AI-Model

FAKE NEWS DETECTION

PHASE 4 : DEVELOPMENT PART 2

Problem Statement:

Fake News Classification with The Help Of Natural Language Processing Technique. Fake news detection is a hot topic in the field of natural language processing. We consume news through several mediums throughout the day in our daily routine, but sometimes it becomes difficult to decide which one is fake and which one is authentic. Our job is to create a model which predicts whether a given news is real or fake.

Problem :

The problem is not only hackers, going into accounts, and sending false information. The bigger problem here is what we call "Fake News". A fake are those news stories that are false: the story itself is fabricated, with no verifiable facts, sources, or quotes.

When someone (or something like a bot) impersonates someone or a reliable source to false spread information, that can also be considered as fake news. In most cases, the people creating this false information have an agenda, that can be political, economical or to change the behavior or thought about a topic.

There are countless sources of fake news nowadays, mostly coming from programmed bots, that can't get tired (they're machines hehe) and continue. to spread false information 24/7.

Serious studies in the past 5 years, have demonstrated big correlations between the spread of false information and elections, the popular opinion or feelings about different topics.

The problem is real and hard to solve because the bots are getting better are tricking us. Is not simple to detect when the information is true or not all the time, so we need better systems that help us understand the patterns of fake news to improve our social media, communication and to prevent confusion in the world.

Process :

In this short code , I'll explain several ways to detect fake news using collected data from different articles. But the same techniques can be applied to different scenarios. For the coders and experts, I'll explain the Python code to load, clean, and analyze data. Then we will do some machine learning models to perform a classification task (fake or not).

Required Libraries :

```
import pandas as pd

import numpy as np

import re

import nltk
```

```
from nltk.corpus import stopwords

from nltk.stem import PorterStemmer, WordNetLemmatizer

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

1. Data Gathering

df = pd.read_csv("News_dataset.csv")

df.head()

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \r\nAn Iranian woman has been sentenced ...	1

2. Data Analysis

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

df['label'].value_counts()

```
1    10413
```

0 10387

Name: label, dtype: int64

df.shape

(20800, 5)

df.isna().sum()

id 0

title 558

author 1957

text 39

label 0

dtype: int64

df = df.dropna() #Handled Missing values by droping those rows

df.isna().sum()

id 0

title 0

author 0

text 0

label 0

dtype: int64

df.shape

(18285, 5)

df.reset_index(inplace=True)

df.head()

	index	id	title	author	text	label
0	0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \r\nAn Iranian woman has been sentenced ...	1

df['title'][0]

'House Dem Aide: We Didn't Even See Comey's Letter Until Jason Chaffetz Tweeted It'

df = df.drop(['id','text','author'],axis = 1)

df.head()

	index	title	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	0
2	2	Why the Truth Might Get You Fired	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	1
4	4	Iranian woman jailed for fictional unpublished...	1

3. Data Preprocessing

1.Tokenization

sample_data = 'The quick brown fox jumps over the lazy dog'

sample_data = sample_data.split()

sample_data

['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

2. Make Lowercase

sample_data = [data.lower() for data in sample_data]

sample_data

['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

3. Remove Stopwords

stopwords = stopwords.words('english')

print(stopwords[0:10])

print(len(stopwords))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]

179

sample_data = [data for data in sample_data if data not in stopwords]

print(sample_data)

len(sample_data)

['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog']

6

4. Stemming

ps = PorterStemmer()

sample_data_stemming = [ps.stem(data) for data in sample_data]

print(sample_data_stemming)

['quick', 'brown', 'fox', 'jump', 'lazi', 'dog']

5. Lemmatization

lm = WordNetLemmatizer()

sample_data_lemma = [lm.lemmatize(data) for data in sample_data]

print(sample_data_lemma)

['quick', 'brown', 'fox', 'jump', 'lazy', 'dog']

```
lm = WordNetLemmatizer()

corpus = []

for i in range (len(df)):

  review = re.sub('^a-zA-Z0-9',' ', df['title'][i])

  review = review.lower()

  review = review.split()

  review = [lm.lemmatize(x) for x in review if x not in stopwords]

  review = " ".join(review)

  corpus.append(review)
```

len(corpus)

18285

df['title'][0]

'House Dem Aide: We Didn’t Even See Comey’s Letter Until Jason Chaffetz Tweeted It'

corpus[0]

'house dem aide: didn’t even see comey’s letter jason chaffetz tweeted'

4. Vectorization (Convert Text data into the Vector)

```
tf = TfidfVectorizer()
```

x = tf.fit_transform(corpus).toarray()

x

array([[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

...,

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.]])

y = df['label']

y.head()

0 1

1 0

2 1

3 1

4 1

Name: label, dtype: int64

Data splitting into the train and test

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 10, stratify = y)

len(x_train),len(y_train)

(12799, 12799)

len(x_test), len(y_test)

(5486, 5486)

5. Model Building

rf = RandomForestClassifier()

rf.fit(x_train, y_train)

RandomForestClassifier()

6. Model Evaluation

y_pred = rf.predict(x_test)

accuracy_score_ = accuracy_score(y_test,y_pred)

accuracy_score_

0.9385709077652206

class Evaluation:

  def _init_(self,model,x_train,x_test,y_train,y_test):

    self.model = model

    self.x_train = x_train

```
    self.x_test = x_test
    self.y_train = y_train
    self.y_test = y_test

def train_evaluation(self):
    y_pred_train = self.model.predict(self.x_train)

    acc_scr_train = accuracy_score(self.y_train,y_pred_train)
    print("Accuracy Score On Training Data Set :",acc_scr_train)
    print()

    con_mat_train = confusion_matrix(self.y_train,y_pred_train)
    print("Confusion Matrix On Training Data Set :\n",con_mat_train)
    print()
    class_rep_train = classification_report(self.y_train,y_pred_train)
    print("Classification Report On Training Data Set :\n",class_rep_train)

        def test_evaluation(self):
```

y_pred_test = self.model.predict(self.x_test)

acc_scr_test = accuracy_score(self.y_test,y_pred_test)

print("Accuracy Score On Testing Data Set :",acc_scr_train)

print()

con_mat_test = confusion_matrix(self.y_test,y_pred_test)

print("Confusion Matrix On Testing Data Set :\n",con_mat_train)

print()

class_rep_test = classification_report(self.y_test,y_pred_test)

print("Classification Report On Testing Data Set :\n",class_rep_train

#Checking the accuracy on training dataset

Evaluation(rf,x_train, x_test, y_train, y_test).train_evaluation()

```
Accuracy Score On Training Data Set : 1.0

Confusion Matrix On Training Data Set :
 [[7252    0]
 [   0 5547]]

Classification Report On Training Data Set :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7252
           1       1.00      1.00      1.00      5547

    accuracy                           1.00     12799
   macro avg       1.00      1.00      1.00     12799
weighted avg       1.00      1.00      1.00     12799
```

#Checking the accuracy on testing dataset

Evaluation(rf,x_train, x_test, y_train, y_test).test_evaluation()

```
Accuracy Score On Testing Data Set : 0.9385709077652206

Confusion Matrix On Testing Data Set :
 [[2825  284]
 [  53 2324]]

Classification Report On Testing Data Set :
              precision    recall  f1-score   support

           0       0.98      0.91      0.94      3109
           1       0.89      0.98      0.93      2377

    accuracy                           0.94      5486
   macro avg       0.94      0.94      0.94      5486
weighted avg       0.94      0.94      0.94      5486
```