FAKE NEWS DETECTION REPORT

Gokul Narayanan

Hariom Solanki

1. Problem Statement:

The objective of this project is to develop a machine learning model that can accurately distinguish between true and fake news articles. This includes various NLP techniques to get the accurate output

2. Methods Used:

Following methods is used to get the required model

a) Data Loading
b) Text Preprocessing (Cleaning, tokenization, Lemmatization)
c) EDA
d) Feature Engineering using TF-IDF and Word2Vec
e) Model Training and Evaluation (Logistic Regression, Random Forest and Decision Tree)

3. Detailed Explanation of Each Steps

1) Data loading

a) Two csv files True.csv and Fake.csv is loaded
b) Data dictionary contains the title of the news article, text of the article and data of publication
c) Head output of data is below

+ Code + Text

```
# Inspect the DataFrame with True News to understand the given data
true.head()
```

	title	text	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	December 29, 2017

```
# Inspect the DataFrame with Fake News to understand the given data
fake.head()
```

	title	text	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	December 29, 2017

Variables Terminal

d) Add new column news_label to both the DataFrames and assign labels. 1 for true news and 0 for fake news
e) True and Fake news dataframes are merged and null values are dropped

2) **<u>Text Preprocessing</u>**

a) A function was written to clean the all text and remove all unnecessary elements. This converted the text to lower case, removed the square bracket, removed punctuation and removed words with numbers

```
# Remove words with numbers

import re
import string

def clean_text(text):
    text = text.lower() #Lower case
    text = re.sub(r'\[.*?\]', '', text) # Remove text in square brackets
    text = text.translate(str.maketrans('', '', string.punctuation)) # Remove punctuation
    text = re.sub(r'\w*\d\w*', '', text) # Remove words with numbers

    return text
```

2.1.2 Apply the function to clean the news text and store the cleaned text in a new column within the new DataFrame. [1 mark]

```
[ ] # Apply the function to clean the news text and remove all unnecessary elements
    # Store it in a separate column in the new DataFrame
    df_clean["cleaned_text"] = merged_cleaned["news_text"].apply(clean_text)
```

b) POS tagging and Lemmatization was performed on data. This filtered stopwords and retained only NN and NNS tags

```
[ ] # Write the function for POS tagging and lemmatization, filtering stopwords and keeping only NN and NNS tags
    import spacy
    from nltk.corpus import stopwords
    from tqdm.notebook import tqdm

    # Load spacy model
    nlp = spacy.load("en_core_web_sm", disable=["ner", "parser"])
    import nltk
    nltk.download('stopwords', quiet=True)
    from nltk.corpus import stopwords
    stop_words = set(stopwords.words('english'))

    def pos_tag_lemmatize(texts):
        lemmatized_texts = []

        # Batch processing using spaCy's pipe
        for doc in tqdm(nlp.pipe(texts, batch_size=50, n_process=1), total=len(texts)):
            tokens = []
            for token in doc:
                if (token.tag_ in ['NN', 'NNS']) and (token.text.lower() not in stop_words) and (token.is_alpha):
                    tokens.append(token.lemma_.lower())
            lemmatized_texts.append(" ".join(tokens))

        return lemmatized_texts
```

c) Applied the POS tagging and lemmatization function to cleaned text and store it in a new column within the new dataFrame.

