UNIVERSITY OF HYDERABAD

PREDICT FAKE NEWS

A thesis presented for the Diploma Degree

By

YASHASVI SHUKLA

(yashasvishukla143@gmail.com)

Enrollment No: 40AIML692-21/1

Github: https://github.com/yashasvi-shukl/FakeNews.git LinkedIn: linkedin.com/in/yashasvi--shukla

Abstract

The proliferation of fake news on social media sites is a serious problem with documented negative impacts on individuals and organizations. This makes detection of fake news an extremely important challenge. A fake news item is usually created by manipulating photos, text or videos that indicate the need for multimodal detection.

Researchers are building detection algorithms with the aim of high accuracy as this will have a massive impact on the prevailing social and political issues. A shortcoming of existing strategies for identifying fake news is their inability to learn a feature representation of multimodal (textual + visual) information.

In this thesis research, we present a novel approach using a Cultural Machine Learning and Deep Learning Algorithm with situational and normative knowledge to detect fake news using text articles. The proposed model's principal innovation is to use the power of natural language processing like sentiment analysis, segmentation process for feature extraction, embedding, vectorization and optimizing it with algorithm. An extensive set of experiments is carried out on real-world news datasets collected from Kaggle and Github.

Acknowledgement

I would like to take this opportunity to express my heartfelt gratitude to all those who not only helped me complete my thesis but also made my experience at this course unforgettable.

Firstly, I would like to express my sincere gratitude to whole team of **Applied Roots**, for there continued help & support, encouragement, inspiration, and tremendous expertise. They gave me the opportunity to choose project of my interset.

Besides the team, I would like to thank faculty and management team of **University of Hyderabad** for offering this course and their insightful comments and inspiration for the completion of this thesis.

I sincerely thank my **parents** for their love and unconditional care. They have always supported and motivated me. They have always believed more in me than myself.

Finally, I would like to offer my sincere thanks to all my **friends** who have been vital support throughout my work.

Content

Literature survey & Data Acquisition

1. Background

The rapid development in technology and the digitization of media have made sharing of information over the geography very quick and easy with growing internet. Now-a-days online social networks act as a primary platform for sharing news and opinions. The reason behind choosing social network platform over conventional social media is because the distribution of news over social media is often more frequent and in no time. Humans prefer image and video contents more than textual reporting's as it provide a strong story-line and attracts more attention to readers. Another reason towards choosing social platform is that it provides short summary report which saves time of a reader. With these advantages large majority of peoples tend to choose social media platform rather than television or newspaper. We are aware of the role of social media in diverse fields such as communication, business, support, situational awareness and aid during disasters, crises, and emergencies. Most of peoples relies on social media for news information and updates so the spread of fake news on social network is a real matter of concern. It is harming our society as well as the world.

Fake news is no new reality. The origin of it existed long ago in society. Fake news is usually created to mislead the readers, damage a community or individual, create chaos, and gain financially or politically. Spreading of fake news in multimedia form is more effective than textual content. Below is an example of disinformation.

Figure 1.1: An example of fake news which claims that the 2019 corona virus outbreak in China could be cured by cocaine consumption[13]

https://breakyourownnews.com/

You might be aware of the spread of fake news during last 3 months of 2016 U.S presidential election. Fake news generated to favor either of individuals was shared more than 37 million times over social media.

2. Problem Definition:

As we know fake news's are intentionally spread to confuse reader and viewers which makes it completely non-trivial to be identified by just textual content and features. Detecting fake news over internet is a very challenging task and this problem still need lots of research, auxiliary knowledge-based information and user-social engagements for improvement.

We define our problem as follows:
Given a set of 'm' news articles which includes textual information, we can represent our data as

$$X = (X_i)_i^m$$

The task of the model is to predict whether given article 'X' is fake or not, i.e., F(X) → 0,1 such that

$$f(X) = \begin{cases} 0, \text{ if 'x' is fake} \\ 1, \text{ Otherwise} \end{cases}$$

Why Fake news detection is an important problem to solve?

1. Readers deserves the truth
2. Fake news destroys credibility
3. Fake news can hurt an individual or group. E.g., Cocaine consumption cures COVID-19
4. Sometimes real news benefits readers.

Real-world impact of solving this problem:

1. It can be a powerful tool for news verification
2. Social platforms will gain trust of readers
3. Avoid misconception and conspiracy.
4. Help to overcome social issues caused because of fake news.

3. What is fake news?

Definition:

Fake news is when hoaxes, opinions, stories, scams or rumors are created to look like legitimate news stories or information. They are presented in such a way to deliberately mislead, deceive and misinform people [1]

Few Facts:

1. 71% of US citizens get news updates from social medias [2] and 68% of Indian audience consume news on smartphones where 24% through social media [3].
2. A fake news website can spread fake news faster than real news.
3. Fake news had more share on Facebook than mainstream news. [4]

Characterization:

Fake news definition is made of two types:

1. **Authenticity:** Means news contain false information but it can be verified as conspiracy theory is not included.
2. **Intent:** Means that the news contains false information with the goal of misleading the readers.

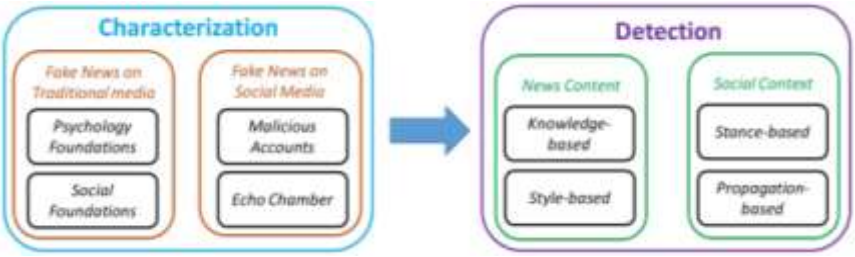


Fig 3.1: Fake news on social media: from characterization to detection [5]

4. Dataset:

To classify news as fake or legitimate news we first need to have labelled and balanced dataset. The need of fake and true news in roughly amount in the dataset is to avoid the frequency of news being used as classifying factor.

In the proposed system, 5 different datasets which are used for classification.

1. **Dataset 1** (**ISOT Fake News Dataset** [6]) contains 44,848 news articles which consists of 23,480 fake news articles and 21,417 real news articles.

Title	Represent the heading/title of the article
Text	Represents body or detailed news of the article
Subject	Represent from which type of news it belongs. E.g., politics news, Government news, etc.
Date	Publish date of article.

2. **Dataset2**(**Fake News** [7]) is another dataset which has 20,800 labelled full-length articles in it. It consists of 10,413 real news articles and 10,387 fake articles. As per source 0 means Real News and 1 indicate Fake News. So, we need to convert label as per problem statement.

Id	Article id
Author	Name of author who written that particular article
Title	Represent the heading/title of the article
Text	Represents body or detailed news of the article
Label	Article is fake or real.

3. **Dataset3** [8] has 30938 labelled articles which consist of 16085 real news articles and 14853 fake articles.

Author	Name of author who written that particular article
Title	Represent the heading/title of the article
Text	Represents body or detailed news of the article
Label	Article is fake or real.

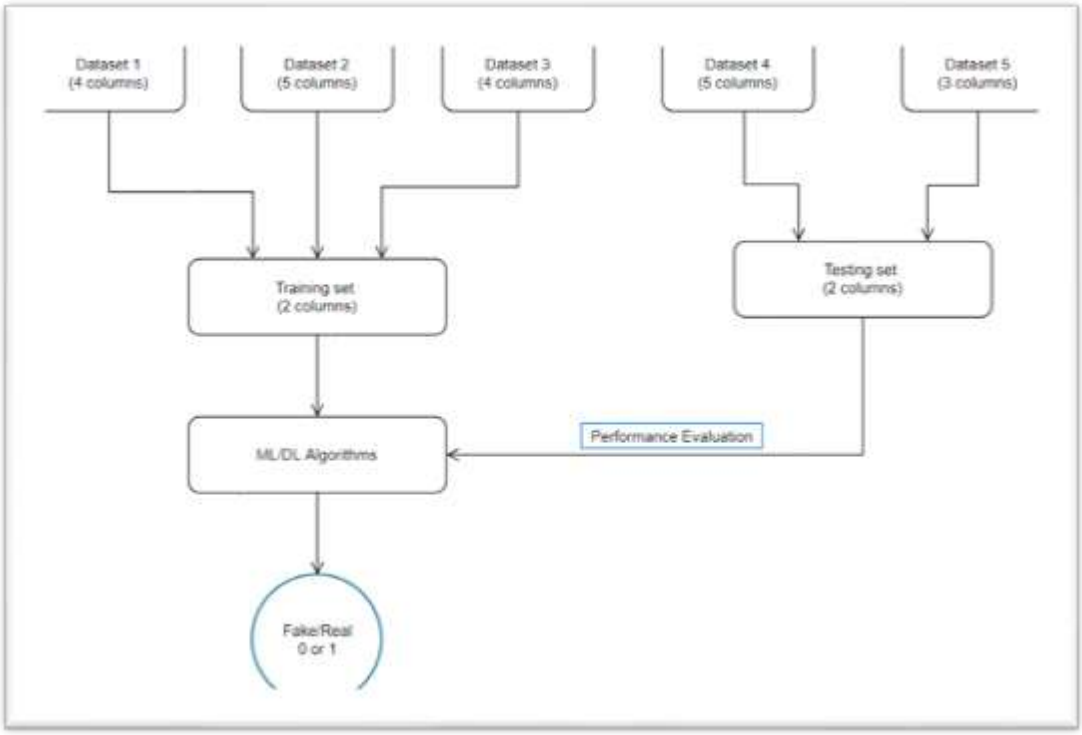
4. **Dataset4** [9] contains 5200 labelled articles out of which 2339 are real news article and 2861 are fake news.

Id	Article id
Author	Name of author who written that particular article
Title	Represent the heading/title of the article
Text	Represents body or detailed news of the article
Label	Article is fake or real.

5. There is another dataset which contains 6335 labeled articles which will be referred as **Dataset5**(**News** [10]) dataset. It consists of 3,171 real news articles and 3,164 fake articles.

Title	Represent the heading/title of the article
Text	Represents body or detailed news of the article
Label	Article is fake or real.

5. Data Cleaning and Processing:



Data size:

Dataset	**Size (MB)**
Training Dataset	300
Test Dataset	53.2

Each model will initially be trained with 80% of **Training dataset**. Remaining 20% of the data will be used for validation. **Test dataset** will only be used for testing the accuracy of trained classifier.

Tools Used:

1.	Scikit-Learn	2.	Tensorflow
3.	Jupyter Notebook	4.	Pandas
5.	Numpy	6.	Matplotlib
7.	NLTK	8.	Seaborn

6. Performance Metrics:

We have used Accuracy, Precision, Recall and F1 score to evaluate the performance of the algorithm. These Metrics are based on the confusion matrix. A confusion matrix is a summary of prediction results on a classification problem which consists of four parameters: true positive, false positive, true negative and false negative [10][11].

6.1. Accuracy Score

It is the ratio of correct prediction to the total number of predictions made (input sample).

Most of the time it works well if data is balanced, it would not be appropriate to use accuracy if data is imbalanced because accuracy would misrepresent the effectiveness of model. It is used when the True Positive and True Negative are more important than False Negative and False positive [12].

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predicition\ made} \quad OR \quad \frac{TP + TN}{TP + TN + FP + FN}$$

High accuracy usually represents a good model but as we are training a classification model, classifying article as true when actually it is fake can have negative impact; similarly classifying article as fake when actually it is not fake could cause trust issue.

Therefore, we have used three other metrics that take incorrectly classified observations into account, i.e., precision, recall, and F1-score.

6.2. Recall

It is the ratio of correctly predicted positive class out of total actual positive class. Also referred to as 'Sensitivity'.

When the cost of false negative is high and the cost of false negative is low, it is better to use recall as an evaluation metric.

$$Recall = \frac{Predicted\ Positive}{Actual\ Positive} \quad OR \quad \frac{TP}{TP + FN}$$

6.3. Precision

It is defined as the ratio of relevant instances made out of all retrieved instances.

When the cost of a false positive is very high and the cost of a false negative is low, it is better to use recall as an evaluation metric.

$$Precision = \frac{Predicted\ Positive}{Actual\ Results} \quad OR \quad \frac{TP}{TP + FP}$$

6.4. F1 – Score

It is defined as the harmonic mean of precision and recall.

It is used when False Negative and False positive is more important than True Positive and True Negative

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

Alternative metrics:

Log loss: It is an appropriate classification metric when outcome of the model is probability of the class. Here, in this case we are dealing with binary classification where we don't care much about probability. Therefore, we are using few ML models which does not output probability.

7. Challenges Faced:

1. Lack of clean data.
2. Challenging problem is to collect the available datasets.
3. In real scenario loss to value of information is very high.
4. Authenticated fact checked data sources. How much we can trust on datasets published on Kaggle and Github?
5. Content based classification of news is just a part of the whole picture.
6. Data cleaning and preparation for model building.
7. Distinguish between click-bait and actual fake news.

8. Related Works:

In this section I will focus on some related works which have been done in past which worth investigating.

1. Supervised Learning for Fake News Detection [13]

Reis et al. used Machine Learning approach to detect Fake News. They evaluated performance of various ML algorithms (k-Nearest Neighbor, Naïve-Bayes, Random Forest, SVM with RBF kernel and XGBoost) on Buzzfeed dataset.

They used lots of features in order to train the model. Those features are: -

- Language Features: BOW (bag-of-words), Parts of Speech (POS) tagging and 31 other features.
- Lexical Features: unique words, frequency of words, common words, stop words, etc.
- Psychological Features [14]: Dictionary formed by text mining software's using Linguistic inquiry and word counts.
- Semantic Features: Toxicity score from Google's API.
- Engagement: Total number of comments within a time interval.

Result:

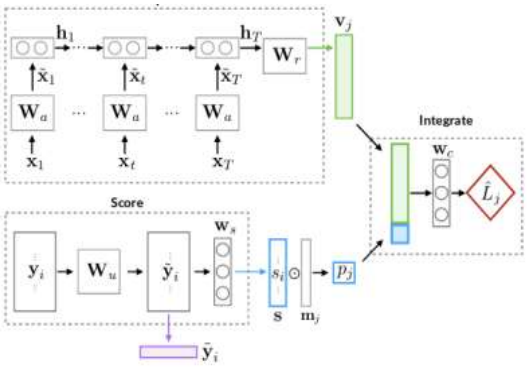
Classifier	AUC	F1
KNN	0.80±0.009	0.75±0.008
NB	0.72±0.009	0.75±0.001
RF	**0.85±0.007**	**0.81±0.008**
SVM	0.79±0.030	0.76±0.019
XGB	**0.86±0.006**	**0.81±0.011**

RF and XGB performed best.

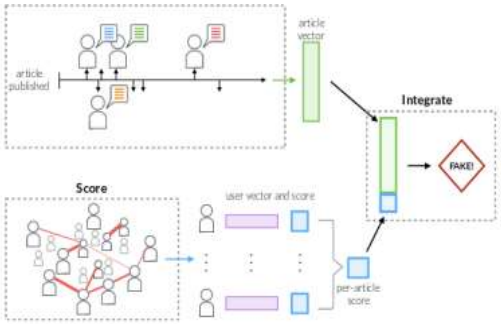
Fig 6.1: Result by Reis et al.

2. CSI: A Hybrid Deep Model for Fake News Detection [15]

Ruchansky et al. used Deep Learning model to classify Fake News. They merged news content features and social engagement features into a single network. They used RNN for feature extraction from news content and full connected neural layer for social feature extraction. The results of both the networks are then concatenated and used for final classification. They used doc2vec [16] as textual feature. Below is the network architecture of CSI Model.



(a) The CSI model specification. The Capture module depicts the LSTM for a single article a_j, while the Score module operates over all users. The output of Score is then filtered to be relevant to a_j.



(b) Intuition behind CSI. Here, Capture receives the temporal series of engagements, and Score is fed an implicit user graph constructed from the engagements over all articles in the data.

Fig 6.2: CSI Model

Result:

They tested model performance on different datasets i.e., Twitter and Weibo (a Chinese equivalent of Twitter). Compared to other models CSI performed better with 6% improvement.

	Twitter		Weibo	
	Accuracy	F-score	Accuracy	F-score
DT-Rank	0.624	0.636	0.732	0.726
DTC	0.711	0.702	0.831	0.831
SVM-TS	0.767	0.773	0.857	0.861
LSTM-1	0.814	0.808	0.896	0.913
GRU-2	0.835	0.830	0.910	0.914
CI	0.847	0.846	0.928	0.927
CI-t	0.854	0.848	0.939	0.940
CSI	**0.892**	**0.894**	**0.953**	**0.954**

Fig 6.3: Result by Ruchansky et al.

3. Some Like it Hoax: Automated Fake News Detection in Social Networks

In this research [16], Tacchini et al. collected two categories (scientific news and conspiracy news) of data from social network data using Facebook Graph API. They used LR (Logistic Regression) and Harmonic BLC [17] (Harmonic Boolean Label Crowdsourcing) for classification.

They used 80% of data for training and remaining 20% of data for testing and 5-fold cross validation. They achieved 99% accuracy in both the cases.

In addition to that they used one-page-out approach an half-page-out approach, they used half page for training the model and the second half for testing the performance. As a result, Harmonic model outperformed Logistic Regression.

