PHASE- 3 ASSIGNMENT

PROJECT TITLE:FAKE NEWS DETECTION USING NLP

PROBLEM DEFINITION

The problem is to develop a fake news detection model using a Kaggle dataset. The goal is to distinguish between genuine and fake news articles based on their titles and text. This project involves using natural language processing (NLP) techniques to preprocess the text data, building a machine learning model for classification, and evaluating the model's performance.

GITHUB LINK

https://github.com/Gunalakshmiselvam/fake-news-detection-using-NLP.git

DATASETLINK

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

DOCUMENT

FAKE NEWS DETECTION USING NLP

INTRODUCTION

Fake news detection using Natural Language Processing (NLP) involves the application of machine learning techniques to analyze and identify misleading or fabricated information within textual content. Here's a step-by-step description of how fake news detection using NLP works.

Preprocessing a dataset is a crucial step in data analysis and machine learning. The specific steps you need to take can vary depending on the type of data and the problem you're trying to solve. However, here are some common preprocessing steps with code examples using Python and popular libraries like Pandas and Scikit-Learn:

Data Loading

- Load your dataset into a Pandas DataFrame.

```python
import pandas as pd

# Load a CSV file
data = pd.read_csv("your_dataset.csv")
```

Handling Missing Values

- Identify and handle missing values. Common techniques include imputation or removal of rows/columns.

```python
# Remove rows with missing values
data = data.dropna()

# Impute missing values
data['column_name'].fillna(value, inplace=True)
```

Encoding Categorical Data

- Encode categorical variables using techniques like one-hot encoding.

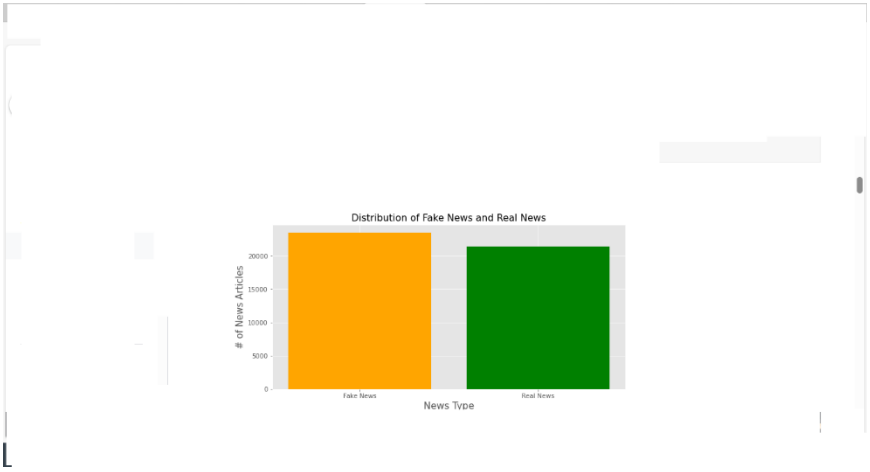
```python
# One-hot encoding
data = pd.get_dummies(data, columns=['categorical_column'])
```

Feature Scaling

- Standardize or normalize numerical features.

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data['numerical_column'] = scaler.fit_transform(data['numerical_column'].values.reshape(-1, 1))
```



Feature Selection

- Select relevant features or drop irrelevant ones.

```python
# Select specific columns
data = data[['feature1', 'feature2']]

# Or, use feature selection techniques
from sklearn.feature_selection import SelectKBest
```

selector = SelectKBest(k=5)

data = selector.fit_transform(data, target)

