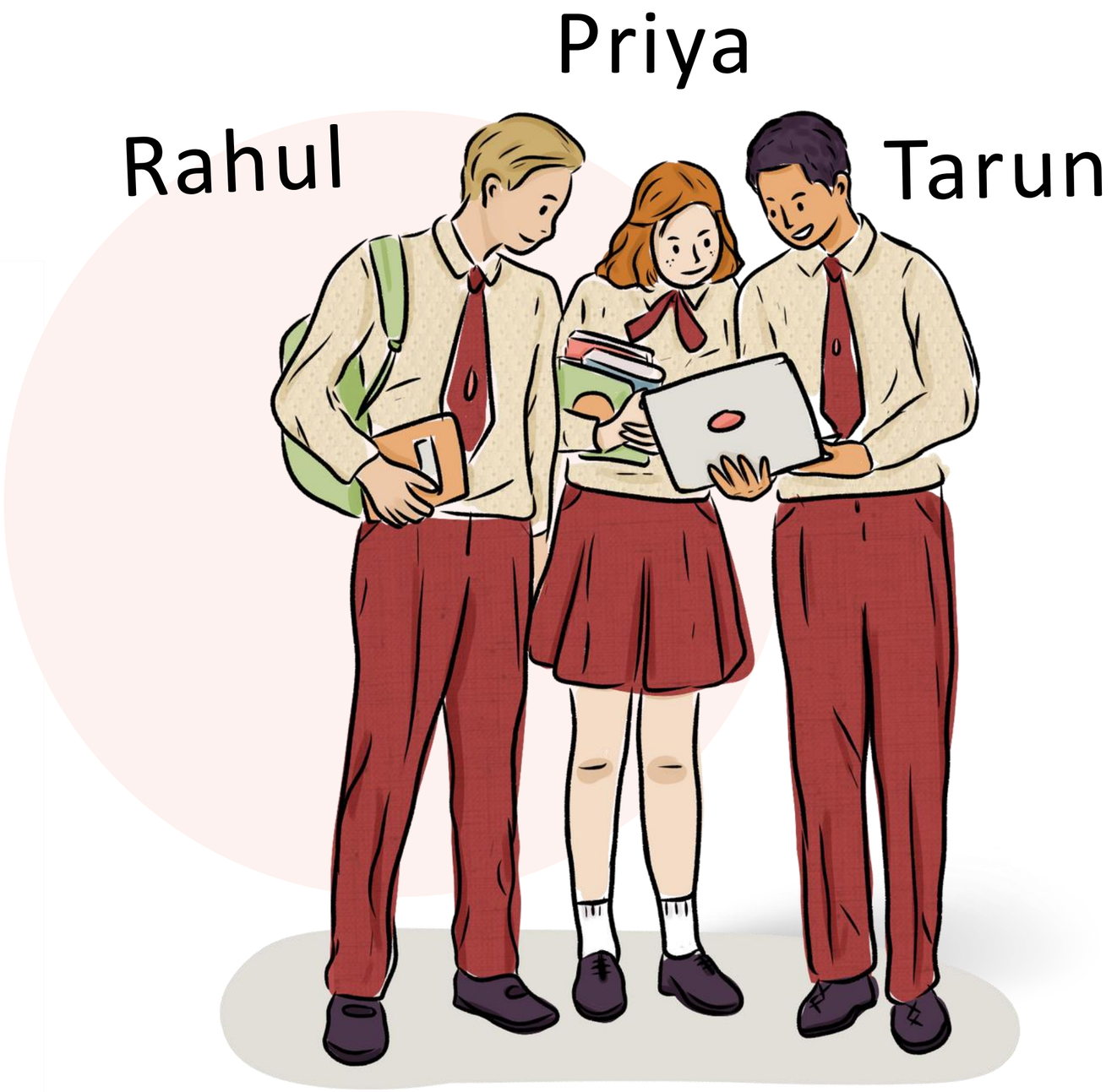
DATA MINING

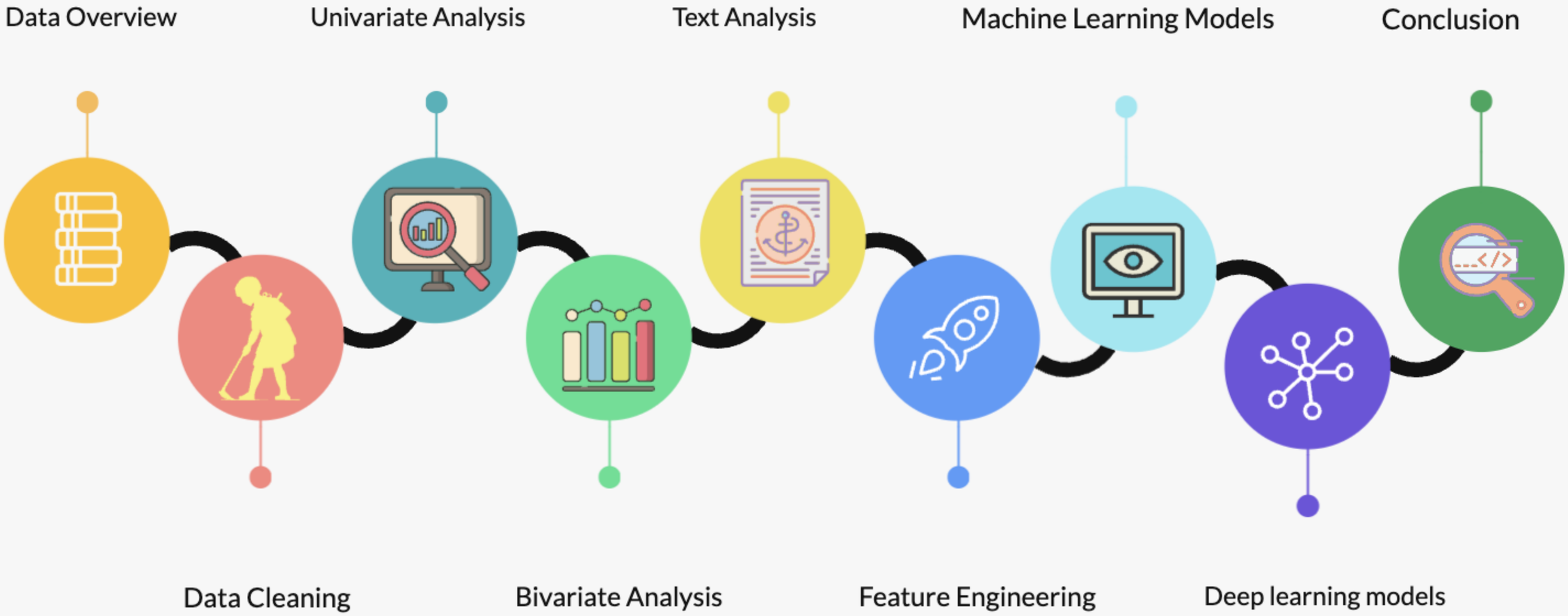
DETECTING FAKE NEWS

OUR

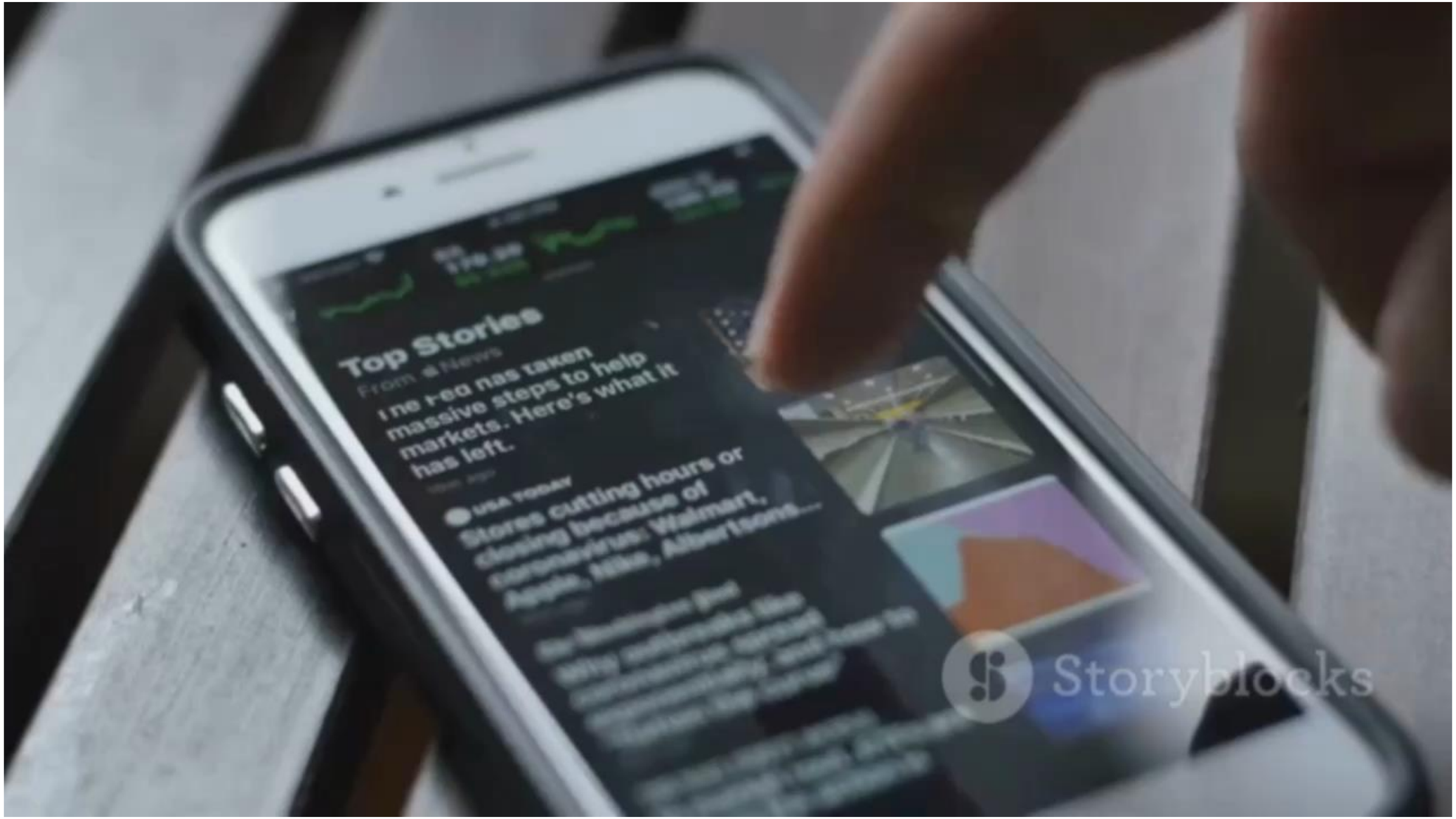
TEAM



ROAD MAP

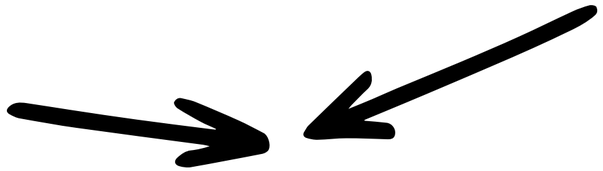

Data Overview
Univariate Analysis
Text Analysis
Machine Learning Models
Conclusion
Data Cleaning
Bivariate Analysis
Feature Engineering
Deep learning models

CHALLENGE STATEMENT



Data Overview & Cleaning

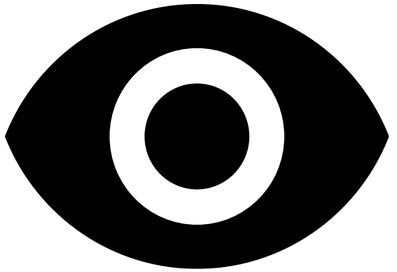
Data Shape and Size: Number of rows and columns in the dataset are 6335, 4



Handling Missing Values

Handling Outliers