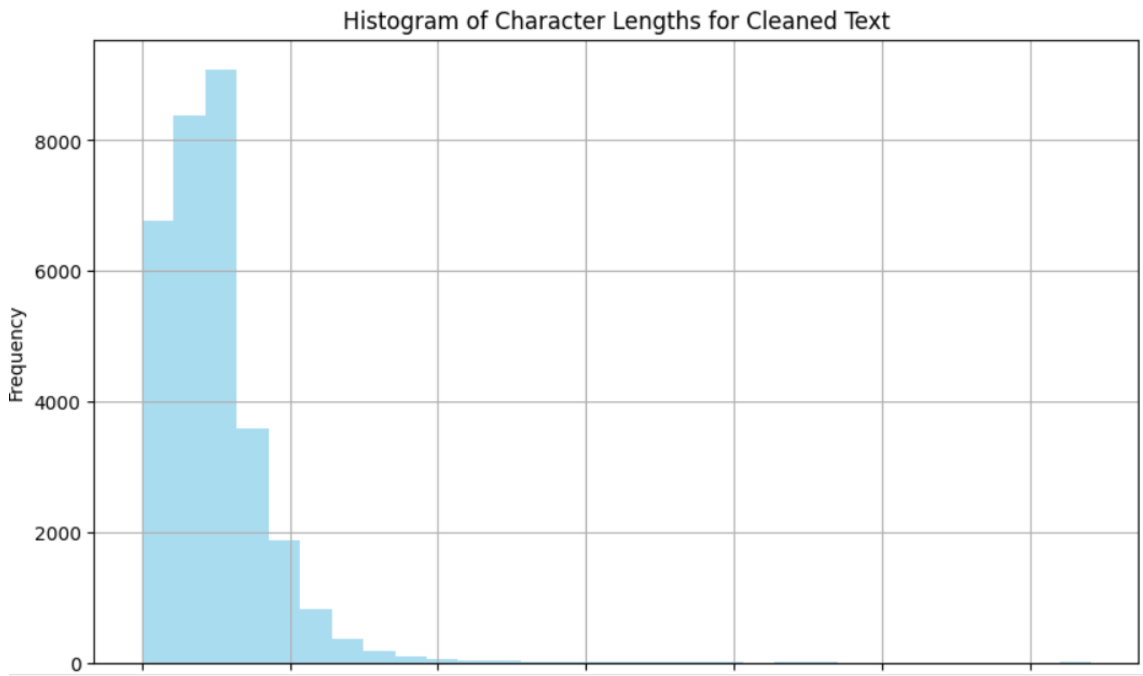
d) Saved the new dataframe as csv file for easy usage in future.

```
[ ] # Check the first few rows of the DataFrame
df_clean.head()
```

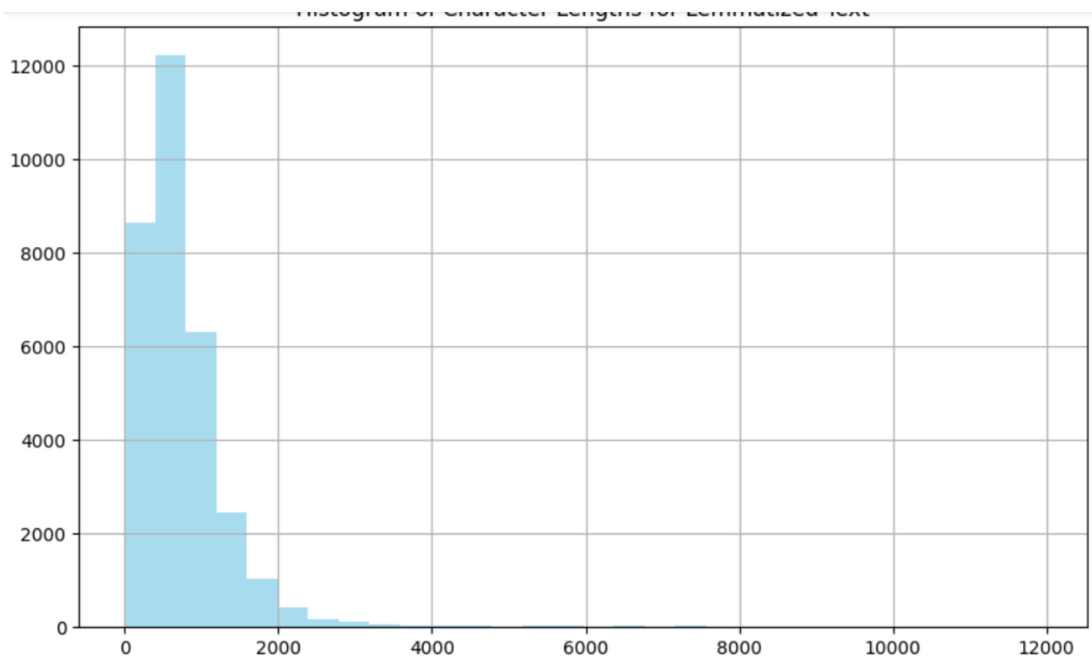
	news_label	cleaned_text	lemmatized_text
0	1	as us budget fight looms republicans flip thei...	budget fight script head faction month expansi...
1	1	us military to accept transgender recruits on ...	military transgender recruit people time milit...
2	1	senior us republican senator let mr mueller do...	mueller job counsel investigation link electio...
3	1	fbi russia probe helped by australian diplomat...	probe diplomat trump campaign adviser diplomat...
4	1	trump wants postal service to charge much more...	trump service service ship package amzno fight...

