Fake News Detection

Team Members:

- Aditya Chirania,181CO104
- Anusha P. Das, 181CO108

Introduction

Fake news can cause considerable misunderstandings in society hence it is imperative to differentiate between Fake and True News.

This project utilises Machine Learning algorithms to perform the classification/prediction of news as Fake or True.

We will depict how good data cleaning techniques can impact the performance of the fake news classifier in this project.

We used text-preprocessing techniques like removing stop words, lemmatization, tokenization, and vectorization before we feed the data to models.

Fed data to various models and compared their performance. These data cleaning techniques fall under Natural Language Processing (NLP).

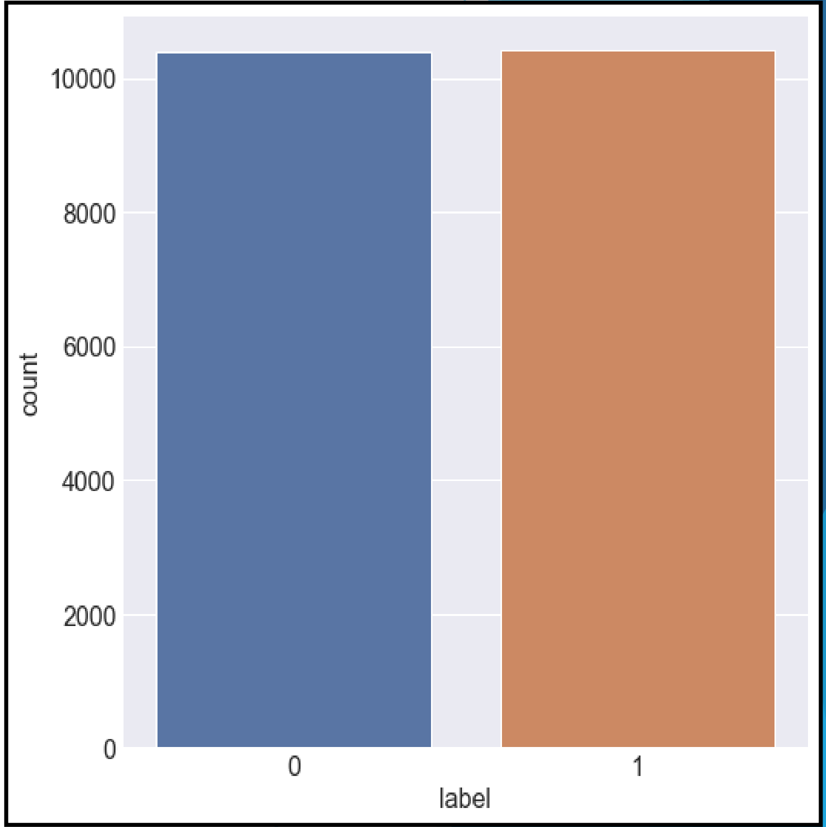
Examining the Dataset

The dataset, created by the University of Tennessee, USA is a collection of about 20800 news articles.

The attributes are:

1. *id*: unique id for a news article
2. *title*: the title of a news article
3. *author*: author of the news article
4. *text*: the text of the article; could be incomplete
5. *label*: a label that marks the article as potentially unreliable
   - 1: Fake News or Unreliable
   - 0: True News or reliable

The adjacent figure shows that the dataset is balanced with 10387 fake and 10413 true news.



Examining the dataset (continued)

The below code snippet shows that:

Total Words	Total Unique Words
6,83,32,444	742

```
In [100]: # Obtain the total words present in the dataset
          list_of_words = []
          for i in train.total:
              for j in i:
                  list_of_words.append(j)

In [101]: len(list_of_words)

Out[101]: 68332444

In [102]: # Obtain the total number of unique words
          total_words = len(list(set(list_of_words)))
          total_words

Out[102]: 742
```

We broadly divide the working of this project into 4 parts:

1. Data Pre-processing & Cleaning

2. Feature Extraction

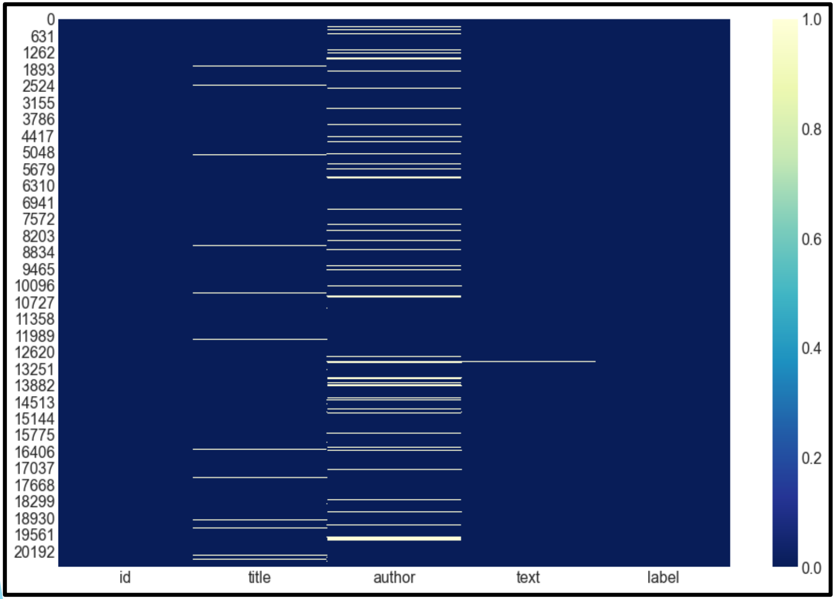
3. Applying various(11) Machine Learning models

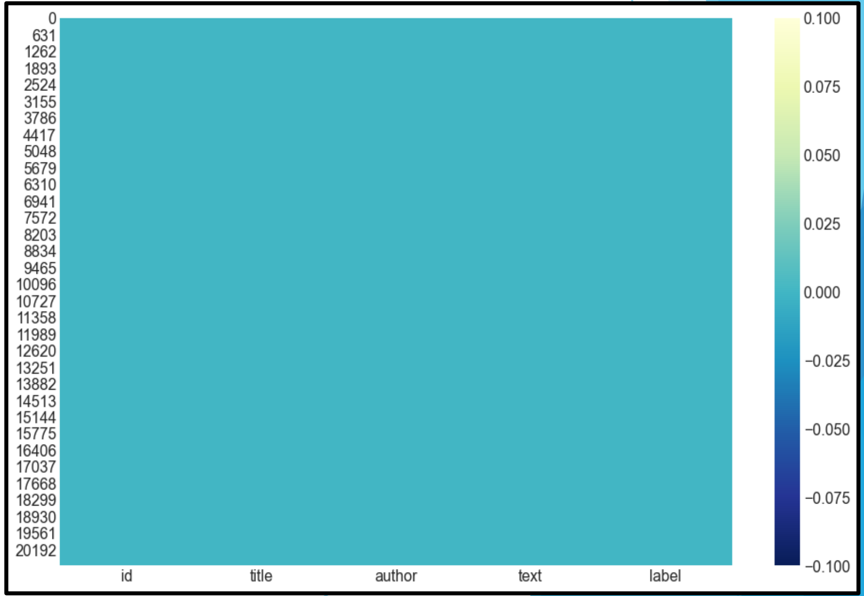
4. Analyzing & comparing the results of the 2 approaches and finally conclusion

Data Pre-Processing

1. *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 the input data prior to modelling the prediction task.





Check number of NULL values in the dataset

In [39]:
```
# how many null values in the dataset
print("Null values in train data:")
print(train.isnull().sum())
print('\n\n')

print("Null values in test data:")
print(test.isnull().sum())
```

```
Null values in train data:
id            0
title       558
author     1957
text         39
label         0
dtype: int64
```

Why is Missing Data Imputation required?

Data Pre-Processing (continued)

2. Merging the attributes into one column for preprocessing the text

In [45]:
```
test['total']=test['title']+' '+test['author']+test['text']
train['total']=train['title']+' '+train['author']+train['text']
```

In [46]:
```
train.head()
```

Out[46]:

	id	title	author	text	label	total
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	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	FLYNN: Hillary Clinton, Big Woman on Campus - ...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	Why the Truth Might Get You Fired Consortiumne...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1	15 Civilians Killed In Single US Airstrike Hav...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	Iranian woman jailed for fictional unpublished...

Data Pre-Processing (continued)

3. Using a Regex to remove special characters

1. Regex

In [48]:
```
#Remove punctuations from the String
sample = "!</> NLP is $$ </>^sh!!!o%%rt &&%$fo@@@r^^^&&!& </>*Natural@# Language&&\ Pro@@@##%^^&cessing!@# %%$"
```

In [49]:
```
# what is gonna get selected we r gonna replace that with the empty string(2nd parameter)
sample = re.sub(r'[^\w\s]','',sample)
```

In [50]:
```
print(sample)
```

```
NLP is  short for Natural Language Processing
```

Data Pre-Processing (Continued)

4. Tokenization of Data

```
In [9]: print("The NLTK tokeniser has tokenised \"Computers are not as great at understanding words as they are numbers.\" inte
        print(nltk.word_tokenize("Computers are not as great at understanding words as they are numbers."))

The NLTK tokeniser has tokenised "Computers are not as great at understanding words as they are numbers." into a list
of tokens

['Computers', 'are', 'not', 'as', 'great', 'at', 'understanding', 'words', 'as', 'they', 'are', 'numbers', '.']
```

Data Pre-Processing (Continued)

5. Removal of stop-words

Before removing stop-words	After removing stop-words
Does this thing really work? Lets see.	['thing', 'really', 'work', 'lets', 'see']

