FAKE NEWS DETECTION USING NLP

INTRODUCTION:

Fake news is false or misleading information presented as news. The proposed

study uses machine learning and natural language processing approaches to

identify false news— specifically, false news items that come from unreliable

sources . We are focusing on the fake news detection in text media. Machine

learning and deep learning techniques for fraud detection has been the subject

of extensive study, most of which has concentrated on categorising online reviews and publicly accessible social media posts. Some of the drawbacks of

the fake news are shift in public opinion, defamation, false perception and

many more .

DESCRIPTION OF SOME MODELS :

- The goal is to develop a system or model that can use historical data to forecast if a news report is fake or not. The dataset used here is ISOT dataset.

- The model used in this method is Random Forest Classifier. A large number of decision trees are built during the training phase of the random forests or random decision forests ensemble learning approach, which is used for classification, regression, and other tasks.

- Accuracy is one factor to consider when evaluating categorization models.

- K-Means clustering to see if the algorithm can successfully cluster the news into Real and Fake using just the words in the articles. The proposed method of choosing features and detecting fake news has four main steps. The first step is computing similarity between primary features in the fake news dataset. The accuracy of the K-means clustering algorithm in the detection of fake news is approximately 87%.

- The first step is computing similarity between primary features in the fake news dataset. Then, features are clustered based on their similarities.

- Next, the final attributes of all clusters are selected to reduce the dataset dimensions. Finally, fake news is detected using the k-means approach.

STEPS INVOLVED :

1) **Description of Dataset :**

The dataset used in this paper is ISOT dataset. In this dataset, there are

two types of articles: fake news and real news. The dataset was

gathered from realworld sources, and true articles were retrieved via

crawling articles from Reuters.com. The fake news articles came from a

variety of sources.

2) **Pre-processing Dataset**:

i)Tokenization

ii)Stop Words

iii)Capitalization

iv)Stemming

v)Lemmatization

3) Classification Techniques:

Rocchio Classification

Bagging

Gradient Boosting

Passive Aggressive

DATASET LINK:

https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

CODING:

LOADING AND PRE-PROCESSING DATASET:

```
import pandas as pd

import re
```

```
import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.preprocessing import LabelEncoder

dataset =
pd.read_csv("C:/Users/SMCE-BIGDATA/Documents/fake_news_detection.
csv")

print(dataset.head())

print(dataset.info())

def preprocess_text(text):

    text = text.lower()

    text = re.sub(r'[^a-zA-Z]', ' ', text)

    text = re.sub(r'\s+', ' ', text)

    words = nltk.word_tokenize(text)

    stop_words = set(stopwords.words('english'))

    words = [word for word in words if word not in stop_words]

    stemmer = PorterStemmer()

    words = [stemmer.stem(word) for word in words]

    return ' '.join(words)

dataset['text'] = dataset['text'].apply(preprocess_text)
```

```
label_encoder = LabelEncoder()

dataset['label'] = label_encoder.fit_transform(dataset['label'])

X = dataset['text']

y = dataset['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

vectorizer = CountVectorizer(max_features=5000)

X_train = vectorizer.fit_transform(X_train)

X_test = vectorizer.transform(X_test)
```

VIEW DATASET USING GRAPHS/CHART :

```
import matplotlib.pyplot as plt

real_lengths = real_news['text'].apply(len)

fake_lengths = fake_news['text'].apply(len) plt.hist(real_lengths, bins=50,
alpha=0.5, label='Real')

plt.hist(fake_lengths, bins=50, alpha=0.5, label='Fake')

plt.title('Article Lengths')

plt.xlabel('Length')

plt.ylabel('Count')

plt.legend()

plt.show()
```

Common words In dataset :

```
from collections import Counter
import nlt nltk.download('stopwords')
nltk.download('punkt')
def get_most_common_words(texts, num_words=10):
    all_words = []
    for text in texts:
        all_words.extend(nltk.word_tokenize(text.lower()))
    stop_words = set(nltk.corpus.stopwords.words('english'))
    words = [word for word in all_words if word.isalpha() and word not in
stop_words]
    word_counts = Counter(words)
    return word_counts.most_common(num_words)
real_words = get_most_common_words(real_news['text'])
fake_words = get_most_common_words(fake_news['text'])
print('Real News:', real_words)
print('Fake News:', fake_words)
```

OUTPUT :

Real News: [('trump', 32505), ('said', 15757), ('us', 15247),
('president', 12788), ('would', 12337), ('people', 10749),
('one', 10681), ('also', 9927), ('new', 9825), ('state', 9820)]