Handling Duplicate Records



Language Detection

TEXT PREPROCESSING

Tokenization

Stopword Removal

HTML Tag Removal

Punctuation Removal

Lemmatization

Removing Special Characters

UNIVARIATE ANALYSIS

Title Column

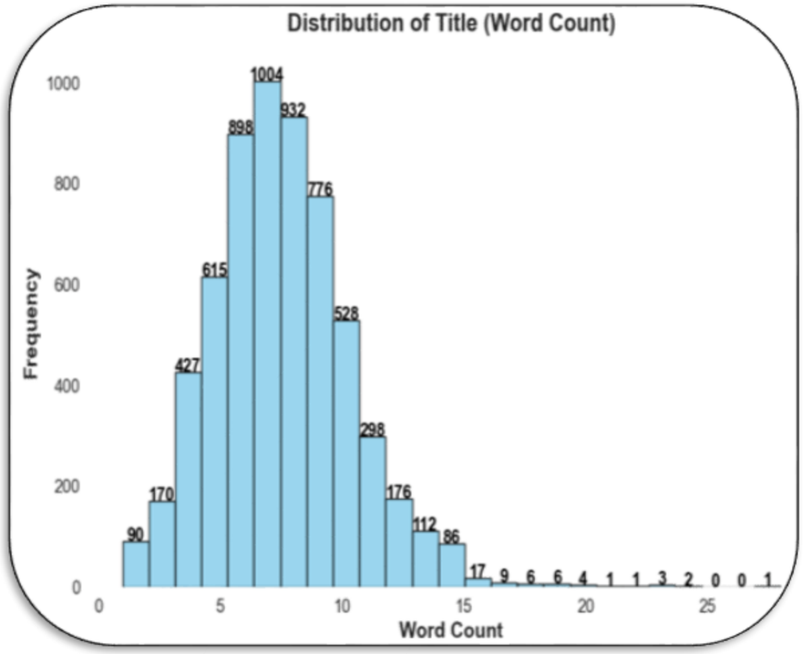
- Title Length Analysis
- Most Common Words in Titles
- Title Sentiment Analysis

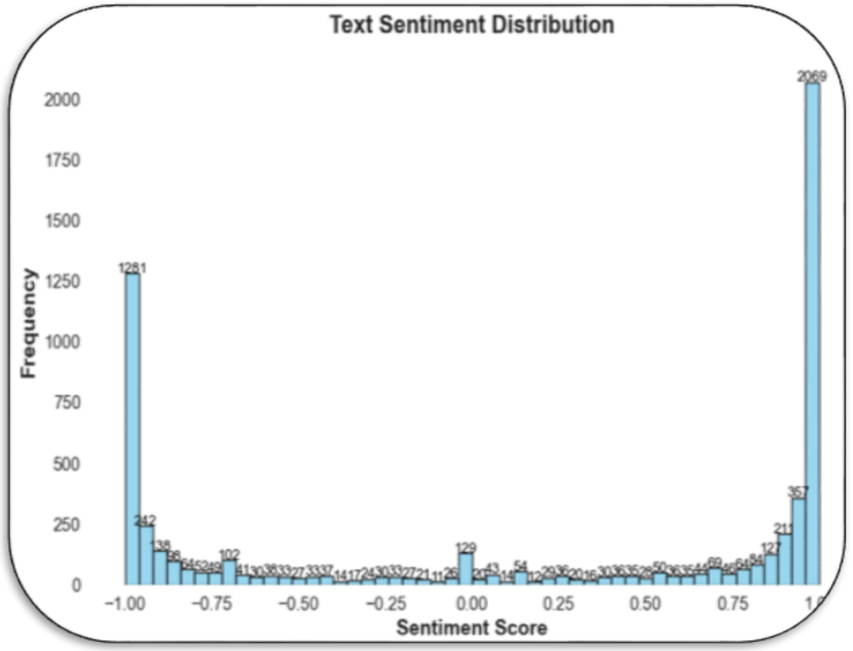
Text Column

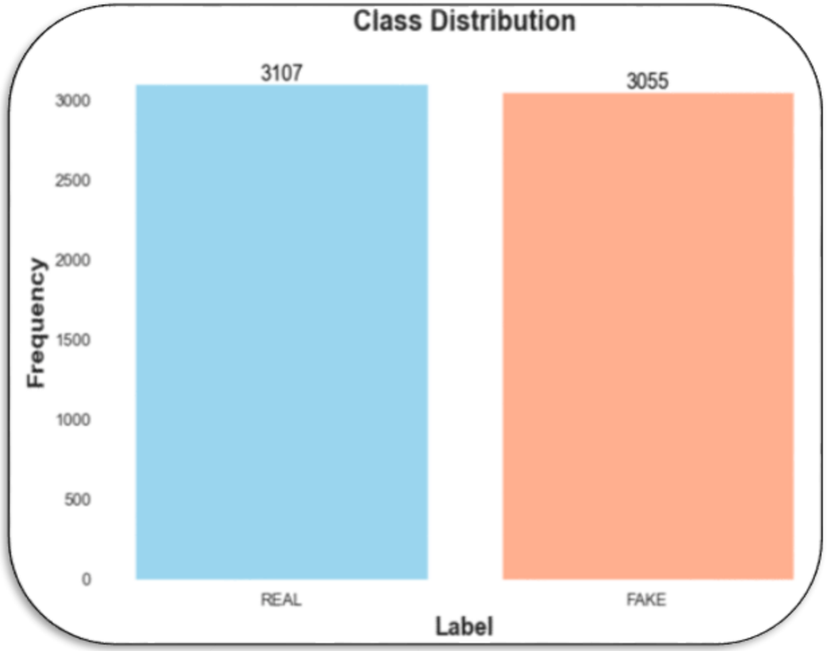
- Title Length Analysis
- Most Common Words in Titles
- Title Sentiment Analysis

Label Column

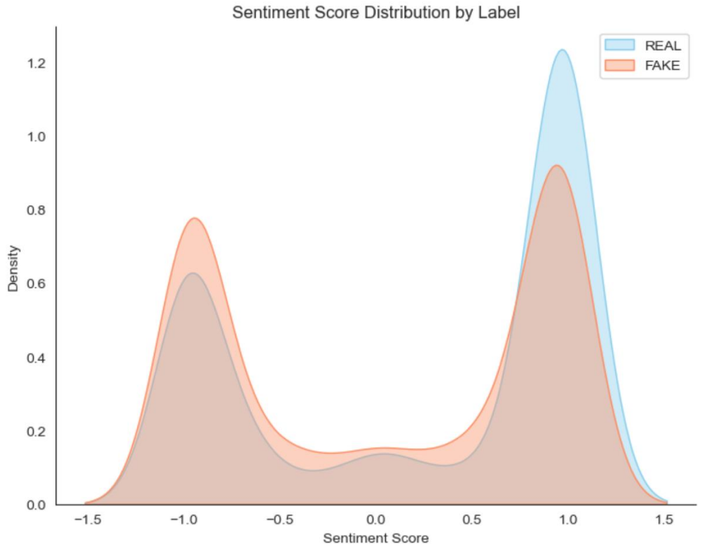
- Class Distribution
- Class Balance Analysis

