Fake News Detection Assignment

Objective:

The objective of this assignment is to develop a Semantic Classification model. We will be using Word2Vec method to extract the semantic relations from the text and develop a basic understanding of how to train supervised models to categorise text based on its meaning, rather than just syntax. We will explore how this technique is used in situations where understanding textual meaning plays a critical role in making accurate and efficient decisions.

In this assignment, we will develop a Semantic Classification model that uses the Word2Vec method to detect recurring patterns and themes in news articles. Using supervised learning models, the goal is to build a system that classifies news articles as either fake or true.

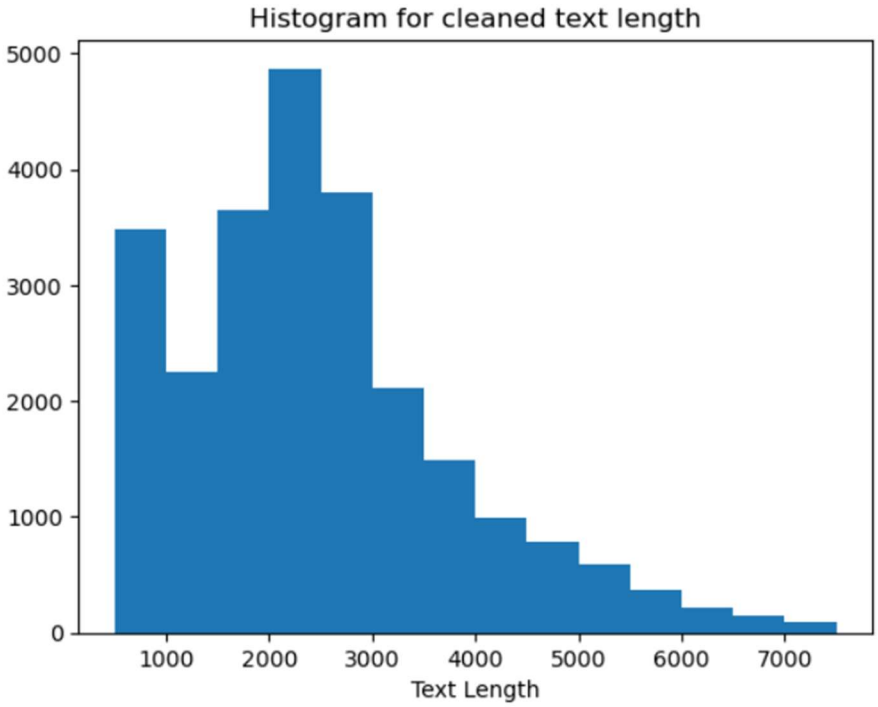
Below is the visual representation/report of the python file and our findings:

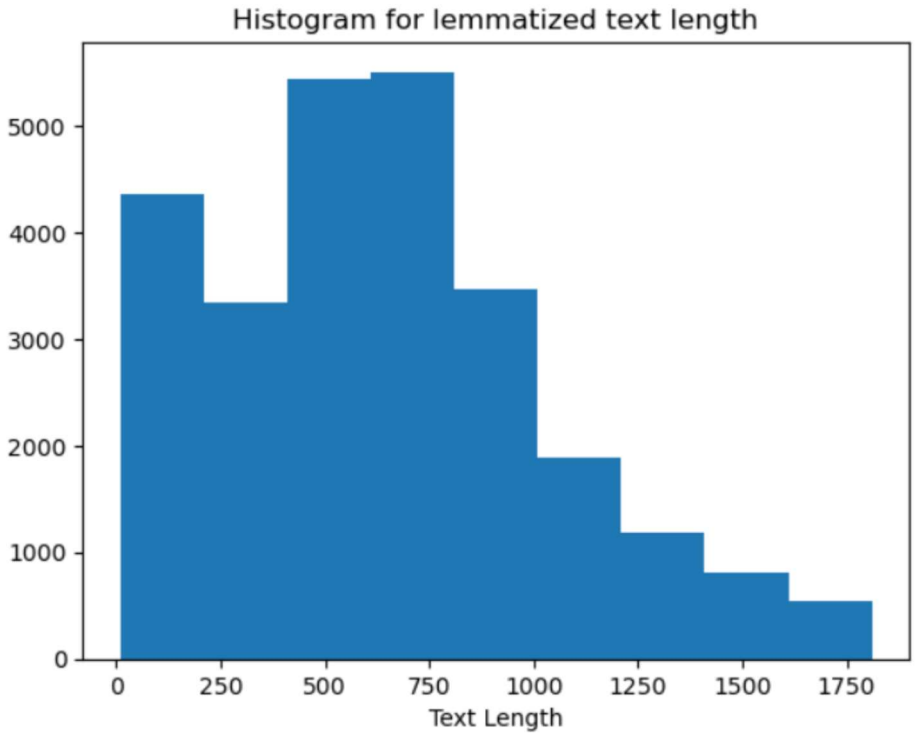
1. Initially we imported the necessary libraries, loaded and prepared the data.