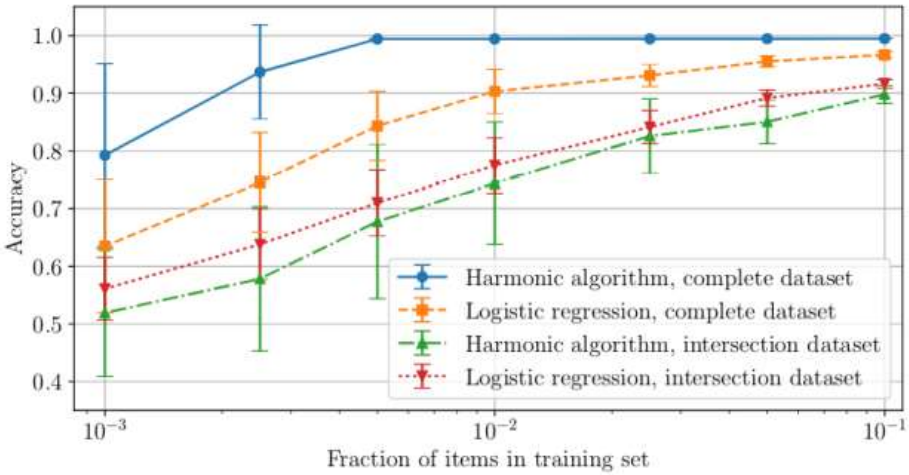


Fig: 6.4: Cross Validation of model

	One-page-out		Half-pages-out	
	Avg accuracy	Stdev	Avg accuracy	Stdev
Logistic regression	0.794	0.303	0.716	0.143
Harmonic BLC	0.991	0.023	0.993	0.002

Fig 6.5: Results by Tacchini et al.

Phase 2: EDA and Feature Extraction

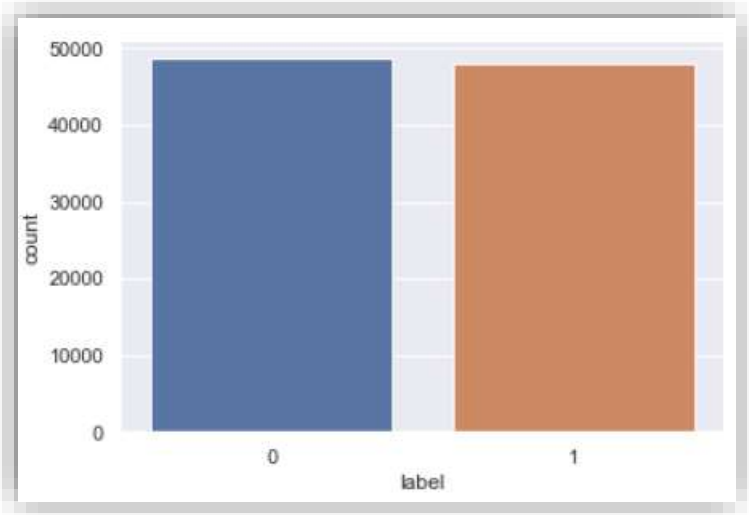
Introduction:

Exploratory Data Analysis and Feature Engineering is the most important part of any model building. EDA helps us to understand the pattern of data and helps us to extract relevant feature from it.

To understand the dataset a thorough analysis has been performed to study the text data. There are some differences between fake and real news, I investigated the dataset from various perspectives to find different hidden features out of it, such as statistical analysis, sentiment analysis, psychological analysis and various other feature extraction techniques. To start with data exploration the first thing to be done is statistical analysis like number of sentences, words, characters, etc. Then it will become possible to take deeper insight of data by plotting in 2D or 3D.

Datasets:

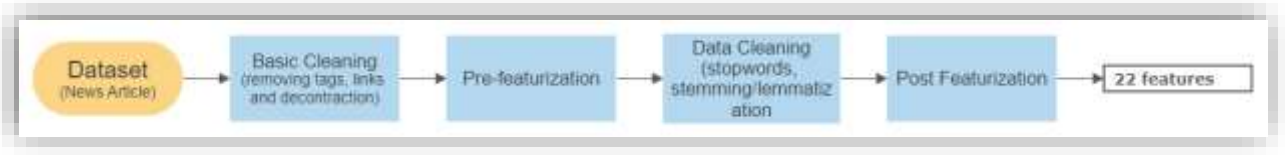
Here I have used few open-source datasets for training and testing purpose. Models will be trained and tested on different combinations of datasets. The final training dataset consist of **96517** rows which contains **48628** fake news and **47889** real news. Datasets has different column information but to classify news as fake or real I will be using only two columns i.e., title and text. To have a better view of data a histogram is given below:



Text Analysis:

Looking at the data I have some interesting findings about the data. There are few rows which has missing values in text and title. It can also be observed that dataset contain lots of meaningless words and special characters. Articles also contains html tags and web links.

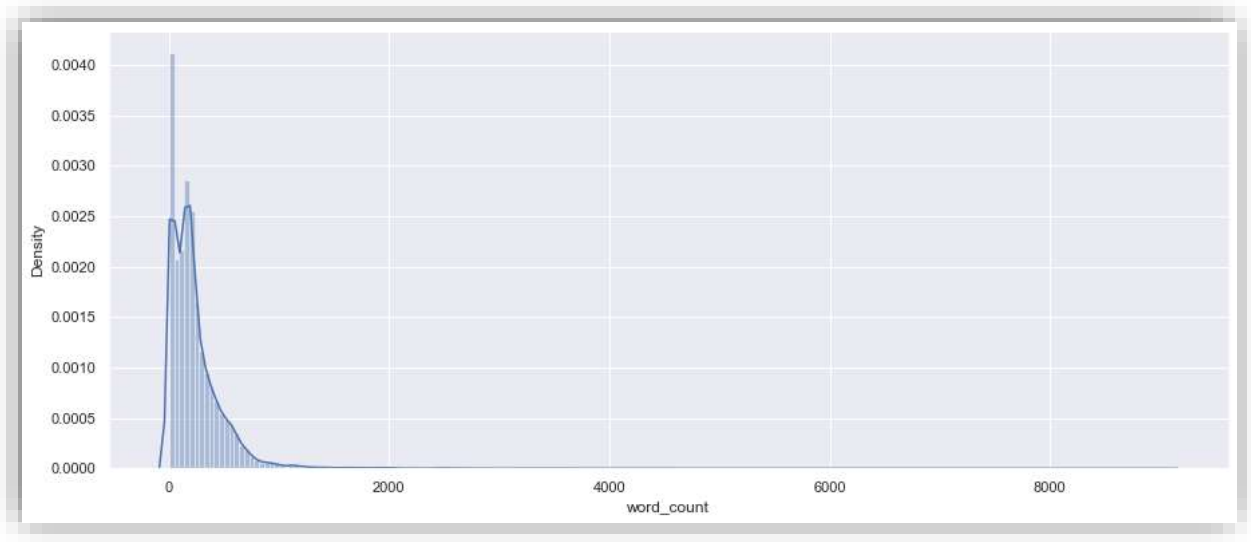
To start with text analysis and featurization it is very important to clean the data and keep only meaningful words in the article. Data cleaning and featurization are performed in four stages, below is the outmap of the same.



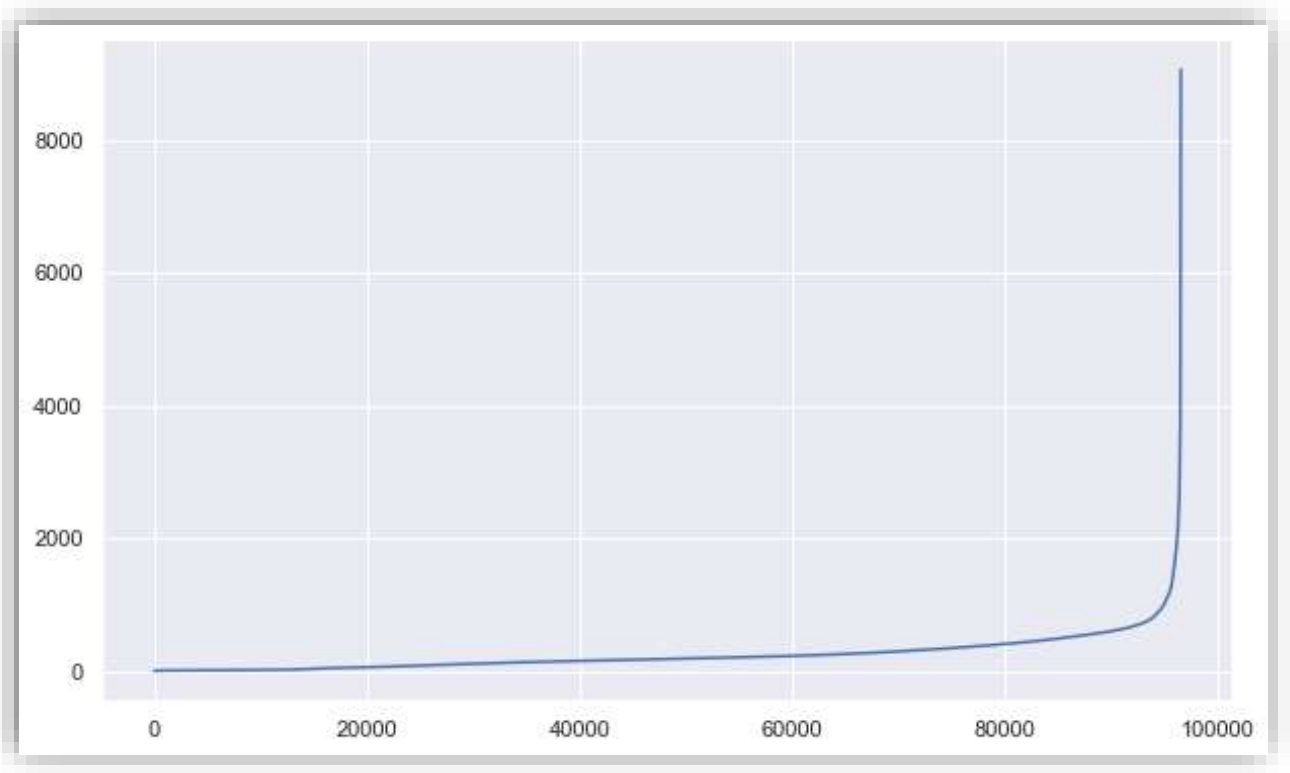
Textual features are computational features derived from the text content. Before the era of modern Machine Learning and Deep Learning, researchers depend heavily on text content and meta data for extracting features.

Univariate Analysis:

1. As we are dealing with classification problem of multi-dimensional data, it is sometimes difficult to find sense out of data using univariate data analysis. But I believe univariate analysis can be best used to detect the presence of outliers in the data. Below is the distribution plot of word count.



In the above plot you can see majority of articles have word length less than 500 and very few numbers of articles have article length greater than 1000. From the above plot it's still not clear exactly how many percentiles of data is outlier. To find that answer let's visualize it further.



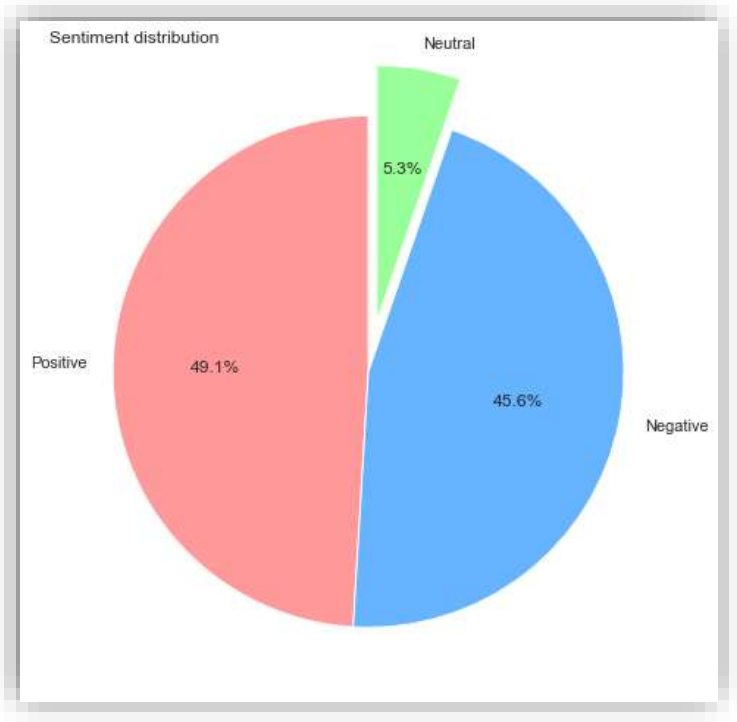

```
75 precentile value is  316.0
80 precentile value is  370.0
85 precentile value is  436.0
90 precentile value is  526.0
95 precentile value is  659.0
100 precentile value is  9073.0
```

```
95 precentile value is  659.0
96 precentile value is  706.0
97 precentile value is  778.0
98 precentile value is  924.0
99 precentile value is  1229.0
100 precentile value is  9073.0
```

Above is sorted line plot of word count. Here we can see there are about 90000 articles with article length less than 900 i.e., approximate to 97 percentiles of data. It can also be seen that there are bulk of articles which has very few (very close to zero) numbers of words in the article i.e., around 14000.

Very few and very large number of words in an article could act like and outliers which decrease out model performance. Therefore, we need to remove this from our final model training data.

2. From below plot we can infer that out data contains majority of positive sentiment article. It is observed that fake news contains more of negative sentiments as fake news creators personally attack on peoples to defame or for personal benefits. However, real news is created with certain limitation on contents. Therefore, real news contains more of positive sentiments.



Bivariate Analysis:

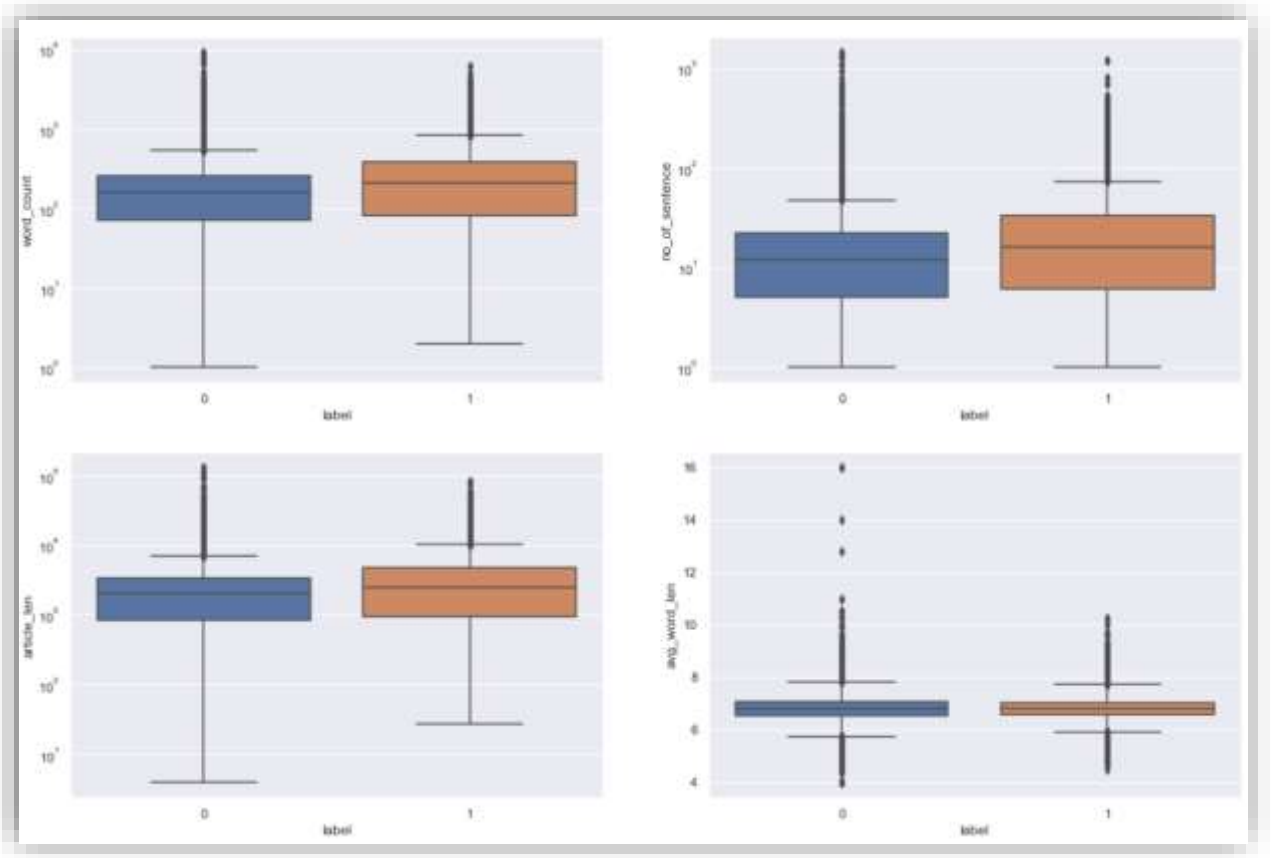
1. Computational/Statistical features:

a) Number of words, sentences and article length: We all knows, liars have complete control over their story but to portrait their story as real sometimes their state of mind leaks through their writing pattern/style. Similar things happen with fake news creators.

From the below figure we can see real news have larger number of words, sentences, article length and avg number of words. There are 263 words on an average in real news, while fake news contains 222 words. Number of words in fake new is distributed over a wide range, which indicate that some fake news contains very few numbers of words and some fake news have very large number of words.

