Fake news detection using Natural language processing

Date	11 October 2023
Project Id	Proj_212171_Team 1
Project name	Fake news detection using nlp
Maximum marks	

Importing libraries and dependencies

```
import numpy as np
import pandas as pd
import re
import nltk from nltk.corpus
 import stopwords
from nltk.stem.porter
 import PorterStemmer
import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model
import LogisticRegression
from sklearn.metrics
import accuracy_score
```

Covering the importance of each library/module/function that we imported:

NumPy : It is a general-purpose array and matrices processing package.

Pandas : It allows us to perform various operations on datasets.

re : It is a built-in RegEx package, which can be used to work with Regular Expressions.

NLTK : It is a suite of libraries and programs for symbolic and statistical natural language processing (NLP).

nltk.corpus : This package defines a collection of corpus reader classes, which can be used to access the contents of a diverse set of corpora.

stopwords : The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. (Example-and, of, are etc.)

PorterStemmer : A package to help us with stemming of words. (More about stemming in the Data Preprocessing section)

Sci-kit Learn (sklearn) : It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

feature_extraction.text : It is used to extract features in a format supported by machine learning algorithms from datasets consisting of text.

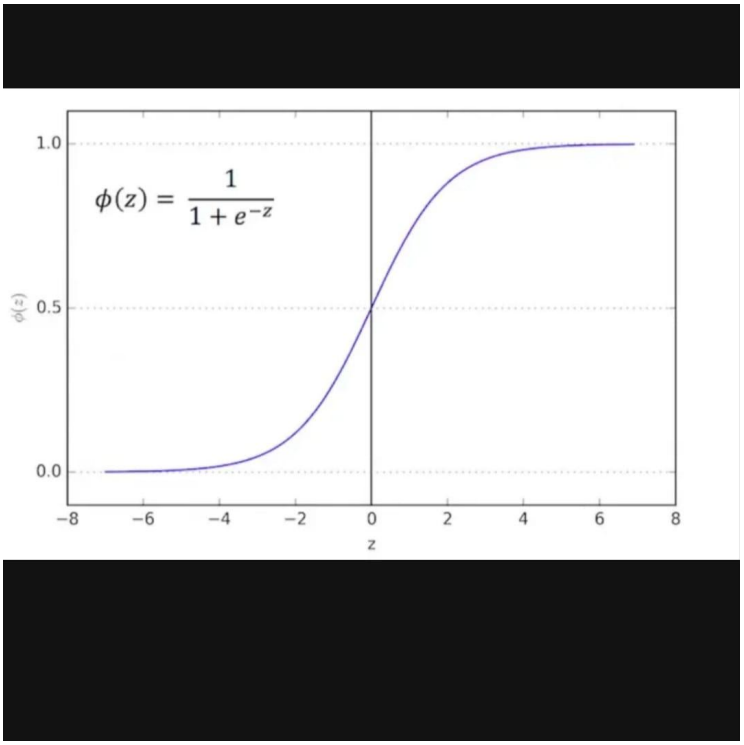
TfidfVectorizer : It transforms text to feature vectors that can be used as input to estimator. (More about TfidfVectorizer in the Data Preprocessing section)

LogisticRegression : A pretty self explanatory part of the code, used to import the Logistic Regression Classifier.

metrics and accuracy_score : To import Accuracy classification score from the metrics module.

logistic Regression

Before diving into the code let us revise the Logistic Regression concept. Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no(binary classification), based on prior observations of a data set. It is a Supervised statistical technique to find the probability of dependent variable. The graph shown below is a Sigmoid Function, which we also call as a Logit. This function converts the probabilities into binary values which could be further used for predictions.



According to this graph, if we obtain the probability value to be less than 0.5, then it is considered to be of the Class 0 and if the value is more than 0.5, then it would be a part of Class 1.

Loading the dataset

I hope you have downloaded the dataset by now. Now you can load the dataset as,

```
data = pd.read_csv('fakenews.csv')
data.head()
```

Here I have renamed my csv file as fakenews.csv and saved it in the same folder as my jupyter notebook. If you saved your dataset and the jupyter notebook in 2 different folders, you can add the path of the dataset file as the prefix in the code as,

```
data = pd.read_csv('/Users/bala/Documents/fakenews.csv')
```

(This is a macbook path, so if you are using Windows or any other OS, the path might look different.)