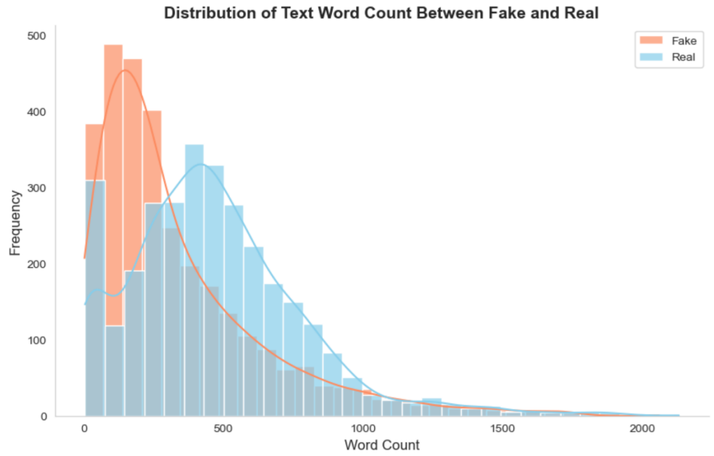


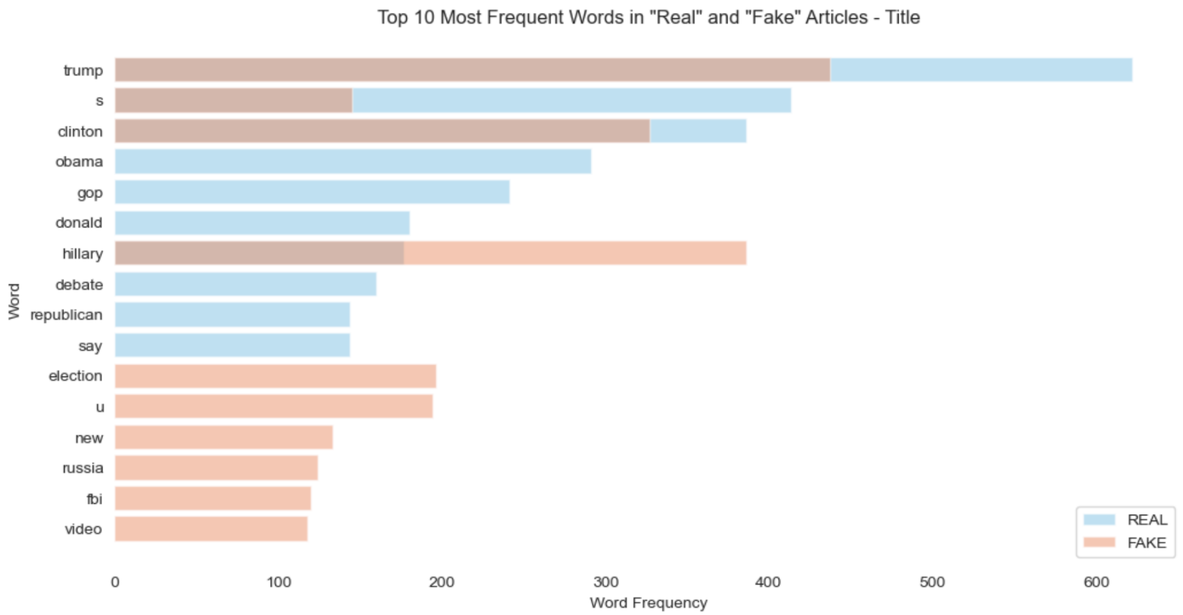


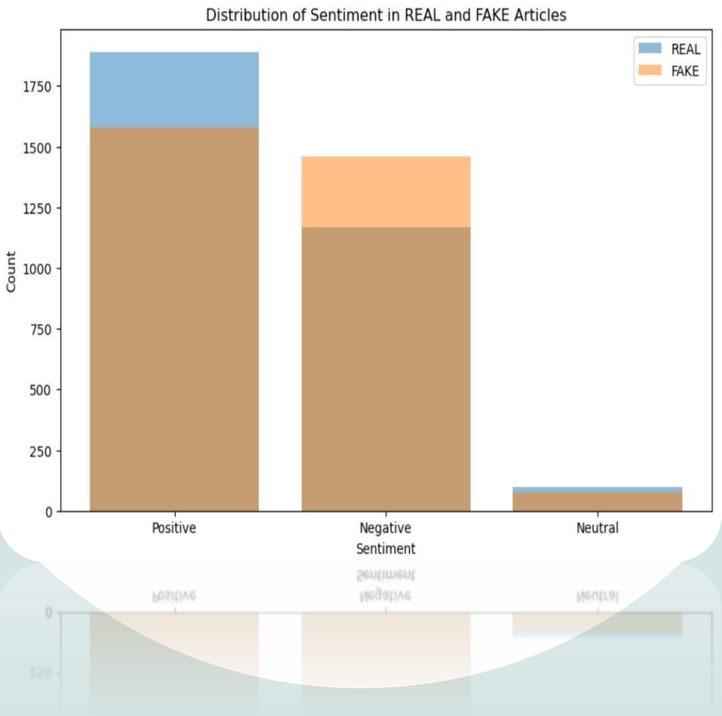


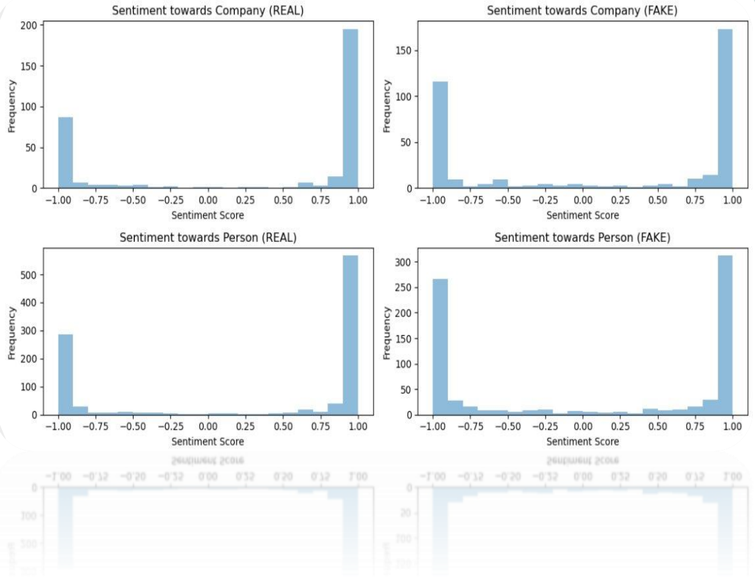
BIVARIATE ANALYSIS

