FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING

TEAM MEMBER

810621104005: ARJUNAN.E

PROJECT TITLE: Fake News Detection NLP

PHASE 3: Development part 1

Topic : Building your project by loading and preprocessing the dataset

INTRODUCTION:

Fake news Net is a repository for an ongoing data collection project for fake news research at ASU. The repository consists of comprehensive dataset of Buzz feed news and policy fact which contains two separate datasets of real and fake news. The Fake news Net consists of multi-dimension information that not only provides signals for detecting fake news but can also be used for researches such as understanding fake news propagation and fake news intervention. However, the repository is very wide and multi-dimensional, In this project, we perform a detailed analysis on Buzz feed news dataset.

The Buzz feed news dataset comprises a complete sample of news published in Facebook from 9 news agencies over a week close to the 2016 U.S. election from September 19 to 23 and September 26 and 27. Every post and the linked article were fact-checked claim-by-claim by 5 Buzz feed journalists. There are two datasets of Buzz feed news one dataset of fake news and another dataset of real news in the form of csv files, each have 91 observations and 12 features/variables.

1.IMPORT LIBRARIES:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

2.LOAD THE DATASET:

Load your dataset into pandas dataframe. You can typically find fake news detection using NLP in csv format.

PROGRAM:

```
df=pd.read_csv('/kaggle/input/emotion-dataset/Emotion_classify_Data.csv')
```

```
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
```

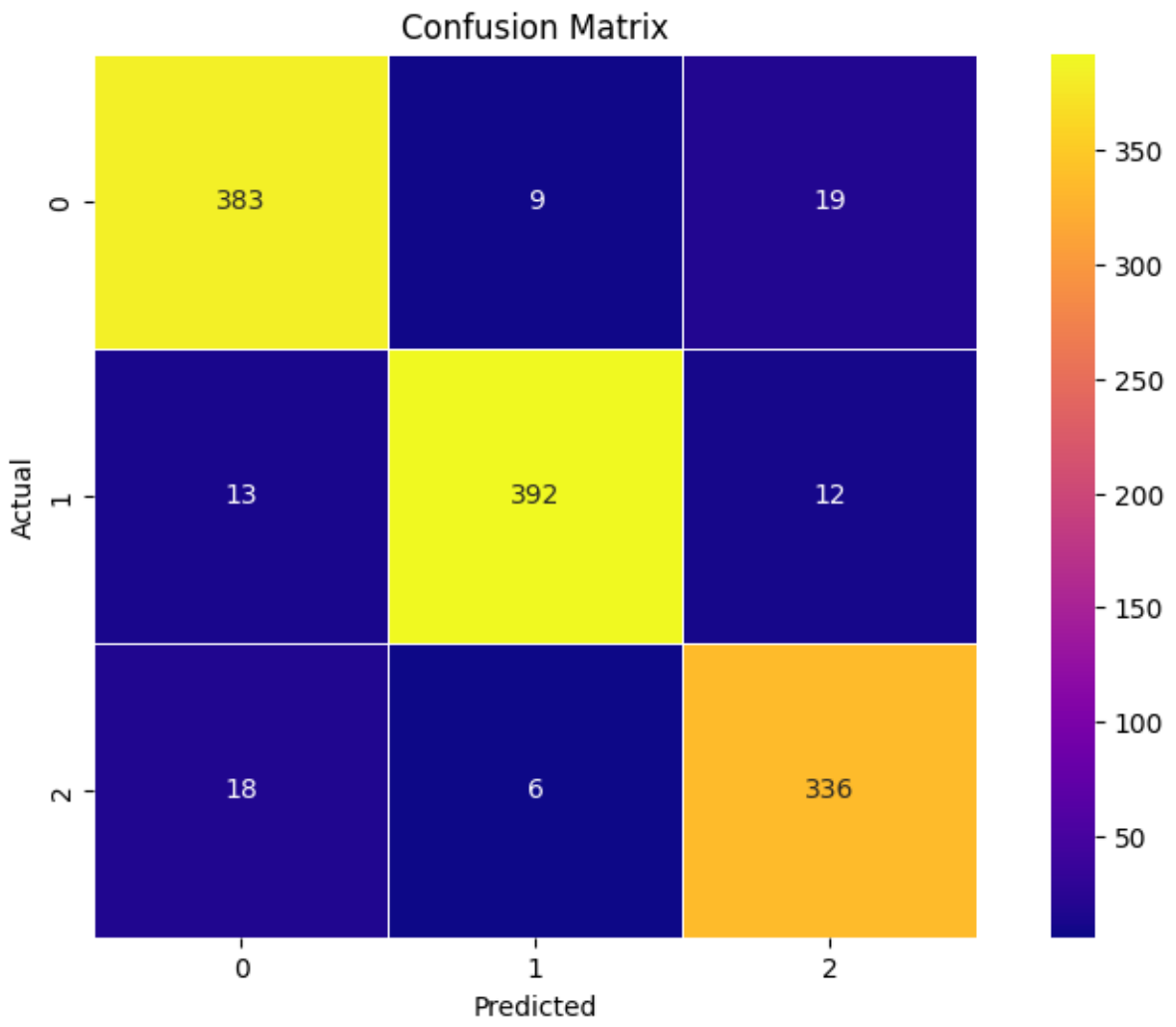
```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
clf = Pipeline([
```

```
    ('vectorizer_tfidf',TfidfVectorizer()),

  ('dec_tree', DecisionTreeClassifier())

])

clf.fit(X_train, Y_train)

prediction = clf.predict(X_test)

print(classification_report(Y_test, prediction)) from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report

from sklearn.tree import DecisionTreeClassifier

from sklearn.feature_extraction.text import TfidfVectorizer



clf = Pipeline([

    ('vectorizer_tfidf',TfidfVectorizer()),

  ('dec_tree', DecisionTreeClassifier())
```

```
])

clf.fit(X_train, Y_train)

prediction = clf.predict(X_test)

print(classification_report(Y_test, prediction))
```


Confusion Matrix
383
9
19
13
392
12
18
6
336
Actual
0
1
2
Predicted
0
1
2
350
300
250
200
150
100
50