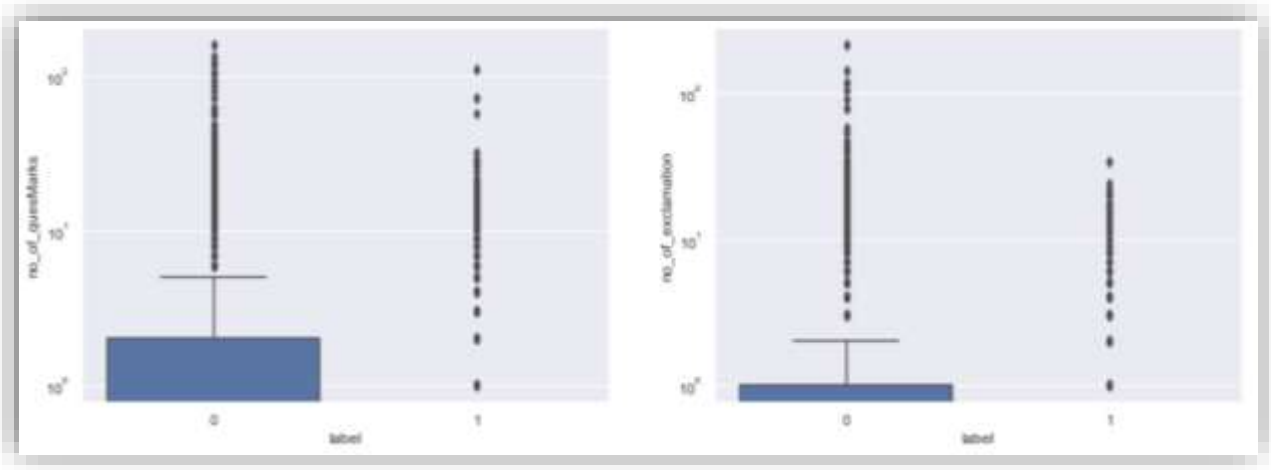
Similarly, we can see count of number of sentences in real news is slightly higher than the fake news. On a count, there are 24 sentences on an average in real news while fake news has average of 19. Average word length and article length also follows same pattern.

From the below box plots we can infer that the variance of all statistical feature in real news is smaller than that of fake news. The reason could be that the article writer or editor of real news have to follow certain set of rules which includes article length. However, fake news creators do not have as such any restriction.



b) Question mark and Exclamation marks: As per the analysis performed on news text data, real news has fewer question marks than real news (as shown in below fig). The reason behind this is that in fake news questions are asked in order to create a dramatic effect or to make a point rather than expecting an answer. E.g., "Who would not like to earn money?"

Also, we can see that the use of exclamation mark is higher in fake new than real news. The reason is that fake news is created in declarative way so that the creator can turn a simple sentence into a strong command. Hence, fake news contains a greater number of exclamation marks than a real news.



c) Cognitive Perception: From cognitive perception, I focused on the presence of negative words in the article. As per the research's [1], real news contains a greater number of negation words than fake news because the writing tries to be more specific and precise to lower chance of being caught into contradiction.

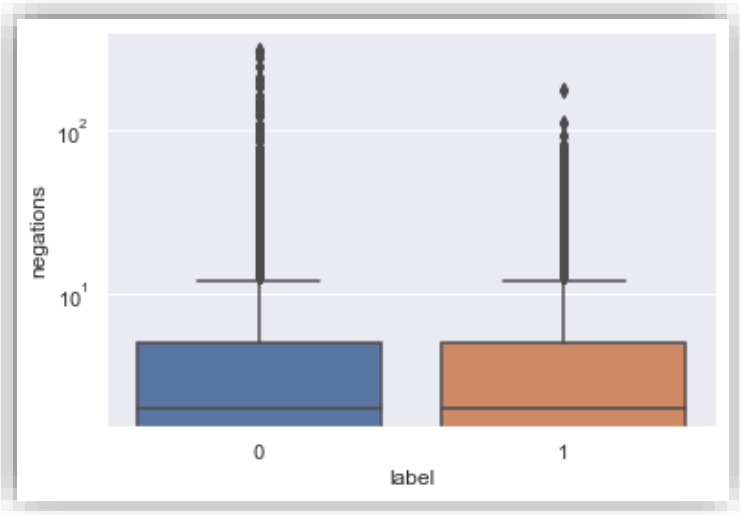
From the below plot we can see that the median of fake news and real now is almost equal in this case. Fake news has slightly higher number of negation words as outlier.

```
Fake News

count    48628.000000
mean         3.826540
std          7.620807
min          0.000000
25%          0.000000
50%          2.000000
75%          5.000000
max        305.000000
Name: negations, dtype: float64
```

```
Real News

count    47889.000000
mean         3.697801
std          5.082761
min          0.000000
25%          0.000000
50%          2.000000
75%          5.000000
max        171.000000
Name: negations, dtype: float64
```

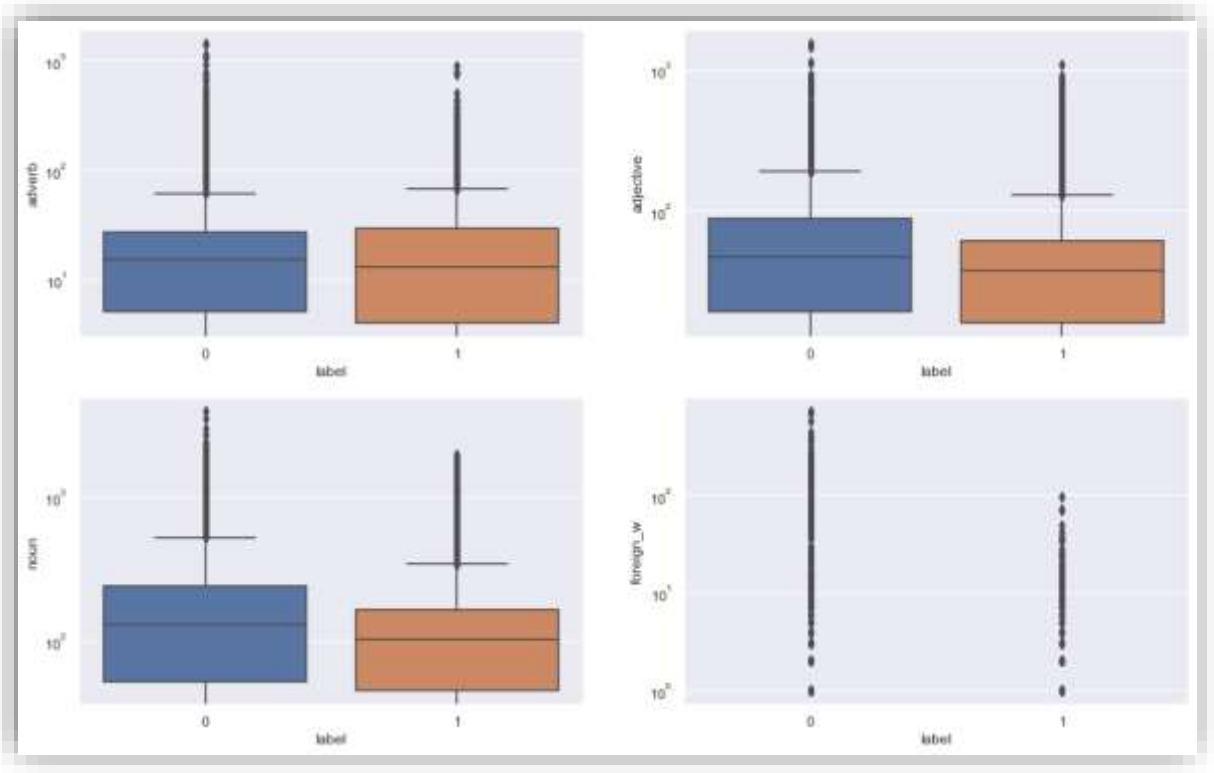


2. **Morphological Analysis (POS tagging):** POS tagging is a process of marking a word present in the corpus to its corresponding part of speech. As pos tagging consider the relation between words, I have used pos tagging for feature engineering.

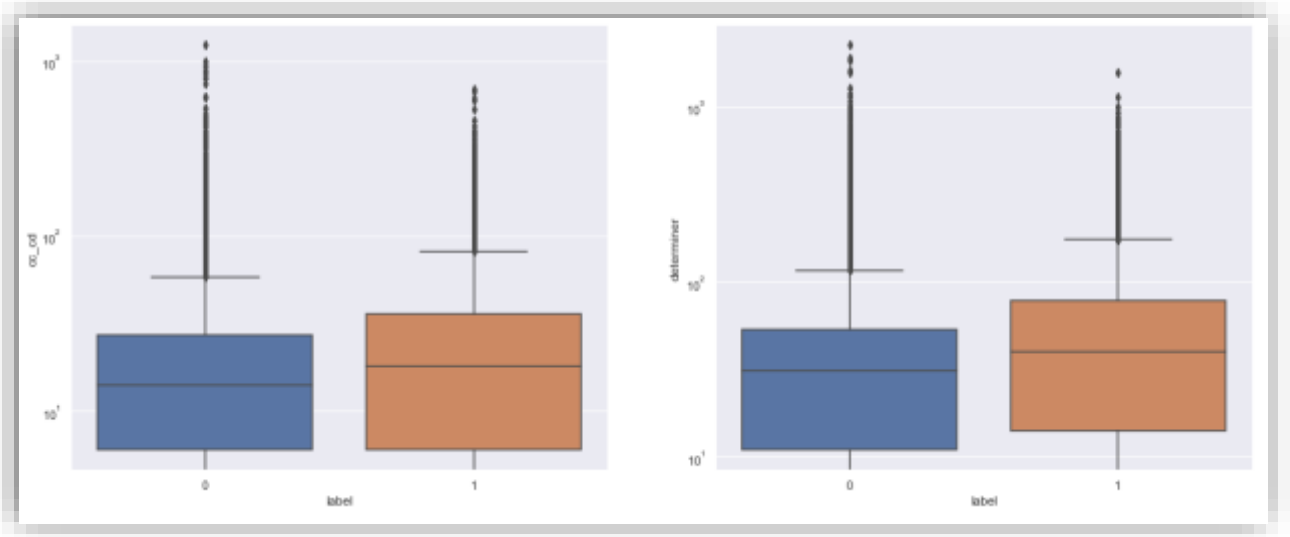
GTAG	POS Tags
group C	CC (coordinating conjunction) , CD (cardinal number)
group D	DT (determiner)
group E	EX (existential there)
group F	FW (foreign word)
group I	IN (preposition, subordinating conjunction)
group J	JJ (adjective) , JJR (adjective, comparative) , JJS (adjective, superlative)
group M	MD (modal)
group N	NN (noun, singular or mass) , NNS (noun plural) , NP (proper noun, singular) , NPS (proper noun, plural)
group P	PDT (predeterminer) , POS (possessive ending) , PP (personal pronoun)
group R	RB (adverb) , RBR (adverb, comparative) , RBS (adverb, superlative) , RP (particle)
group T	TO (infinitive 'to')
group U	UH (interjection)
group V	VB (verb be, base form) , VBD (verb be, past tense) , VBG (verb be, gerund/present participle) , VBN (verb be, past participle) , VBP (verb be, sing. present, non-3d) , VBZ (verb be, 3rd person sing. present)
group W	WDT (wh-determiner) , WP (wh-pronoun) , WPS (possessive wh-pronoun) , WRB (wh-abverb)

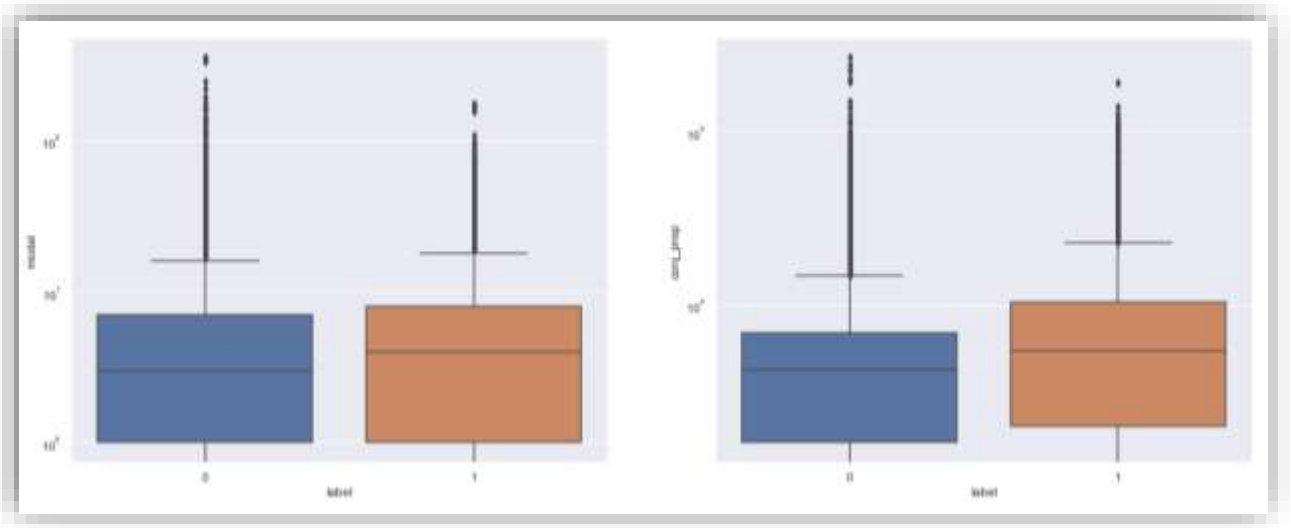
POS Tag Groups Representation by (Kapusta, Hájek, Munk, & Benko, 2020)

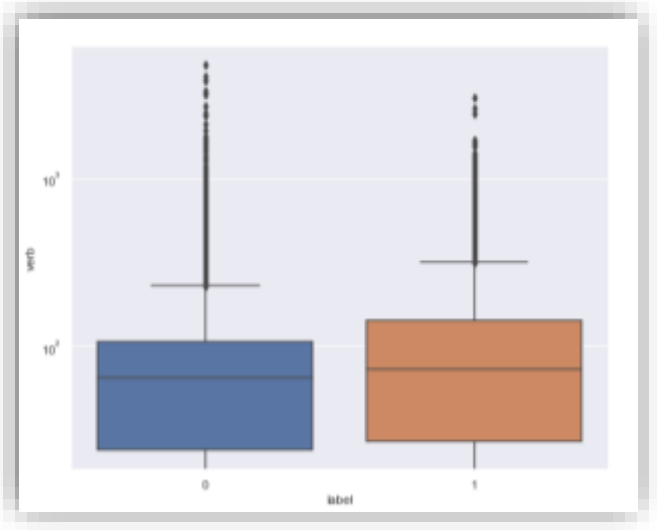
I have generated the count of post tags present in the news article. In feature engineering I have excluded rarely occurring groups (group E, P, T, U and W) to avoid sparse matrix. Below is the box plot representation of pos tags created during feature engineering.



It can be observed that the occurrence of group N (Nouns), F (Foreign Words), R (Adverbs) and J (Adjectives) in favor of fake news. Reason lies behind is that the fake news creators in usual directly attack on people, group or community which involves use of more noun and adjective whereas groups CC (Coordinating Conjunction), CD (Cardinal Number), I (Prepositions), M (modal), D (Determiner) and V (Verbs) in favor of real news. However, in all the cases we can see fake news contains large number of outliers.

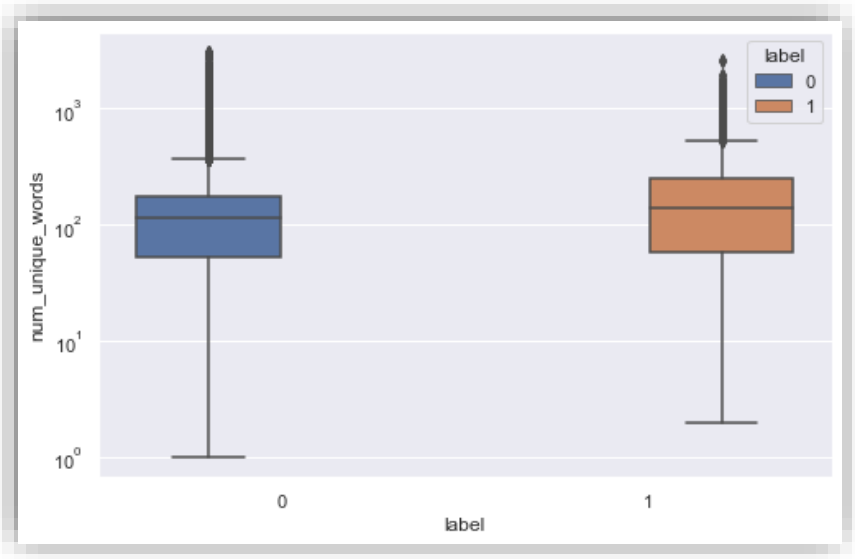






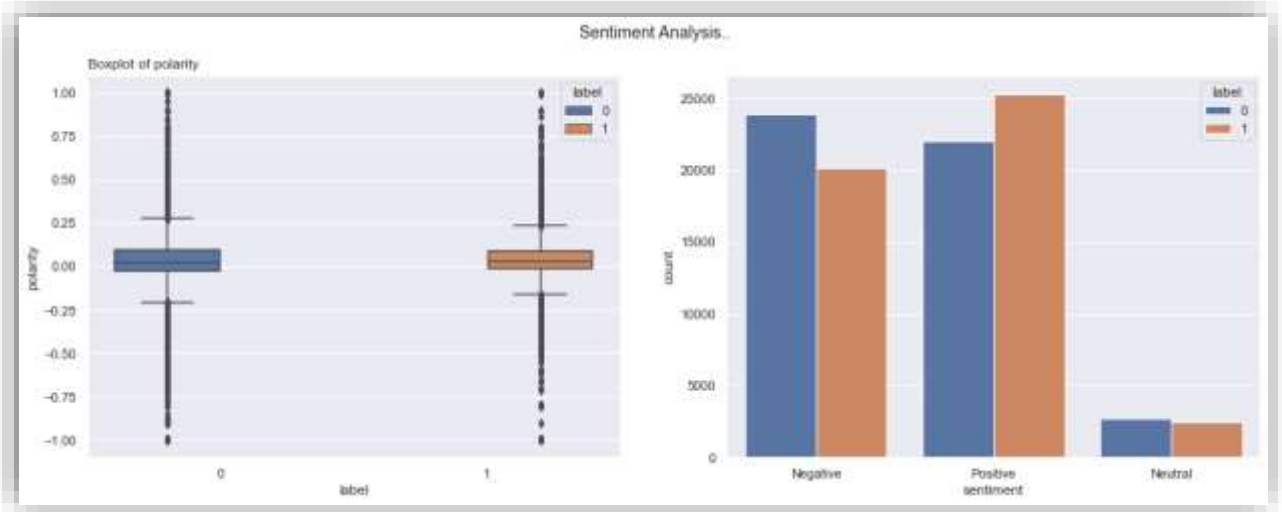
3. **Lexical Diversity:** It is the measure of number of unique words used in the article [20]. In many researches it is observed that the rich news has more diversity in content. That means real news contains a greater number of unique words.