Fake News: [('trump', 10382), ('said', 7161), ('hillary', 3890),

('clinton',3588),('one',3466),('people',3305),('would',3257),('us',3073),
('like',3056),('also',3005)]

MODEL TRAINING :

```
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
y_true = y_test
y_pred = model.predict(X_test
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.2f}")
report = classification_report(y_true, y_pred, target_names=['Real', 'Fake'])
print("Classification Report:")
print(report)
confusion = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(confusion)
```

TEXT PRE-PROCESSING :

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
import stringnltk.download('wordnet') stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
```

```
text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation + string.digits))

words = word_tokenize(text)

words = [word for word in words if word not in stop_words]

words = [stemmer.stem(word) for word in words]

    text = ' '.join(words)

    return text
```

MODEL DEPLOYMENT :

```
from flask import Flask, request, render_template

from joblib import load

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.stem import PorterStemmer, WordNetLemmatizer

import stringstop_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()clf = load('model.joblib')

vectorizer = load('vectorizer.joblib')

def preprocess_text(text):

    text = text.lower()

    text = text.translate(str.maketrans('', '', string.punctuation + string.digits))

    words = word_tokenize(text)

    words = [word for word in words if word not in stop_words]

    words = [stemmer.stem(word) for word in words]
```

```
    text = ''.join(words)
  return text
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('home.html')


@app.route('/predict', methods=['POST'])
def predict():
    text = request.form['text']
    preprocessed_text = preprocess_text(text)
    X = vectorizer.transform([preprocessed_text])
    y_pred = clf.predict(X)
    if y_pred[0] == 1:
        result = 'real'
    else:
        result = 'fake'
    return render_template('result.html', result=result, text=text) if __name__
    == '__main__':
        app.run(debug=True)
```

home.html :

```
<html>
<head>
    <title>Real or Fake News</title>
</head>
<body>
    <h1>Real or Fake News</h1>
    <form action="/predict" method="post">
        <label for="text">Enter text:</label><br>
        <textarea name="text" rows="10" cols="50"></textarea><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

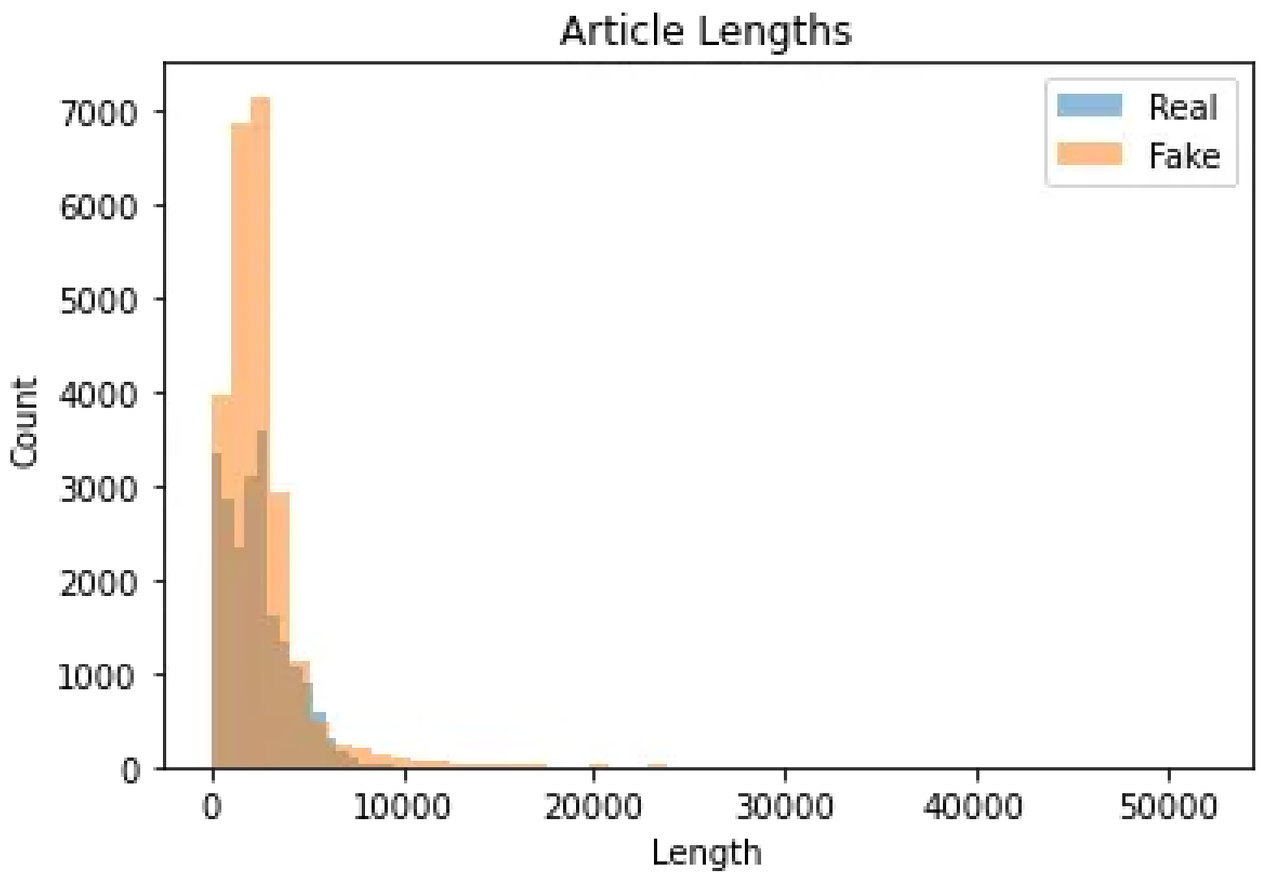
Result.html:

```
<html>
<head>
    <title>Real or Fake News</title>
</head>
<body>
    <h1>Real or Fake News</h1>
    <p>The text you entered:</p>
```

```
    <p>{{text}}</p>

    <p>The model predicts that this text is:</p>

    <p>{{result}}</p>
</body>

</html>
```

