Fake News Detection

Developed By Burhan Ahmed / Email:2ba30122004@gmail.com / Phone#: 03079925825

Description:

A simple Python project to classify news as FAKE or REAL using Logistic Regression.

Libraries:

1. pandas
2. numpy
3. scikit-learn

Install these libraries using pip command.

Dataset

Searched for the proper dataset in Kaggle.
Link: Detecting Fake News

File Name: ‘fake_or_real_news.csv’

Columns:

1. title
2. text
3. label (FAKE or REAL)

Rows: 7795

Working:

1. Created new python jupyter file of name “task1.ipynb”.

2. Ensure ‘fake_or_real_news.csv’ is in the same folder.

3. **Steps:**

1) Import the rquired libraries:

```
Generate  + Code  + Markdown

import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

[9]  ✓ 0.0s

Generate  + Code  + Markdown
```

2) Load the dataset: (Also handling errors with exceptions)

```
try:
    df = pd.read_csv('fake_or_real_news.csv')
except FileNotFoundError:
    print("Error: 'fake_or_real_news.csv' not found. Ensure the file is in the same directory.")
    exit(1)

[10]  ✓ 1.2s
```

3) Create a DataFrame from the loaded dataset.

```
df = pd.DataFrame(data)

✓ 0.0s
```

4) Remove rows with missing or empty 'text' or 'label' values.

```
df = df.dropna(subset=['text', 'label'])
df = df[df['text'].str.strip() != '']
df = df[df['label'].str.strip() != '']

✓ 0.0s
```

5) Map 'FAKE' to 0 and 'REAL' to 1 in the 'label' column.

```
df['label'] = df['label'].map({'FAKE': 0, 'REAL': 1})

✓ 0.0s
```

6) Check for unmapped labels and exit if any are found.

```
if df['label'].isnull().any():
    print("Error: Some labels could not be mapped. Check the dataset for unexpected values.")
    exit(1)

✓ 0.0s
```

7) Define a function to clean text by converting to lowercase, removing punctuation, and normalizing whitespace.

```
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra whitespace
    return text
```

0.0s

Generate + Code + Markdown

8) Apply the clean_text function to the 'text' column.

```
df['text'] = df['text'].apply(clean_text)
```

3.0s

9) Initialize a TfidfVectorizer with stop words, max features, and document frequency thresholds.

```
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000, max_df=0.95, min_df=2)
```

0.0s

10) Transform the text data into TF-IDF features and extract labels.

```
X = vectorizer.fit_transform(df['text'])
y = df['label']
```

4.9s

11) Split the data into training and test sets with stratified sampling.

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

0.0s

Generate + Code + Markdown

12) Print the sizes and label distributions of training and test sets.

```
print(f"Training set size: {X_train.shape[0]} samples")
print(f"Test set size: {X_test.shape[0]} samples")
print(f"Training labels distribution: {np.bincount(y_train)} (0=FAKE, 1=REAL)")
print(f"Test labels distribution: {np.bincount(y_test)} (0=FAKE, 1=REAL)")
```

✓ 0.0s

```
Training set size: 5039 samples
Test set size: 1260 samples
Training labels distribution: [2502 2537] (0=FAKE, 1=REAL)
Test labels distribution: [626 634] (0=FAKE, 1=REAL)
```

13) Train a logistic regression model with a maximum of 1000 iterations.

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

✓ 0.1s

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

14) Evaluate the model on the test set and print accuracy and classification report.

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {accuracy * 100:.2f}%")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['FAKE', 'REAL'], zero_division=0))
```

✓ 0.0s

```
Accuracy: 92.14%

Classification Report:
              precision    recall  f1-score   support

        FAKE       0.91      0.93      0.92       626
        REAL       0.93      0.91      0.92       634

    accuracy                           0.92      1260
   macro avg       0.92      0.92      0.92      1260
weighted avg       0.92      0.92      0.92      1260
```

15) Predict the label for a new article and print the result.

```
# new_article = ["NASA announces discovery of water ice on Mars, raising prospects for future human exploration."]
# new_article = ["Secret government files reveal Bigfoot is real and living in Yellowstone National Park!"]
new_article = ["European Union leaders agree on new climate targets to reduce emissions by 50% by 2030."]
new_article_cleaned = [clean_text(new_article[0])]
new_article_transformed = vectorizer.transform(new_article_cleaned)
prediction = model.predict(new_article_transformed)
print(f"\nNew article prediction: {'REAL' if prediction[0] == 1 else 'FAKE'}")
```

[15] ✓ 0.0s

```
New article prediction: REAL
```

Notes

- Model predicts FAKE or REAL based on text patterns in the dataset.
- Change new_article in the last cell to test new articles.
- Use a larger dataset for better accuracy.
- Try models like Naive Bayes for improved performance.
- Text cleaning (lowercase, punctuation removal) is key for consistent results.