   In the below boxplot we can see real news contains primarily a greater number of unique words compared to fake news. The presence of outlier is more in fake news than in real news.



On an average, the number of unique words present in the data is 179 for real news whereas 147 for fake news.

4. **Sentiment Analysis:** It is an NLP approach which measures writers' emotion behind the text. NLTK provides a package called VADER (Valence Aware Dictionary for Sentiment Reasoning) which is used for Sentiment Analysis. VADER compute sentiment score by summing up the emotion of each word. In the experiment we classified all articles with compound score greater than 0.05 as positive, less than -0.05 as negative, else neutral.

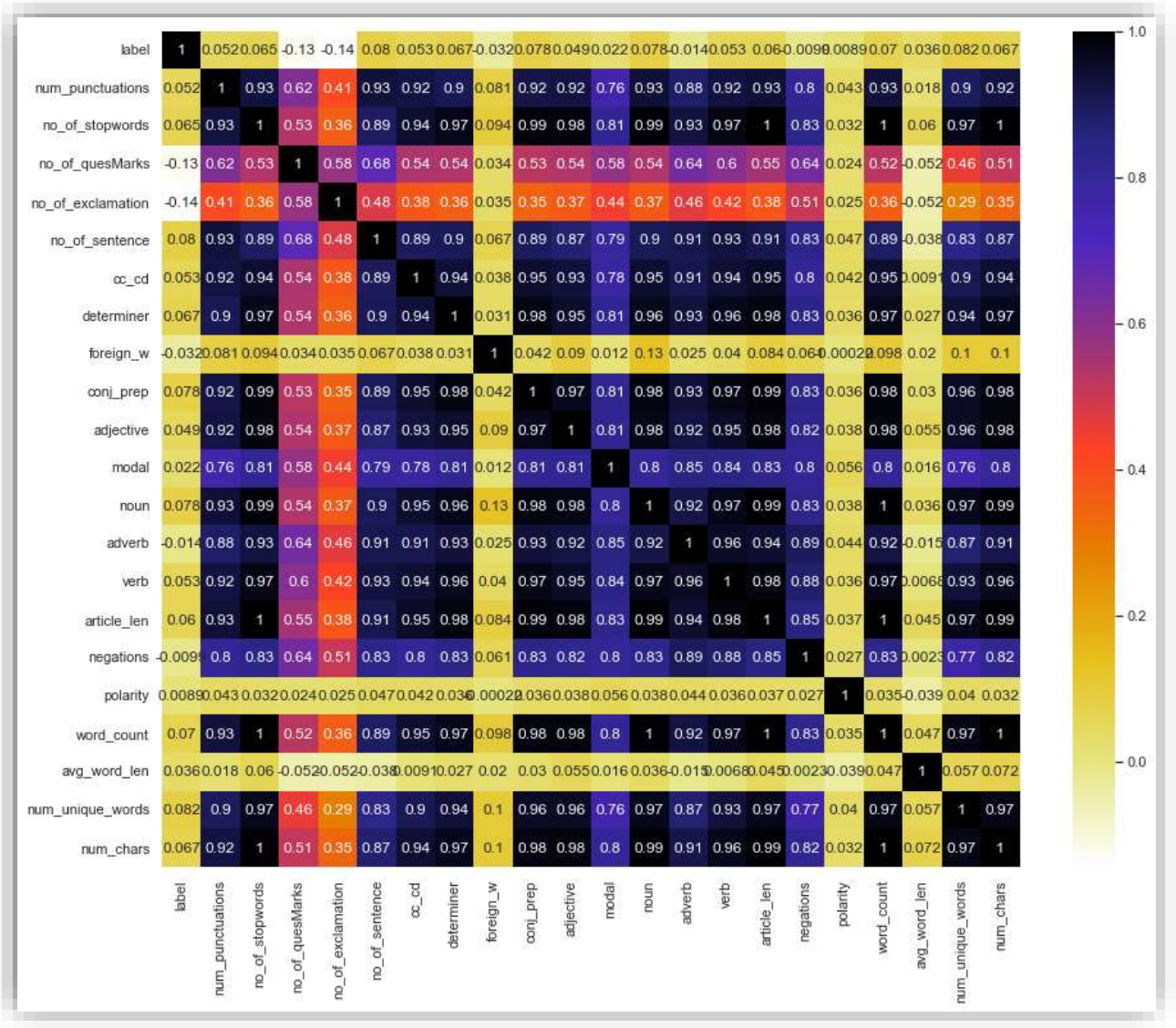


From the above boxplot we can see that the polarity of data is almost same. Even though polarity of data is equally distributed we can observe significant difference in the sentiment of the articles. From the above count plot, we can understand that real news contains more of positive article whereas fake news contains more of negative sentiment sentences.

Multivariate Analysis:

It is a kind of data analysis in which more than one type of measurement or observation is involved.

1. **Correlation heatmap:** It is used to find pairwise correlation between different columns of data.

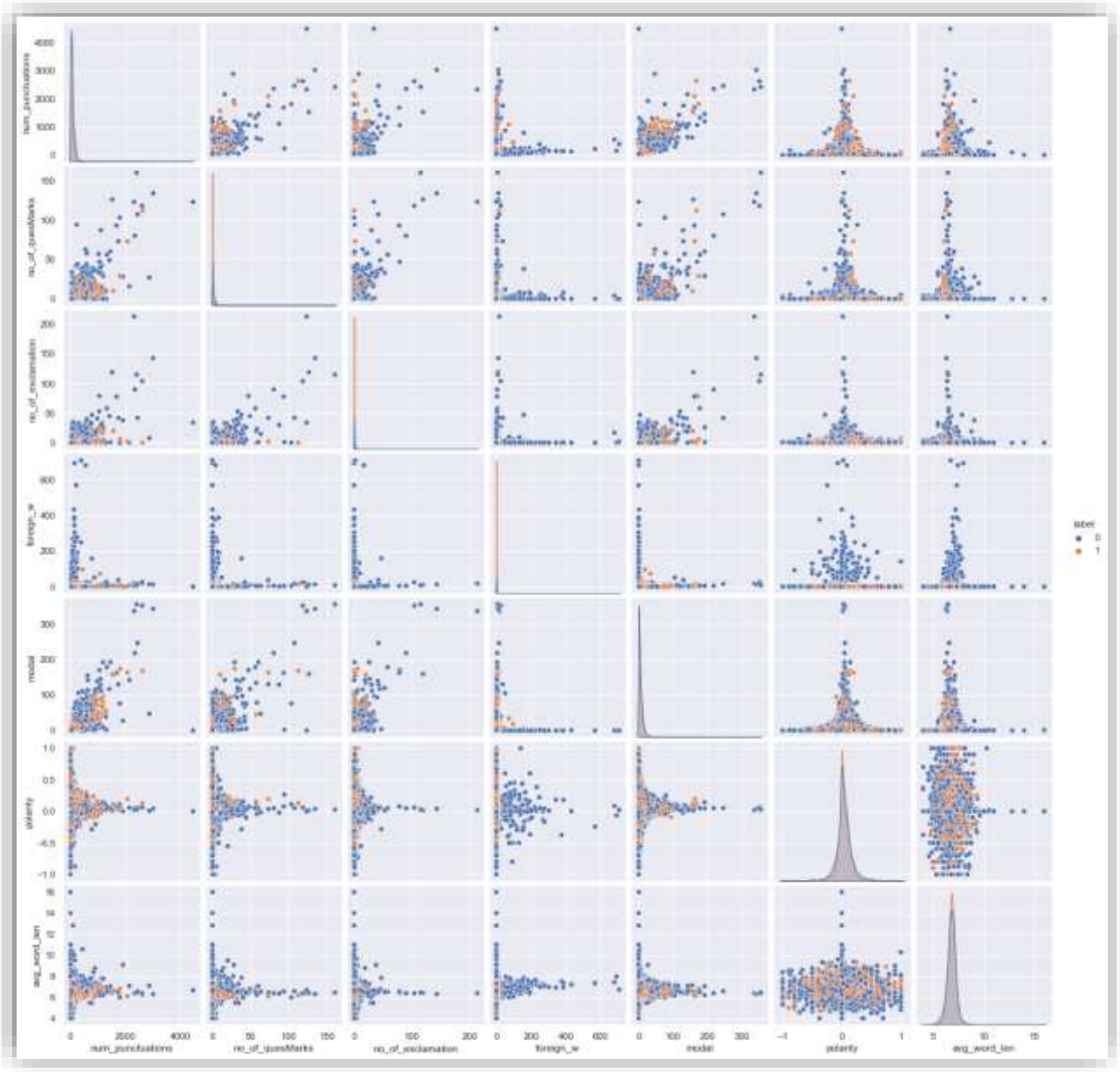


In the above correlation heatmap we can clearly see there are number of columns which are highly corelated. Correlation ranges from -1 (perfect negative correlation), 0 (no correlation) to +1 (perfect positive correlation). If two features are positively correlated then change in one will impact other in same direction whereas if two features are negatively correlated then change in one will impact other in opposite direction.

According to above analysis, features **'adjective', 'adverb', 'article_len', 'cc_cd', 'conj_prep', 'determiner', 'negations', 'no_of_sentence', 'no_of_stopwords', 'noun', 'num_chars', 'num_unique_words', 'verb' and 'word_count'** are highly correlated. Keeping all these high correlated features in training data is not that usefu l. Therefore, we will drop these features from model training data.

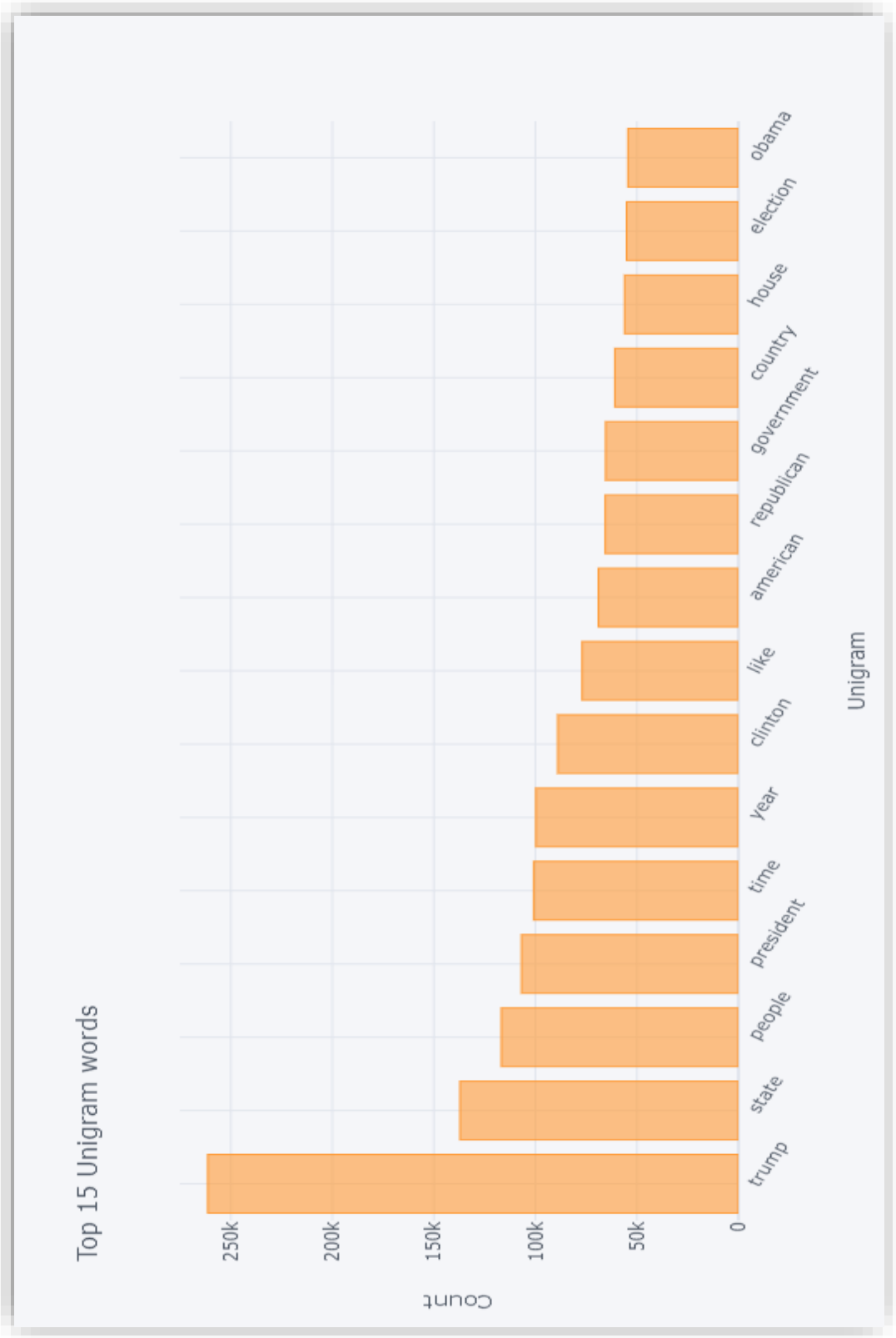
2. Pair Plot:

Plot represent pairwise bivariate relationships between dataset features.

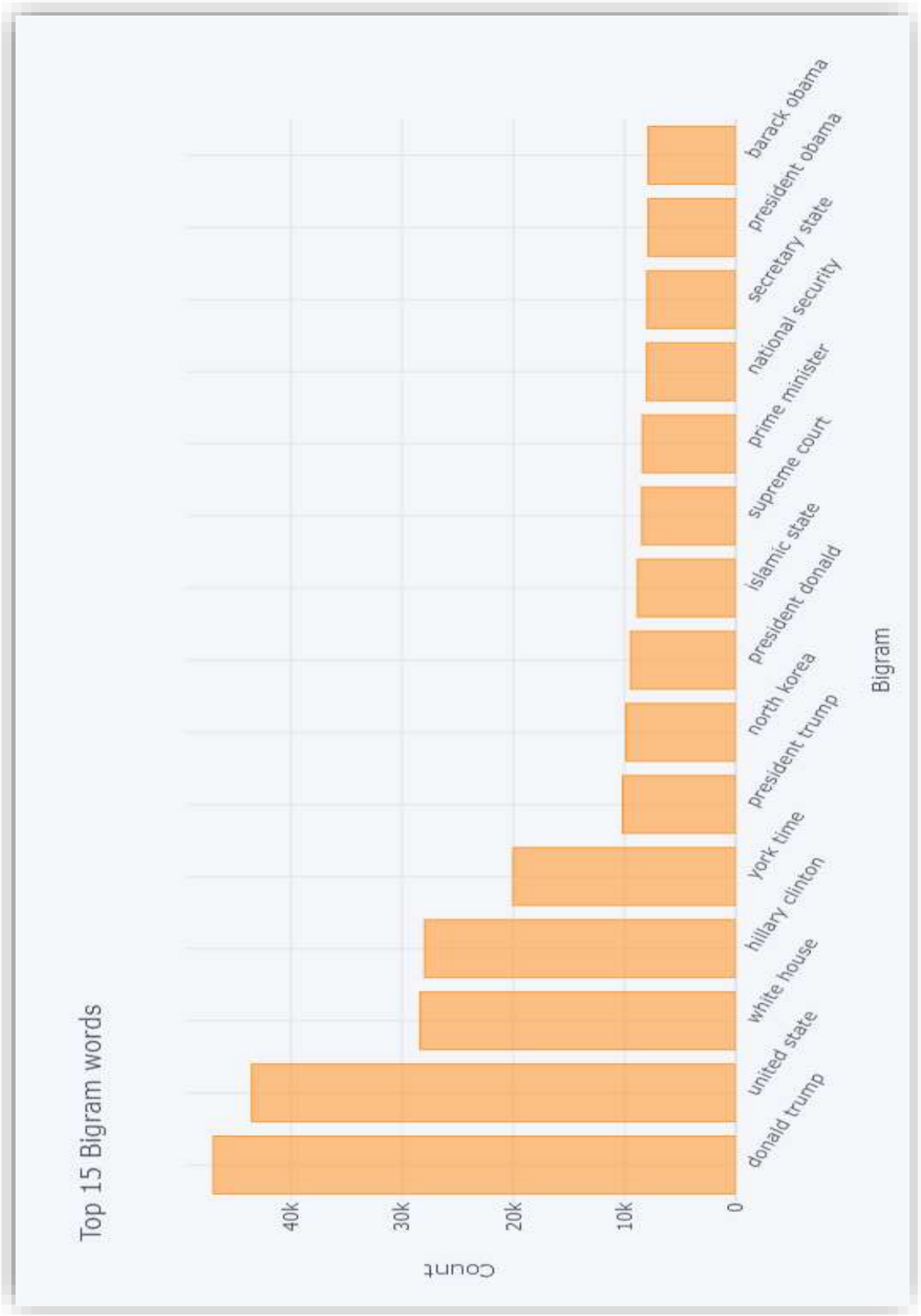


Looking at the above plot we see lots of classified points between few features which can be useful for creating a good model for classification. Plots forming with features foreign_w, polarity and avg_word_len seems to be more useful for classifying fake and real news.