3) **Exploratory Data Analysis (EDA)**

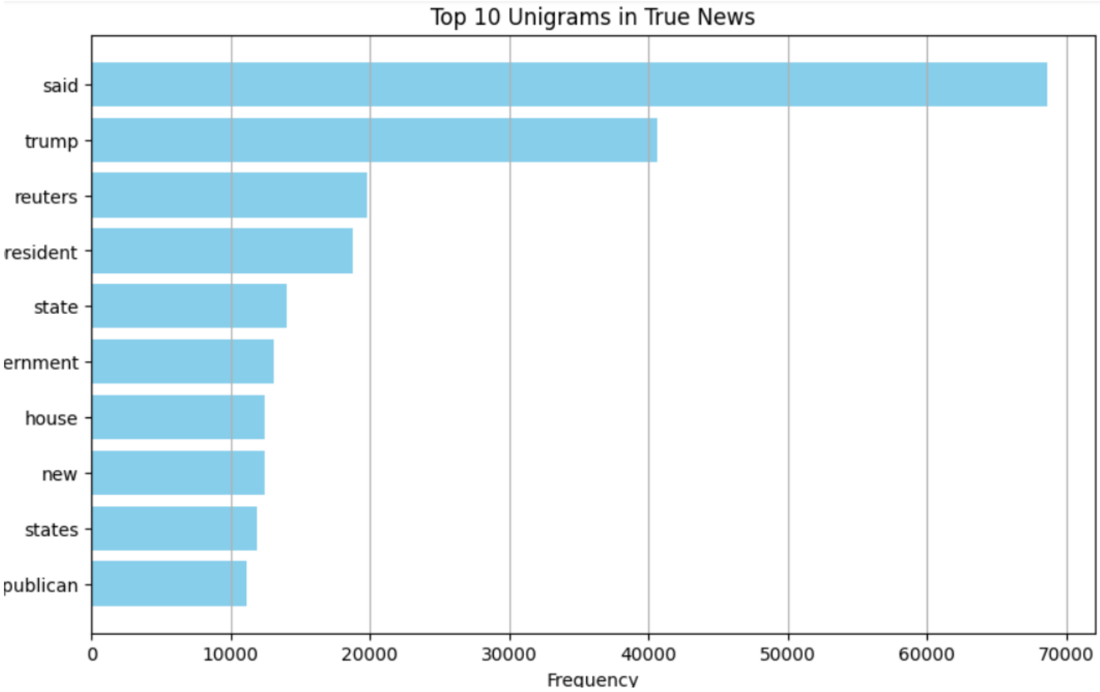
a) Histogram of cleaned text is plotted