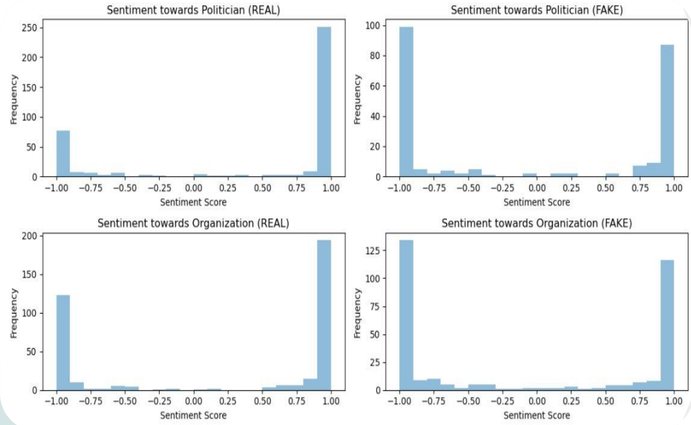






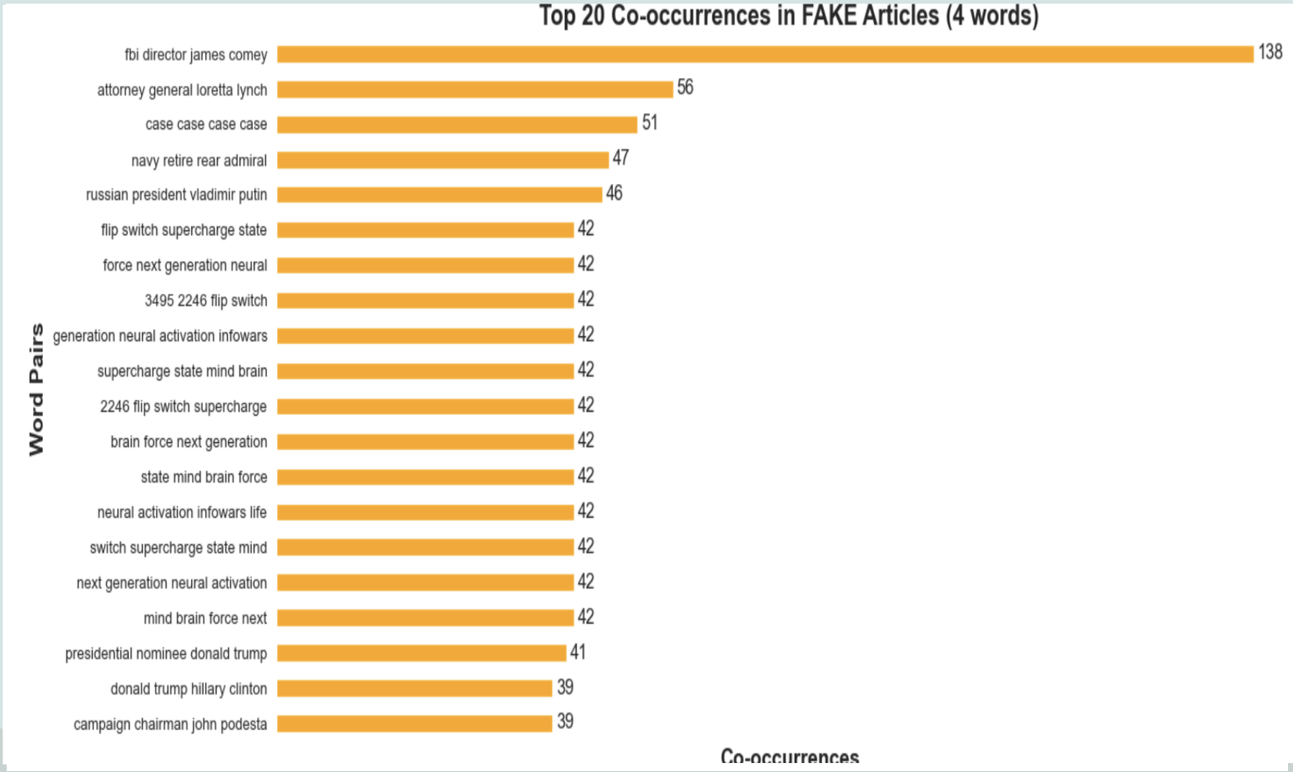


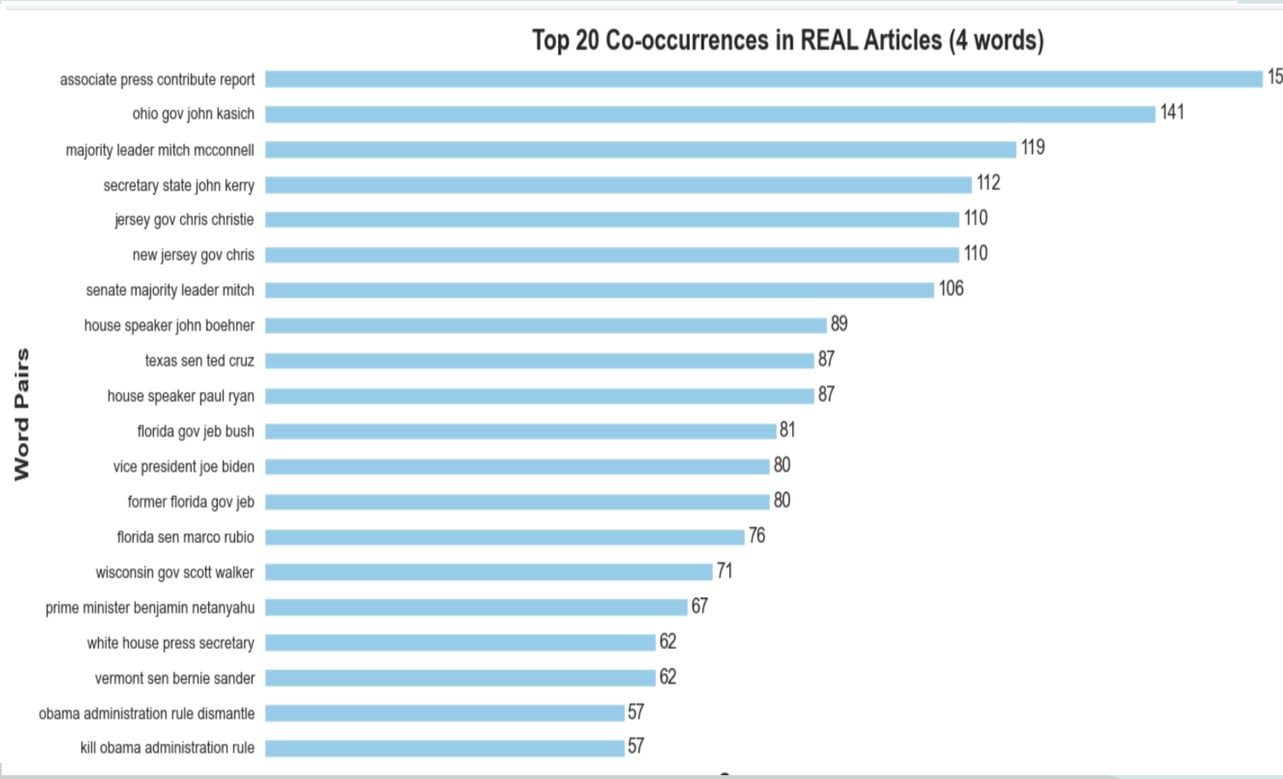




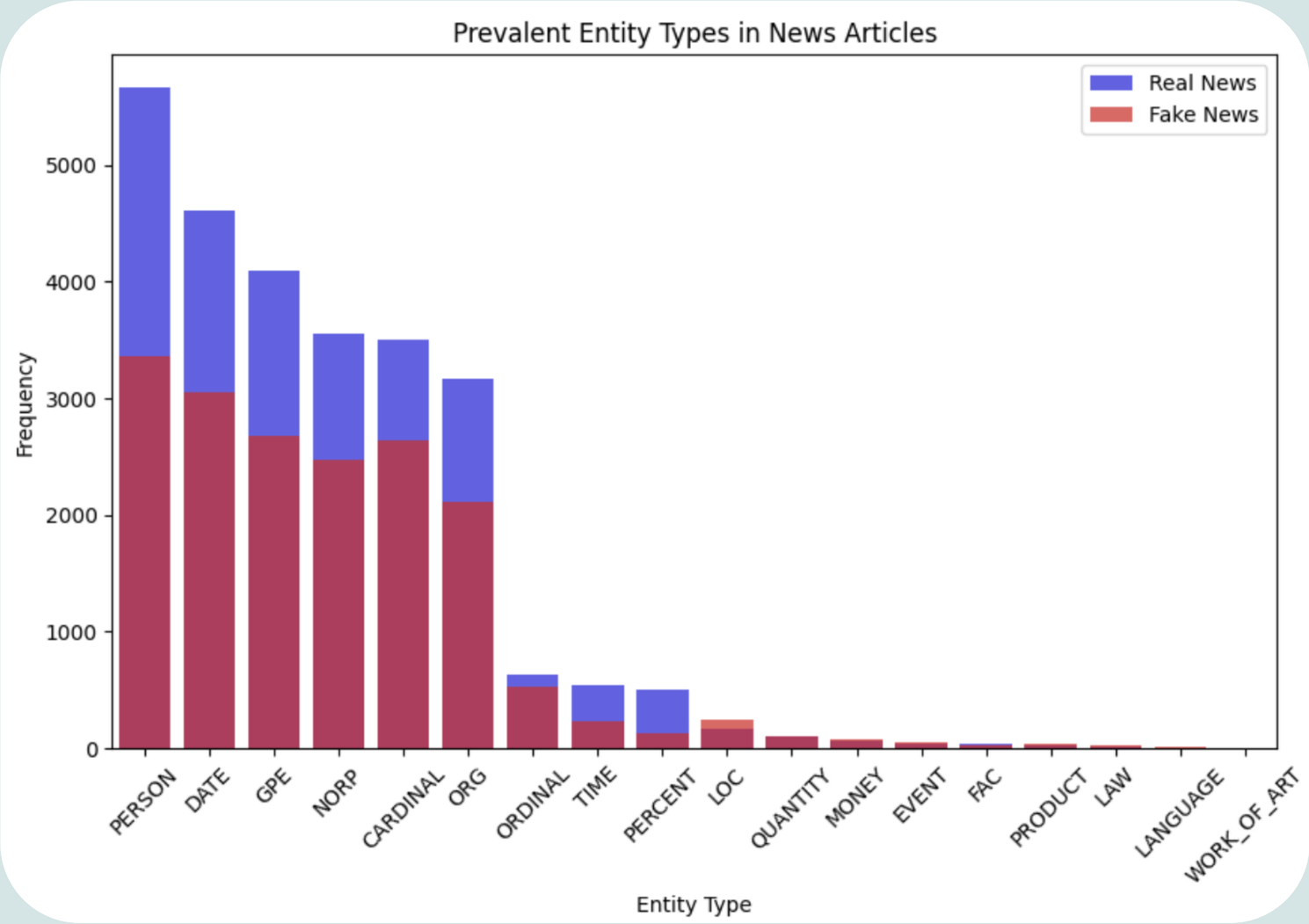
SENTIMENT ANALYSIS