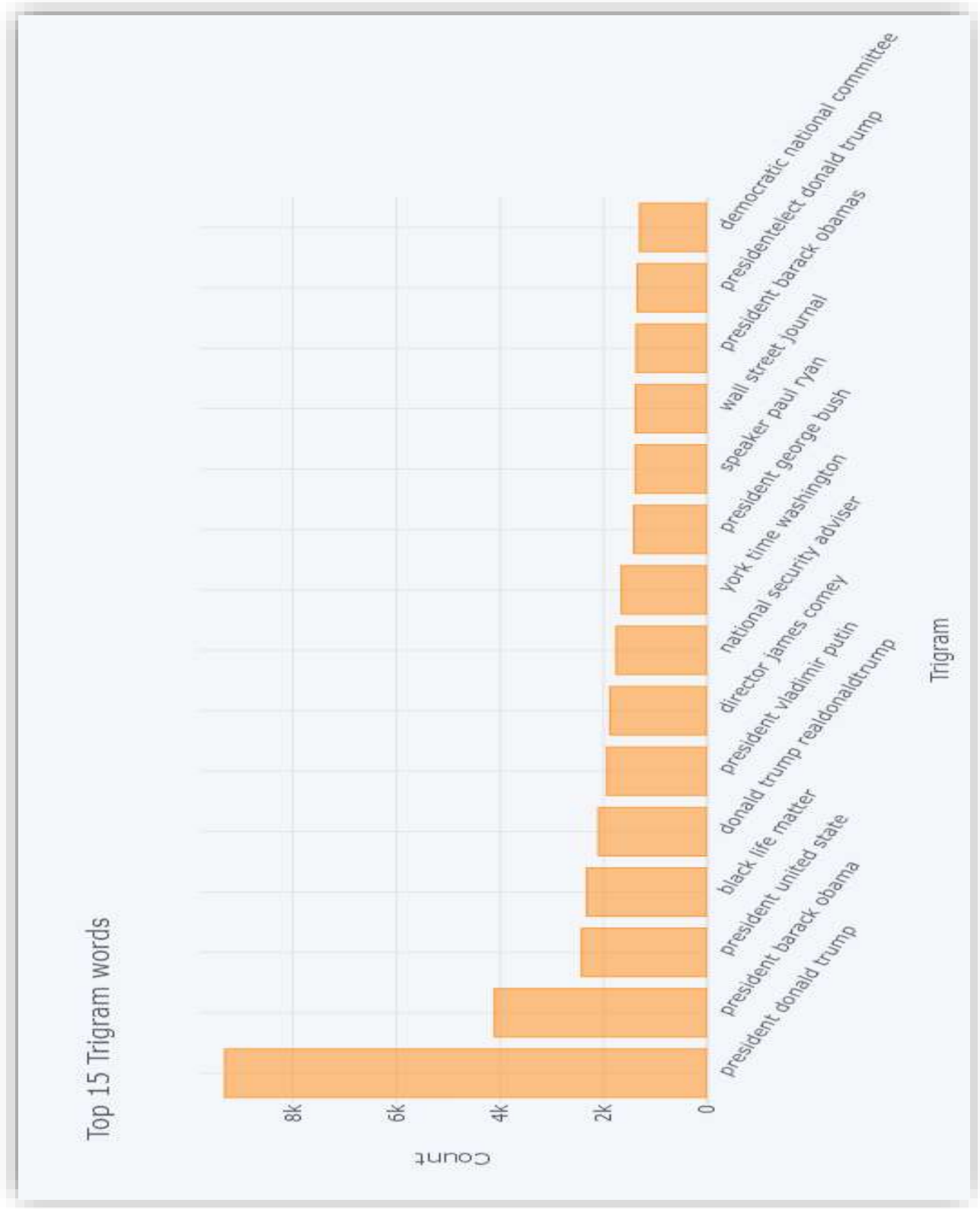
Unigram Analysis:



Bigram Analysis:



Trigram Analysis:



TSNE (T-distributed Stochastic Neighbor Embedding) Visualization:

t-SNE [1] is a dimensionality reduction technique which is well suited for visualizing high dimensional data into 2d or 3d plots.

In the above T-SNE 3d plot we can see words with similar semantic meaning (e.g., threatened, threatening, threat, danger, risk and potentially) and words which are more often used together (e.g., credit, card, loan, bank, etc.) appears close on the plot.

Phase 3: Modeling and Error Analysis

Machine Learning Techniques

3.1. Introduction:

In this phase, we will focus on the traditional Machine Learning techniques used for classifications such as Logistic Regression, Naïve Bayes, Decision Trees, SVM and so on. These models will serve as a baseline for comparing the performance with advance models that will be analyzed in next phase i.e., LSTM, BERT, etc.

Machine Learning modeling will be divided into two parts. In first part we will perform modelling on Numerical features which we have created in last phase while feature engineering and in second part we will be working with text data, indeed, in order to use machine learning algorithms on texts, a mathematical representation of these texts is required which we will achieve using word embedding techniques like Count Vectorization and TF-IDF.

3.2. Methodology:

All the models presented will be tested in three different ways. To be more precise, in the first case in the first case, the models will be trained on a training set and finally tested using test set. In second case two different datasets will be used just to test the accuracy of models.

3.3. Algorithms:

3.3.1. Logistic Regression [24]

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes.

Linear Regression Equation,

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

Where, y is dependent variable and x1, x2 ... and Xn are explanatory variables.

Sigmoid Function:

$$p = 1/1 + e^{-y}$$

Apply Sigmoid function on linear regression:

$$p = 1/1 + e^{-(\beta 0 + \beta 1 X1 + \beta 2 X2 ..... \beta n Xn)}$$

Pros:

1. Logistic Regression is very efficient, easy to implement, interpretable, requires very less computation power and very straightforward in nature.
2. It does not require feature scaling.
3. It provides a probability score.

Cons:

1. It cannot handle large number of categorical features.
2. Vulnerable to overfitting.
3. Can't handle non-linear data. Therefore, we need feature transformation.
4. LR works well when features are correlated with target variable. It does not perform well when features are correlated to each other.

3.3.2. KNN [25]

The K-Nearest Neighbor is a supervised machine learning algorithm which works on assumption that similar thing exists close to each other. Hence, the K-NN predict values based on similarity between training data and new data points.

Pros:

1. KNN is much faster to train.
2. There is no need to train a model for generalization. It performs well even on non-linear data.

Cons:

1. The testing phase of K-nearest neighbor classification is slower and costlier in terms of time and memory.
2. Scaling of feature is important as KNN works on distance metrics.

3.3.3. Linear SVM [26]

Linear SVM is a method for large linear classification. Given pairs of features-label (xi , yi), yi ∈ {−1, 1}, it solves the following unconstrained optimization problem.

$$\min_{w} \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + \mathbf{C}\sum_{i=1}^{l}\xi(\mathbf{w}; \mathbf{x_i}, y_u)$$

Where ξ is a loss function, in this case L2 loss function has been used, and C > 0 a penalty parameter.

Pros:

1. Effective model in high-dimensional spaces.
2. it is memory efficient as it stores only support vector for prediction.

Cons:

1. Sensitive to the choice of the kernel parameters.
2. There is no direct probabilistic interpretation.

3.3.4. Random Forest [27]

Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.

Each Decision tree works by recursively selecting features and splitting the dataset on those features. These features can either be nominal or continuous.

In order to find the best split, I have used gini impurity,

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i))$$

Where p(i) is the probability of class 'i' in the current branch. The best split is chosen as the one that decreases the most of impurity.

Pros:

1. The random forest classifier doesn't face the overfitting issue
2. This algorithm provides relative feature importance

Cons:

1. It is slow and not efficient for real time prediction.
2. Difficult to interpret.

3.3.5. Gradient Boosting [28] [29]

Gradient Boosting is a machine learning algorithm, used for both classification and regression problems. It works on the principle that many weak learners (e.g., shallow trees) can together make a more accurate predictor. The goal is to find some function that best approximates the output variable from the values of input variables.

$$F_m(x) = F_{m-1}(x) + \underset{h_m \in \mathcal{H}}{\arg\min} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right],$$

where $h_m \in \mathcal{H}$ is a base learner function.

Unfortunately, choosing the best function h at each step for an arbitrary loss function L is a computationally infeasible optimization problem.

Pros:

1. Create a new tree at a time which correct error of last tree.
2. Usually performs better than Random Forest.

Cons:

1. Model becomes very expensive as it creates one tree at a time.
2. Harder to tune hyperparameter and more prone to overfitting.

3.3.6. XGBoost (Extreme Gradient Boosting) [30]

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model.

Pros:

1. XGBoost is reliant on the performance of a model and computational speed.
2. It provides various benefits, such as parallelization, distributed computing, cache optimization, and out-of-core computing.

Cons:

1. Expensive to build the model.
2. The method is too dependent on outliers.

3.4. Models on Numerical Features

Before moving to actual model building, it is very important to scale the data and handle categorical features. Here, I am using OnHotEncoding to encode categorical feature and StandardScaler for data scaling.

OneHotEncoding:

It is a method of converting categorical feature into binary value. It creates individual vector for all categories where each vector values are zeros except the rows which belongs to that particular category.

In the dataset, "Sentiment" column contains categorical data. It contains 3 categories of values i.e., "Positive", "Negative" and "Neutral". Therefore, we need to encode this column into separate feature.

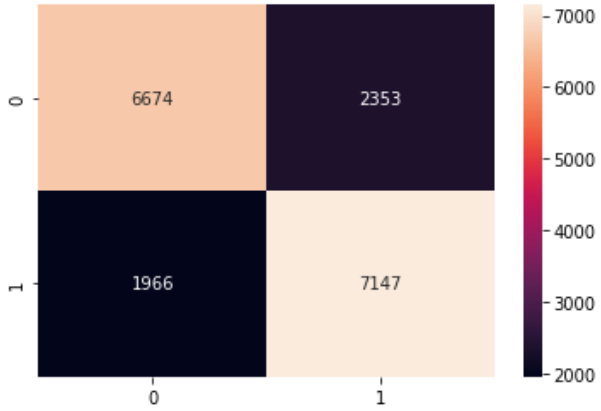
StandarScaler:

It is a technique of removing the mean i.e., 0 and scaling each feature to unit variance i.e., 1.

1. **Logistic Regression**

   In the case of Logistic Regression there was three parameters to tune up, which is the penalty parameter, max_iter (Maximum number or iteration) and C (inverse regularization strength).

   Below is the confusion matrix of train-test set after hyperparameter tuning.



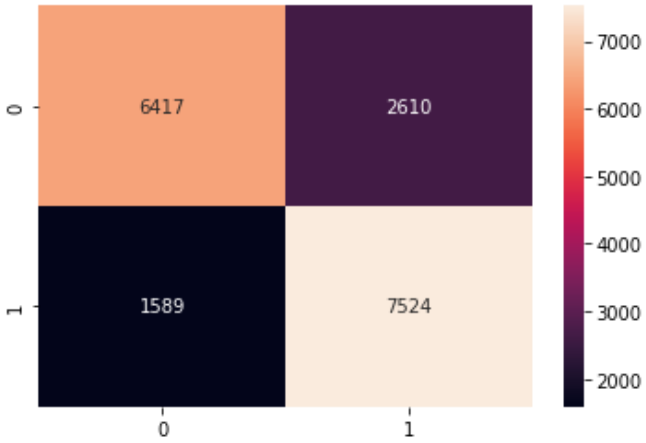
Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	76.19	75.23	78.43	75.55
Test_Data1	56.77	49.11	71.67	58.28
Test_Dataset2	61.50	62.13	61.50	61.81

2. **KNN**

In this case we have only on parameter to tune i.e., n_neighbors (nearest neighbors). The best value of nearest neighbor was 5.

Below is the confusion matrix of train-test set after hyperparameter tuning.



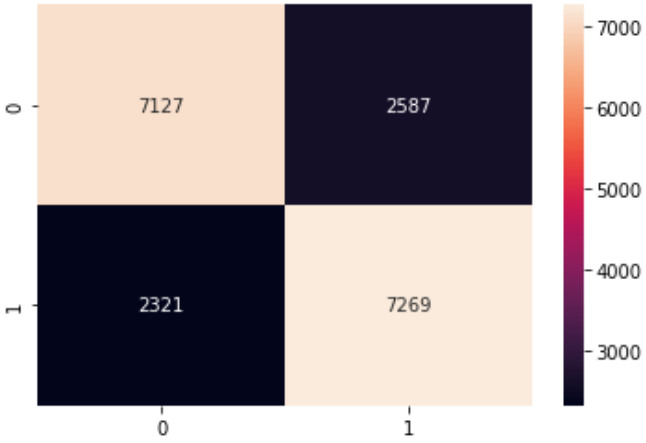
Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	76.85	74.25	82.56	78.18
Test_Data1	61.50	52.97	77.01	62.77
Test_Dataset2	70.92	72.34	67.87	70.03

3. **Linear SVC**

In the case of linear SVC there is one parameter to tune up i.e., C = 1 value and max_iter = 1500.

Below is the confusion matrix of train-test set after hyperparameter tuning.



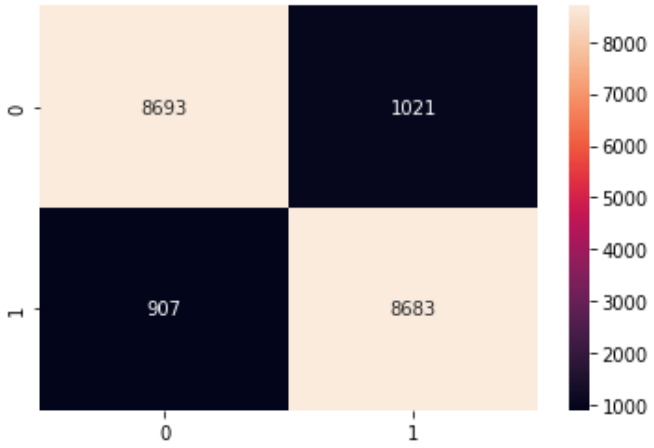
Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	74.58	73.75	75.80	74.76
Test_Data1	56.42	51.27	67.22	58.17
Test_Dataset2	61.84	62.51	59.48	60.96

4. Random Forest

With trees, it is possible to reduce overfitting by pruning the tree. We can do either pre-pruning or post pruning. Pre-pruning means that a stopping criterion is used to stop tree growth earlier and post pruning cut the tree once it has been fully grown. It this case pre-pruning is done by limiting the maximum depth of the tree.
In this model I am tuning three parameters i.e., n_estimators=210, max_depth=27 and criterion='entropy'

Below is the confusion matrix of train-test set after hyperparameter tuning.



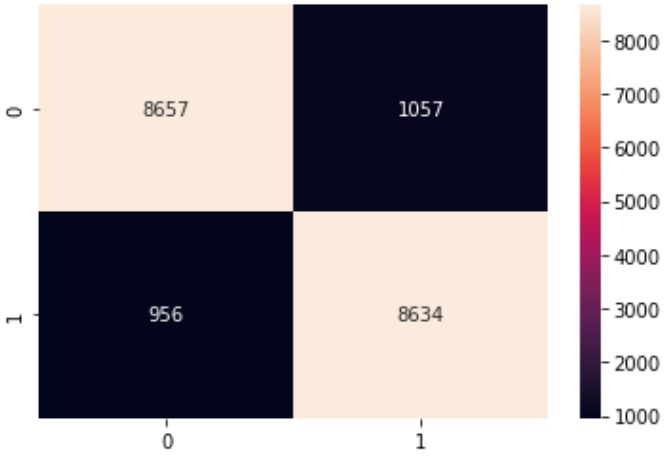
Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	90.09	89.49	90.71	90.09
Test_Data1	62.69	57.01	70.09	62.88
Test_Dataset2	77.47	89.14	62.63	73.57

5. Gradient Boosting

Same as Random Forest, we will per pruning in this model as well. In this case we will tune 4 parameters, those are n_estimators=170, learning_rate=0.75, max_depth=27 and min_samples_leaf = 2.