The list of stopwords available in the NLTK library are:

```
In [59]: stop=stopwords.words("english")
         print(stop)

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your',
'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "i
t's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
t', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'havi
ng', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'o
f', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'abov
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'onc
e', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'so
me', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just',
'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'could
n', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn',
"isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Data Pre-Processing (Continued)

6. Lemmatization

Before Lemmatization	After Lemmatization
kites	kite
babies	baby
languages	language
cities	city
mice	mouse

Data Pre-Processing (Continued)

7. Count Vectorization

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]]

8. TF-IDF Transformation

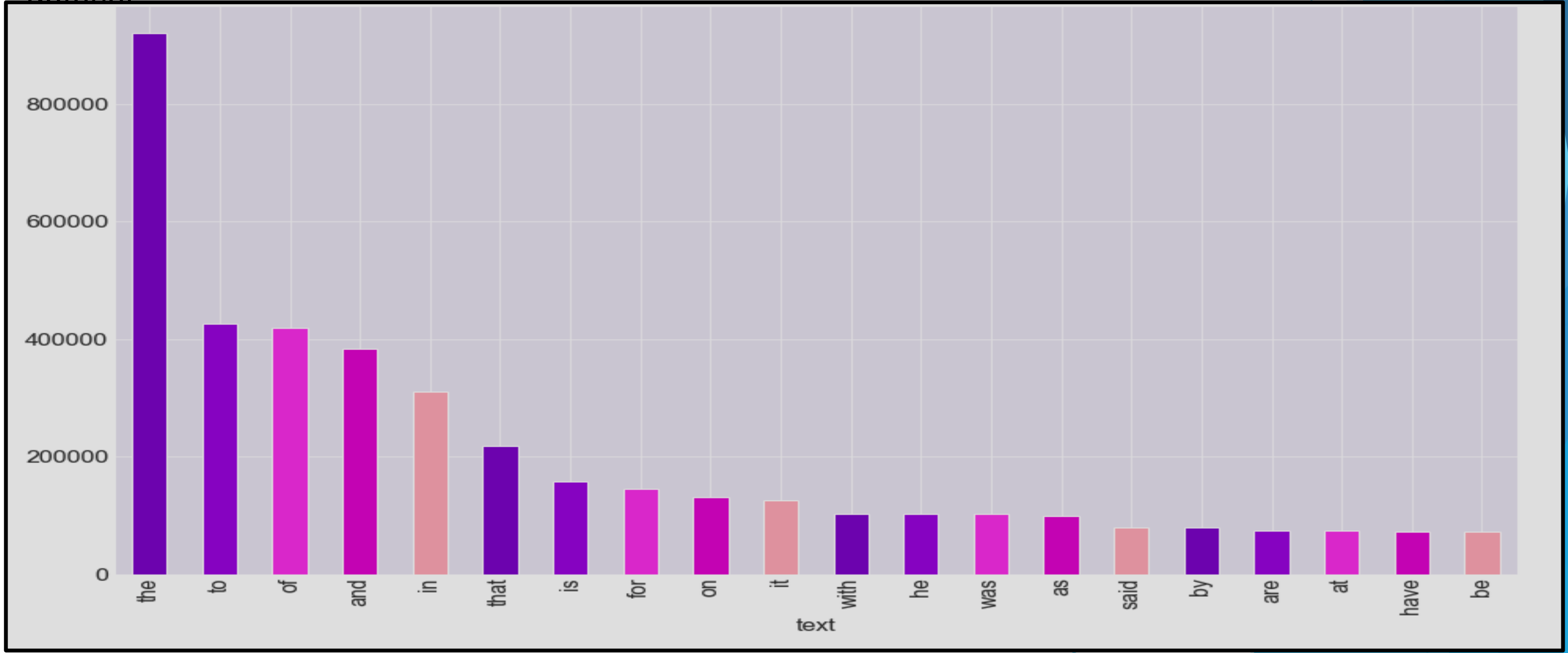
- *Term Frequency (TF)*