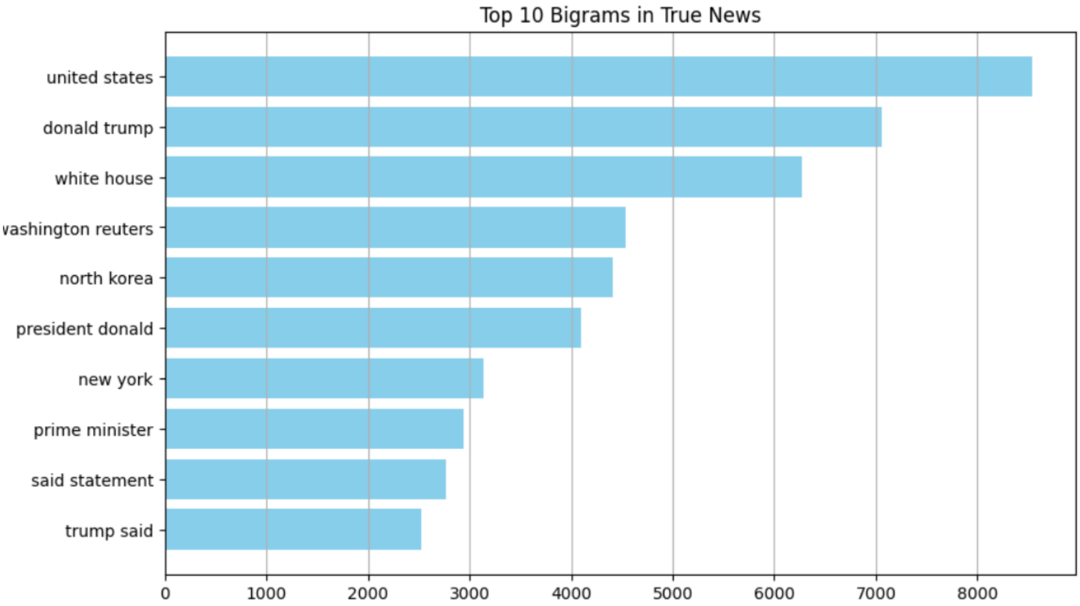


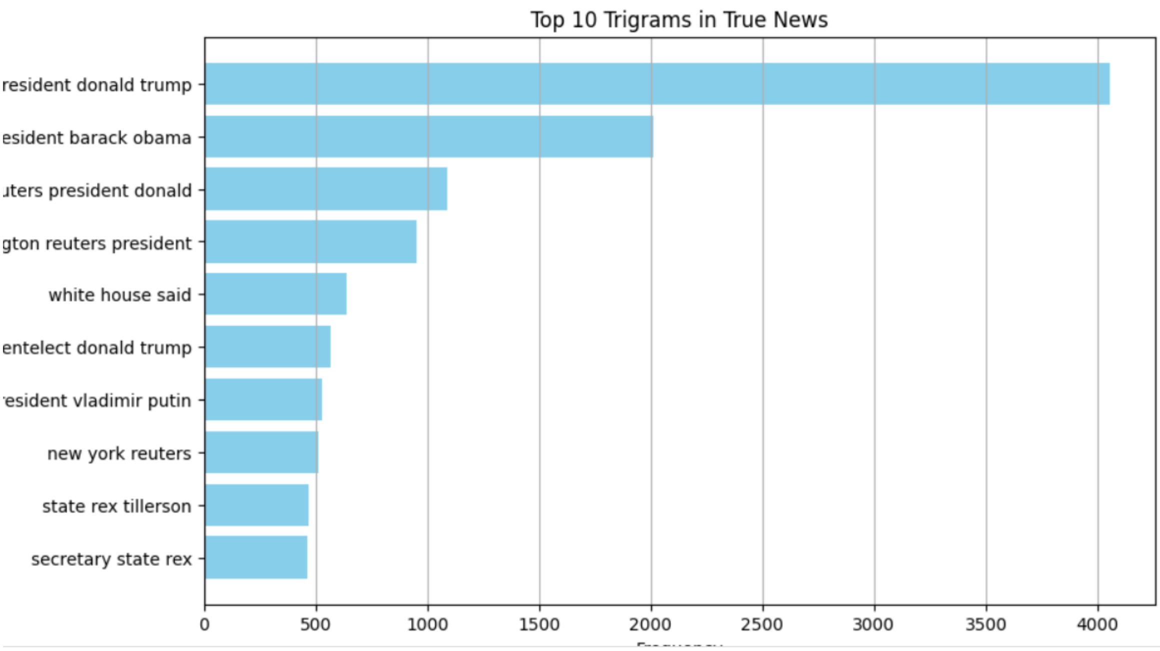
b) Histogram of lemmatized texts is plotted

