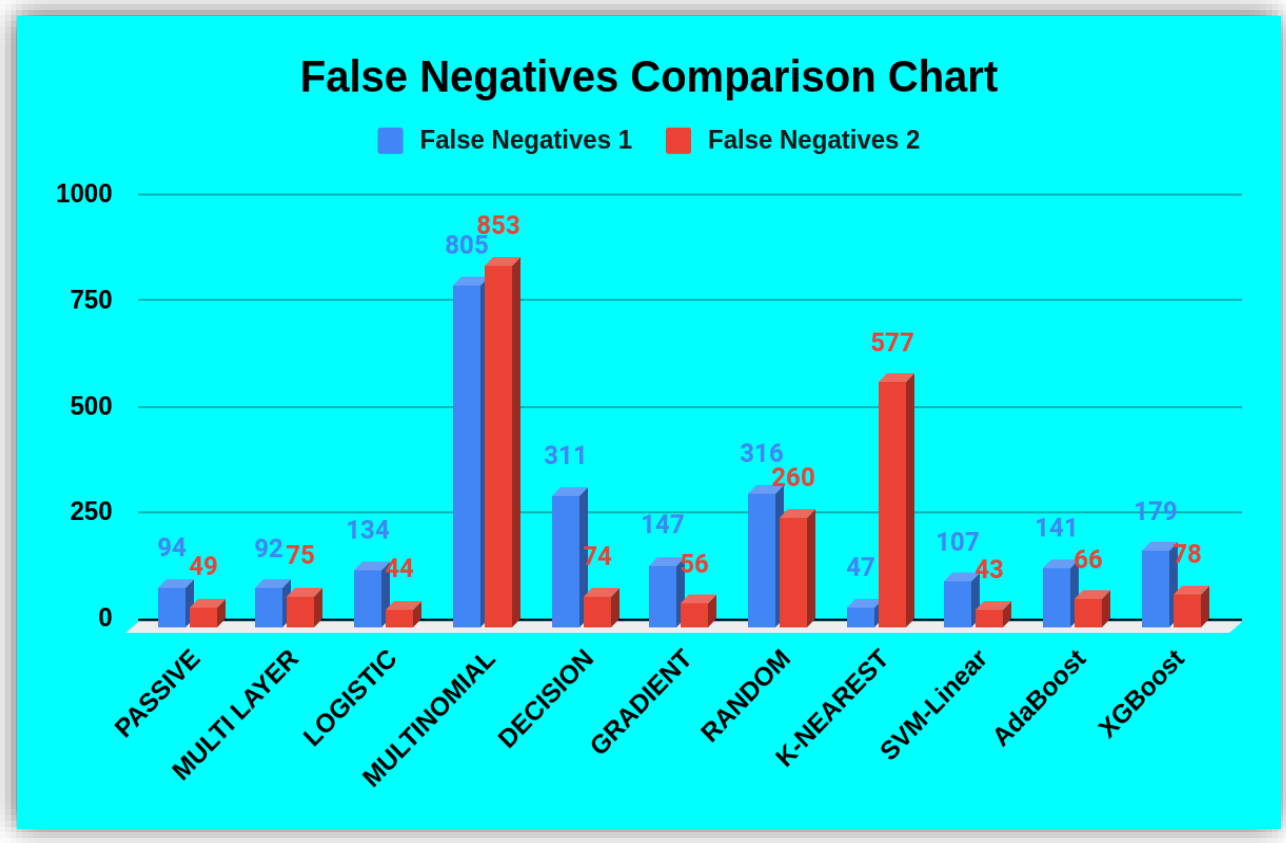
In statistical analysis of binary classification, the **F-score** or **F-measure** is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$



5. FALSE NEGATIVES COMPARISON CHART

It is really important to understand that if any false news is predicted as true, it could have catastrophic results. Often rumors can create huge misunderstandings and can also potentially can cause a million dollar bank to shut down as well. Hence we monitor and compare the false negatives of all models.



CONCLUSION

Classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. We have classified fake news articles using 11 machine learning models. This Fake News Detection aims to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using Natural Language Processing for text preprocessing and also, Feature Extraction tools like 'CountVectorizer' and 'TF-IDF Transformer' and used the feature set as an input to the models. The learning models were trained. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. A Fake News Classifier should essentially ensure at least the following measure:

1) High accuracy
2) The number of False Negatives must be minimum. The value of False Negative indicates how many actually Fake News has been classified/predicted as Real news by the Machine Learning model. Clearly, this situation is not desirable because the results of fake news classified as true news may be catastrophic.

We have made some concrete conclusions at the end of our experiments:

- 10 out of 11 models showed better accuracy, recall, precision and f1-score in the second approach. 9 out of 11 models showed lower number of false negatives in the second approach. This implies that processes like removal of stop words, lemmatization and inclusion of all attributes do significantly impact performance of a machine learning model of a fake news classifier.

- We conclude that Passive Aggressive Classifier, Logistic Regression, Gradient Boosting Classifier, and SVM models show the best performance with respect to accuracy, recall, precision, f1-score and false negative values. They exhibit relatively higher values of accuracy with relatively lower values of false negatives. Hence, these models are better choices for the sake of fake news classification.
- KNN scores an accuracy of 66% along with 47 false negatives as per the first approach. Despite increase in its accuracy in the second approach to 86%, it has very high number of false negative values which is clearly very undesirable. Hence KNN is not an apt model for fake news classification.
- Multinomial Naive Bayes, with relatively lower accuracies of 84% and 83% in the first and second approach respectively, have significantly high false negative values of 805 and 853. Hence Multinomial Naive Bayes is not an apt model for fake news classification.

Fake news detection has many open issues that require the attention of researchers. For instance, to reduce the spread of fake news, identifying critical elements involved in the spread of news is essential. Machine learning techniques can be employed to identify the critical sources involved in spreading fake news. Likewise, real-time fake news identification in videos can be another possible future direction.