- *Inverse Document Frequency (IDF)*
- *TF-IDF Value = tf * idf*

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

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

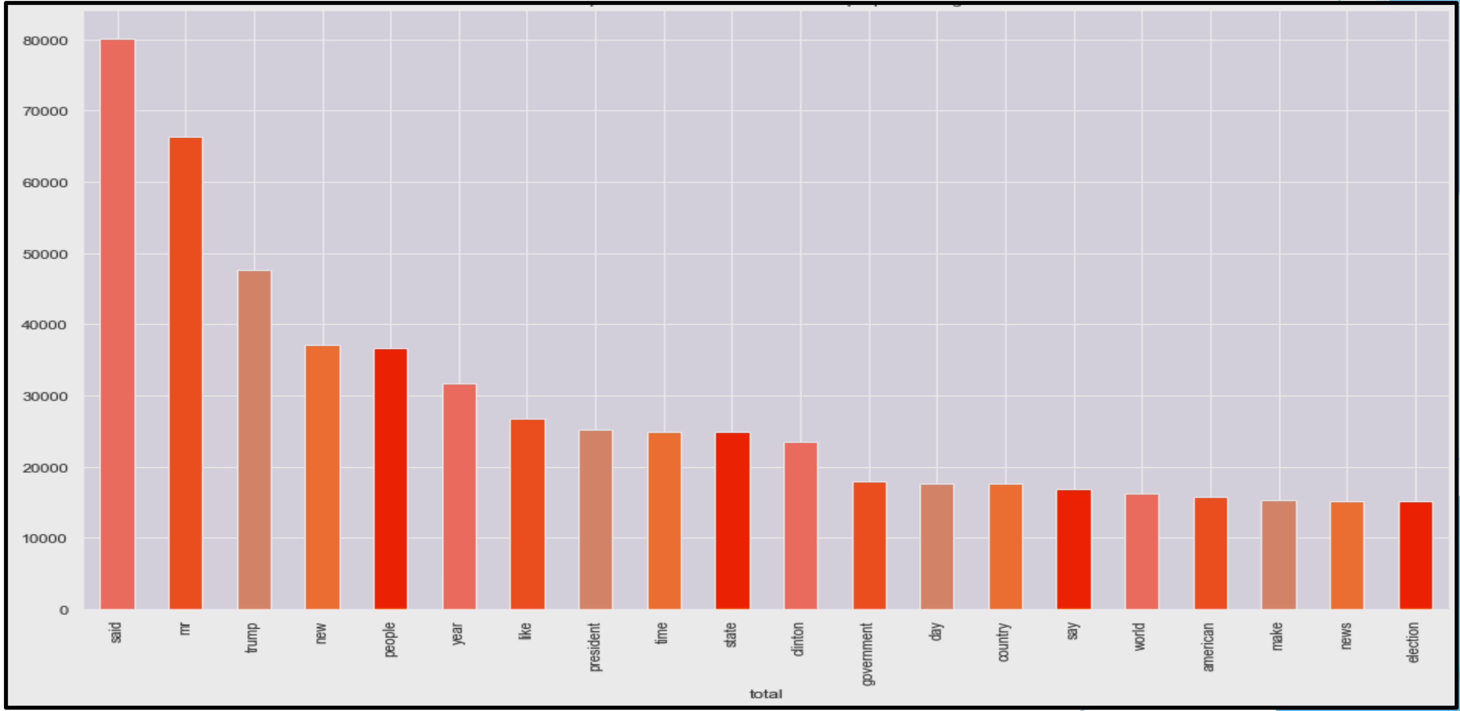
Top 20 unigrams before stop words removed

Stop words like “the”, ”to”, ”of”, ”and”, etc. are amongst the most frequent words present in the dataset

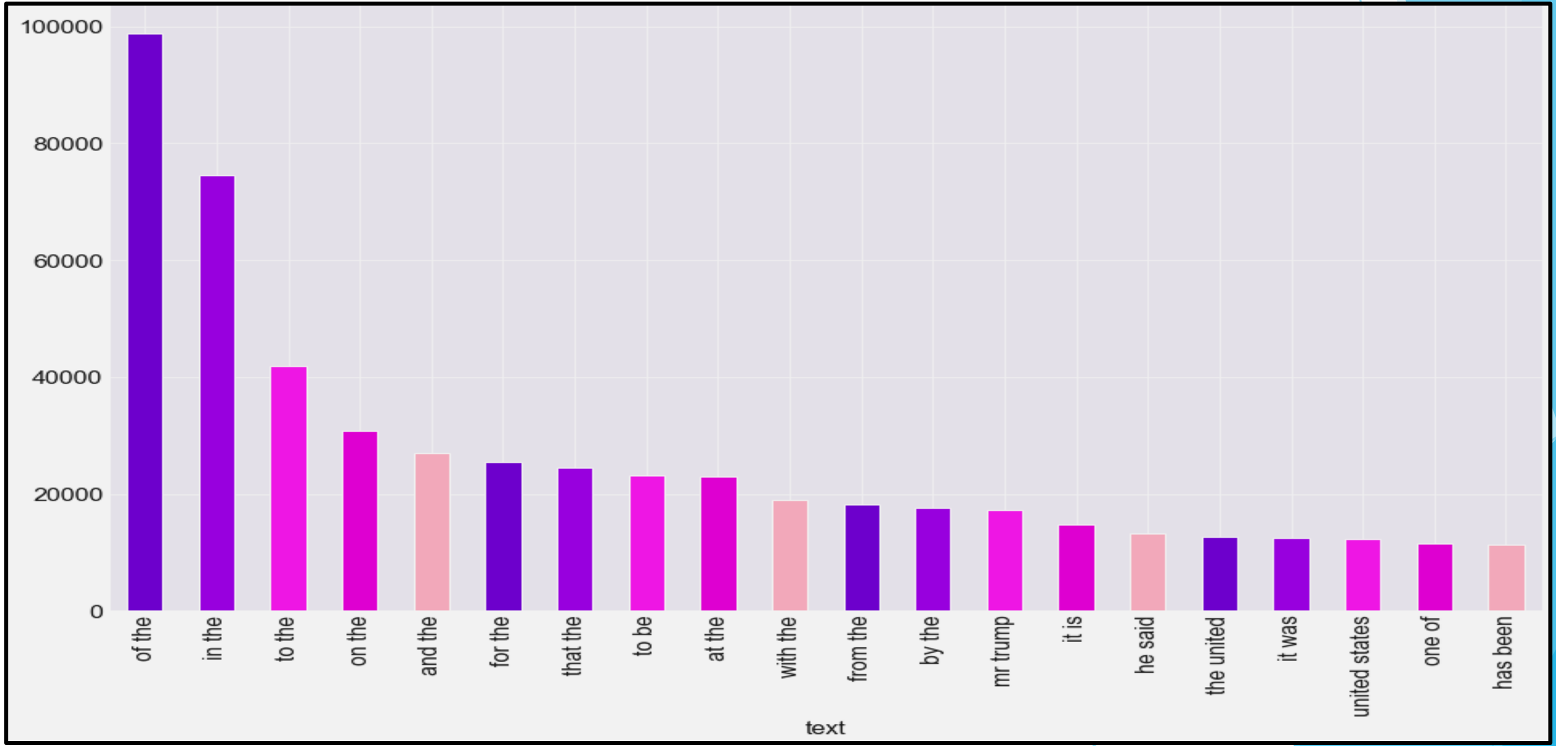


Top 20 Unigrams after stop words removal

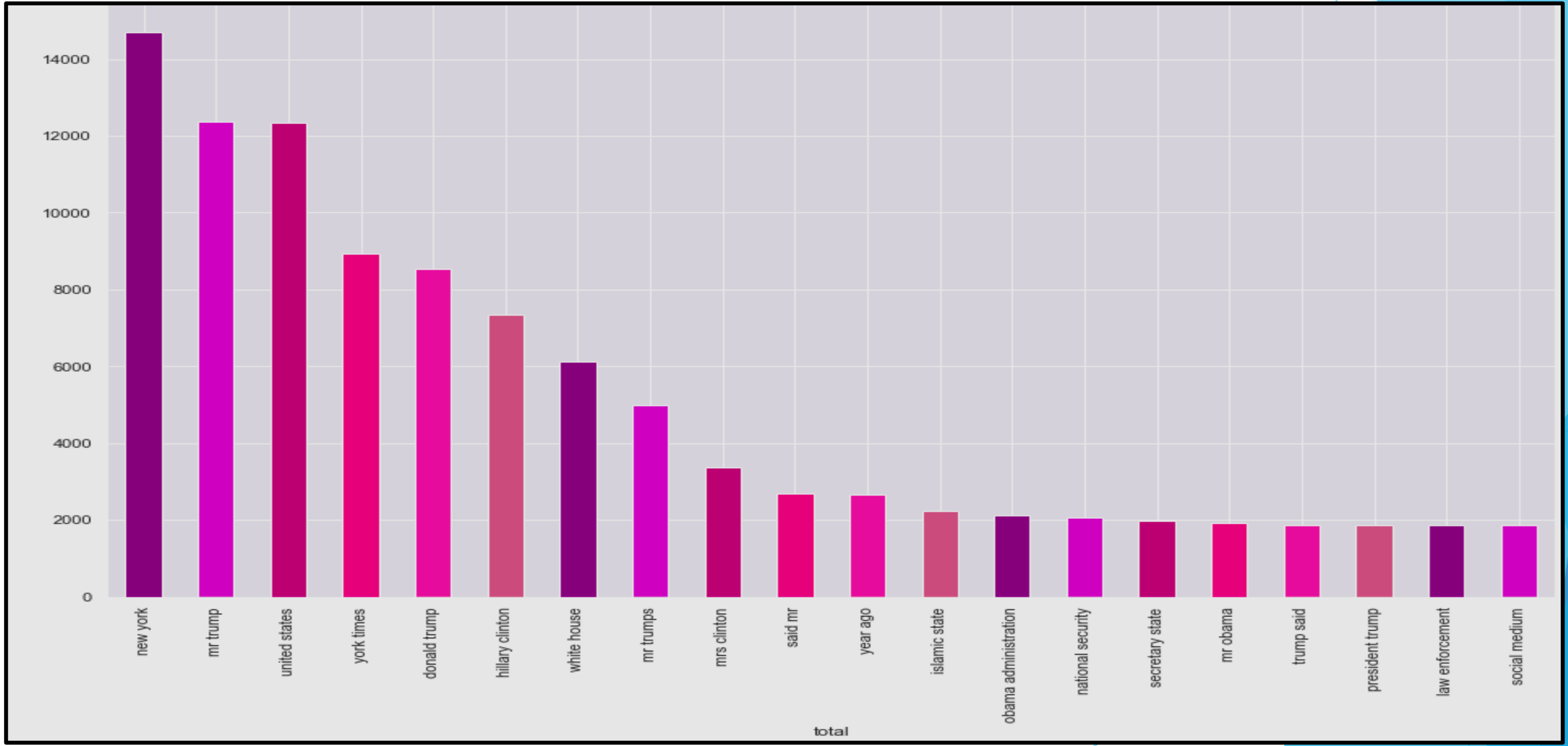
Words like “said”, “mr”, “trump”, “new”, “people”,”year” etc are most frequent

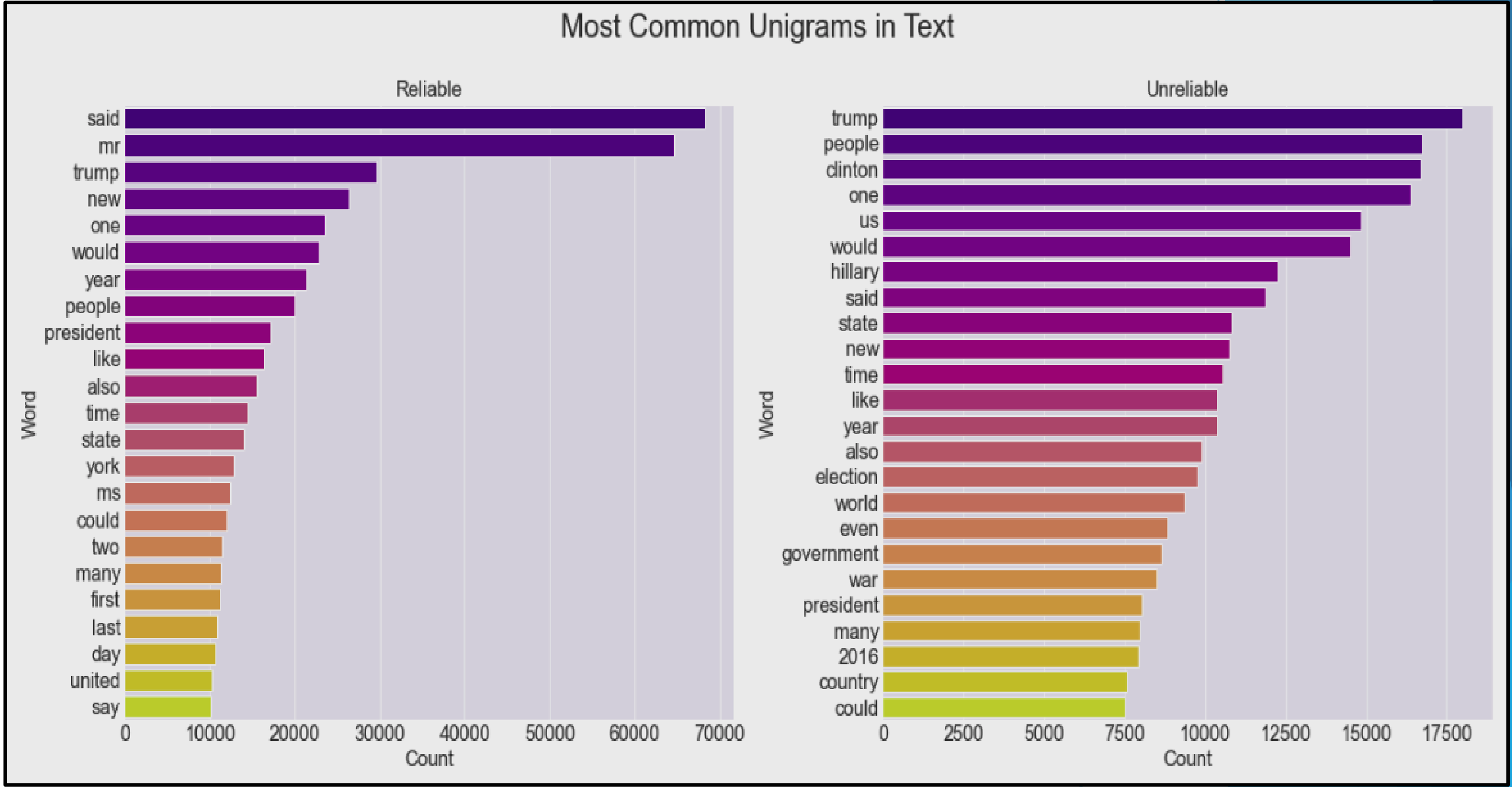


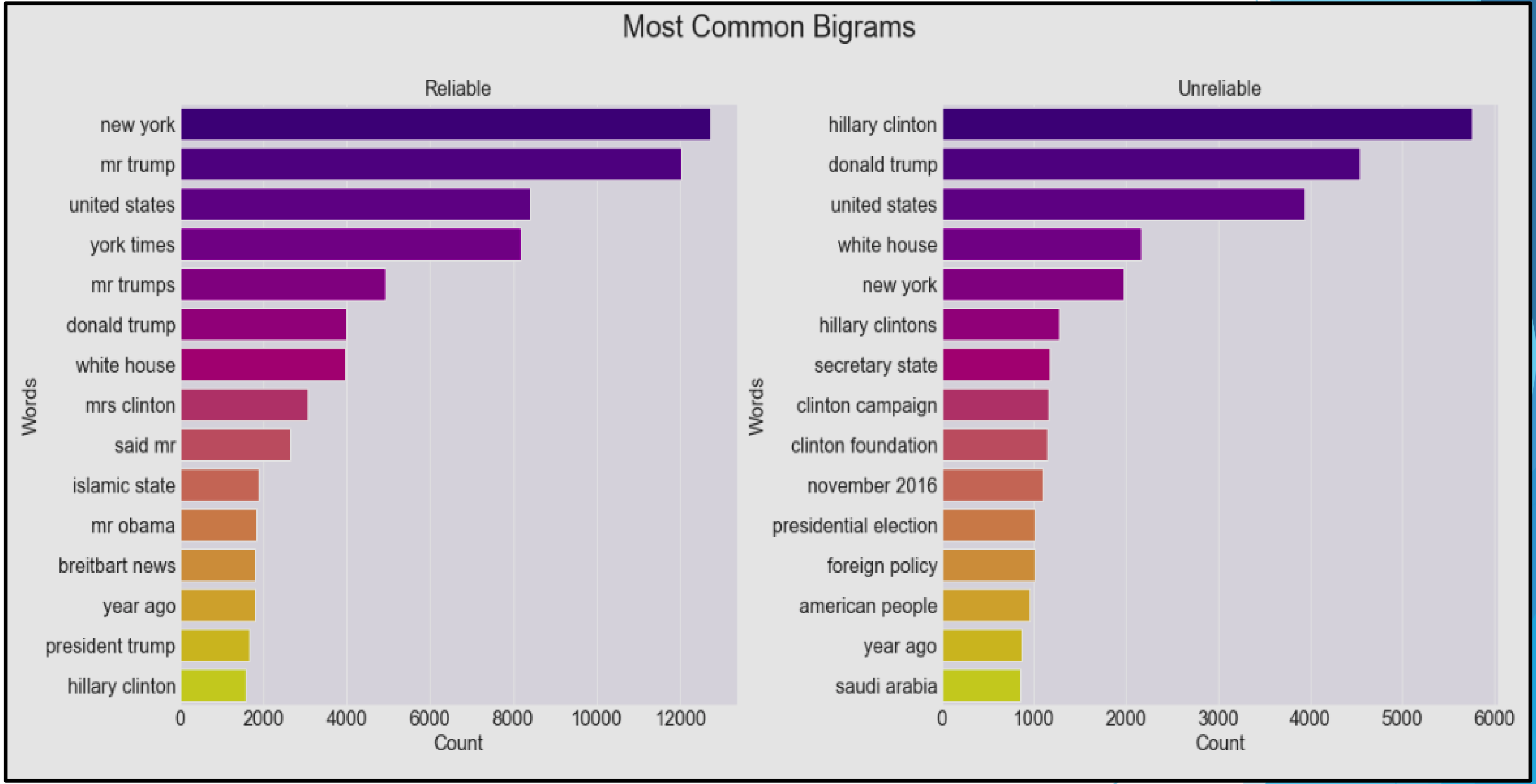
Top 20 bigrams before removing stop words



Top 20 bigrams after removing stop words




Most Common Unigrams in Text
Reliable
said
mr
trump
new
one
would
year
people
president
like
also
time
state
york
ms
could
two
many
first
last
day
united
say
Word
0
10000
20000
30000
40000
50000
60000
70000
Count
Unreliable
trump
people
clinton
one
us
would
hillary
said
state
new
time
like
year
also
election
world
even
government
war
president
many
2016
country
could
Word
0
2500
5000
7500
10000
12500
15000
17500
Count



Most Common Bigrams