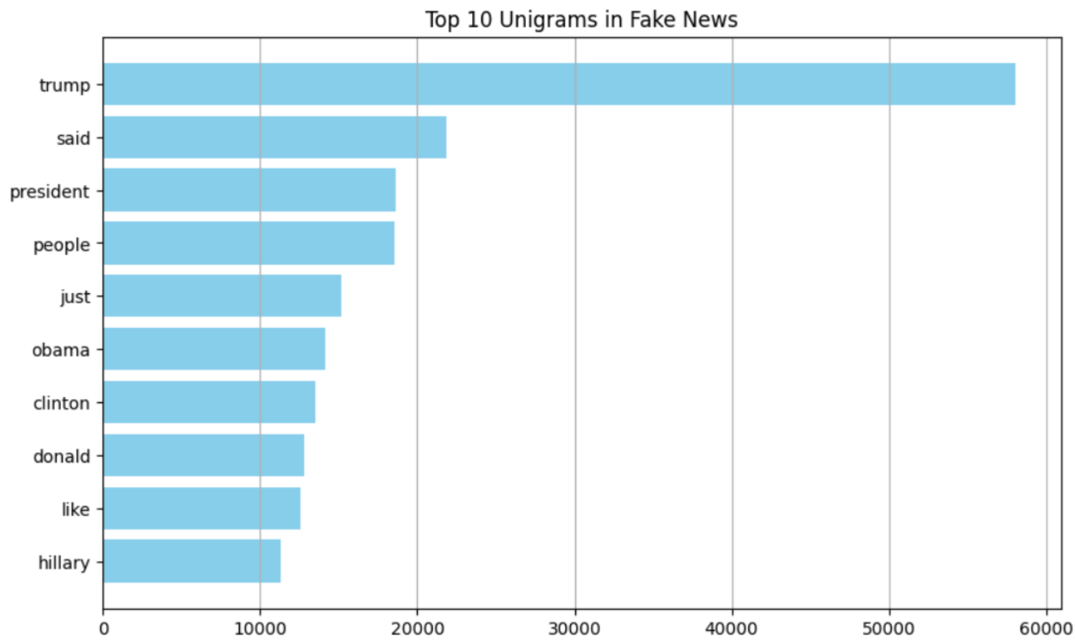


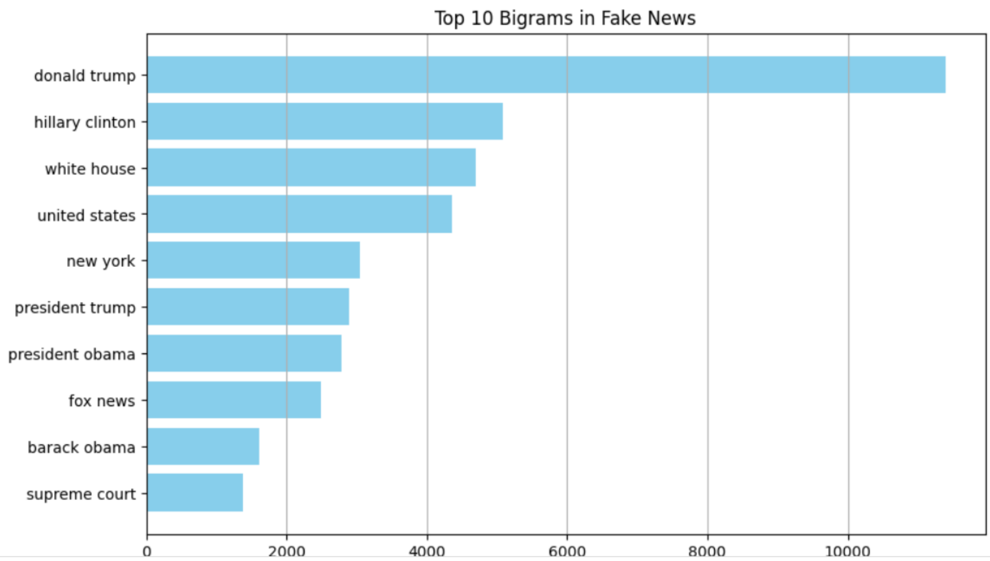
c) Displayed Top 10 unigrams, bigrams and trigrams in both True news and Fake news