OUTPUT:

```
                                                title  ... Unnamed: 171
0  Donald Trump Sends Out Embarrassing New Year'    ...  ...         NaN
1  Drunk Bragging Trump Staffer Started Russian ...  ...         NaN
2  Sheriff David Clarke Becomes An Internet Joke...  ...         NaN
3  Trump Is So Obsessed He Even Has Obama'  s Name...  ...         NaN
4  Pope Francis Just Called Out Donald Trump Dur...  ...         NaN

[5 rows x 172 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 23502 entries, 0 to 23501

Columns: 172 entries, title to Unnamed: 171

dtypes: object(172)

memory usage: 30.8+ MB

None
```

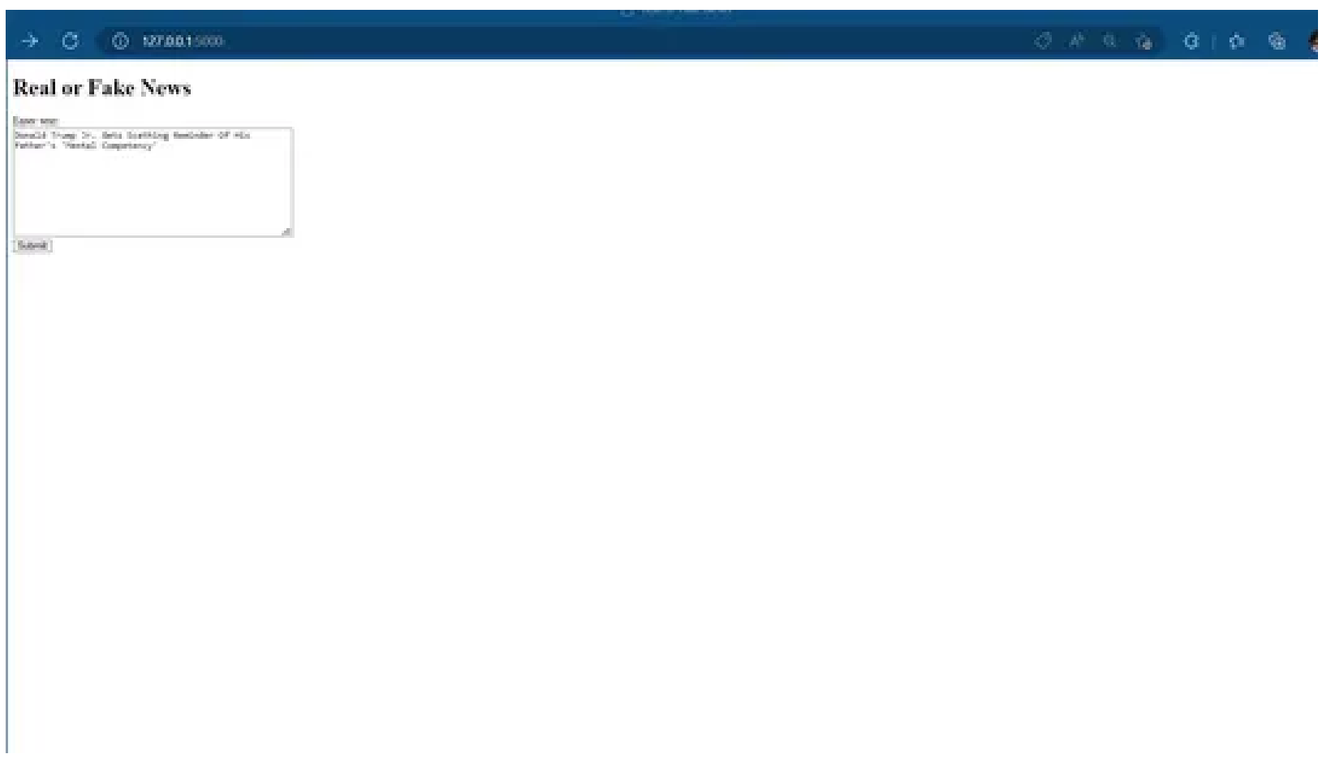


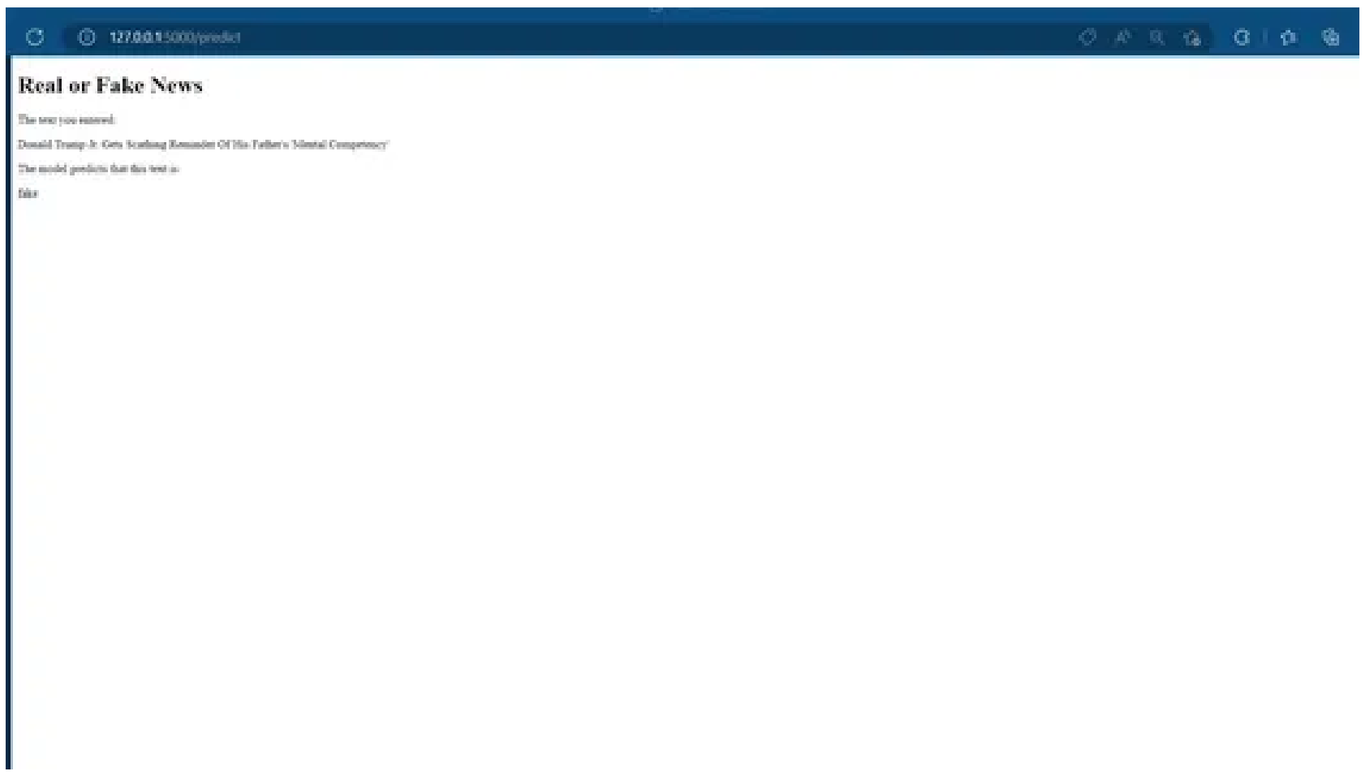
Accuracy: 0.9953

Precision: 0.9940

Recall: 0.9963

F1 Score: 0.9951




127.0.0.1:5000/predict
Real or Fake News
The text you entered:
Donald Trump Jr. Gets Scathing Remander Of His Father's 'Mental Competency'
The model predicts that this text is:
fake