Reliable
Unreliable
Words
Count
new york
mr trump
united states
york times
mr trumps
donald trump
white house
mrs clinton
said mr
islamic state
mr obama
breitbart news
year ago
president trump
hillary clinton
0
2000
4000
6000
8000
10000
12000
hillary clinton
donald trump
united states
white house
new york
hillary clintons
secretary state
clinton campaign
clinton foundation
november 2016
presidential election
foreign policy
american people
year ago
saudi arabia
0
1000
2000
3000
4000
5000
6000

Models Applied

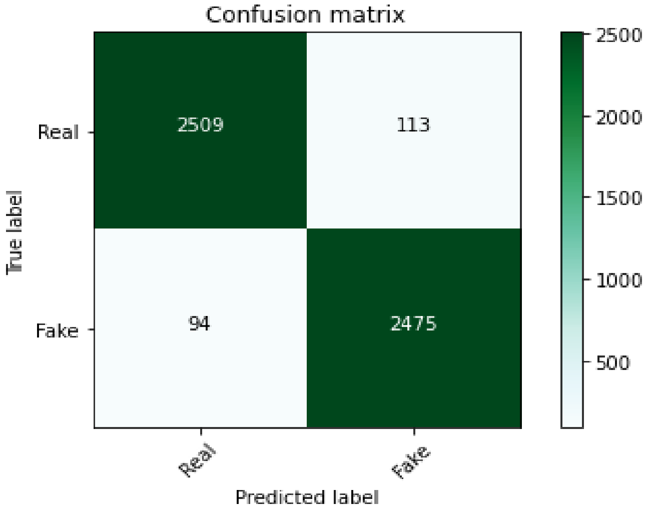
We have applied models with 2 approaches to data cleaning:

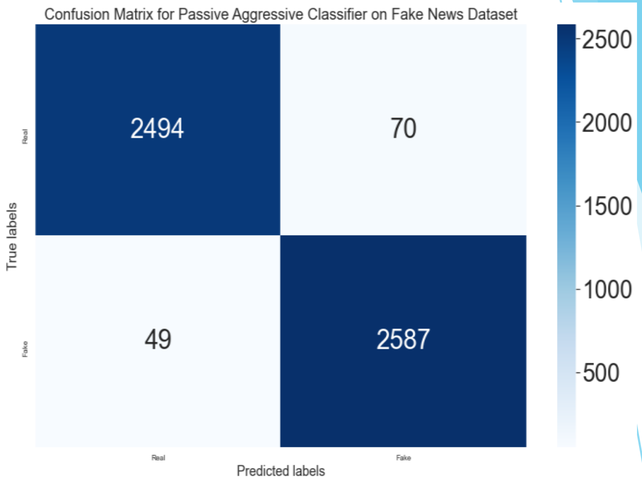
Approach 1: In the first approach, we have selected only one feature i.e the news text and have directly done TF-IDF vectorization after removing punctuation marks and eliminating rows with null values from the text.

Approach 2: In the second approach, we have followed the steps of removal of stop words, replacing null values with spaces, lemmatization, and combined all attributes including “author” , “text” and “title” into one column.

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.

1. Passive Aggressive Classifier:





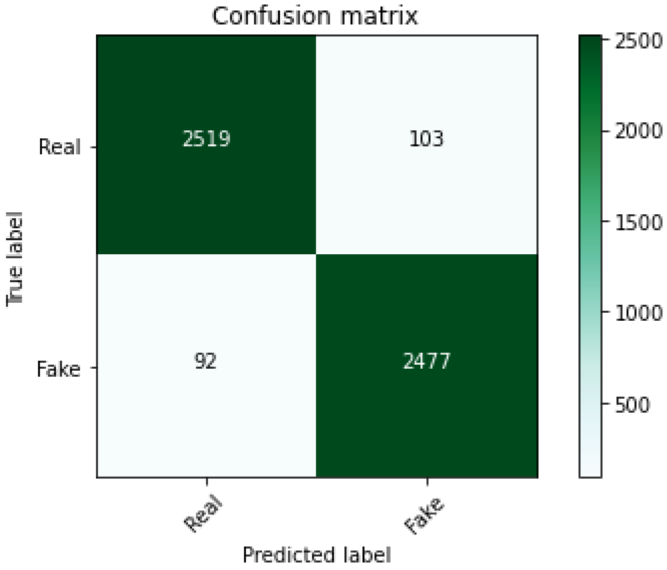
Approach 1:

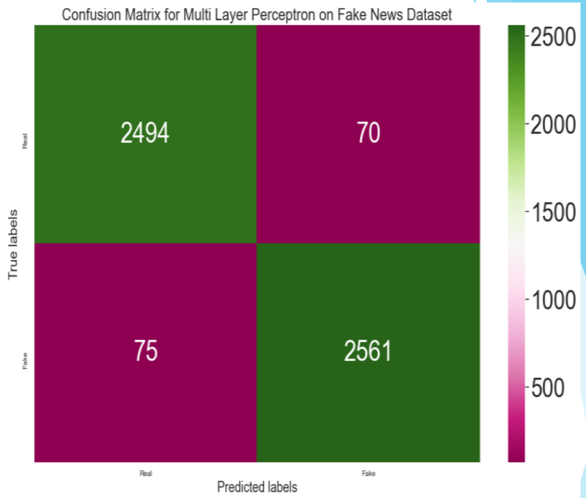
	precision	recall	f1-score	support
0	0.96	0.97	0.97	2630
1	0.96	0.96	0.96	2561
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

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

2. Multi Layer Perceptron





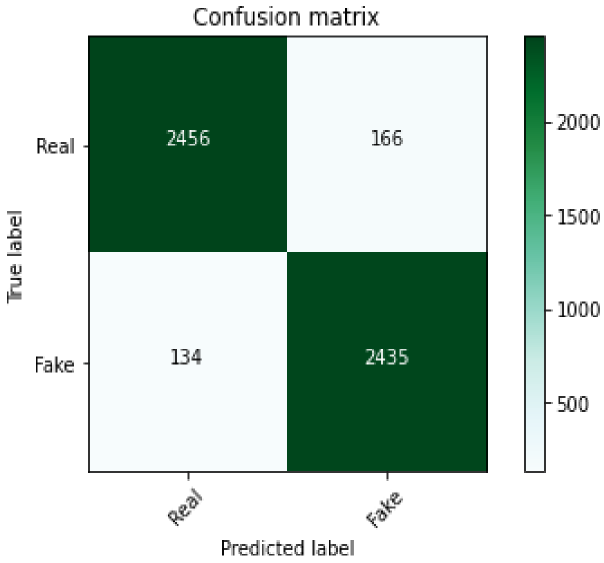
Approach 1:

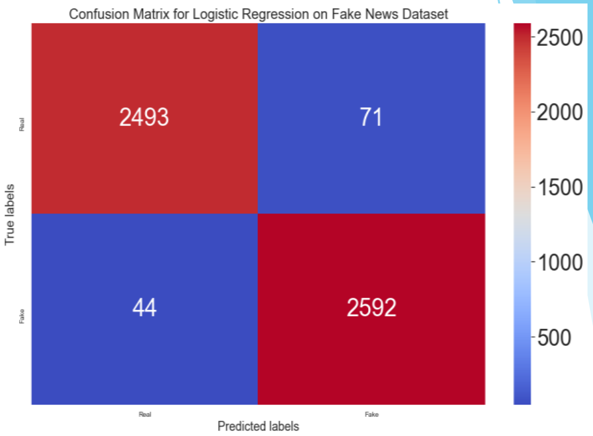
	precision	recall	f1-score	support
0	0.96	0.96	0.96	2622
1	0.96	0.96	0.96	2569
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

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

3. Logistic Regression





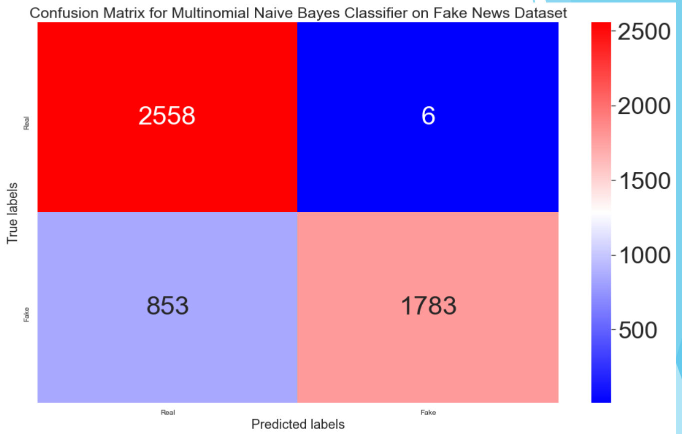
Approach 1:

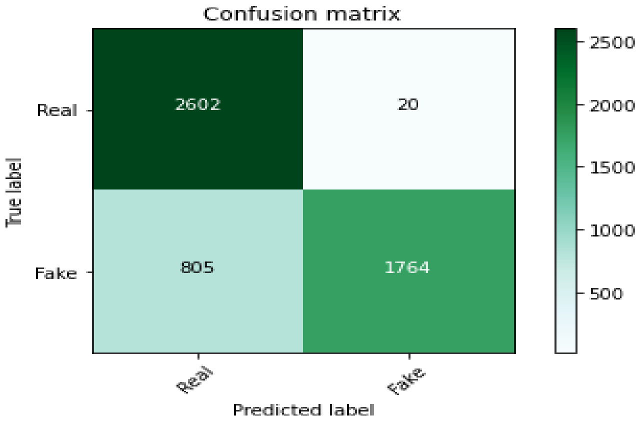
	precision	recall	f1-score	support
0	0.95	0.94	0.94	2622
1	0.94	0.95	0.94	2569
accuracy			0.94	5191
macro avg	0.94	0.94	0.94	5191
weighted avg	0.94	0.94	0.94	5191

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

4. MultinomialNB





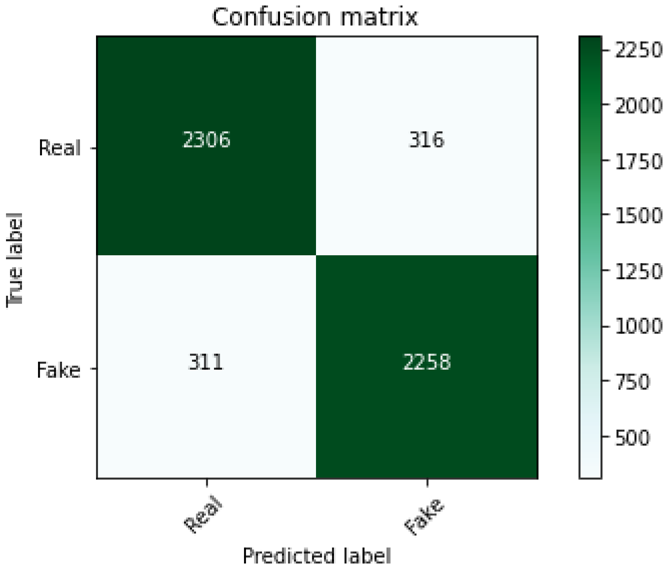
Approach 1:

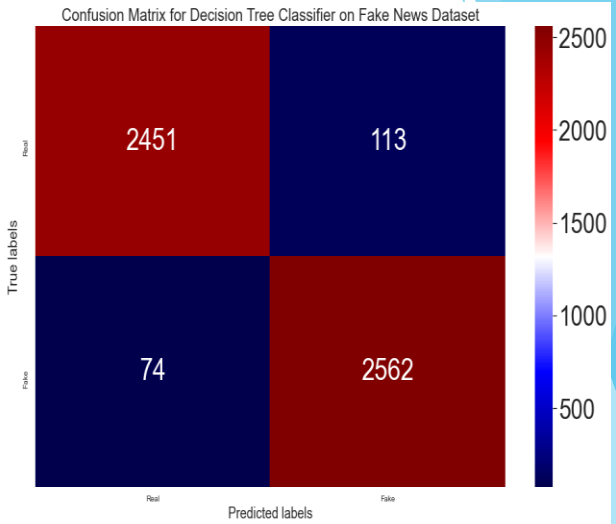
	precision	recall	f1-score	support
0	0.76	0.99	0.86	2622
1	0.99	0.69	0.81	2569
accuracy			0.84	5191
macro avg	0.88	0.84	0.84	5191