CO-OCCURRENCE ANALYSIS

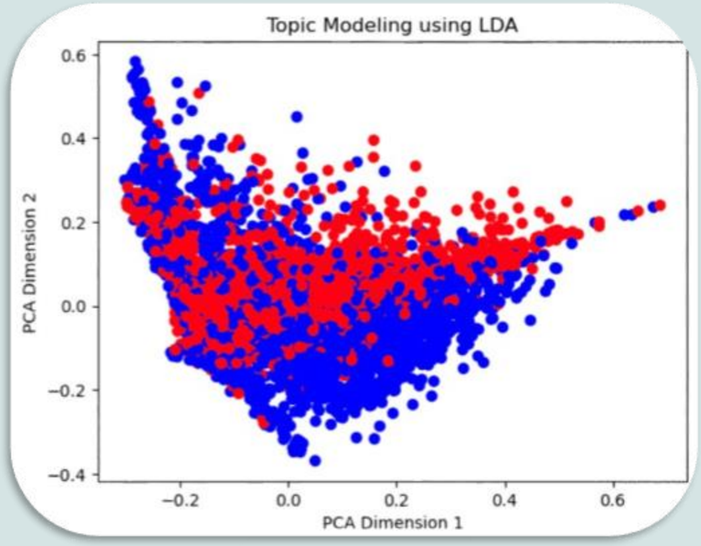


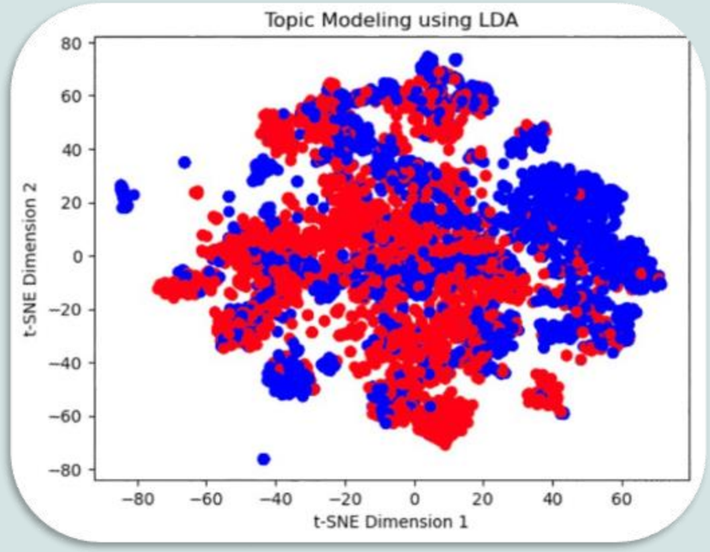


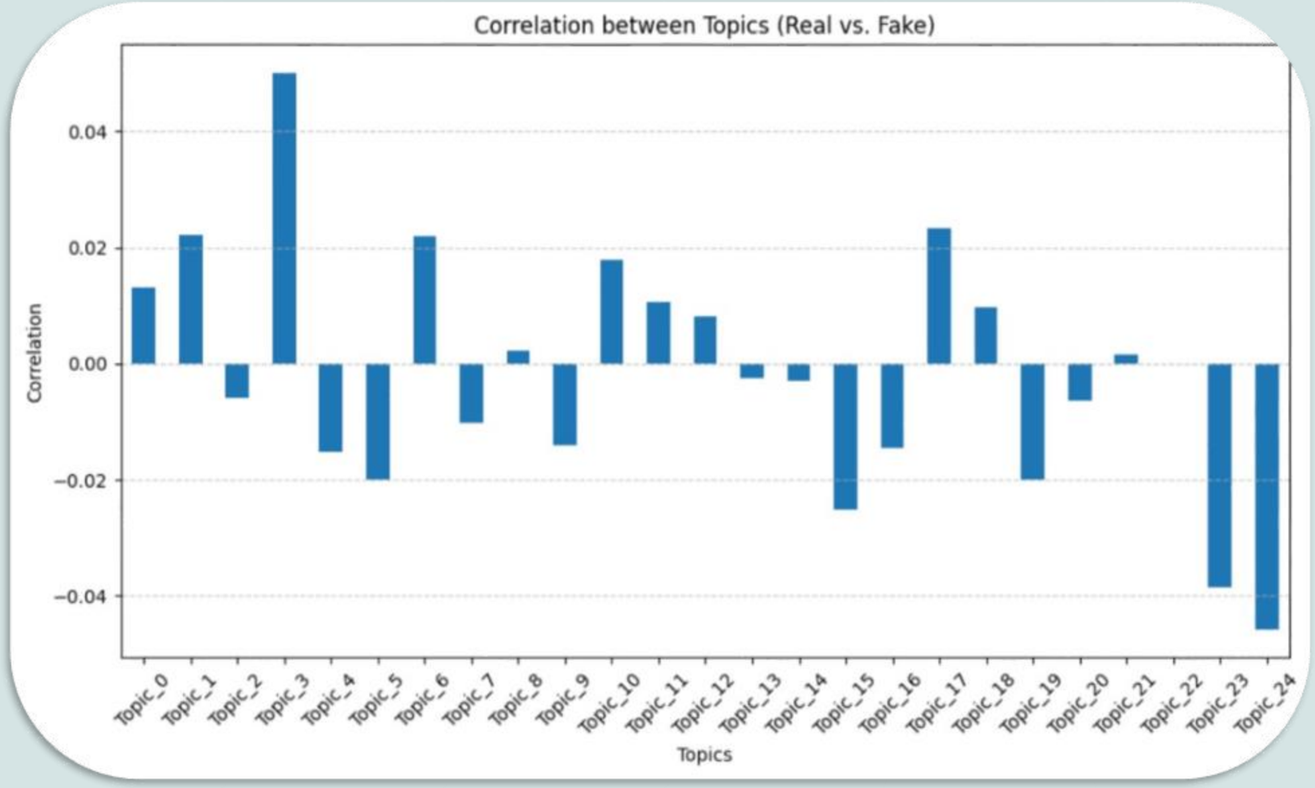
NAMED ENTITY RECOGNITION (NER)