Below is the confusion matrix of train-test set after hyperparameter tuning.



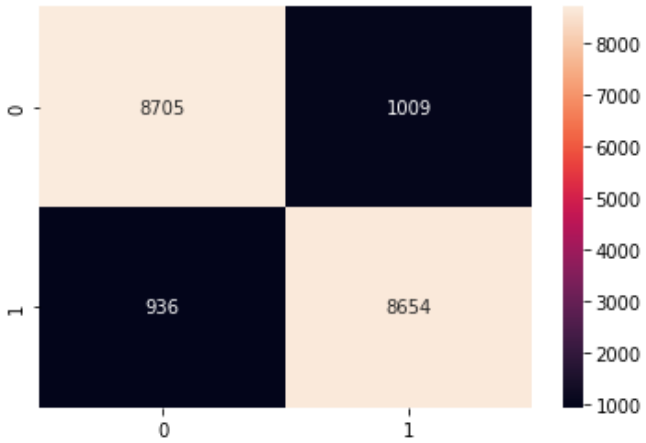
Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	89.57	89.09	90.03	89.56
Test_Data1	61.74	56.05	70.22	62.34
Test_Dataset2	77.81	89.07	63.45	74.11

6. XGBoost

In case of XGBoost we are using three features for tuning including pre-pruning. Tuned parameters are learning_rate =0.1, n_estimators=210, and max_depth=19. This could be tuned further with more hyperparameters but because of lack of compute resource I am not fine tuning as of now.

Below is the confusion matrix of train-test set after hyperparameter tuning.



Model Performance:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Training Data	89.92	89.56	90.24	89.90
Test_Data1	61.12	55.58	68.59	61.41
Test_Dataset2	77.24	88.82	62.38	73.29

Result of Machine Learning Algorithms on Numerical Features:

Algorithms	Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Logistic Regression	Training Data	76.19	75.23	78.43	76.80
	Test_Data1	56.77	49.11	71.67	58.28
	Test_Dataset2	61.50	62.13	61.50	61.81
KNN	Training Data	76.85	74.25	82.56	78.18
	Test_Data1	61.50	52.97	77.01	62.77
	Test_Dataset2	65.76	65.77	67.61	66.68
SVM	Training Data	75.96	75.01	78.22	76.58
	Test_Data1	56.65	49.02	71.18	58.05
	Test_Dataset2	61.29	61.90	61.43	61.66
Passive Aggressive	Training Data	71.57	71.93	71.18	71.56
	Test_Data1	51.56	43.71	51.92	47.46
	Test_Dataset2	56.04	58.17	47.16	52.09
Random Forest	Training Data	90.67	90.16	91.40	90.77
	Test_Data1	63.59	55.01	74.64	63.34
	Test_Dataset2	76.71	87.79	62.77	73.20
Gradient Boosting	Training Data	90.27	89.79	90.98	90.38
	Test_Data1	62.25	53.79	73.93	62.27
	Test_Dataset2	77.50	88.78	63.65	74.14
XGBoost Classifier	Training Data	90.63	90.41	91.00	90.70
	Test_Data1	61.22	52.92	72.48	61.17
	Test_Dataset2	76.14	87.24	61.97	72.47

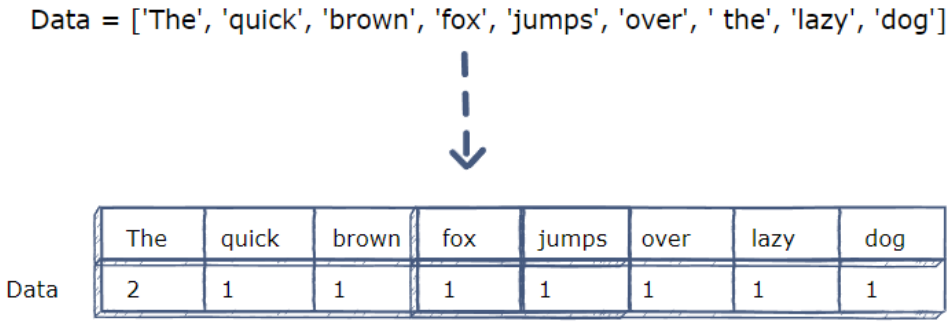
From the above performance report, we can infer that tree-based algorithms like **Random Forest, Gradient Boosting and XGBoost** performing so well on datasets. However, Random Forest an ensemble-based model out performing over all other models. There is very less performance difference between Random Forest and Boosting algorithms. I believe fine tuning XGBoost can give more better result.

3.5. Models on Embedded features:

To fit model on text features, it is very important to textual data into machine understandable language i.e., numeric vector. This is achieved by word embedding.

Word Embedding: Word embedding [7] is a form of word representation that is used to bridge the gap between human language and machine. It represents the words, phrases or vectors in n-dimensional space called as vectors or encoded vectors. Word embedding is an approach which learns from corpus of data and finds the encoded vectors for each word in such a manner that words with similar meaning or context have similar vector representation. Here in this project, we are using two techniques for embedding and they are:

1. **Count Vectorization:** Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts.



2. **TF-IDF**: In order to compute tf-idf, it is separated in two parts, the first one being term frequency and the second one inverse document frequency. We have that,

$$tf_{ij} = \#(W_j | W_j \in D_i)$$

That it is the number of times the word j appears in the document i. Secondly, we have that

$$idf_j = \log\left(\frac{\#D}{\#(D_i | W_j \in D_i)}\right)$$

this is the log of the total number of documents, over the number of documents that contains the word j. Finally, the value tfidf value is computed by

$$tf - idf_{ij} = tf_{ij} * idf_j$$

Result of Machine Learning Algorithms on TF-IDF vectors:

Algorithms	Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Logistic Regression	Training Data	91.07	90.58	91.25	90.91
	Test_Data1	62.67	57.68	64.61	60.95
	Test_Dataset2	77.71	91.86	60.86	73.22
KNN	Training Data	79.33	74.48	87.88	80.63
	Test_Data1	58.85	53.69	63.50	58.18
	Test_Dataset2	79.98	84.09	74.01	78.73
SVM	Training Data	87.82	87.59	87.50	87.55
	Test_Data1	62.78	57.89	64.01	60.80
	Test_Dataset2	75.82	85.61	62.13	72.00
Passive Aggressive	Training Data	93.07	92.68	93.21	92.94
	Test_Data1	61.30	56.33	62.99	59.47
	Test_Dataset2	78.09	95.79	58.81	72.88
Random Forest	Training Data	93.27	92.20	94.22	93.20
	Test_Data1	65.43	59.83	70.99	64.93
	Test_Dataset2	81.99	94.99	67.58	78.98
Gradient Boosting	Training Data	88.68	89.89	86.60	88.22
	Test_Data1	62.69	57.27	67.91	62.14
	Test_Dataset2	72.42	86.29	53.39	65.97
XGBoost Classifier	Training Data	92.45	93.13	91.31	92.21
	Test_Data1	63.55	58.37	66.84	62.32
	Test_Dataset2	74.89	94.84	52.70	67.75

From the above performance report, we can see Random Forest outperforming over all other algorithms. Passive Aggressive algorithm providing accuracy similar to Random Forest on test set. However, Random Forest is performing well on all test datasets.

Result of Machine Learning Algorithms on TF-IDF values + Numeric Features:

Algorithms	Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Logistic Regression	Training Data	91.71	91.18	91.97	91.57
	Test_Data1	62.82	57.71	65.64	61.42
	Test_Dataset2	78.18	91.28	62.38	74.11
KNN	Training Data	78.77	74.67	85.71	79.81
	Test_Data1	62.76	56.69	73.81	64.13
	Test_Dataset2	70.89	71.85	68.81	70.30
SVM	Training Data	88.83	88.57	88.61	88.59
	Test_Data1	62.80	57.56	66.62	61.76
	Test_Dataset2	74.54	83.75	60.96	70.56
Passive Aggressive	Training Data	93.28	93.44	92.79	93.11
	Test_Data1	61.89	56.79	64.78	60.52
	Test_Dataset2	80.28	95.81	63.39	76.30
Random Forest	Training Data	93.44	92.59	94.13	93.35
	Test_Data1	66.66	60.92	72.70	66.29
	Test_Dataset2	82.49	94.87	68.75	79.72
Gradient Boosting	Training Data	90.03	90.04	89.54	89.79
	Test_Data1	63.01	57.56	68.42	62.52
	Test_Dataset2	73.64	87.10	55.57	67.85
XGBoost Classifier	Training Data	93.68	93.57	93.51	93.54
	Test_Data1	63.11	57.76	67.65	62.32
	Test_Dataset2	75.56	95.67	53.61	68.71

Similar as above Random Forest is performing better than all other algorithm when applied on combined features of TFIDF values and Numeric Features.

Result of Machine Learning Algorithms on Count Vectorization:

Algorithms	Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Logistic Regression	Training Data	93.49	93.65	93.22	93.44
	Test_Data1	60.89	56.04	61.53	58.66
	Test_Dataset2	76.26	95.77	55.00	69.87
KNN	Training Data	74.07	78.18	66.31	71.76
	Test_Data1	62.67	59.50	53.87	56.55
	Test_Dataset2	71.60	82.09	55.35	66.11
SVC	Training Data	92.82	94.12	91.25	92.66
	Test_Data1	61.62	56.80	62.22	59.38
	Test_Dataset2	75.56	95.82	53.52	68.68
Passive Regressive	Training Data	90.46	89.60	91.41	90.49
	Test_Data1	58.52	53.48	61.45	57.19
	Test_Dataset2	78.44	91.96	62.38	74.33
Random Forest	Training Data	93.37	92.13	94.74	93.42
	Test_Data1	63.92	58.18	71.07	63.98
	Test_Dataset2	83.19	94.43	70.58	80.78
Gradient Boosting	Training Data	88.80	90.83	86.14	88.42
	Test_Data1	62.38	57.07	66.84	61.57
	Test_Dataset2	71.43	86.37	50.96	64.10
XGBoost Classifier	Training Data	91.84	93.26	90.08	91.65
	Test_Data1	63.17	57.97	66.62	61.99
	Test_Dataset2	72.52	93.23	48.63	63.92

From above observation we can clearly see Random Forest is performing best of all other algorithms when applied on Count Vectorized data.

Result of Machine Learning Algorithms on Count Vectorization + Numerical Features:

Algorithms	Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Logistic Regression	Training Data	93.60	93.48	93.62	93.55
	Test_Data1	61.64	56.76	62.69	59.58
	Test_Dataset2	76.42	96.05	55.16	70.07
KNN	Training Data	75.59	77.93	70.95	74.28
	Test_Data1	60.66	56.75	53.62	55.14
	Test_Dataset2	71.82	82.23	55.76	66.45
SVC	Training Data	93.14	93.91	92.16	93.03
	Test_Data1	62.05	57.18	63.07	59.98
	Test_Dataset2	75.93	96.08	54.12	69.24
Passive Regressive	Training Data	91.10	92.66	89.14	90.87
	Test_Data1	60.20	55.31	61.10	58.06
	Test_Dataset2	76.18	93.64	56.23	70.27
Random Forest	Training Data	93.52	92.27	94.91	93.57
	Test_Data1	66.80	60.87	73.81	66.72
	Test_Dataset2	88.63	94.42	71.52	81.39
Gradient Boosting	Training Data	89.83	90.42	88.96	89.68
	Test_Data1	62.96	57.50	68.38	62.47
	Test_Dataset2	73.01	86.15	54.90	67.06
XGBoost Classifier	Training Data	93.90	93.80	93.92	93.86
	Test_Data1	63.28	58.12	66.45	62.01
	Test_Dataset2	75.82	96.48	53.64	68.95

Here count vectorized text data are merged with numeric features and then machine learning algorithms are applied over it. We can see XGBoost, Logistic Regression and Random Forest are very close to each other in performance. However, Random Forest is performing better than other algorithm when tested on other two testing datasets.

Conclusion:

When various machine learning algorithms are applied on dataset features, we can see there are many algorithms which are performing better on test set which is created during train-test split. However, performance of models on test dataset is deciding factor in this case. We can clearly see Random Forest classifier is performing better than any other machine learning algorithm in call cases. Random Forest has given 93+% accuracy in cases, however highest accuracy of RF is seen when model is applied on combined features of Count Vector and Numeric Features.

Phase 4: Advanced Modeling and Feature Engineering

Deep Learning Techniques

4.1. Introduction:

In this phase of project development, we will deal with Deep Learning algorithms. We will train our Deep Learning RNN model on our training text data to solve our real-world problem.

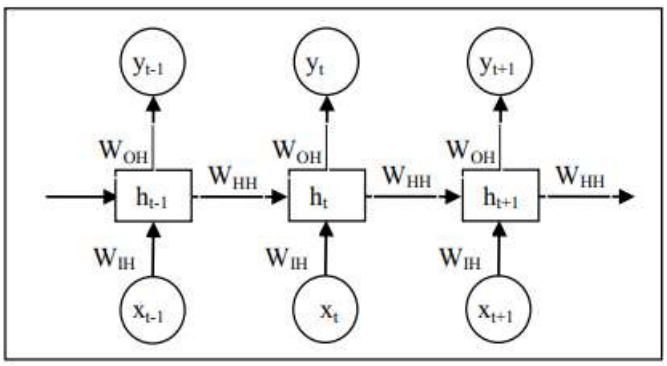
Deep Learning, on the other hand, is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.

We will be using RNN based Deep Learning model i.e., LSTMS as we are dealing with text classification problem. To understand LSTM's, it is important to know about RNN.

4.2. Recurrent Neural Network (RNN):

Recurrent Neural Network is a feed-forward artificial neural network. RNNs handle a variable-length sequence input by comprising a recurrent hidden layer whose activation at each time is dependent on the previous time. Hence RNNs are better choice for long-distance contextual information. A basic RNN can be formalized as follows: given an input sequence (x1,x2,…,xT) and for each timestamp, the model update the hidden state (h1,h2,…,hT) and output (y1, y2,…,yT).

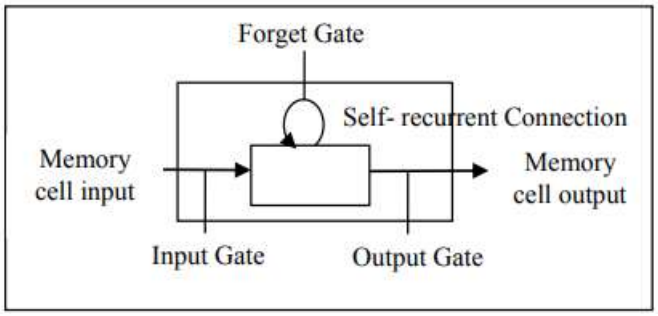
Below figure shows the general architecture of basic RNN. The vectors xt and yt are input and output vector at timestamp t. The three connection weight matrices are WIH, WHH and WOH represent the weight corresponding to input, hidden and output vectors respectively



General Architecture of RNN [1]

4.3. Long Short-Term Memory (LSTM) - Recurrent Neural Network [31]:

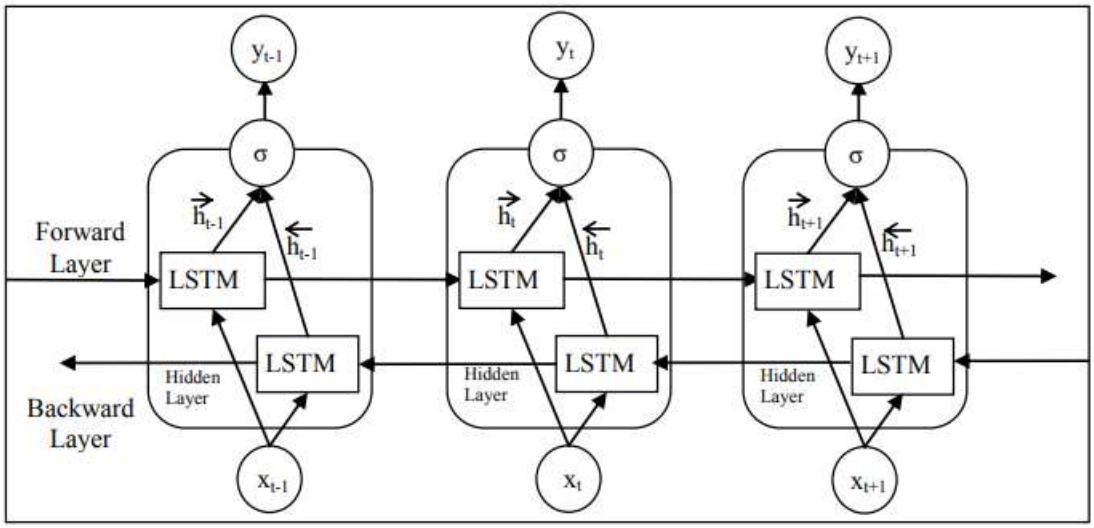
Long Short-Term Memory networks (LSTM) are a special type of RNN competent in learning long-term dependencies. LSTM is a very effective solution for addressing the vanishing gradient problem. In LSTM-RNN the hidden layer of basic RNN is replaced by an LSTM cell.



Structure of LSTM cell [2]

4.4. Bi-directional Long Short-Term Memory – RNN [32]:

LSTMs help to preserve the error that can be back-propagated through time and in lower layers of a deep network. Bi-directional processing is an evident approach for a large text sequence prediction and text classification. As shown in below figure, a Bi-Directional LSTM network steps through the input sequence in both directions at the same time.