weighted avg	0.88	0.84	0.84	5191

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

5. Decision Tree





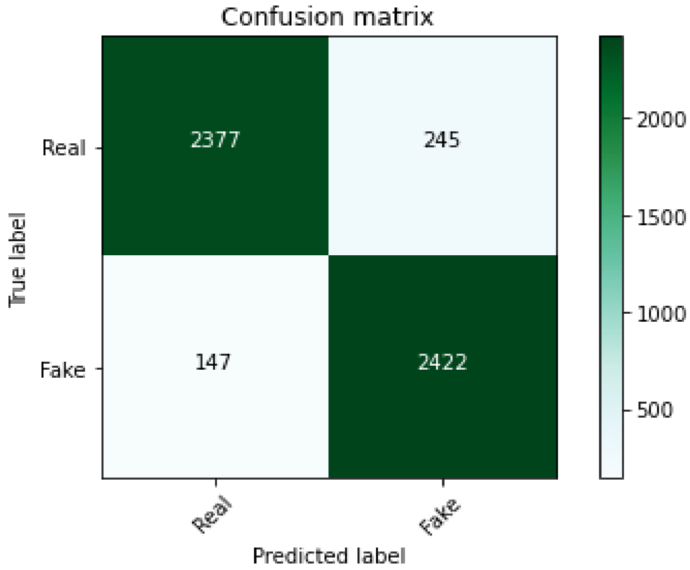
Approach 1:

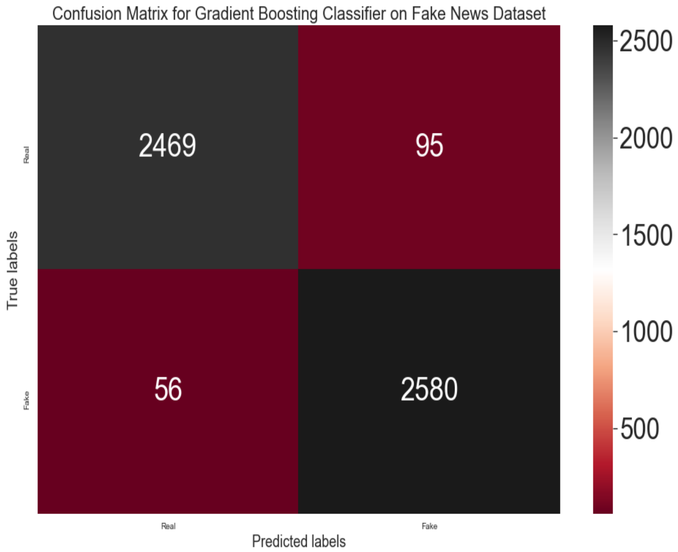
	precision	recall	f1-score	support
0	0.88	0.88	0.88	2622
1	0.88	0.88	0.88	2569
accuracy			0.88	5191
macro avg	0.88	0.88	0.88	5191
weighted avg	0.88	0.88	0.88	5191

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

6. GradientBoostingClassifier





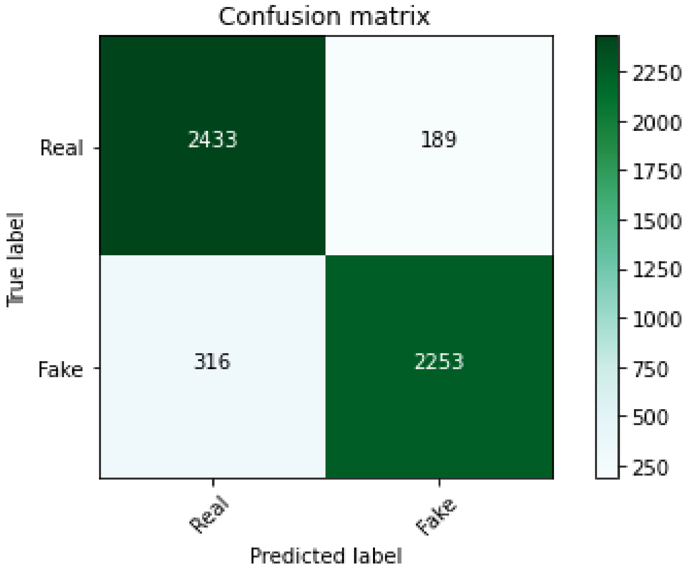
Approach 1:

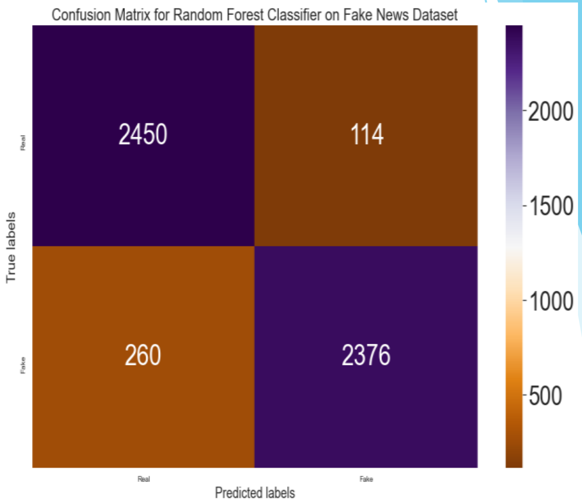
	precision	recall	f1-score	support
0	0.94	0.91	0.92	2622
1	0.91	0.94	0.93	2569
accuracy			0.92	5191
macro avg	0.92	0.92	0.92	5191
weighted avg	0.93	0.92	0.92	5191

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

7. Random Forest Classifier





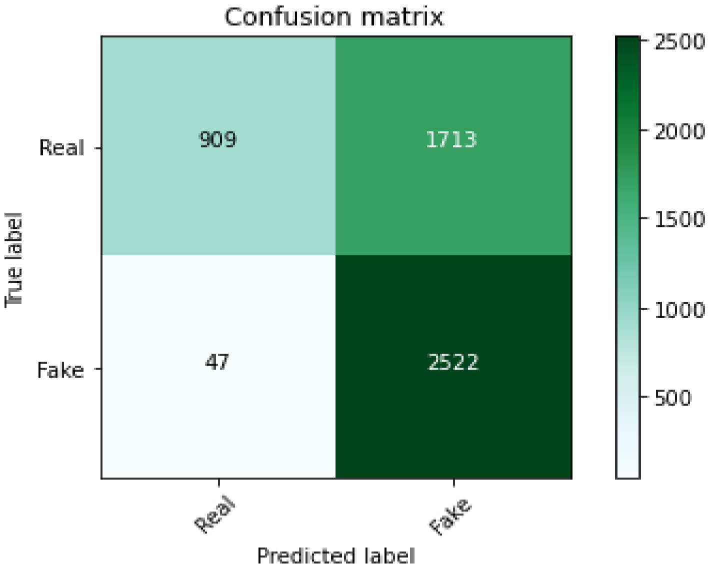
Approach 1:

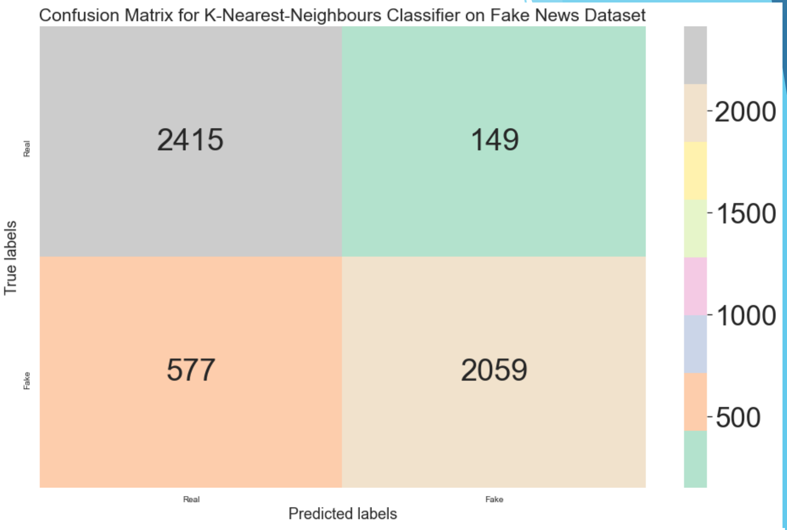
	precision	recall	f1-score	support
0	0.89	0.93	0.91	2622
1	0.92	0.88	0.90	2569
accuracy			0.90	5191
macro avg	0.90	0.90	0.90	5191
weighted avg	0.90	0.90	0.90	5191

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

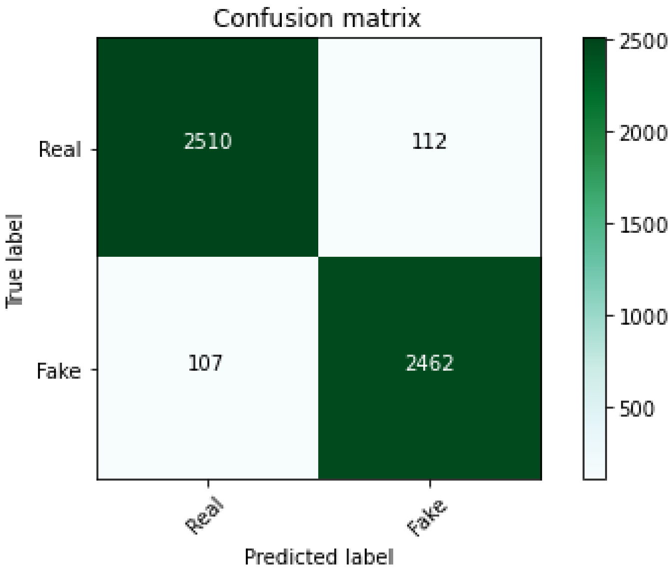
8. KNN

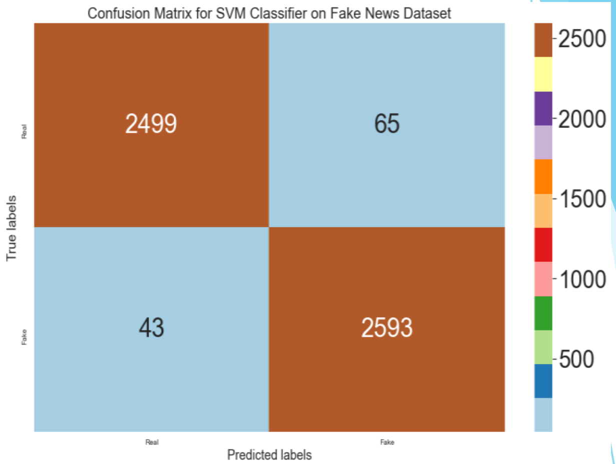