During data preparation, we labelled the dataset with 1 for true news and 0 for fake news. We then checked for null values and removed any missing entries. Following that, we cleaned the text to eliminate irrelevant elements, applied Part-of-Speech (POS) tagging, and performed lemmatization. Finally, we split the data into training and testing sets.

2. Exploratory Data Analysis

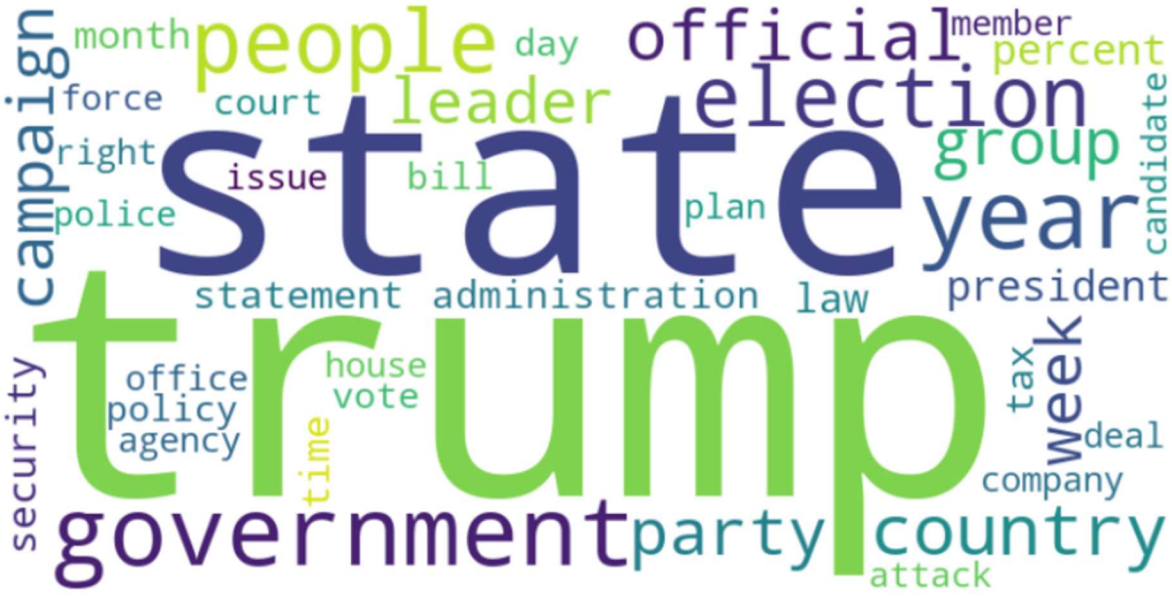
We created Histograms to check the character lengths for cleaned text