```

## Splitting Data

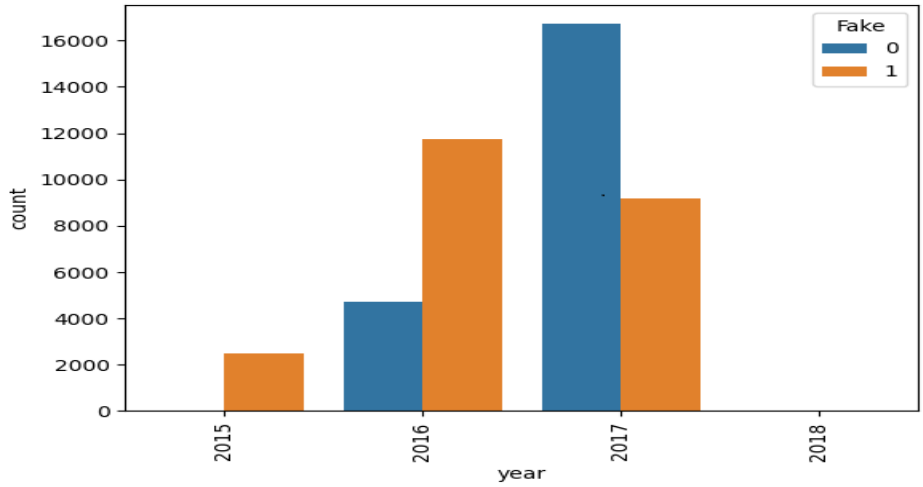
- Split your data into training and testing sets.

```python

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=42)

```



Data Transformation

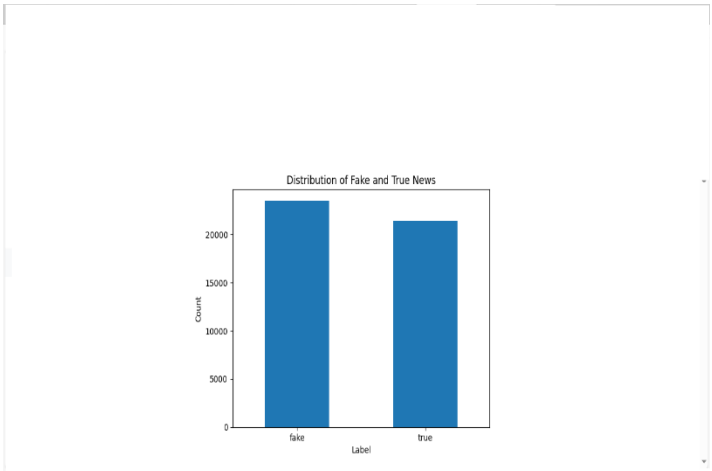
- Apply any necessary transformations to the data.

```python

# For example, applying logarithmic transformation to a feature

import numpy as np

data['feature'] = np.log(data['feature'])




```

Handling Outliers

- Detect and deal with outliers, which may involve capping or removing extreme values.

```python
# Removing outliers using z-scores
from scipy import stats
z_scores = np.abs(stats.zscore(data))
data = data[(z_scores < 3).all(axis=1)]
```

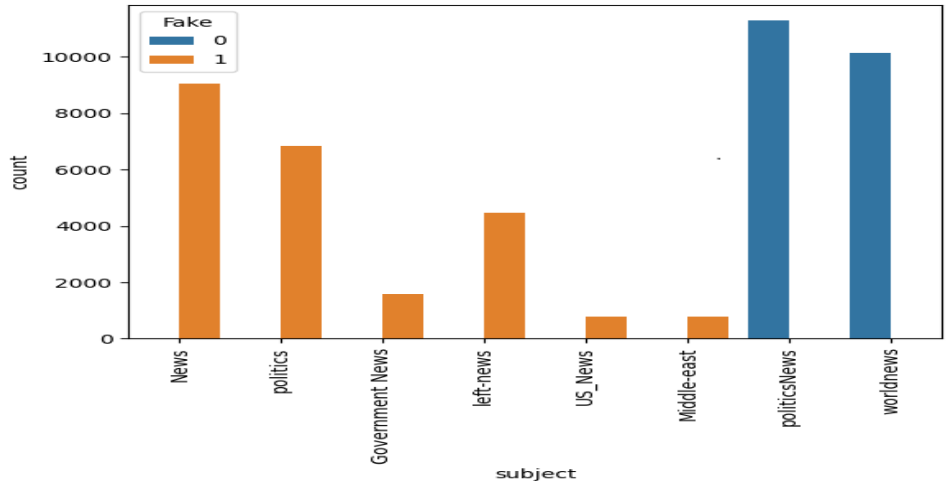
Text Data Preprocessing (if applicable)

- Tokenization, stemming, and other text-specific preprocessing steps.

Save Preprocessed Data (Optional)

- Save the preprocessed data to a file for future use.

```python
data.to_csv("preprocessed_data.csv", index=False)
```
```



SUMBITTED BY

711221104018

au711221104018