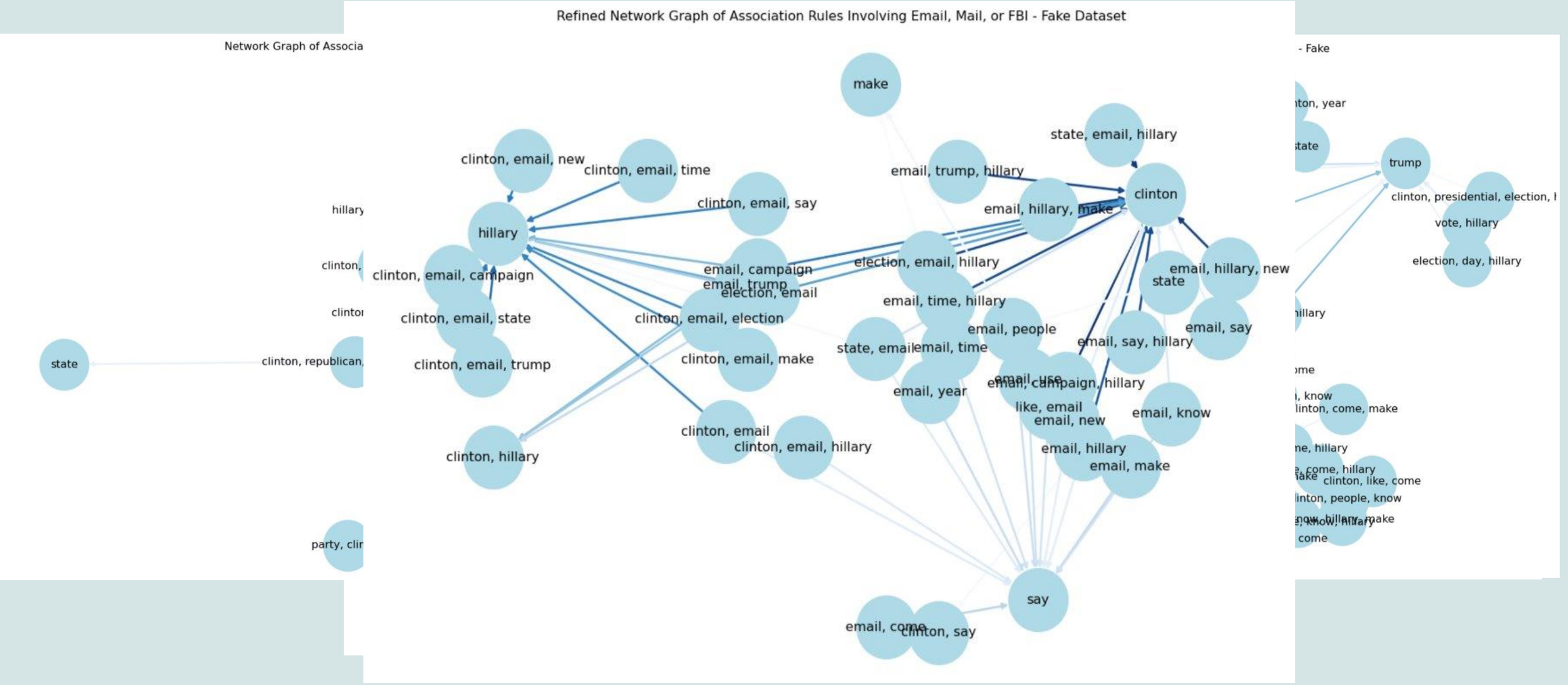
TOPIC MODELING







ASSOCIATION RULE MINING

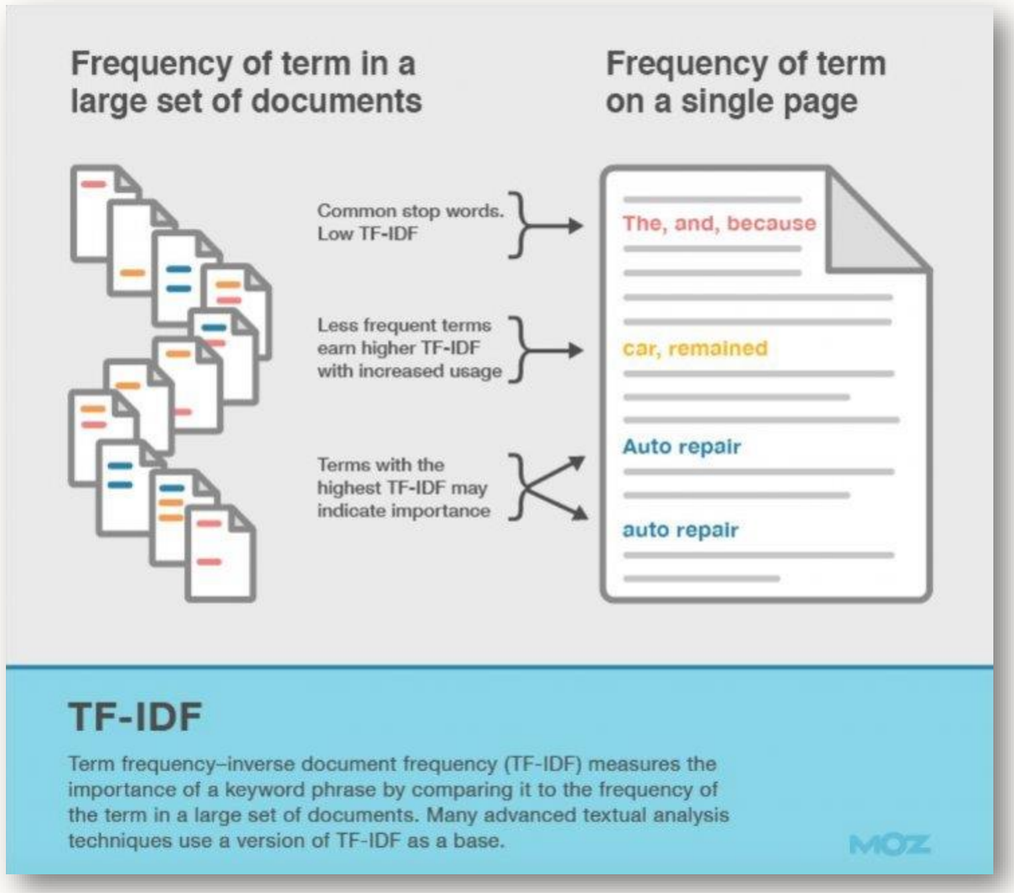


SUMMARY OF EDA

- Descriptive Statistics
- Duplicate Removal
- Language Detection
- Data Cleaning
- Text Preprocessing
- Text Length Analysis
- Readibilty Analysis
- Co-Occurrences Analysis
- Sentiment Distribution
- Named Entity Recognition (NER)
- Topic Modeling
- Association Rule Mining

EDA CONCLUSION

FEATURE ENGINEERING:(TF-IDF)



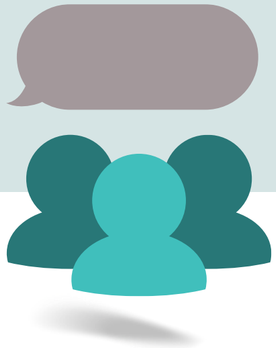
1. Data Splitting: Spliting the given dataset into training and testing sets, with 80% of the data used for training and 20% for testing.

2. Text Vectorization: We used TF-IDF vectorization technique to convert the text features ('title' and 'text') into numerical representations that machine learning models can process. The TfidfVectorizer from sklearn.feature_extraction.text is used for this purpose.

3. Label Encoding and Saving: We performed label encoding on the target variable ('label') to convert the categorical labels into numeric values. We used Label Encoder from sklearn.preprocessing for this task. Additionally, we also saved the vectorizer object using joblib.dump() for future use in transforming new data.

Word Embeddings

(Word2Vec)



- PURPOSE: PRE-TRAINED WORD EMBEDDINGS THAT CAPTURE SEMANTIC MEANINGS OF WORDS.

- USE CASE: SUITABLE FOR DEEP LEARNING MODELS, ESPECIALLY WHEN CAPTURING CONTEXTUAL RELATIONSHIPS IS IMPORTANT.

MACHINE LEARNING MODEL

Logistic Regression (LR)

```
Classification Report:
              precision    recall  f1-score   support

        Fake       0.92      0.94      0.93       611
        Real       0.94      0.92      0.93       622

    accuracy                           0.93      1233
   macro avg       0.93      0.93      0.93      1233
weighted avg       0.93      0.93      0.93      1233
```

ENSEMBLE MACHINE LEARNING MODEL

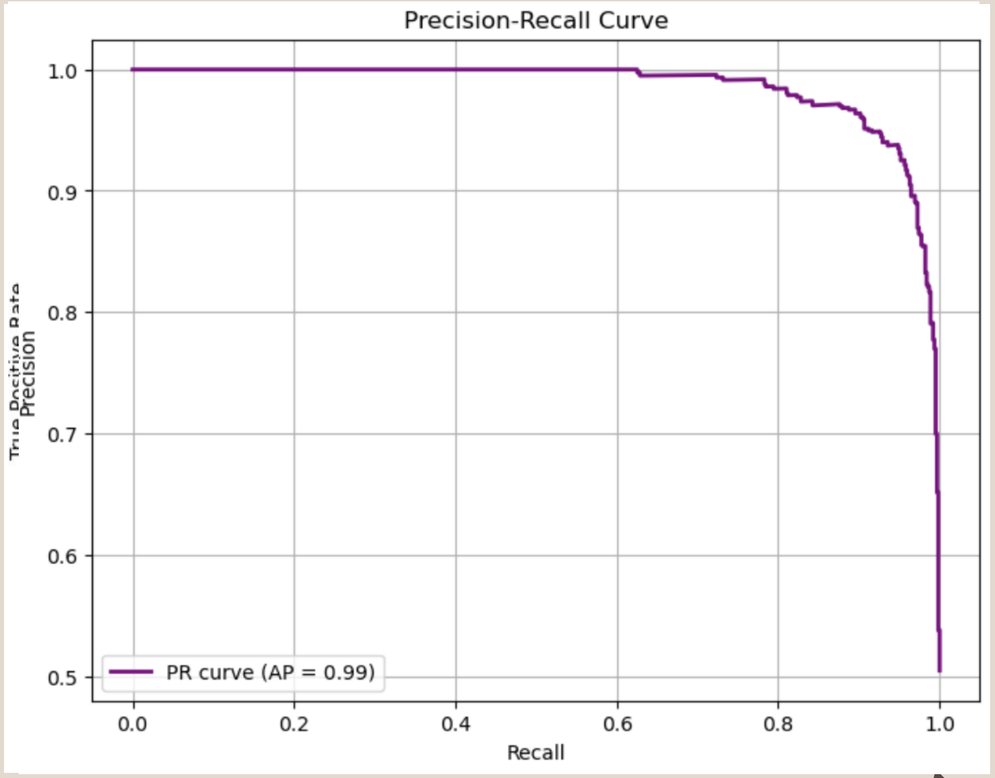
Random Forest(RF)

```
Classification Report:
              precision    recall  f1-score   support

        Fake       0.91      0.92      0.92       611
        Real       0.92      0.91      0.92       622

    accuracy                           0.92      1233
   macro avg       0.92      0.92      0.92      1233
weighted avg       0.92      0.92      0.92      1233
```