General architecture of Bi-directional LSTM- RNN [2]

4.5. Proposed Work:

The proposed fake news detection model based on LSTMs are shown below. The news articles are first split into lists of words and then tokenized using Tokenizer. Tokenized list of articles is turned into space-separated padded sequences of tokens.

Word2Vec: Gensim is designed for data streaming, handle large text collections and efficient incremental algorithms or in simple language – Gensim is designed to extract semantic topics from documents automatically in the most efficient and effortless manner. Gensim word2vec model has been trained on the text articles with window of 5 words.

The embedding layer will load the weights from word2vec model instead of loading random weights. Gensim applies globally aggregated co-occurrence statistics across all words in the news article corpus. The resulting representations formalize significant linear substructures of the word vector space. The transformed vector represented data is partitioned into train, validation and test data. The training is carried out on the news article corpus. Validation data set is used for fine-tuning the model. Further, the test data is used to know the predicted label of news article based on trained model. The model selection among variation of RNN as LSTM-RNN, and Bi-directional LSTM-RNN is carried out to detect fake news.

4.6. Results: LSTM:

Architecture:

Layer (type)	Output Shape	Param #
embedding_13 (Embedding)	(None, 500, 500)	146289500
dropout_36 (Dropout)	(None, 500, 500)	0
lstm_13 (LSTM)	(None, 512)	2074624
dropout_37 (Dropout)	(None, 512)	0
dense_23 (Dense)	(None, 128)	65664
dropout_38 (Dropout)	(None, 128)	0
dense_24 (Dense)	(None, 1)	129

Total params: 148,429,917
Trainable params: 2,140,417
Non-trainable params: 146,289,500

Result:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Test Set	94.28	92.83	96.08	94.43
Test_Data1	63.09	57.99	65.85	61.67
Test_Dataset2	78.47	96.21	59.32	73.39

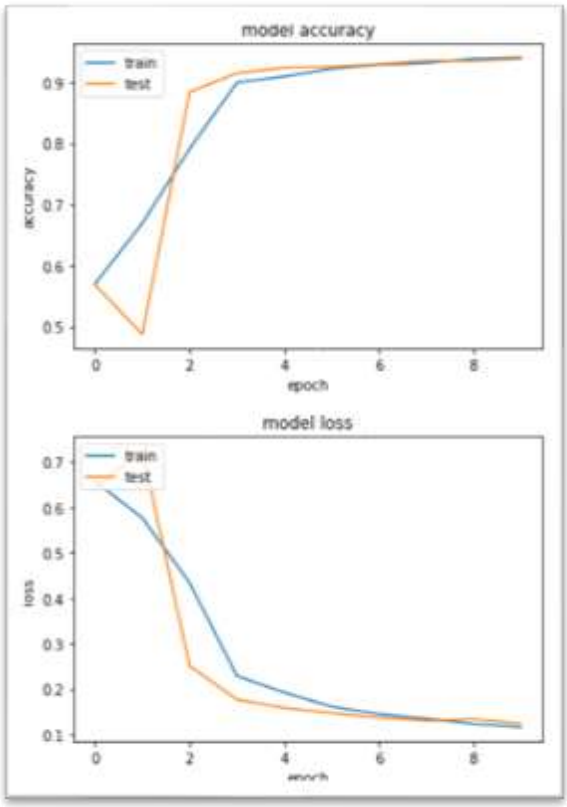
Inference:

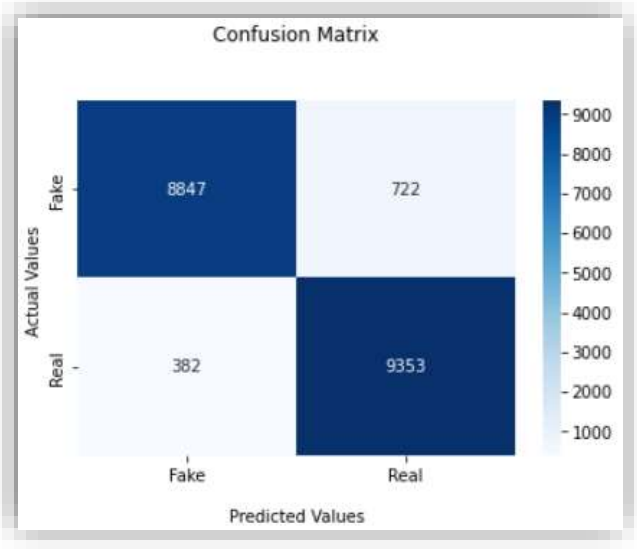
We can observe that LSTM model is give an accuracy of 94.28% on test set. We can also see the precision and recall of test set if quite close in result. There is no much difference. So, we can say out LSTM model is working well on the given data.

Moving further to the test dataset we can see LSTM model is giving accuracy of 63.09% on first Test dataset. It can also be seen there is sight difference in precision and recall. Recall is higher than precision which means fake news are getting classified as real news more.

Similarly, on second Test dataset accuracy is 78.47%. However, there is huge difference in precision and recall. High precision indicates a greater number of real news are getting classified as fake.

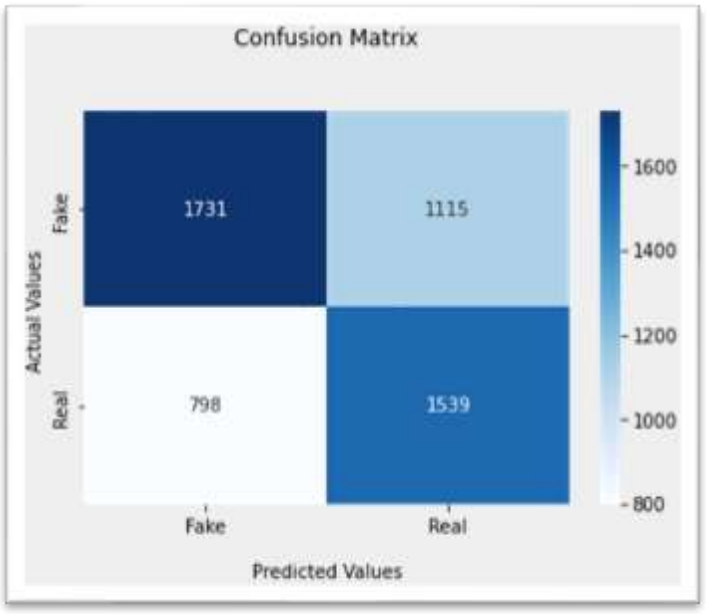
Uni-LSTM Accuracy & loss Vs epoch:





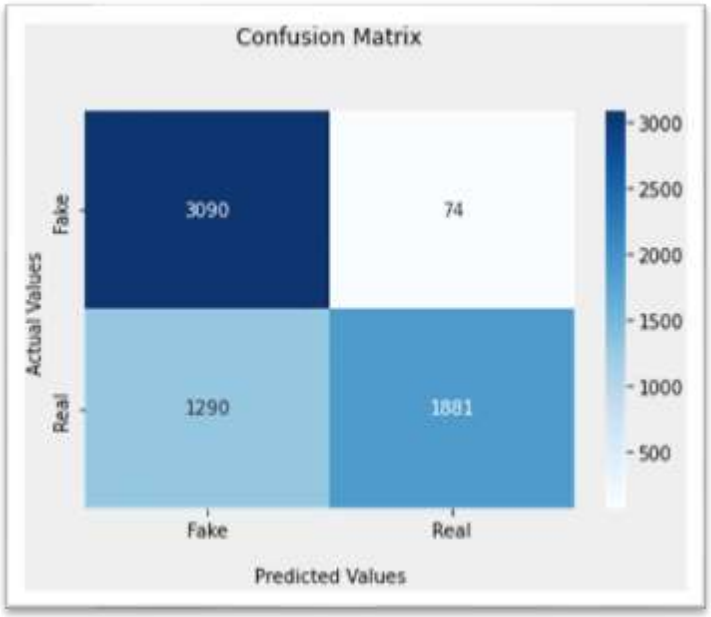
From the above plot of accuracy Vs loss and confusion matrix we can see out model is neither under fitting or over fitting. Classification is not biased. False Positive and False Negative score is balanced.

Test Dataset-1:



From the above confusion matrix, we can infer that the greater number of fake news are incorrectly getting classified as real news.

Test Dataset-2



From the above confusion matrix, it can be seen greater number of real news are incorrectly getting classified as fake news.

Bi-directional LSTM:

Architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, 500, 500)          146289500
_________________________________________________________________
dropout_30 (Dropout)         (None, 500, 500)          0
_________________________________________________________________
bidirectional_10 (Bidirectio (None, 200)               480800
_________________________________________________________________
dropout_31 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_19 (Dense)             (None, 32)                6432
_________________________________________________________________
dropout_32 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 33
=================================================================
Total params: 146,776,765
Trainable params: 487,265
Non-trainable params: 146,289,500
_________________________________________________________________
```

Result:

Dataset	Accuracy %	Precision %	Recall %	F1-Score %
Test Set	93.65	93.71	93.70	93.71
Test_Data1	62.15	57.56	61.10	59.28
Test_Dataset2	71.14	96.09	44.15	60.50

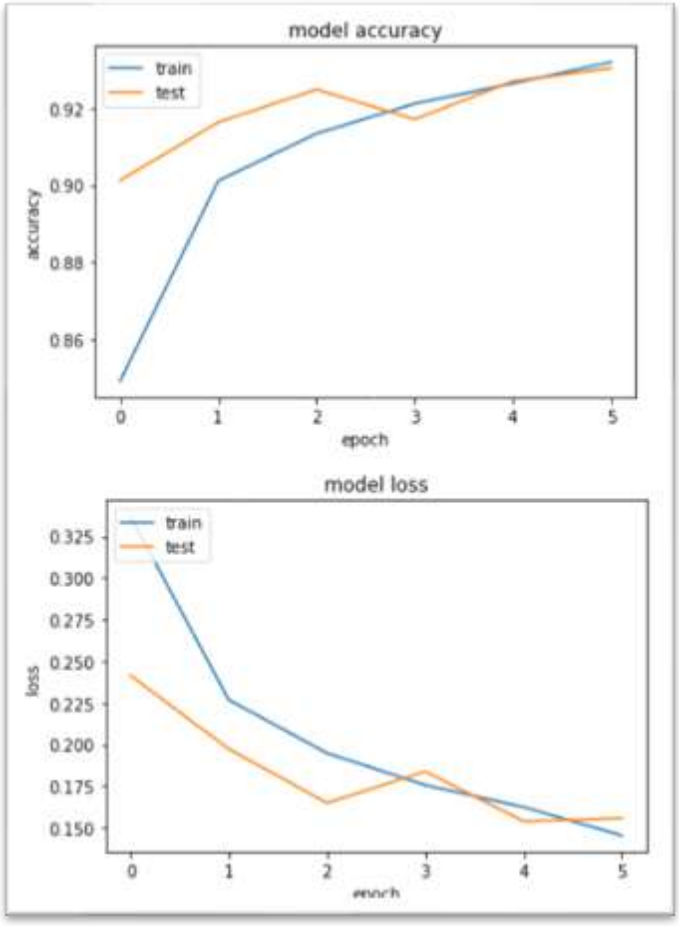
Inference:

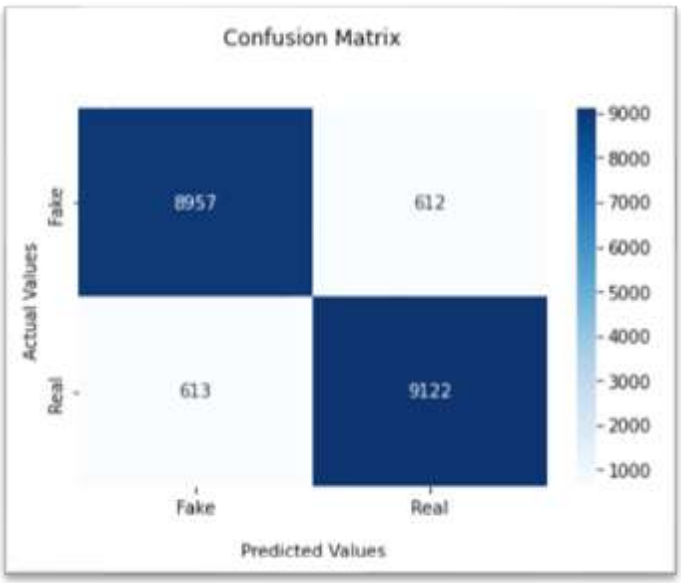
We can observe that LSTM model is give an accuracy of 93.65% on test set. We can also see the precision and recall of test set if very close in result. There is no much difference. So, we can say out Bi-LSTM model is working perfect on the given data.

Moving further to the test dataset we can see LSTM model is giving accuracy of 62.15% on first Test dataset. It can also be seen there is sight difference in precision and recall. Recall is higher than precision which means fake news are getting classified as real news more.

Similarly, on second Test dataset accuracy is 71.14%. However, there is huge difference in precision and recall. High precision indicates a greater number of real news are getting classified as fake.

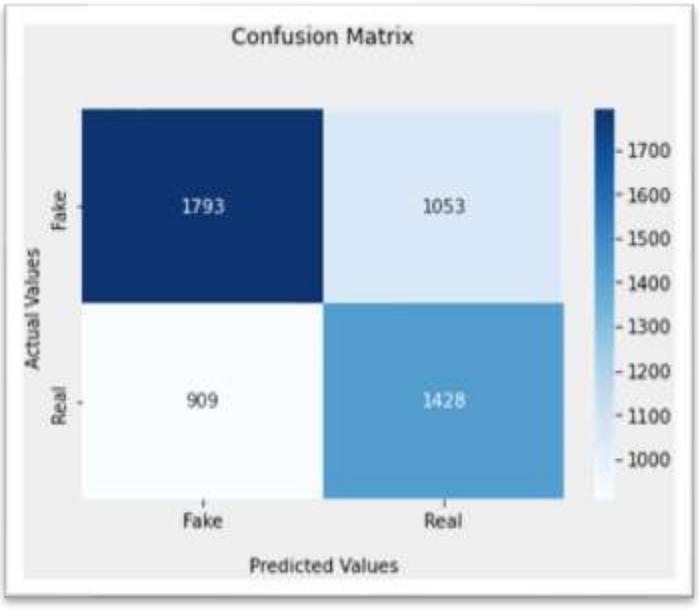
Bi-directional LSTM Accuracy & loss Vs epoch:





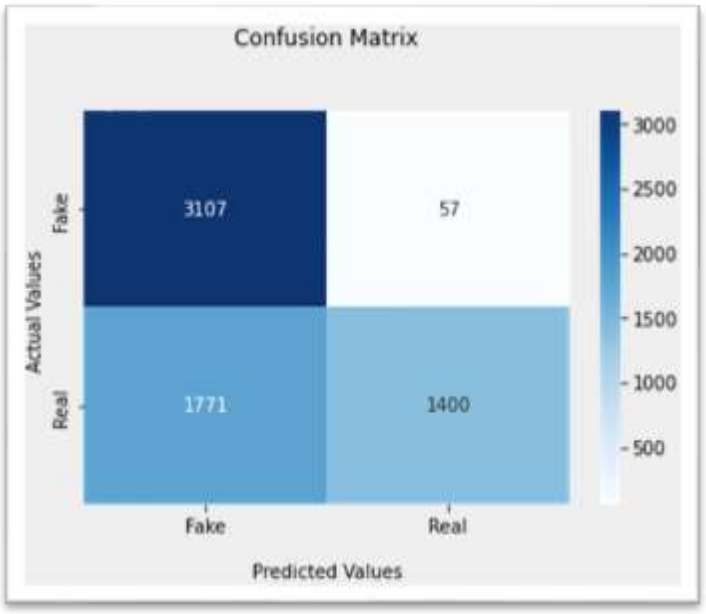
We can see there is good balance in accuracy score, precision and recall. This indicates that LSM model is neither under fitting or over fitting. False Positive and False Negative score is almost balanced.

Test Dataset-1:



From the above confusion matrix, it can be seen that greater number of fake news are incorrectly getting classified as real news.

Test Dataset-2



From the above confusion matrix, it can be seen that greater number of real news are incorrectly getting classified as fake news.

Phase 5: Deployment and Productionization

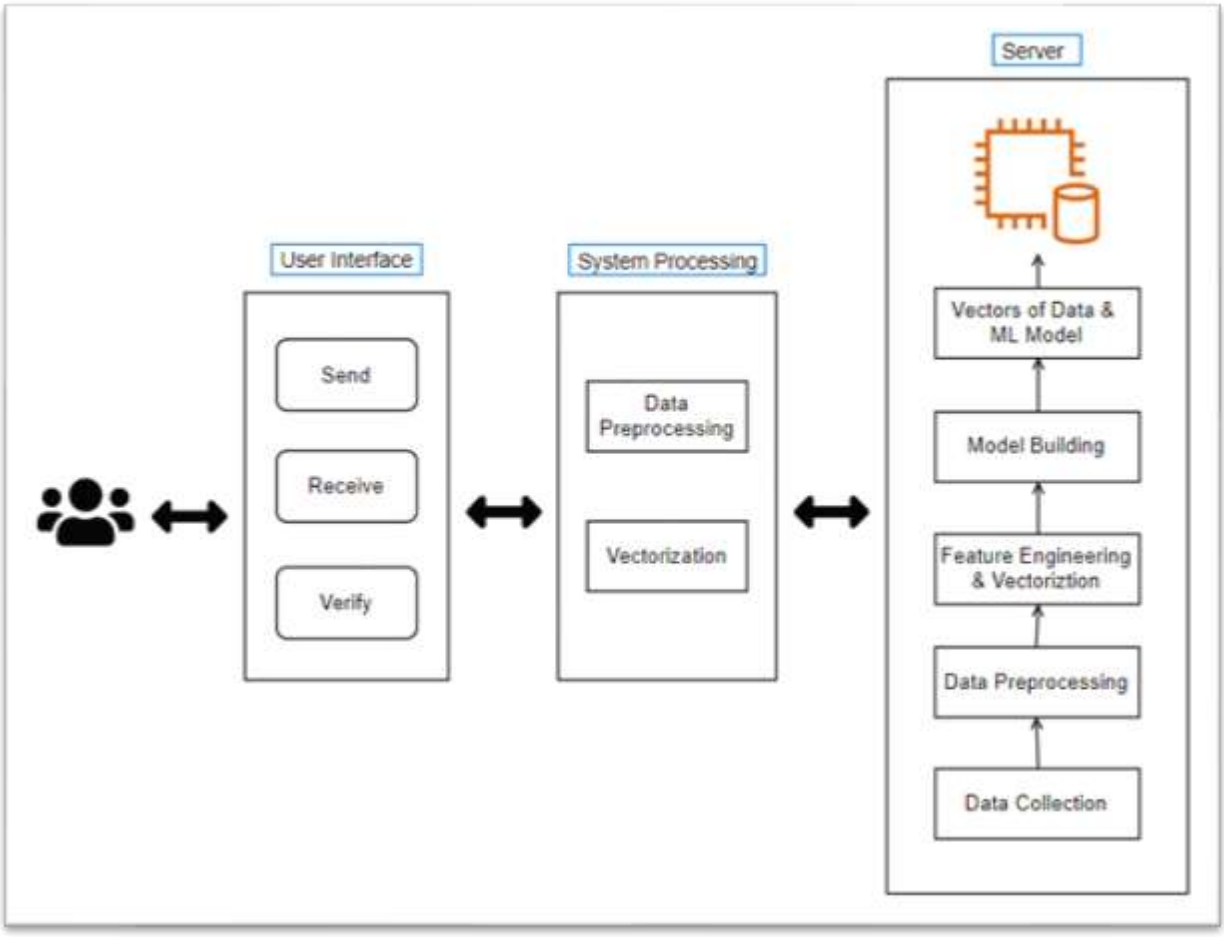
Deployment is the final phase of this project. So, what does productionization mean? It means deploying the whole machine learning pipeline into a production system, into a real-time scenario.

So far,

1. We have collected the data from available sources
2. Performed cleaning of the data
3. Next, we performed data featurization
4. And at stage 4 we have trained different Machine learning and Deep Learning model on top of all those features.

Now as a final stage we need to deploy this pipeline to put of research available to end users for use. The model will be deployed in real world scenario where it takes continuous raw input from real world and classify the article as fake or real.

1. System Architecture:



1. **User application:** This module is nothing but a web application developed using Streamlit. It provides a functionality to the user for entering news article for classification.

2. **System Server:** This module is responsible for identification of received message either 'real' or 'fake' before reverting back the message to the user. Before sending back, the message goes through number of stages as follows:

    - **Data Pre-processing**: We will be removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.
    - **Vectorization:** The converted message into standardized form is transformed into numerical vector using pre-trained machine learning model based on word embedding technique namely Count Vectorization.

3. **Database Server:** This module is responsible for storing the data in vector form. The data gathered from different sources are converted into vectors before storing into the database.
    - **Data Collection:** The data is collected from different sources like Kaggle.
    - **Data Pre-processing**: This module deals with pre-processing of the data. Different operations like tokenization and standardization are performed to convert all the data into standard form.
    - **Data Vectors:** The web embedding techniques like word2vec and USE are used to convert the data in text form into numerical vector form. This transformed data is then stored into the database.

2. Which model to choose for productionization?

When we talk about model deployment the initial question will be which model to deploy? However, in real model selection is completely depends upon business requirement. In this project we will be classifying news article as "fake" or "real" depending upon article content.

As per growing consumption of news content, we want our deployed environment to be

1. Very efficient for real world scenario
2. Requires less computational power
3. Easy to implement and interpret
4. Less time for classification
5. Takes less storage space

In this case I feel Logistic Regression model will be more efficient to use as it stands on above requirements. If you look at the test result conducted in Phase 3, Random Forest and Logistic Regression has almost same accuracy, precision and recall on test set created in train-test-split. However, Random Forest is too heavy as compared to LR (LR model size is 2.3mb whereas RF model size is 770mb).

Therefore, I will be choosing Logistic Regression model for production environment.

3. User Interface:

As part of user interface, I will be developing a Streamlit app. Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. Streamlit allows you to write an app the same way you write a python code.

Let look at the functioning of Streamlit app. Initially, an article input will be given from web interface. Secondly, that input will be cleaned and then featured for prediction. So, for featurization purpose we will be choosing Count Vector and for prediction as we already discussed we will be using LR model.

How do we use Count Vector and LR trained model in Streamlit app?

There are many methods to save and load Count Vector and LR trained model but we will be using "Pickle" module to do so in this project.

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk and then unpickled to retain its properties. Pickling is a way to convert a python object (list, dict, etc.) into a character stream and "Unpickling" is exactly opposite of "Pickling".

Saving/Pickling Count Vector and LR model:

```
import pickle
pickle.dump(log_cls, open("LR_model.pkl", "wb"))
pickle.dump(vectorizer, open("Count_vector.pkl", "wb"))
```

Unpickling Count Vector and LR model:

```
vectorizer = pickle.load(open("Count_vector.pkl", "rb"))
```

```
model = pickle.load(open("LR_model.pkl", "rb"))
```

4. Tools and Libraries:

1. Streamlit	6. Git
2. Heroku	7. Beautiful Soup
3. NLTK	8. Numpy
4. Pickle	9. Scikit learn
5. Re	10. Spyder

5. Deployment:

We will be deploying our project on free tire Heroku platform. Heroku is a common platform-as-a-service solution that enables you to deploy application quickly and without endless infrastructure headaches. It is based on a managed containers (called dynos within the Heroku paradigm) system. It has integrated data services and a powerful ecosystem for deploying and running modern applications.

Steps:

1. Install Git and the Heroku CLI

2. Create requirement.txt file, Procfile file, setup.sh file.
   - **requirements.txt**: Create a requirements.txt file (No capital letters!) and add all the libraries to that file you are using in your python script.
   - **setup.sh**: The next file is the setup.sh file. This is a shell file, if you don't add this file, you will get an error message when you deploy your app to Heroku.
   - **Procfile**: Inside your directory, create a text file and call it "Procfile".

     web: sh setup.sh && streamlit run app.py

     The "web" means that it's a web app. The Procfile pretty much specifies the commands once you run the app on Heroku.

3. Before you can deploy your app to Heroku, initialize a local Git repository and commit your application code to it.

   The following example demonstrates initializing a Git repository for an app that lives in the Deployment directory.

```
(base) C:\Users\Monster>activate spyder_gpu

(spyder_gpu) C:\Users\Monster>cd "Machine Learning\ML Projects\Deployment"
```

```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>git init
Initialized empty Git repository in C:/Users/Monster/Machine Learning/ML Projects/Deployment/.git/
```

```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>git add .
```

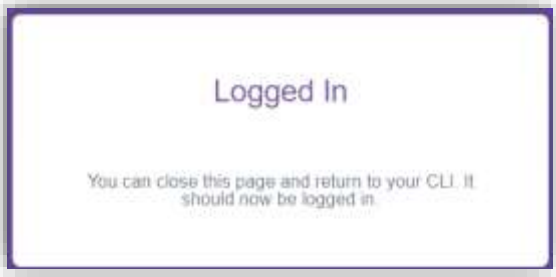
```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>git commit -m "Initial Commit"
[master (root-commit) b1dd81c] Initial Commit
```

4. Login to your Heroku account from the terminal. Run the following command in your terminal:

```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>heroku login
```

You will be redirected to Heroku login page on web. Enter the credentials and login into it.

You will see below screen after successful login and on terminal you will see your email id.



5. Create a new empty Heroku app.

```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>heroku create -a fake--news--classification
Creating ⬚ fake--news--classification... done
https://fake--news--classification.herokuapp.com/ | https://git.heroku.com/fake--news--classification.git
```

```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>git remote -v
heroku  https://git.heroku.com/fake--news--classification.git (fetch)
heroku  https://git.heroku.com/fake--news--classification.git (push)
```

Here I am creating an app called “fake—news—classification” and then confirming remote connection.

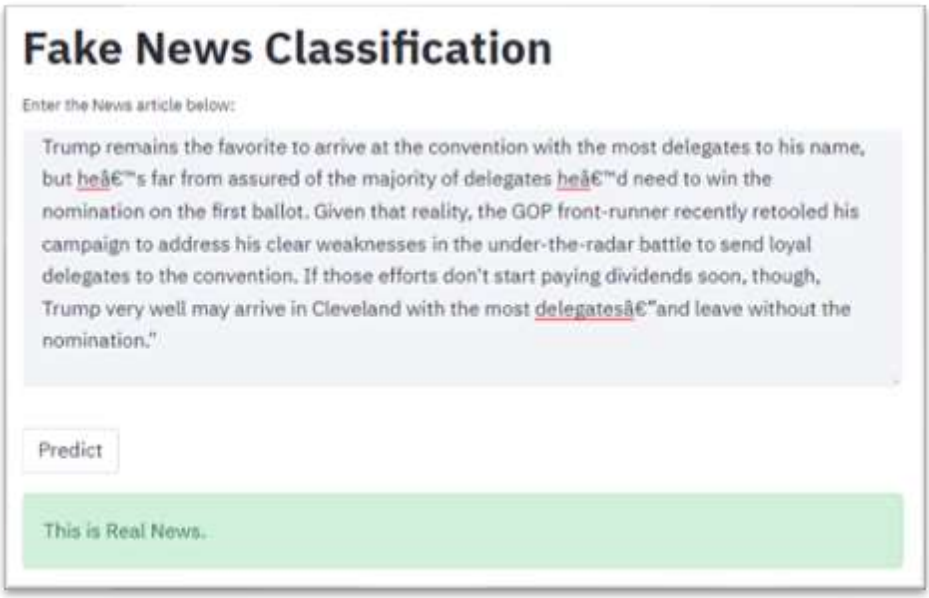
6. The last step is to push our files to Heroku.

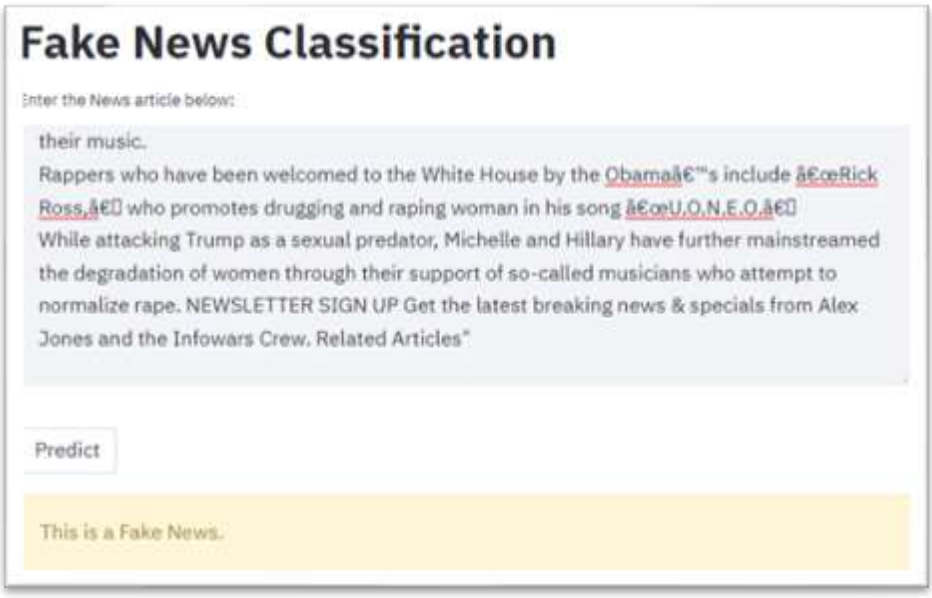
```
(spyder_gpu) C:\Users\Monster\Machine Learning\ML Projects\Deployment>git push heroku master
Enumerating objects: 11, done.
```

Once you executed the command above, it will take a couple of seconds for Heroku to install all the libraries and set up your app. You will see a URL in the output, and that will be the URL of your app.

```
remote: -----> Launching...
remote:        Released v3
remote:        https://fake--news--classification.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/fake--news--classification.git
 * [new branch]      master -> master
```

7. Copy and open the URL on web browser. Below is the screenshot of user interface.





Conclusion:

Result Analysis:

Some hypothesis can be made on why model is working very well on training dataset and does not work well on test datasets.

1. The first thing we can think of is that the original hypothesis on different styles of writing between fake and reliable news is only verified in one dataset, and it is the most logical one.
2. Secondly, Original dataset consist of news articles from US presidential election whereas test datasets consist of news articles of various context.
3. Test datasets is different from training dataset and both belongs to different period of time.

Limitation:

1. Can’t handle all category of data as model is trained on 2016 US political news data.
2. Limited scalability.
3. Data and Security issue
4. Limited storage and bandwidth.
5. Periodic model training is needed for better performance.

Future Scope:

In future work, we would like to expand the research outlined in this thesis. Incorporating more user-independent features will be one important addition. These include demographics, sex, age, place, to learn more about the readers, and their related patterns of reading. We would also like to see if the user features could help us in identifying users who are more likely to believe and spread fake news.

Also, I will be using transformer-based model for classification like BERT. Bidirectional Encoder Representations from Transformers is a transformer-based machine learning technique for natural language processing pre-training developed by Google.

Bibliography:

[1] https://www.visma.com/blog/what-is-fake-news/

[2] https://www.pewresearch.org/journalism/2021/01/12/news-use-across-social-media-platforms

[3] https://timesofindia.indiatimes.com/india/68-of-indian-users-consume-news-on-smartphones

[4] Craig Silverman and Lawrence Alexander. How teens in the balkans are duping trump supporters with fake news. Buzzfeed News, 3, 2016.

[5] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1):22–36, 2017

[6] https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

[7] https://www.kaggle.com/mohit28rawat/fake-news?select=train.csv

[8] https://www.kaggle.com/ahmedsmara/full-fake-news

[9] https://www.kaggle.com/c/fake-news/data?select=test.csv

[10] https://www.kaggle.com/hassanamin/textdb3

[11] https://machinelearningmastery.com/confusion-matrix-machine-learning/

[12] https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

[13] Julio CS Reis, Andr´e Correia, Fabr´ıcio Murai, Adriano Veloso, Fabr´ıcio Benevenuto, and Erik Cambria. Supervised learning for fake news detection. IEEE Intelligent Systems, 34(2):76–81, 2019.

[14] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. Mahway: Lawrence Erlbaum Associates, 71(2001):2001, 2001.

[15] Natali Ruchansky, Sungyong Seo, and Yan Liu. Csi: A hybrid deep model for fake news detection. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pages 797–806. ACM, 2017.

[16] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. Some like it hoax: Automated fake news detection in social networks.

[17] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 24, pages 1953–1961. Curran Associates, Inc., 2011.

[18] Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. Ti-cnn: Convolutional neural networks for fake news detection.

[19] Simon Lorent: Fake News Detection using Machine Learning

[20] TI-CNN: Convolutional Neural Networks for Fake News Detection (arxiv.org/pdf/1806.00749.pdf)

[21] Jozef Kapustaa, Petr Hájekc , Michal Munka , Ľubomír Benko. Comparison of fake and real news based on morphological analysis. Procedia Computer Science 171

[22] Xu, K., Wang, F., Wang, H., and Yang, B. (2018) A First Step Towards Combating Fake News over Online Social Media. in: Springer, Cham, pp. 521–531.

[23] Braşoveanu, A.M.P. and Andonie, R. (2019) Semantic Fake News Detection: A Machine Learning Perspective. in: Springer, Cham, pp. 656–667

[24] Bing Liu. Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1):1–167, 2012.

[24] Understanding Logistic Regression in Python - Avinash Navlani

[25] KNN Classifier For Machine Learning: Everything You Need to Know - Pavan Vadapalli

[26] Fake News Detection Using Machine Learning - Simon Lorent

[27] Random Forest Classifier

[28] An Introduction to Gradient Boosting Decision Trees - Machine Learning Plus

[29] An Introduction to Gradient Boosting Decision Trees - Gaurav

[30] Zhang, Ye & Rahman, Md Mustafizur & Braylan, Alex & Dang, Brandon & Chang, Heng-Lu & Kim, Henna & McNamara, Quinten & Angert, Aaron & Banner, Edward & Khetan, Vivek & McDonnell, Tyler & Nguyen, An & Xu, Dan & Wallace, Byron & Lease, Matthew. (2016). Neural Information Retrieval: A Literature Review

[31] Hassan and A. Mahmood. (2018). "Convolutional Recurrent Deep Learning Model for Sentence Classification." IEEE Access 6:13949- 13957 [18] Hochreiter, J. Schmidhuber. (1997). "Long Short-Term Memory", Neural Computation, 9(8):1735-1780.

[32] K. Greff, R. K.Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber. (2015). "LSTM: A search space odyssey."IEEE Transactions on Neural Networks and Learning Systems