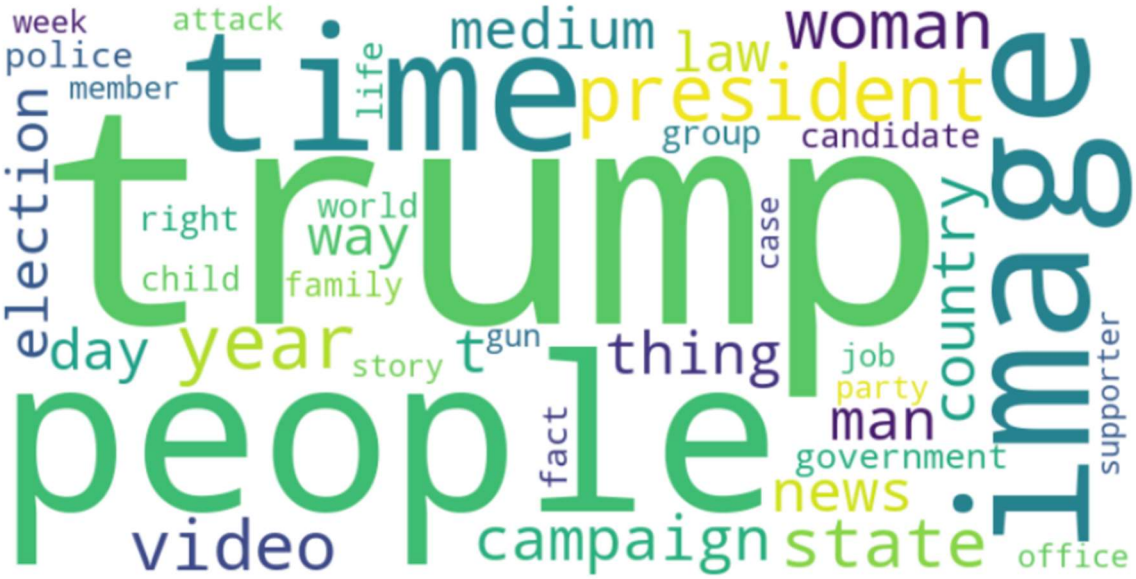


Word cloud for True News:

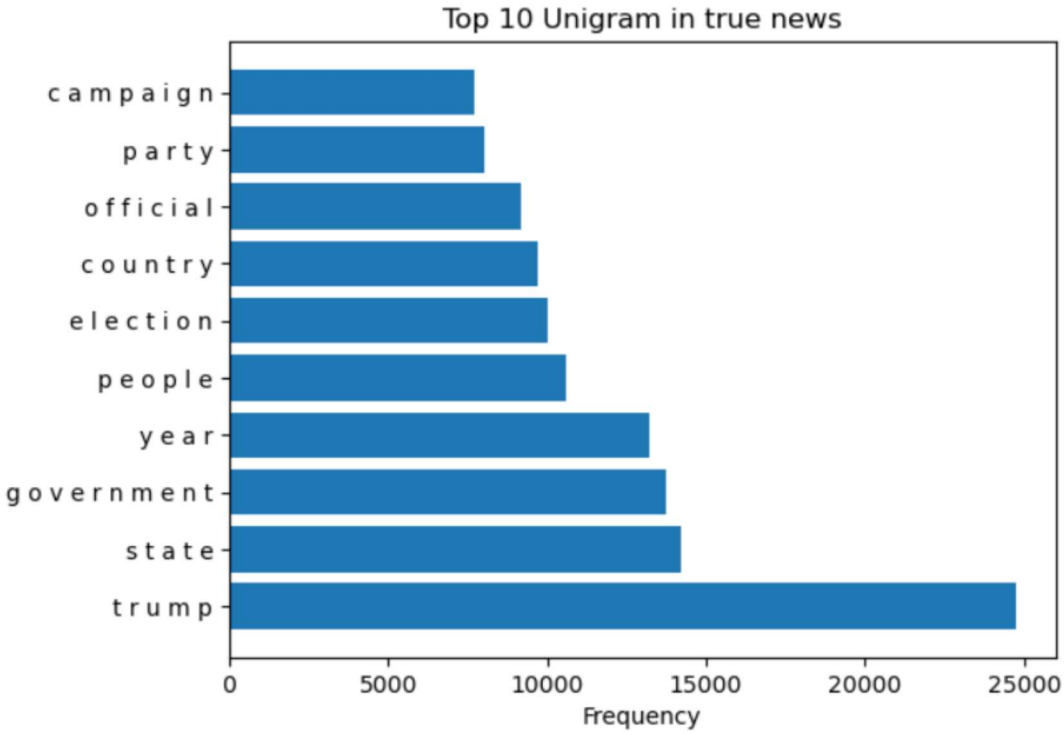
True news word cloud

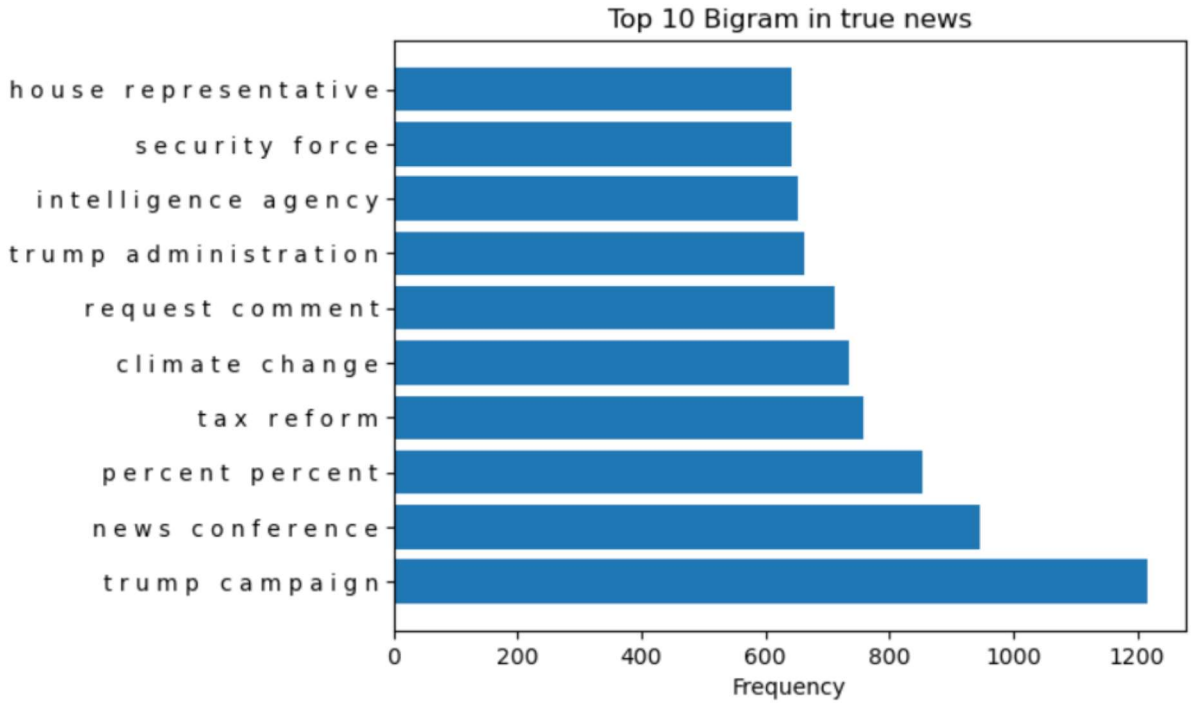


Fake news word cloud

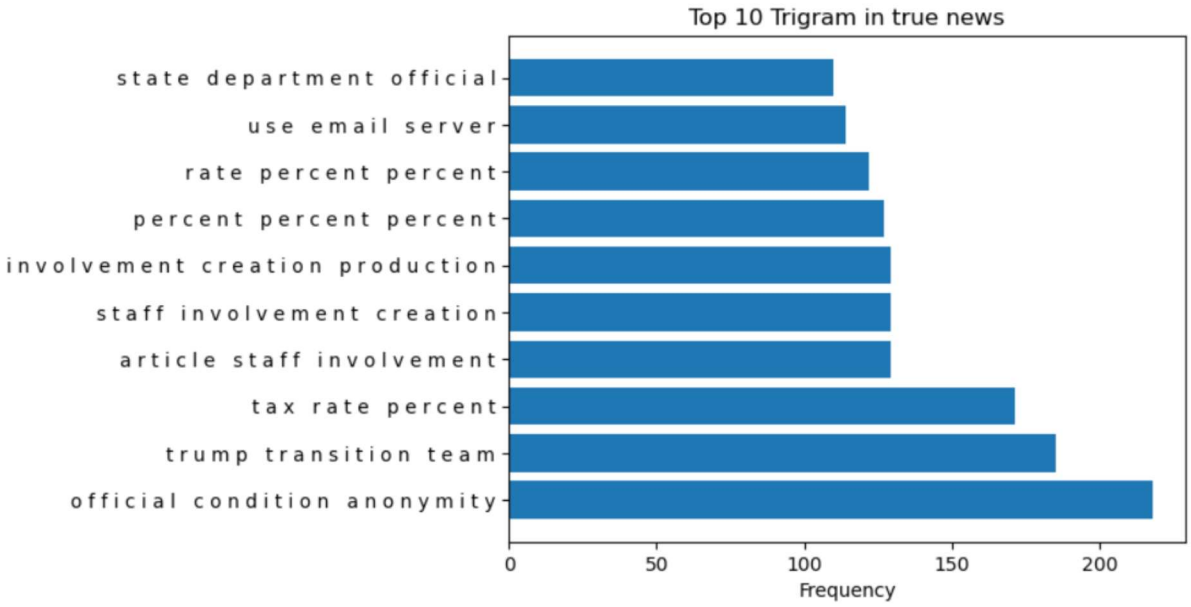


Top ten Unigrams in True news:

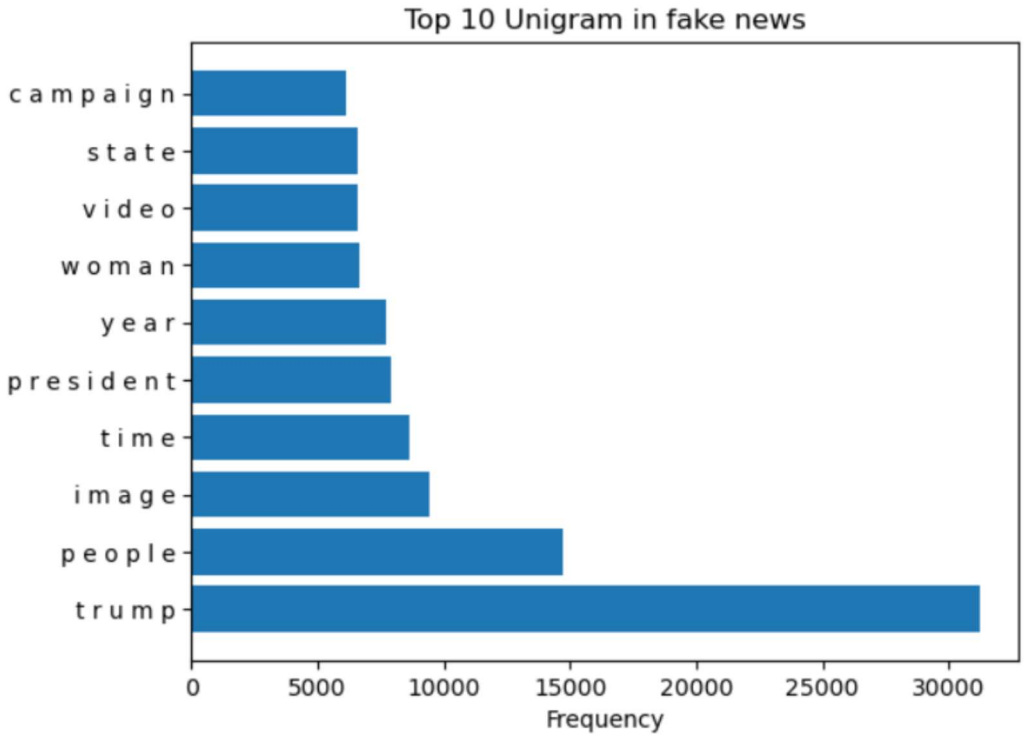




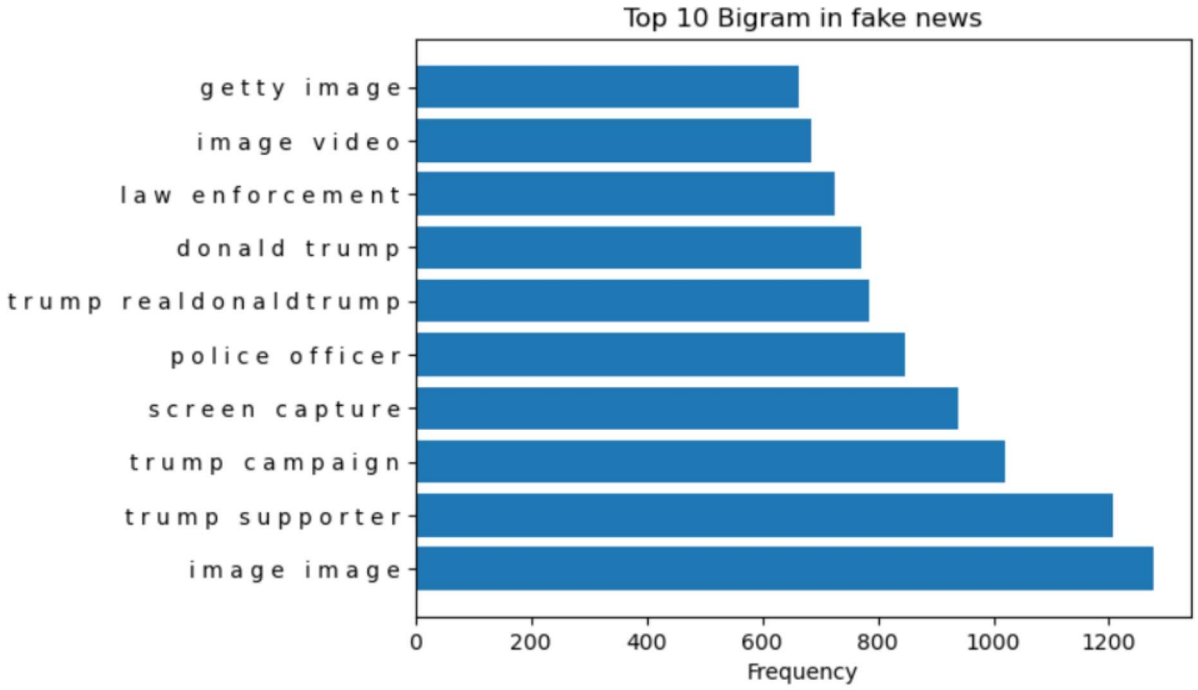
Top ten Trigrams in True news:



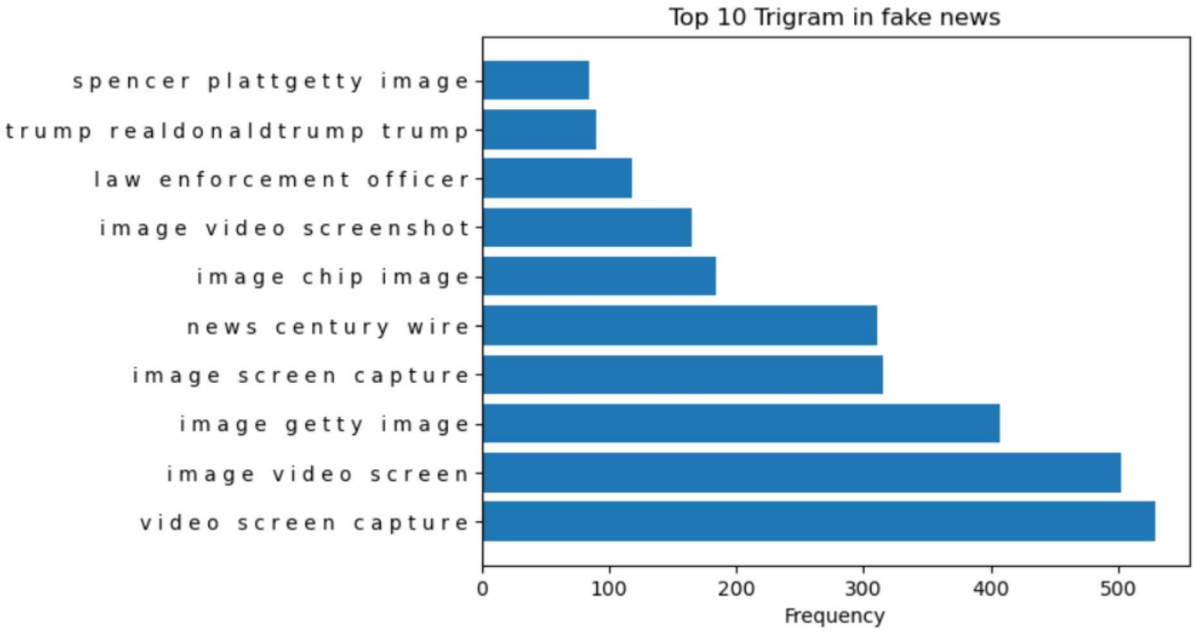
Top ten Unigrams in Fake news:



Top ten Bigrams in Fake news:



Top ten Trigrams in fake news:



Next we proceeded for Feature extraction

As part of this we use the Word2Vec Vectorizer to create vectors from textual data. Word2Vec model captures the semantic relationship between words. We extract vectors for cleaned news data.