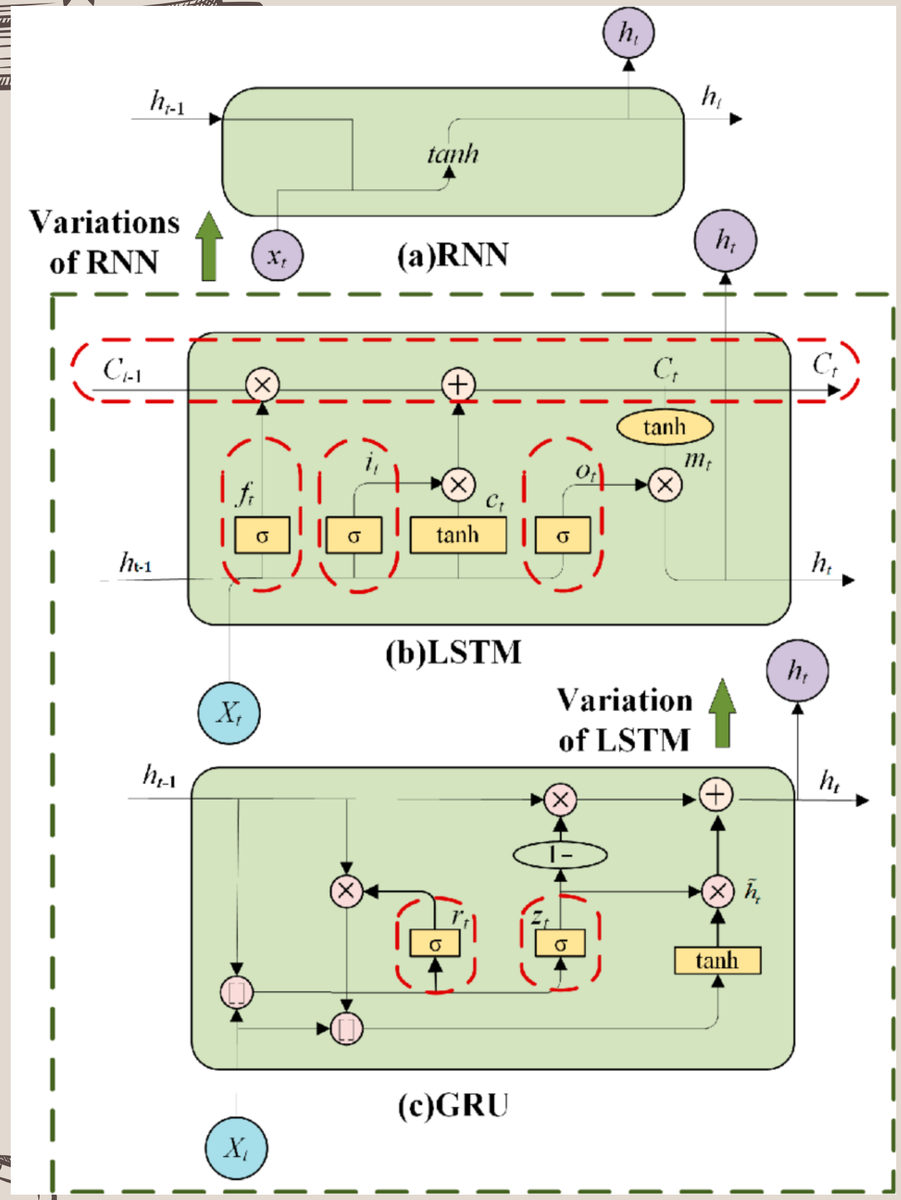
DEEP LEARNING MODEL

Multi-Layer Perceptron (MLP)

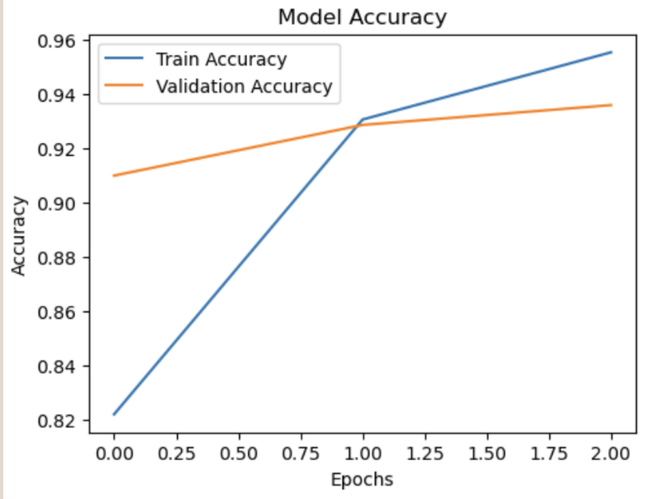
```
Classification Report:
              precision    recall  f1-score   support

        Fake       0.94      0.93      0.94       611
        Real       0.93      0.95      0.94       622

    accuracy                           0.94      1233
   macro avg       0.94      0.94      0.94      1233
weighted avg       0.94      0.94      0.94      1233


Confusion Matrix:
[[569  42]
 [ 34 588]]
```

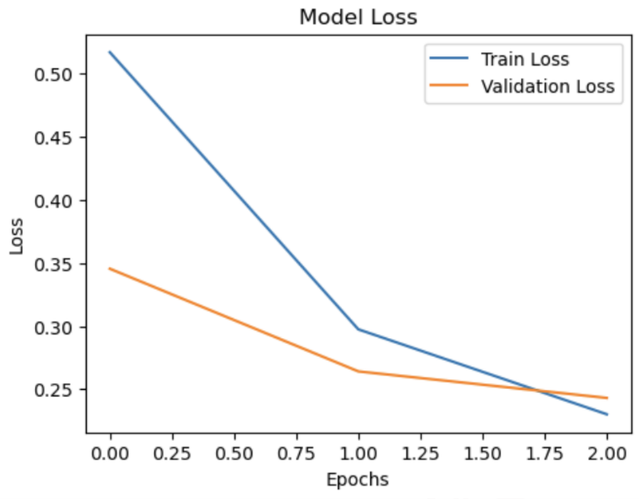


RNN

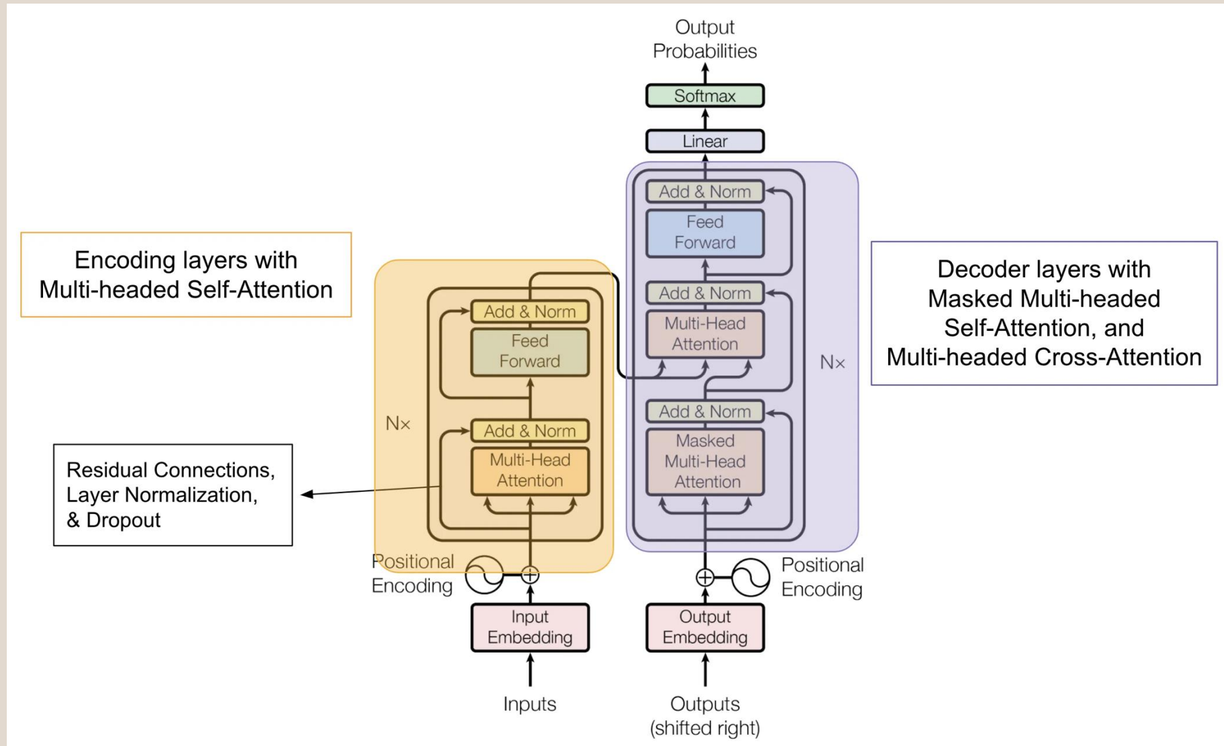


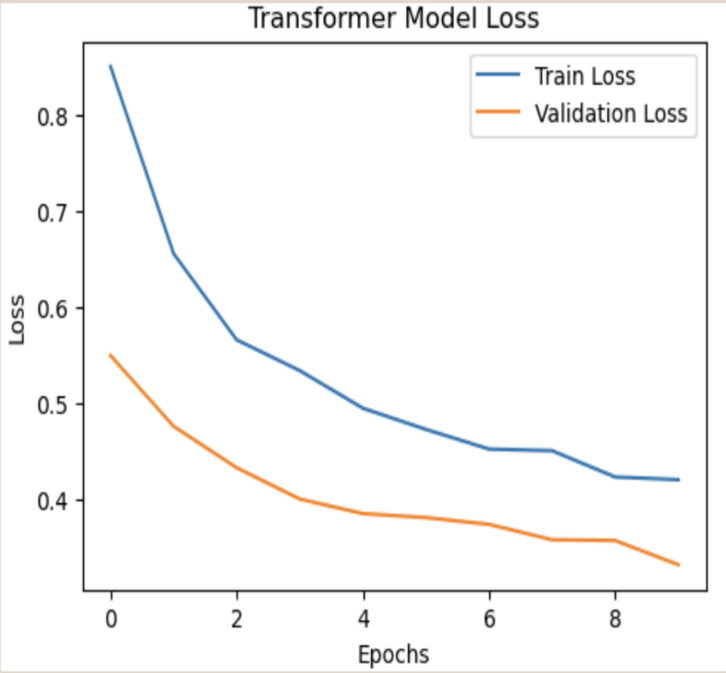

```
Classification Report:
              precision    recall  f1-score   support

        Fake       0.93      0.94      0.94       611
        Real       0.94      0.94      0.94       622

    accuracy                           0.94      1233
   macro avg       0.94      0.94      0.94      1233
weighted avg       0.94      0.94      0.94      1233


Confusion Matrix:
[[572  39]
 [ 40 582]]
```