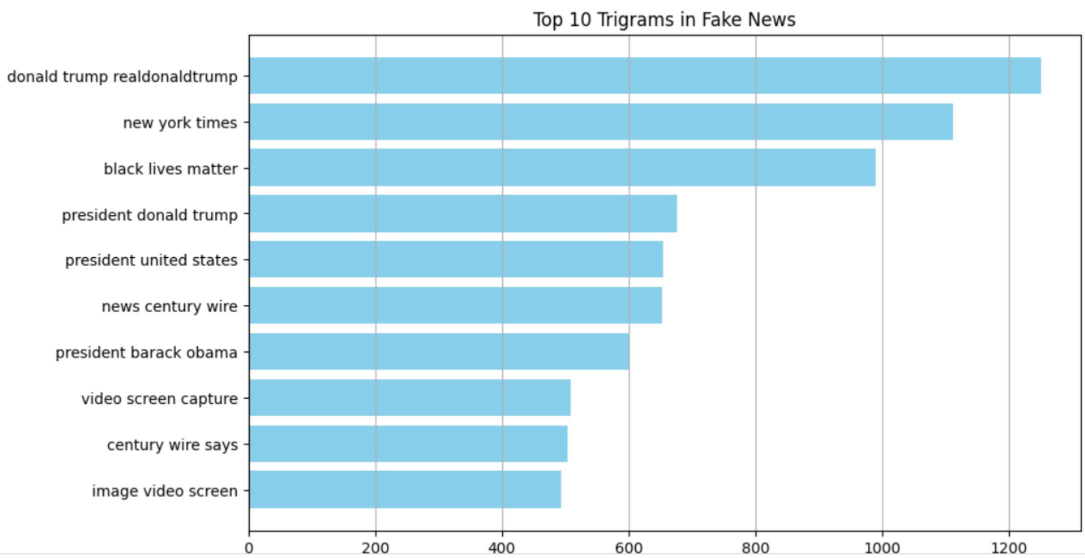



Top 10 Bigrams in True News
united states
donald trump
white house
vashington reuters
north korea
president donald
new york
prime minister
said statement
trump said
0
1000
2000
3000
4000
5000
6000
7000
8000


Top 10 Trigrams in True News
resident donald trump
esident barack obama
uters president donald
gton reuters president
white house said
entelect donald trump
esident vladimir putin
new york reuters
state rex tillerson
secretary state rex
0
500
1000
1500
2000
2500
3000
3500
4000


Top 10 Unigrams in Fake News
trump
said
president
people
just
obama
clinton
donald
like
hillary
0
10000
20000
30000
40000
50000
60000


Top 10 Bigrams in Fake News
donald trump
hillary clinton
white house
united states
new york
president trump
president obama
fox news
barack obama
supreme court
0
2000
4000
6000
8000
10000


Top 10 Trigrams in Fake News
donald trump realdonaldtrump
new york times
black lives matter
president donald trump
president united states
news century wire
president barack obama
video screen capture
century wire says
image video screen
0
200
400
600
800
1000
1200

4) Feature Engineering

a) Initialized the word2vec model

```
## Write your code here to initialise the Word2Vec model by downloading "word2vec-google-news-300"
!pip install gensim
import gensim.downloader as api
from gensim.models import KeyedVectors
print("Downloading Word2Vec model")
word2vec_model = api.load("word2vec-google-news-300")
```

```
Collecting gensim
  Downloading gensim-4.3.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.1 kB)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.12.0)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.3.0.post1)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open>=1.8.1->gensim) (1.17.3)
Downloading gensim-4.3.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.6/26.6 MB 77.5 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.3.3
Downloading Word2Vec model
[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

b) Extracted vectors for cleaned Data

```
## Write your code here to extract the vectors from the Word2Vec model for both training and validation data
def doc_vector(text):
    words = text.split()
    word_vectors = []
    for word in words:
        if word in word2vec_model:
            word_vectors.append(word2vec_model[word])
    if len(word_vectors) == 0:
        return np.zeros(300)  # if no word is found, return a zero vector
    else:
        return np.mean(word_vectors, axis=0)

X_train_vectors = np.vstack(train_data['cleaned_text'].apply(doc_vector))
X_val_vectors = np.vstack(val_data['cleaned_text'].apply(doc_vector))

## Extract the target variable for the training data and validation data
y_train = train_data['news_label']
y_val = val_data['news_label']
```

5) Model Training and Evaluation

a) Initially the model training and evaluation is performed for Logistic regression