Post this we use various models and evaluated them one by one. Train data:

Logistic Regression

Accuracy_train : 0.9144986845951476

Precision_train : 0.9143899204244031

Recall_train : 0.9293031405849845

f1-score_train : 0.9217862156561267

Accuracy_test : 0.910044338335607

Precision_test : 0.9120571783716594

Recall_test : 0.923101116527756

f1-score_test : 0.9175459163735834

```
# Classification Report
print("classification Report - Logistic Regression Model: \n", classification_report(y_test,y_pred_lr_test))
```

```
classification Report - Logistic Regression Model:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      5369
           1       0.91      0.92      0.92      6359

    accuracy                           0.91     11728
   macro avg       0.91      0.91      0.91     11728
weighted avg       0.91      0.91      0.91     11728
```

Decision Tree model

Accuracy_dtm_train : 0.8513592516807951

Precision_dtm_train : 0.8545562286015275

Recall_dtm_train : 0.8747135732578515

f1-score_dtm_train : 0.8645174182375275

Accuracy_dtm_test : 0.7907571623465212

Precision_dtm_test : 0.7965072133637054

Recall_dtm_test : 0.8248152225192641

f1-score_dtm_test : 0.8104140914709518

classification Report Decision Tree Model:

	precision	recall	f1-score	support
0	0.78	0.75	0.77	5369
1	0.80	0.82	0.81	6359
accuracy			0.79	11728
macro avg	0.79	0.79	0.79	11728
weighted avg	0.79	0.79	0.79	11728

Random Forest Model

Accuracy_train_rfm : 0.9215141771411868

Precision_train_rfm : 0.9150313971742543

Recall_train_rfm : 0.9427820460978569

f1-score_train_rfm : 0.9286994622585142

Accuracy test : 0.8832708049113234

Precision test : 0.8788338900698451

Recall test : 0.9102060072338418

f1-score test : 0.8942448821938973

```
classification Report - Random Forest Model:
              precision    recall  f1-score   support

           0       0.89      0.85      0.87      5369
           1       0.88      0.91      0.89      6359

    accuracy                           0.88     11728
   macro avg       0.88      0.88      0.88     11728
weighted avg       0.88      0.88      0.88     11728
```

Conclusion:

As observed from the word clouds of true and fake news, the top 40 most frequent words do not show significant differences.
To explore deeper patterns, we performed Unigram, Bigram, and Trigram analyses on both true and fake news datasets.

- **The Unigram analysis revealed minimal distinction, essentially reflecting the word cloud results in bar chart format.**
- **In the Bigram analysis, however, we observed some notable differences — phrases such as 'police officer', 'getty image', and 'image image' appeared more frequently in fake news articles.**
- **The Trigram analysis provided a clearer picture, highlighting combinations involving images and videos that were more commonly associated with fake news content.**

Among the three models we trained, the basic Logistic Regression model showed relatively better performance in terms of precision, recall, and F1-score. Please refer to the comparison table below for detailed metrics.

	Metric	decision_tree	random_forest	logistic_regression
0	train_Accuracy	0.851359	0.921514	0.914499
1	test_Accuracy	0.790757	0.883271	0.910044
2	train_Precision	0.854556	0.915031	0.914390
3	test_Precision	0.796507	0.878834	0.912057
4	train_Recall	0.874714	0.942782	0.929303
5	test_Recall	0.824815	0.910206	0.923101
6	train_F1-Score	0.864517	0.928699	0.921786
7	test_F1-Score	0.810414	0.894245	0.917546

Logistic Regression Model:

Accuracy_test : 0.910044338335607

Precision_test : 0.9120571783716594

Recall_test : 0.923101116527756

f1-score_test : 0.9175459163735834