	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 \nAn Iranian woman has been sentenced to...	1

Here Label indicates whether a news article is fake or not, 0 denotes that it is Real and 1 denotes that it is Fake.

Data preprocessing

After importing our libraries and the dataset, it is important to preprocess the data before we train our ML model since there might be some anomalies and missing datapoints which might make our predictions a bit skewed from the actual values. Now, we can check the size of the dataframe/table as it would decide whether we can drop the rows with null values without affecting the size of our dataset or not.

data.shape

This gives us (20800, 5) which means that we have 20800 number of entries and 5 columns (features).

Checking the total number of missing values in each of the columns.

data.isnull().sum()

Imagine---)))

```
id          0
title     558
author   1957
text       39
label       0
dtype: int64
```

From this we can see that we will have to delete a minimum of 1957 lines to remove all the null values so it would be better to fill these null values with an empty string. For that we can use fillna.

df1 = data.fillna('')

After this step we no longer have any missing datapoints, you can check that using the isnull().sum()

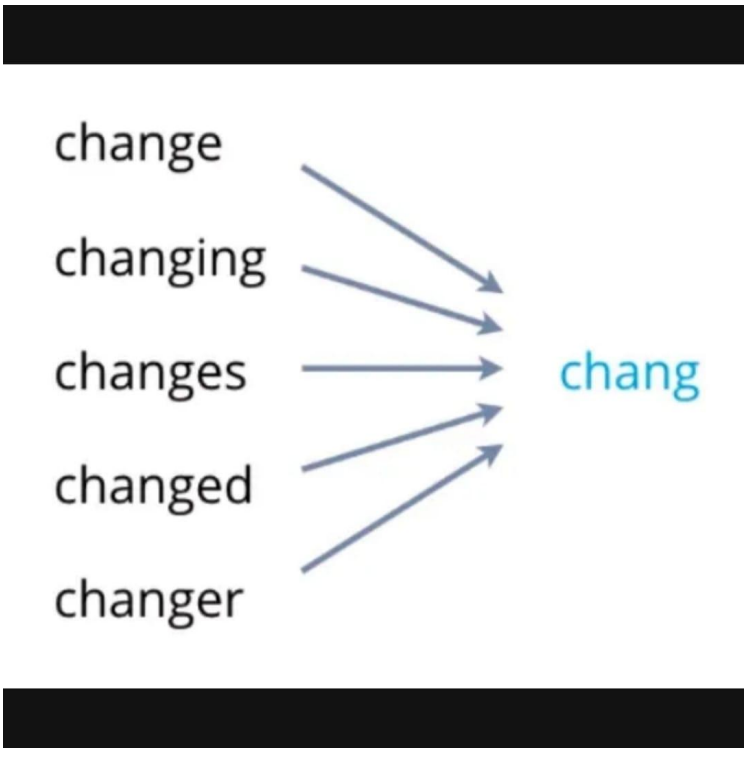
Now, we'll try to reduce those 5 columns to only 2 columns since it will be easier for us to train the model. For that we'll combine the title and the author columns into one, naming it as content. We can drop the other columns as they don't have much effect on determining whether the article is fake or not. This step will leave us with 2 columns - content and label.

df1['content'] = df1['author'] + ' ' + df1['title']

Stemming

Now coming to the stemming part, it basically is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words.

Image-)))



Stemming of words might or might not end up with a root word with meaning, like in this example chang doesn't mean change or anything as a matter of fact. For the root word to have meaning we use lemmatization. But for this project stemming works just fine.

stemmer = PorterStemmer()
[10/10, 6:20 PM] Bala Frd: We create a new Porter stemmer for us so that we can use the function without explicitly typing PorterStemmer() every time.