```
## Initialise Logistic Regression model
log_reg = LogisticRegression(max_iter=1000, random_state=42)
## Train Logistic Regression model on training data
log_reg.fit(X_train_vectors, y_train)
## Predict on validation data
y_pred = log_reg.predict(X_val_vectors)
```

The Evaluation Data for Logistic Regression is as below

```
print("\n Logistic Regression Metrics for Val Data ---")
print(f"Accuracy: {accuracy_log_reg:.4f}")
print(f"Precision: {precision_log_reg:.4f}")
print(f"Recall: {recall_log_reg:.4f}")
print(f"F1-Score: {f1_log_reg:.4f}")
```

```
 Logistic Regression Metrics for Val Data ---
Accuracy: 0.9584
Precision: 0.9504
Recall: 0.9641
F1-Score: 0.9572
```

Classification report for Logistic Regression is as below

```
[ ] # Classification Report
    print("\nClassification Report for Logistic Regression:\n")
    print(classification_report(y_val, y_pred))
```

```
Classification Report for Logistic Regression:

              precision    recall  f1-score   support

           0       0.97      0.95      0.96      6978
           1       0.95      0.96      0.96      6498

    accuracy                           0.96     13476
   macro avg       0.96      0.96      0.96     13476
weighted avg       0.96      0.96      0.96     13476
```

b) Secondly the model evaluation is performed on Decision Tree

```
[ ] ## Initialise Decision Tree model
    dt_model = DecisionTreeClassifier(random_state=42)
    ## Train Decision Tree model on training data
    dt_model.fit(X_train_vectors, y_train)
    ## Predict on validation data
    dt_preds = dt_model.predict(X_val_vectors)
```

Evaluation Data and Classification Report snapshot for Decision Tree is as below

```
print("\n Decision Tree Performance")
print(f"Accuracy: {accuracy_dt:.4f}")
print(f"Precision: {precision_dt:.4f}")
print(f"Recall: {recall_dt:.4f}")
print(f"F1 Score: {f1_dt:.4f}")
```

```
 Decision Tree Performance
Accuracy: 0.8983
Precision: 0.8985
Recall: 0.8983
F1 Score: 0.8983
```

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.89      0.92      0.90      6978
           1       0.91      0.88      0.89      6498

    accuracy                           0.90     13476
   macro avg       0.90      0.90      0.90     13476
weighted avg       0.90      0.90      0.90     13476
```

c) Finally, the model is trained using Random Forest

```
## Initialise Random Forest model
rf_model = RandomForestClassifier(random_state=42)
## Train Random Forest model on training data
rf_model.fit(X_train_vectors, y_train)
## Predict on validation data
rf_preds = rf_model.predict(X_val_vectors)
```

The evaluation result and classification report for Random Forest is as below

```python
print("\n Random Forest Performance")
print(f"Accuracy: {accuracy_rf:.4f}")
print(f"Precision: {precision_rf:.4f}")
print(f"Recall: {recall_rf:.4f}")
print(f"F1 Score: {f1_rf:.4f}")
```

```
 Random Forest Performance
Accuracy: 0.9576
Precision: 0.9576
Recall: 0.9576
F1 Score: 0.9576
```

```
Classification Report for Random Forest
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      6978
           1       0.96      0.95      0.96      6498

    accuracy                           0.96     13476
   macro avg       0.96      0.96      0.96     13476
weighted avg       0.96      0.96      0.96     13476
```

6) CONCLUSION

a) Semantic Analysis helped to better grasp the inner meaning and patterns beyond just keyword detection

b) Multiple Models like Random Forest, Logistic Regression, Decision Tree was used for this activity

c) Random Forest and Logistic regression could give accuracy around 96 percent. But the accuracy for Decision Tree was on lower side

d) However Random Forest was chosen as the best model. Random Forest was preferred for its flexibility and slightly better consistency across all classes

e) Assessment Conclusion

1) The Approach achieved 96 percent accuracy in Detecting Fake news which is a good number

2) Random Forest helped minimized the risk of overfitting

3) F1-score ensured model is not biased towards one class

4) Implementation of this model helped in detecting Fake news early