TRANSFORMERS





MODEL PERFORMANCE COMPARISON: REKE

Model & Representation	Accuracy	Precision	Recall	F1 Score
Bi-dir RNN (TF-IDF)	0.94	0.93	0.94	0.94
Bi-dir LSTM (TF-IDF)	0.87	0.85	0.90	0.87
Bi-dir GRU (TF-IDF)	0.93	0.93	0.92	0.93
RNN (Word2Vec)	0.88	0.88	0.88	0.88
LSTM (Word2Vec)	0.88	0.89	0.87	0.88
GRU (Word2Vec)	0.87	0.89	0.85	0.87
Bi-dir RNN (Word2Vec)	0.88	0.87	0.89	0.88
Bi-dir LSTM (Word2Vec)	0.90	0.91	0.88	0.90
Bi-dir GRU (Word2Vec)	0.89	0.90	0.87	0.89
Transformer (TF-IDF)	0.90	0.86	0.95	0.90
Transformer (Word2Vec)	0.91	0.93	0.91	0.92

TESTING OUR MODEL - MLP PREDICTIONS

```python
# Example 1: Real
news_article = "three clintons iowa glimpse fire eluded hillary clintons campaign cedar rapids iowa  i one wonderful rallies entire career rig
prediction, confidence = predict_news(news_article)
print(f'Prediction: {prediction}, Confidence: {confidence:.2f}')
```

```
Prediction: Real, Confidence: 0.94
```

```python
# Example usage: Fake
news_article = "hillary clinton huge trouble america noticed sick thing hidden picture liberty writers news 0 hillary clinton barely lost pres
prediction, confidence = predict_news(news_article)
print(f'Prediction: {prediction}, Confidence: {confidence:.2f}')
```

```
Prediction: Fake, Confidence: 1.00
```

```python
# Example usage: Real
news_article = "kerry go paris gesture sympathy us secretary state john f kerry said monday stop paris later week amid criticism top american
prediction, confidence = predict_news(news_article)
print(f'Prediction: {prediction}, Confidence: {confidence:.2f}')
```

```
Prediction: Real, Confidence: 1.00
```

```python
# Example Unclead: Real
news_article = "What's in that Iran bill that Obama doesn't like?"
prediction, confidence = predict_news(news_article)
print(f'Prediction: {prediction}, Confidence: {confidence:.2f}')
```

```
Prediction: Fake, Confidence: 0.99
```

```python
# Example Unclead:Real
news_article = "The bill would give Congress a chance to hold hearings, host briefings and pave the way to a vote on a joint resolution that c
prediction, confidence = predict_news(news_article)
print(f'Prediction: {prediction}, Confidence: {confidence:.2f}')
```

```
Prediction: Fake, Confidence: 1.00
```

BERT

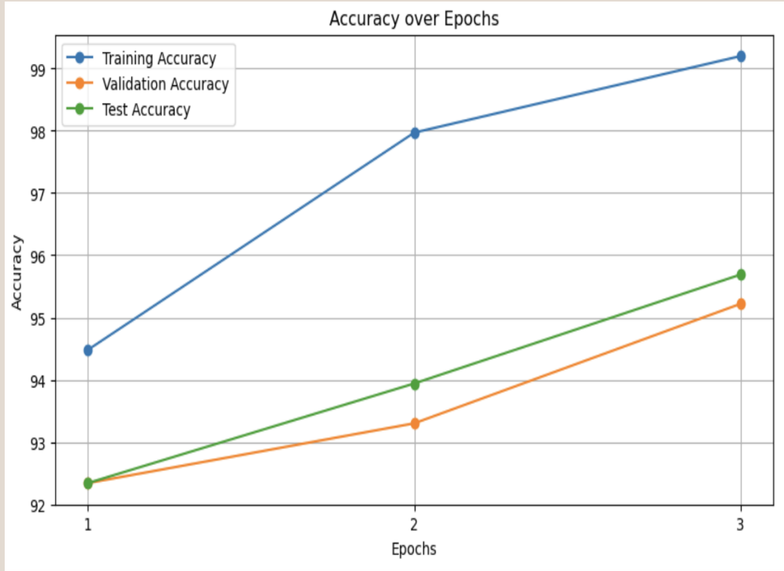
2 Next sentence prediction (NPS)

Binary classification task

Learn the relationships between sentences and predict the next sentence given the first one.

Sentence A	The man went to the store.
Sentence B	He bought a gallon of milk.
Label	IsNextSentence

Sentence A	The man went to the store.
Sentence B	Penguins are flightless.
Label	NotNextSentence



TESTING OUR MODEL - BERT PREDICTIONS

[81]:
```python
#example cleaned: Fake

from transformers import BertTokenizer
import torch

# Example text for prediction
text_to_predict  = "hillary clinton huge trouble america noticed sick thing hidden picture liberty writers news 0 hillary clinton barely lost

# Initialize the tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize and encode the input text
tokenized_input = tokenizer.encode_plus(
    text_to_predict,
    truncation=True,
    padding=True,
    return_tensors='pt'
)

# Move input tensors to the device (GPU or CPU)
input_ids = tokenized_input['input_ids']
attention_mask = tokenized_input['attention_mask']

# Ensure the model is in evaluation mode
loaded_model.eval()

# Make predictions
with torch.no_grad():
    outputs = loaded_model(input_ids, attention_mask=attention_mask)

# Get the predicted class probabilities
logits = outputs.logits
probabilities = torch.nn.functional.softmax(logits, dim=1)

# Get the predicted class
predicted_class = torch.argmax(probabilities, dim=1).item()

# Decode the predicted class using label encoder
predicted_label = label_encoder.classes_[predicted_class]

print(f"Predicted Label: {predicted_label}")
print(f"\nClass Probabilities: {probabilities.squeeze().tolist()}")
```

```
Predicted Label: FAKE

Class Probabilities: [0.9964332580566406, 0.0035667517222464085]
```

FINAL CONCLUSION

1. Significant Impact of Fake News
2. Data-Driven Insight
3. Effective Feature Engineering
4. Model Performance
5. Future Directions

FUTURE SCOPE

- Real-Time Detection & Domain-Specific Fine-Tuning
- Hybrid Models & Advanced Hyperparameter Tuning
- Enhancing Interpretability.
- In-Depth Error Analysis & Continuous Learning
- Scalable, Ethical, & Integrated Misinformation Solutions

REFERENCES

Topic	Reference Website	Description
Data Overview	Kaggle	A platform with datasets and kernels for data analysis and machine learning.
Data Cleaning	Towards Data Science	Articles on data cleaning techniques and best practices in Python.
Univariate Analysis	Analytics Vidhya	Tutorials on univariate analysis methods and visualizations.
Bivariate Analysis	DataCamp	Courses on statistical analysis and visualization techniques for exploring relationships between variables.
Hypothesis Testing	StatQuest	Clear explanations of hypothesis testing concepts and methods.
Sentiment Analysis	NLTK Documentation	Resources for performing sentiment analysis using the Natural Language Toolkit (NLTK).
Named Entity Recognition (NER)	spaCy Documentation	Comprehensive guide on using spaCy for NER tasks.
Topic Modeling	Gensim Documentation	Information on topic modeling techniques like LDA and NMF using Gensim library.
Association Rule Mining	Towards Data Science - Association Rules	Articles explaining association rule mining techniques like Apriori algorithm.
Co-occurrence Analysis	Medium - Co-occurrence Analysis	Insights into co-occurrence analysis methods in text data.

Topic	Reference Website	Description
TF-IDF	Scikit-learn Documentation	Comprehensive guide on implementing TF-IDF in Python using Scikit-learn.
GloVe Vectors	GloVe Official Website	Information and resources for GloVe word embeddings, including pre-trained models.
Logistic Regression	Towards Data Science - Logistic Regression	Tutorial on implementing logistic regression with Scikit-learn.
Random Forest	Random Forests - Scikit-learn Documentation	Detailed explanation and implementation guide for Random Forests in Scikit-learn.
Multi-Layer Perceptron (MLP)	Deep Learning with Python	Book that covers MLPs and their applications in text classification.
Convolutional Neural Networks (CNN)	KDNuggets - CNN for Text Classification	Overview of using CNNs for text classification tasks.
Recurrent Neural Networks (RNN)	Understanding LSTMs - Colah's Blog	In-depth explanation of RNNs and their variants, including LSTMs.
Long Short-Term Memory (LSTM) Networks	Towards Data Science - LSTM Networks	Tutorial on LSTM networks and their applications in sequential data analysis.
Transformers	The Illustrated Transformer	Visual explanation of the Transformer architecture and its components.
BERT	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	Original research paper introducing BERT, detailing its architecture and performance on NLP tasks.

SPECIAL THANKS
TO
PROF.ROBERTA SICILIANO AND PROF.GIUSEPPE LONGO