Detecting Fake News with Python

AIM: To build a model to accurately classify a piece of news as REAL or FAKE.

This advanced python project of detecting fake news deals with fake and real news. Using sklearn, we build a TfidfVectorizer on our dataset. Then, we initialize a PassiveAggressive Classifier and fit the model. In the end, the accuracy score and the confusion matrix tell us how well our model fares.

### Prerequisites

### 1. Install Jupyter Lab and then install a library scikit also called “sklearn” in python.

pip install numpy pandas sklearn

### Steps for detecting fake news with Python

1. Make necessary imports:

import numpy as np

import pandas as pd

import itertools

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import PassiveAggressiveClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

2. Now, let’s read the data into a DataFrame, and get the shape of the data and the first 5 records.

#Read the data

df=pd.read\_csv(C:\\Users\\User\\Desktop\\news.csv')

#Get shape and head

df.shape

df.head()

3. Get the labels from the DataFrame:

#Get the labels

labels=df.label

labels.head()

4. Split the dataset into training and testing sets:

#Split the dataset

x\_train,x\_test,y\_train,y\_test=train\_test\_split(df['text'], labels, test\_size=0.2, random\_state=7)

5. Let’s initialize a TfidfVectorizer with stop words from the English language and a maximum document frequency of 0.7 (terms with a higher document frequency will be discarded). Stop words are the most common words in a language that are to be filtered out before processing the natural language data. And a TfidfVectorizer turns a collection of raw documents into a matrix of TF-IDF features.

Now, fit and transform the vectorizer on the train set, and transform the vectorizer on the test set:

#Initialize a TfidfVectorizer

tfidf\_vectorizer=TfidfVectorizer(stop\_words='english', max\_df=0.7)

#Fit and transform train set, transform test set

tfidf\_train=tfidf\_vectorizer.fit\_transform(x\_train)

tfidf\_test=tfidf\_vectorizer.transform(x\_test)

6. Next, we’ll initialize a PassiveAggressiveClassifier. This is. We’ll fit this on tfidf\_train and y\_train.

Then, we’ll predict on the test set from the TfidfVectorizer and calculate the accuracy with accuracy\_score() from sklearn.metrics.

#Initialize a PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max\_iter=50)

pac.fit(tfidf\_train,y\_train)

# Predict on the test set and calculate accuracy

y\_pred=pac.predict(tfidf\_test)

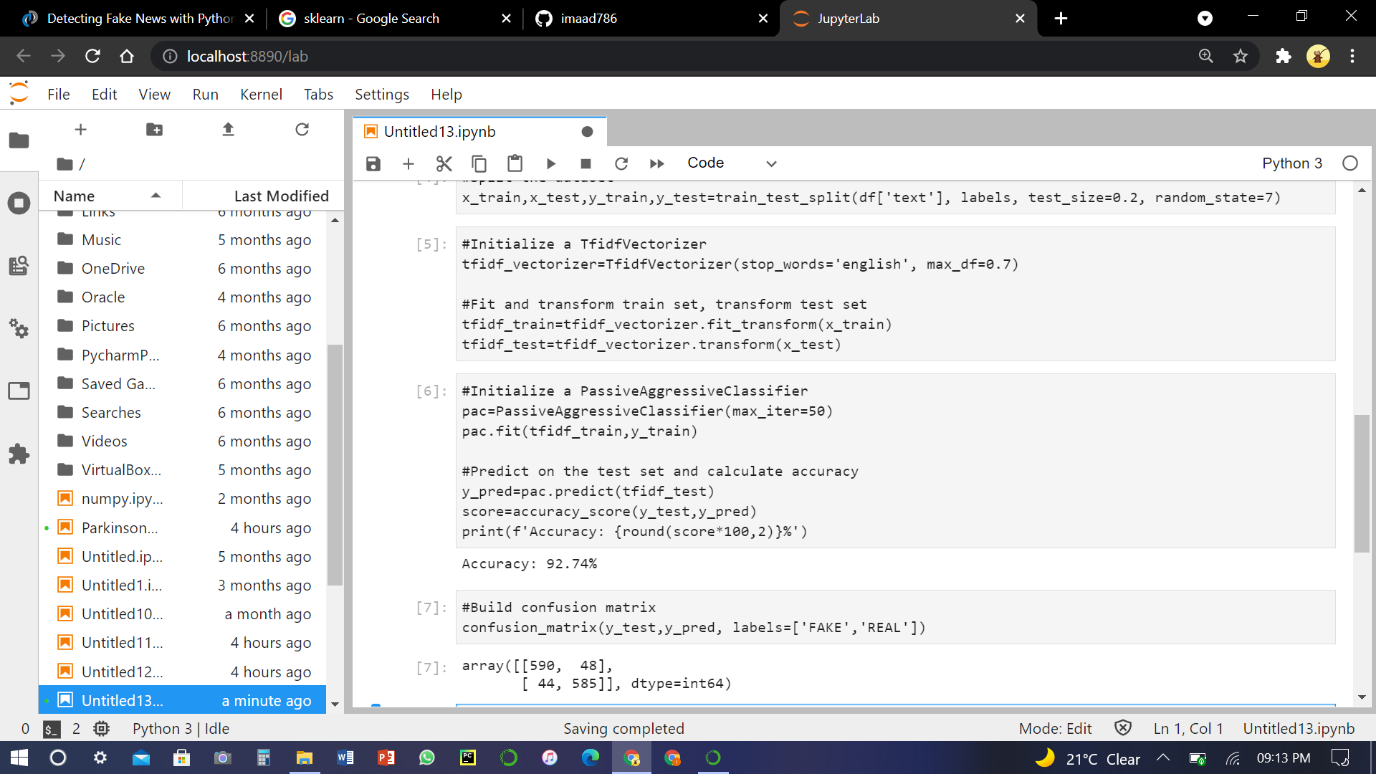
score=accuracy\_score(y\_test,y\_pred)

print(f'Accuracy: {round(score\*100,2)}%')

7. We got an accuracy of 92.82% with this model. Finally, let’s print out a confusion matrix to gain insight into the number of false and true negatives and positives.

# Build confusion matrix

confusion\_matrix(y\_test,y\_pred, labels=['FAKE','REAL'])



So with this model, we have 589 true positives, 587 true negatives, 42 false positives, and 49 false negatives with 92.74% accuracy.