Code

```
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ', content) #1
    stemmed_content = stemmed_content.lower() #2
    stemmed_content = stemmed_content.split() #3
    stemmed_content = [stemmer.stem(word) for word in stemmed_content if not
word in stopwords.words('english')] #4
    stemmed_content = ' '.join(stemmed_content) #5
    return stemmed_content #6
```

Okay, so let's go in depth and see what this function actually does. I have numbered each line from 1 to 6 so that you can easily distinguish between different lines of code and understand each line's use.

#1 First we use the re package and remove everything that is not a letter (lower or uppercase letters).

#2 We then convert every uppercase letter to a lower one.

#3 We then split the each sentence into a list of words.

#4 Then we use the stemmer and stem each word which exists in the column and remove every english stopword present in the list.

#5 We then join all these words which were present in the form of a list and convert them back into a sentence.

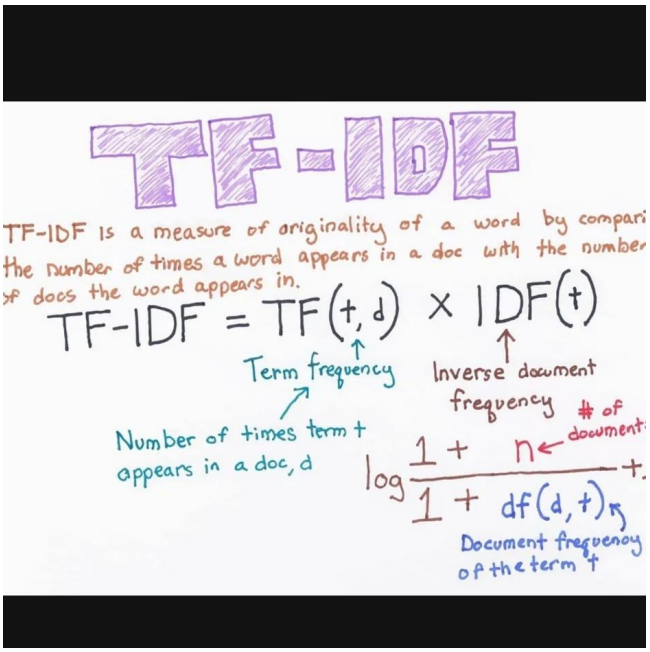
#6 Finally we return the stemmed_content which has been preprocessed.
Applying this function to our dataset,

```
df1['content'] = df1['content'].apply(stemming)
df1['content'].head()
```

```
0    darrel lucu hous dem aid even see comey letter...
1    daniel j flynn flynn hillari clinton big woman...
2               consortiumnew com truth might get fire
3    jessica purkiss civilian kill singl us airstri...
4    howard portnoy iranian woman jail fiction unpu...
Name: content, dtype: object
```

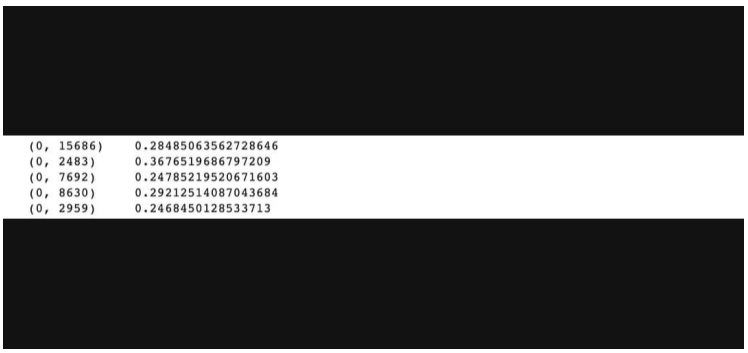
Next step is to name our input and output features

X = df1.content.values
y = df1.label.values
Our last preprocessing step would be to transform our textual X to numerical so that our ML model can understand it and can work with it. This is where TfidfVectorizer comes into play. Here is a picture explaining it in brief,



X = TfidfVectorizer().fit_transform(X)
print(X)

Output of this code is



Splitting the Dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, stratify = y, random_state = 2)

This means that we have divided our dataset into 80% as training set and 20% as test set. stratify = y implies that we have made sure that the division into train-test sets have around equal distribution of either classes (0 and 1 or Real and Fake). random_state = 2 will guarantee that the split will always be the same.