```
Approach 1:
              precision    recall  f1-score   support

           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
```

9. SVM-Linear Kernel





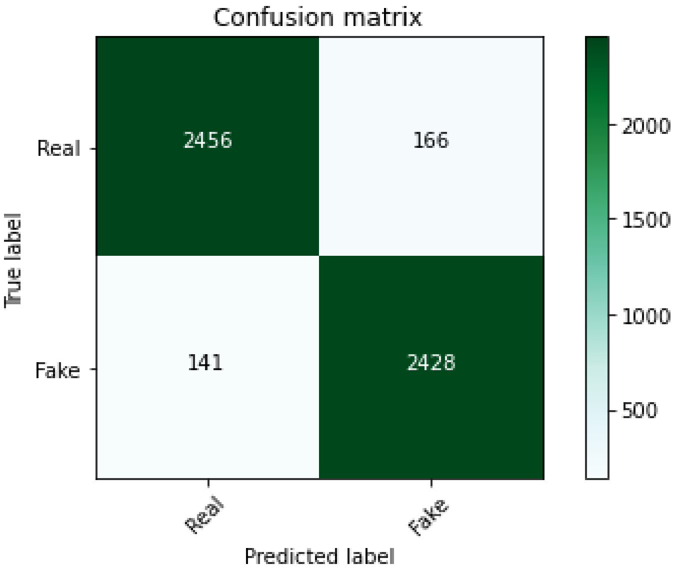
Approach 1:

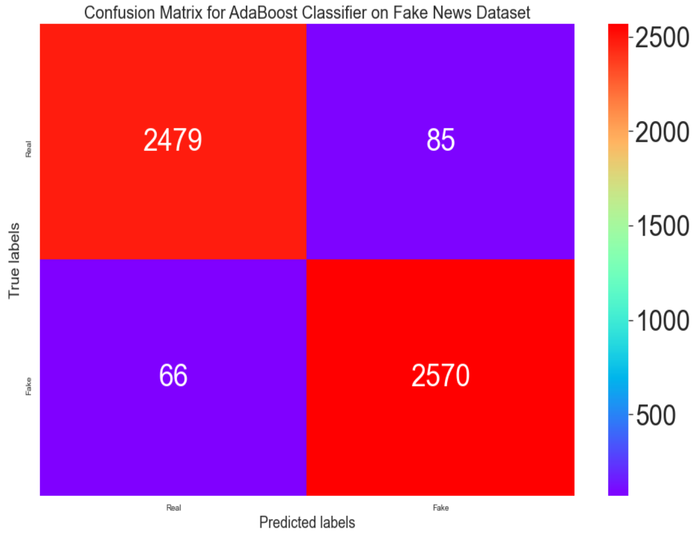
	precision	recall	f1-score	support
0	0.96	0.96	0.96	2622
1	0.96	0.96	0.96	2569
accuracy			0.96	5191
macro avg	0.96	0.96	0.96	5191
weighted avg	0.96	0.96	0.96	5191

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

10. AdaBoost





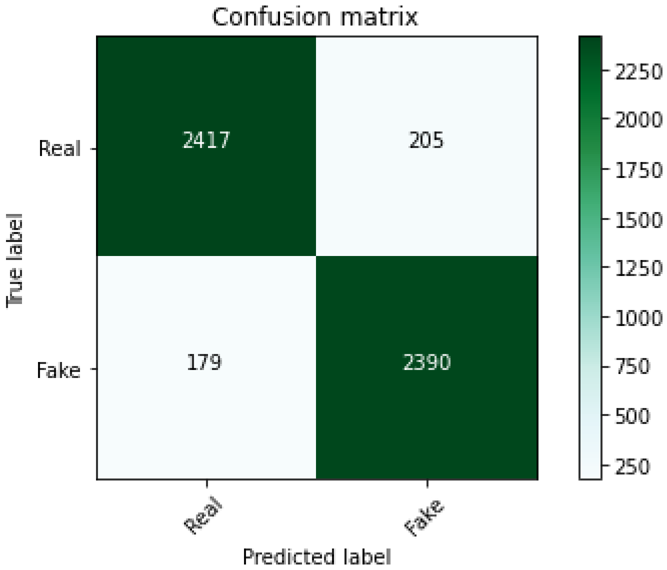
Approach 1:

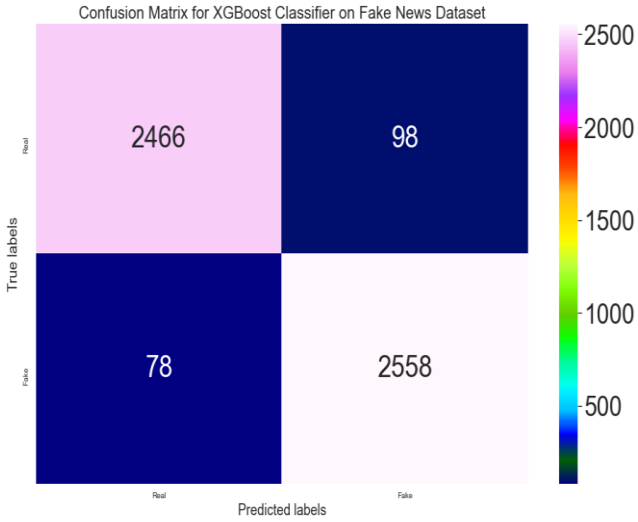
	precision	recall	f1-score	support
0	0.95	0.94	0.94	2622
1	0.94	0.95	0.94	2569
accuracy			0.94	5191
macro avg	0.94	0.94	0.94	5191
weighted avg	0.94	0.94	0.94	5191

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

11. XGBoost



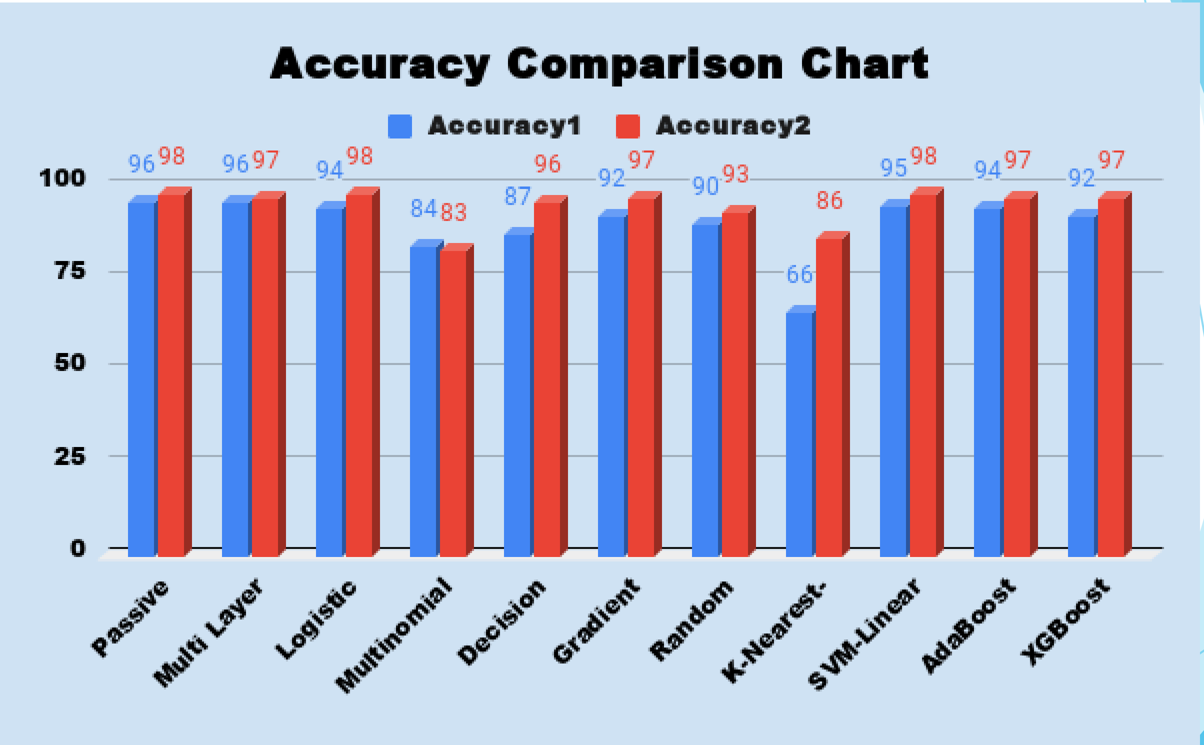


Approach 1:

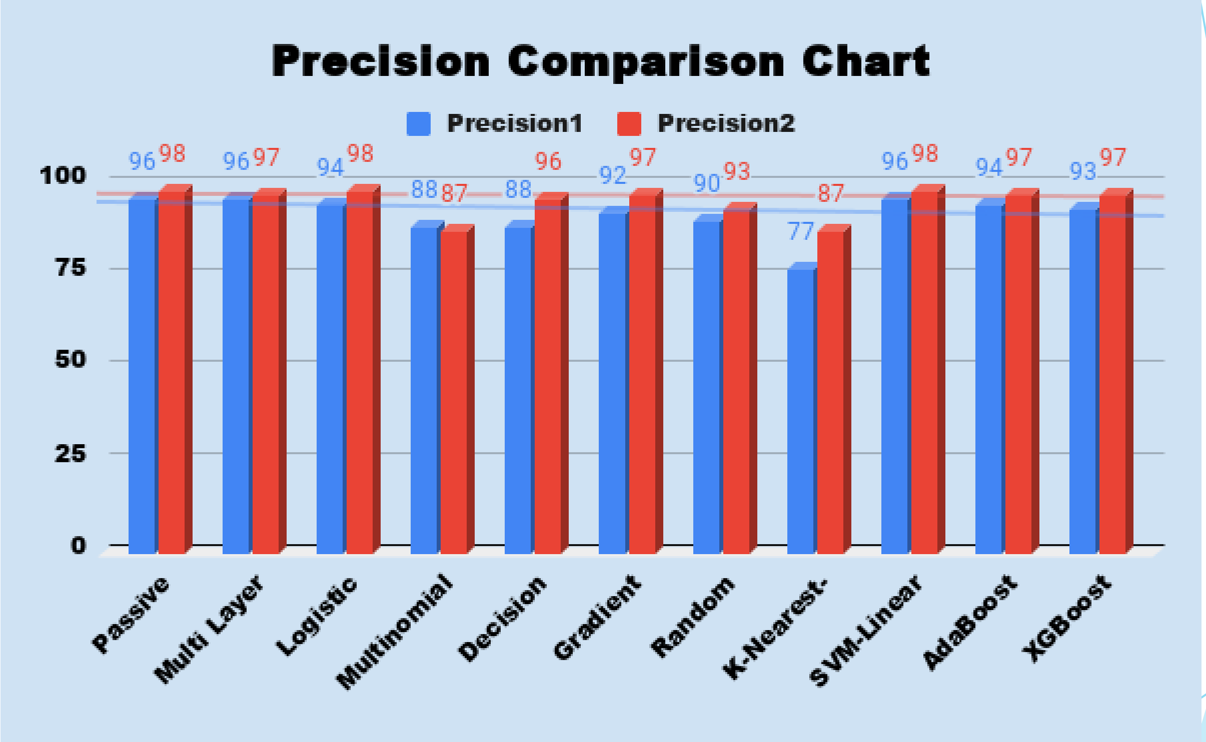
	precision	recall	f1-score	support
0	0.93	0.92	0.93	2622
1	0.92	0.93	0.93	2569
accuracy			0.93	5191
macro avg	0.93	0.93	0.93	5191
weighted avg	0.93	0.93	0.93	5191

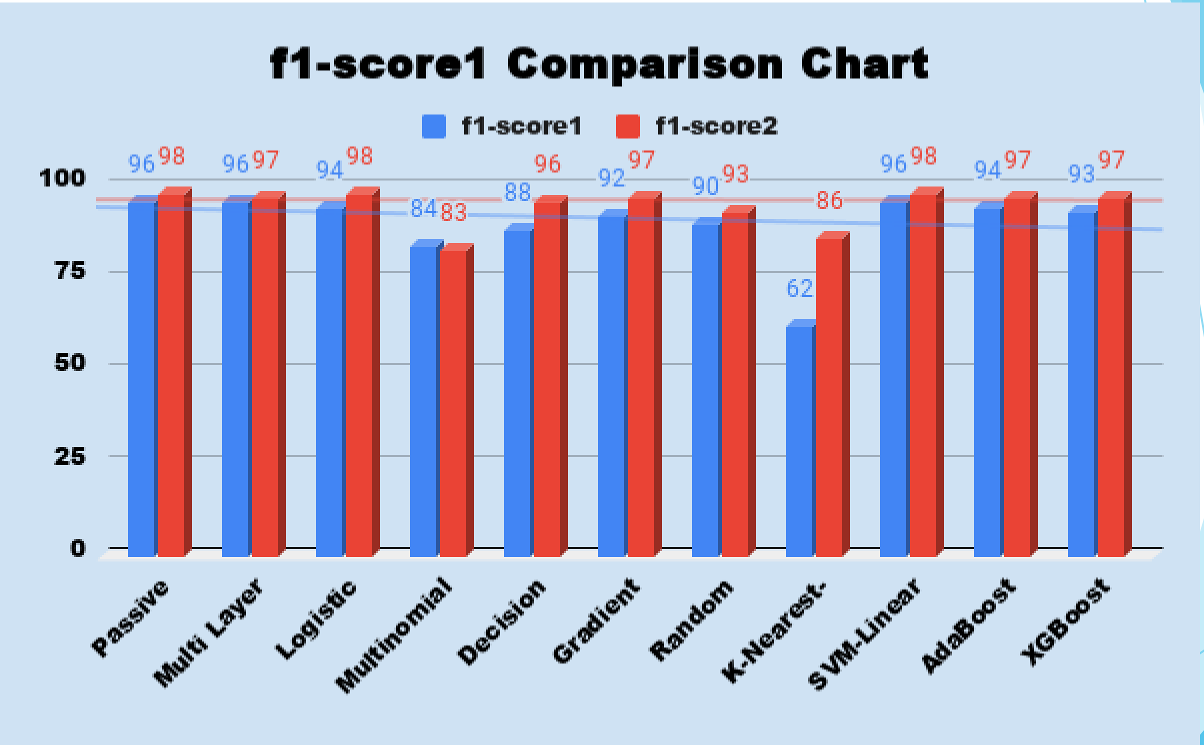
Approach 2:

	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

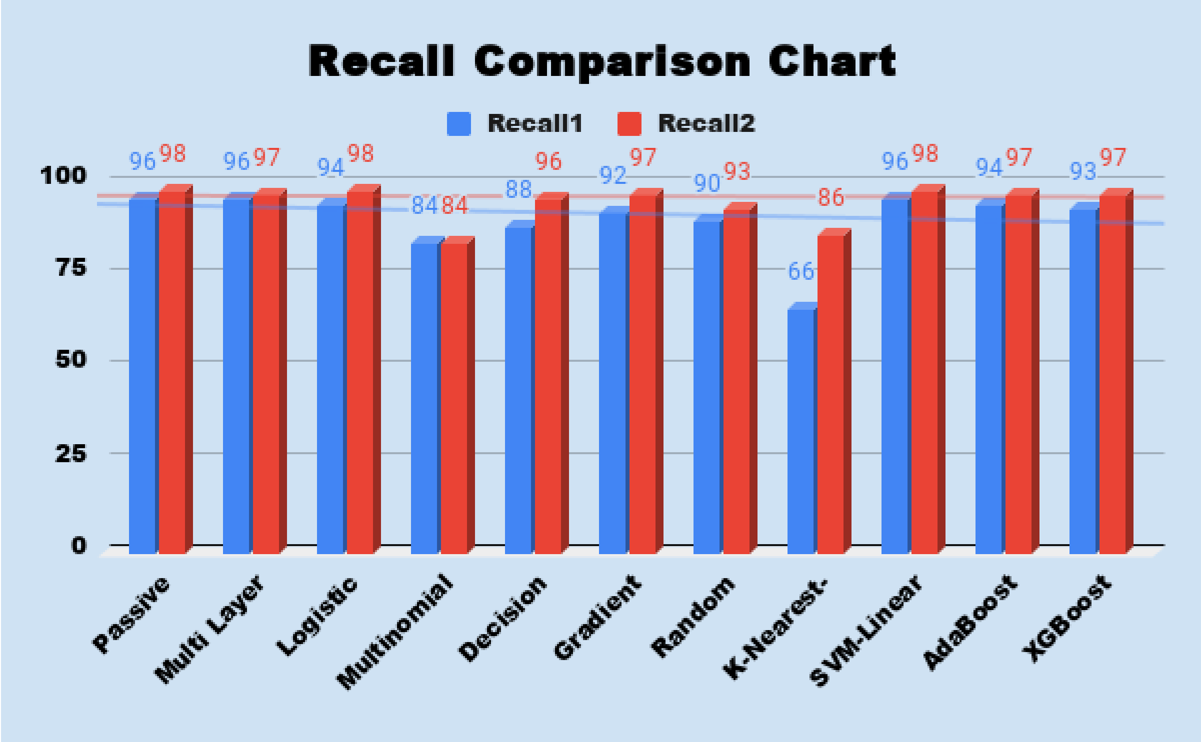


Accuracy Comparison Chart
Accuracy1
Accuracy2
100
75
50
25
0
96 98
96 97
94 98
84 83
87 96
92 97
90 93
66 86
95 98
