# Fake News Detection Project

## 1. Import Packages and Data

This section imports necessary Python libraries and datasets:

- Uses pandas for data manipulation

- NLTK for text processing (stopwords, stemming)

- Scikit-learn for machine learning models and vectorization

- Word2Vec for word embeddings

```python

import pandas as pd

import nltk

from sklearn.feature\_extraction.text import TfidfVectorizer

# ... other imports ...

```

## 2. Data Preparation

### Load and Clean Data

- Combines fake (label 1) and real (label 0) news datasets

- Removes empty text entries

- Drops irrelevant columns keeping only text and class

```python

fake\_news['Class'] = 1

true\_news['Class'] = 0

news = pd.concat([fake\_news, true\_news])

```

## 3. Text Preprocessing Pipeline

### Key Cleaning Steps:

1. \*\*Lowercasing\*\*: Standardizes text case

2. \*\*Special Character Removal\*\*: Cleans URLs, HTML tags, and non-alphanumeric characters

3. \*\*Tokenization\*\*: Splits text into individual words

4. \*\*Stopword Removal\*\*: Eliminates common uninformative words

5. \*\*Stemming\*\*: Reduces words to root forms

```python

def clean\_text(text):

text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

# ... other cleaning steps ...

return text.strip()

```

## 4. Feature Extraction Methods

### Three Vectorization Techniques:

1. \*\*Binary Count Vectorizer\*\*: Tracks word presence

2. \*\*TF-IDF\*\*: Weights words by importance

3. \*\*Word2Vec\*\*: Captures semantic meaning (300-dimension embeddings)

```python

vectorizers = [CountVectorizer(binary=True),

TfidfVectorizer(),

Word2Vec(vector\_size=300)]

```

## 5. Model Training & Evaluation

### Tested Classifiers:

- Logistic Regression (with L1/L2 regularization)

- Naive Bayes

- Support Vector Machines

- Decision Trees

\*\*Best Performers:\*\*

| Model | Vectorizer | Accuracy |

|--------------------|----------------|----------|

| Logistic Regression| Binary | 98.9% |

| SVM | TF-IDF | 98.8% |

## 6. Hyperparameter Optimization

Grid Search on Logistic Regression found:

- Optimal regularization strength (C=0.447)

- Newton-CG solver with tight tolerance (1e-5)

```python

random\_search = RandomizedSearchCV(

estimator=log\_reg,

param\_distributions=param\_grid,

n\_iter=50

)

```

## 7. Model Testing with Custom Examples

### Fake News Detection:

```python

test\_text = ["Trump announces Mars colonization plan..."]

prediction = model.predict(vectorizer.transform(clean\_text))

# Output: 1 (Fake)

```

### Real News Detection:

```python

test\_text = ["Colorado DNA analyst faces data tampering charges..."]

# Output: 0 (Real)

```

## 8. Key Predictive Features

### Strong Fake News Indicators:

- "via" (3.24)

- "sen" (1.89)

- "gop" (1.81)

### Real News Indicators:

- "said" (-2.39)

- "dont" (-1.76)

- "thursday" (-1.76)

## Conclusion

This pipeline achieves 98.9% accuracy using Logistic Regression with binary vectorization. The model effectively identifies political keywords and proper nouns as fake news markers, while neutral verbs and dates indicate real news.

```

### Overview of Key Components:

1. \*\*Importing Libraries and Data Handling\*\*: The initial phase involves importing essential packages such as pandas for data handling, NLTK for natural language processing tasks, and scikit-learn for the implementation of machine learning algorithms. The datasets for fake and real news articles are combined, with clear labeling to facilitate accurate classification.

2. \*\*Data Preparation\*\*: The document elaborates on the data cleaning process, where irrelevant columns are dropped, and only text and class labels are retained. This step also includes the removal of empty entries to ensure data integrity.

3. \*\*Text Preprocessing Pipeline\*\*: Text undergoes rigorous preprocessing, including lowercasing, removal of special characters and stopwords, tokenization, and stemming. These steps ensure the data is uniform and reduces noise, allowing the model to focus on significant textual elements.

4. \*\*Feature Extraction\*\*: Three methods are deployed for vectorization:

- \*\*Binary Count Vectorizer\*\*: This method simply records the presence of words.

- \*\*TF-IDF (Term Frequency-Inverse Document Frequency)\*\*: This technique assigns weight to words based on their significance across documents.

- \*\*Word2Vec\*\*: It produces continuous vector space representations that capture the semantic meanings of words via 300-dimensional embeddings.

5. \*\*Model Training and Results\*\*: The project evaluates several classifiers, including Logistic Regression, Naive Bayes, and Support Vector Machines (SVM). Notably, Logistic Regression with Binary Count Vectorization yields an impressive accuracy of 98.9%, with SVM slightly trailing at 98.8%. The document emphasizes the importance of hyperparameter tuning through grid search methods, optimizing crucial parameters for better performance.

6. \*\*Model Testing\*\*: The model is applied to sample news articles to demonstrate its predictive capabilities. It accurately distinguishes fake news, characterized by specific keywords, from real news which often features different linguistic patterns.

7. \*\*Key Predictive Features\*\*: The analysis identifies strong indicators for both fake and real news. Terms like "via" and "sen" serve as markers for fake news, while neutral verbs and specific days of the week provide clues towards real news articles.