94 97
92 97
Passive
Multi Layer
Logistic
Multinomial
Decision
Gradient
Random
K-Nearest-
SVM-Linear
AdaBoost
XGBoost

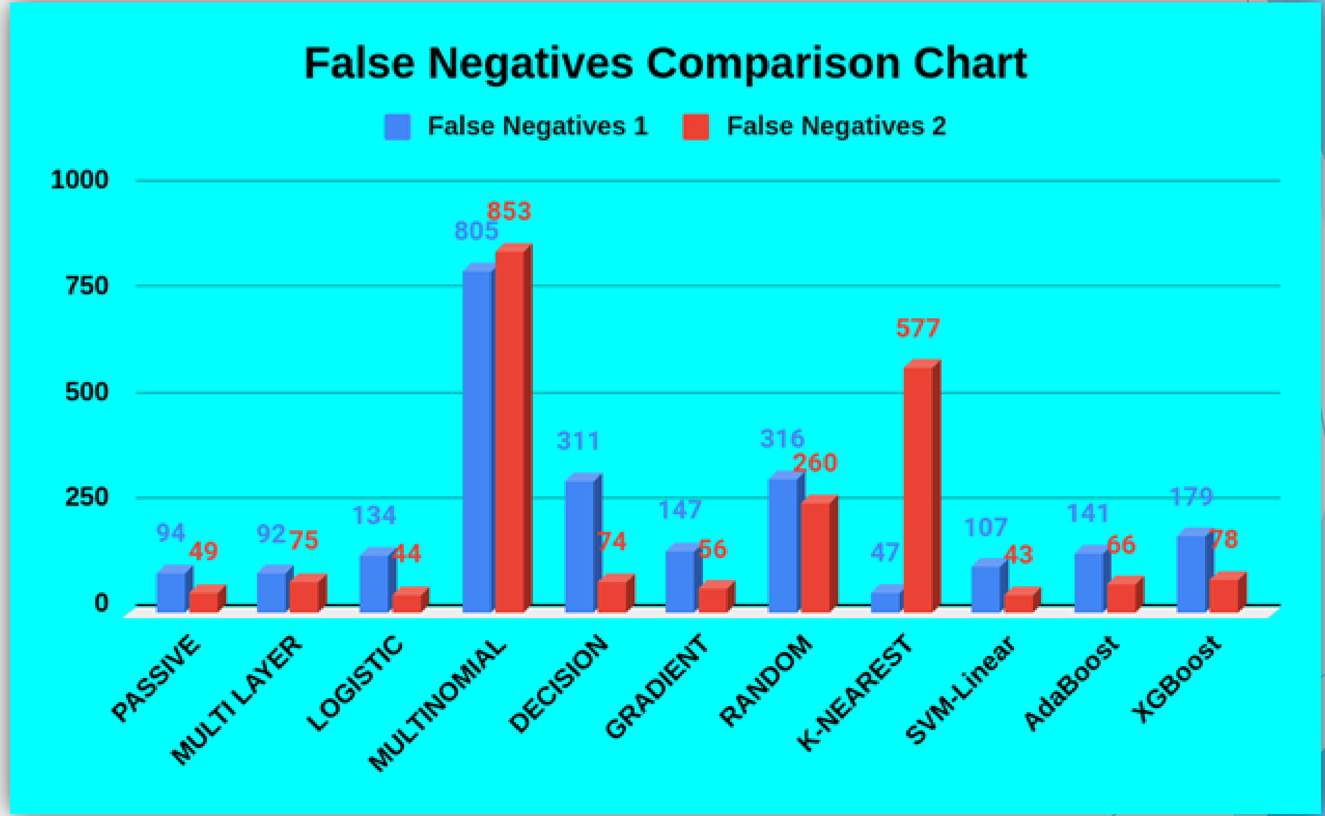

Precision Comparison Chart
Precision1
Precision2
96 98
96 97
94 98
88 87
88 96
92 97
90 93
77 87
96 98
94 97
93 97
100
75
50
25
0
Passive
Multi Layer
Logistic
Multinomial
Decision
Gradient
Random
K-Nearest-
SVM-Linear
AdaBoost
XGBoost



f1-score1 Comparison Chart
f1-score1
f1-score2
100
75
50
25
0
96
98
96
97
94
98
84
83
88
96
92
97
90
93
62
86
96
98
94
97
93
97
Passive
Multi Layer
Logistic
Multinomial
Decision
Gradient
Random
K-Nearest.
SVM-Linear
AdaBoost
XGBoost



Recall Comparison Chart
Recall1
Recall2
100
75
50
25
0
96
98
96
97
94
98
84
84
88
96
92
97
90
93
66
86
96
98
94
97
93
97
Passive
Multi Layer
Logistic
Multinomial
Decision
Gradient
Random
K-Nearest.
SVM-Linear
AdaBoost
XGBoost



False Negatives Comparison Chart
False Negatives 1
False Negatives 2
1000
750
500
250
0
94
49
92
75
134
44
805
853
311
74
147
56
316
260
47
577
107
43
141
66
179
78
PASSIVE
MULTI LAYER
LOGISTIC
MULTINOMIAL
DECISION
GRADIENT
RANDOM
K-NEAREST
SVM-Linear
AdaBoost
XGBoost

Conclusion

A Fake News Classifier should essentially ensure at least the following measure:

1) High accuracy
2) The number of False Negatives must be minimum.

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.

Thank You