DATE	**25/10/2023**
PROJECT_ID	**AU212171**
NAME	**Fake news detection using NLP**

ABSTRACT:

Fake news is a peculiarity which is fundamentally effecting our public activities through social media platforms. The data acquracy on internet particularly via online media. The fake news detection is a subtask of text classification and is often defined as task of classifying news as real or fake. There are different social media platforms that area cessible to these users. A human being is unable to detect all these fake news. So there is a need for a machine learning classifiers that can detect these fake news automatically. This strategy utilizes NLP Classification model(logistic Regression) to anticipate whether the news from the social media is real or fake . With this undertaking we are attempting to get high exactness and furthermore decrease an opportunity to distinguish the Fake News. Key Words: Conterfeit news, Real, Fake, Logical Regression. I. INTRODUCTION: Humans are ureliable detector of fake news. This is because people are susceptible to bias. People tend to believe those that does not contradict their preconceived ideas. The fake news is a wonder which is altogether influencing our public movement. Fake news area is a rising investigation district which is getting interest with the resourcesavailable. In this paper we have displayed anacknowledgement model for fake news using logical regressio

metholodologies.In the last decade, Fake News phenomenon has experienced a very significant spread, favored by social networks. This fake newscan be broadcasted for different purposes. Hence there is a requirement of a technology where the human being can understand and react to such fake news. Hence the fake news detection is a bang for all of it .

PROPOSED METHOD:

There are some past researches conducted in conjunction with fake news detection in various platform which uses the capability for Logistic Regression that makes the predictions in text classifications. Below are some existing findings that described the usage of this machine learning algorithm. The Logistic Regression is quite goodin solving binary classifications due to its predictive power inprobability values are taken. Logistic Regression detection model works well in dealing and also short input text and the range of accuracy can be achieved is within 79.0% to 89.0% based on the data on the table. The algorithm that used to predect is depends on logical regression, and the binary variable that contains the code yes, success etc or no, falure, etc for yes it takes 0 and for no it takes 1, and in other words, the logistic regression model predits as P(y=1) as a function of x.

Logistic Regression:

Before diving into the code let us revise the Logistic Regression

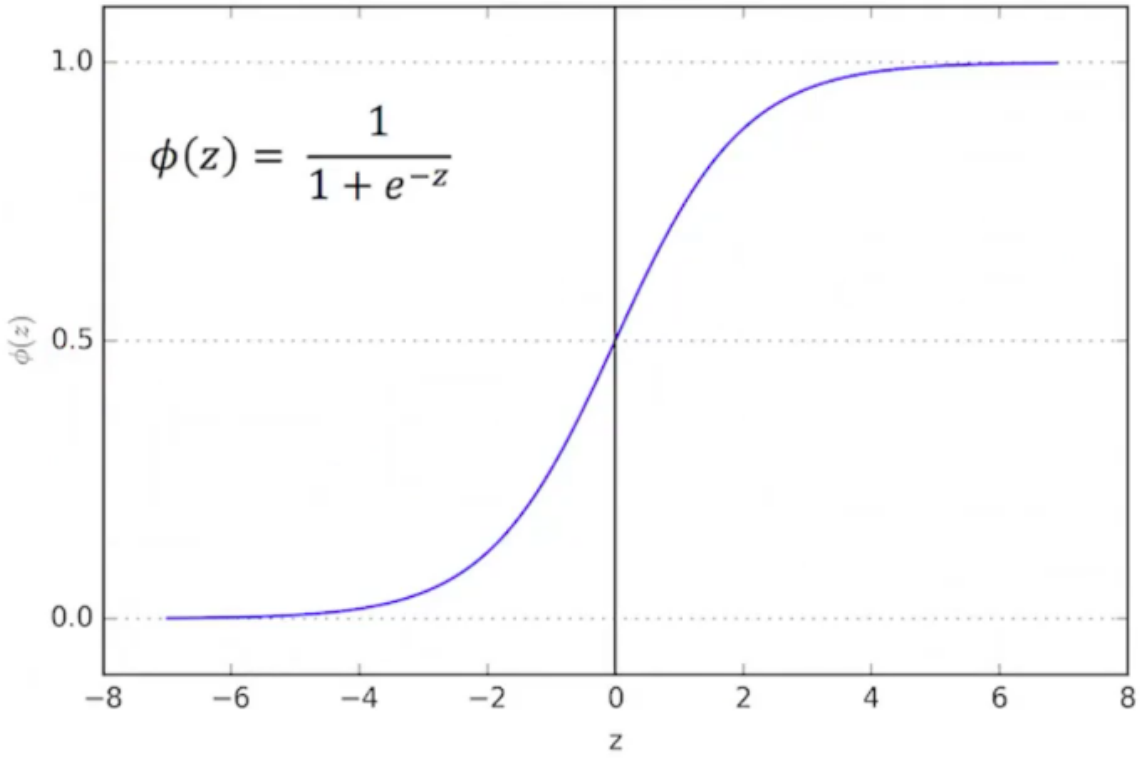
concept. Logistic regression is a statistical analysis method to predict a

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.

Graphical representation:



Sigmoid Function from rasbt

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**.

Code:

Now finally starting off with our code, you can either write it in your Jupyter Notebook or Google Colab or any other platform you like.

Also download your dataset from ***here***. (I only used the training dataset so you can also download that itself.)

Importing libraries/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_scor
```

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.)

The dataframe will look like this,

	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()
```

```
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('')
```

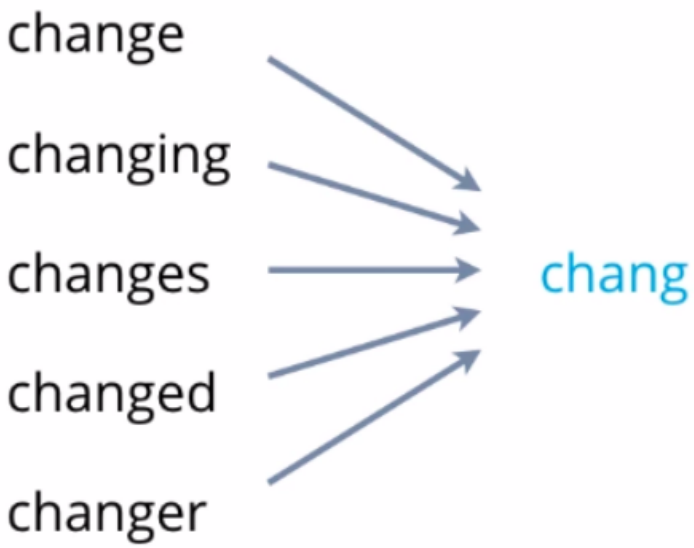
After this step we no longer have any missing datapoints, you can check that using the isnull().sum()

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.



Stemming example

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()
```

We create a new Porter stemmer for us so that we can use the function without explicitly typing PorterStemmer() every time.

```
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
```

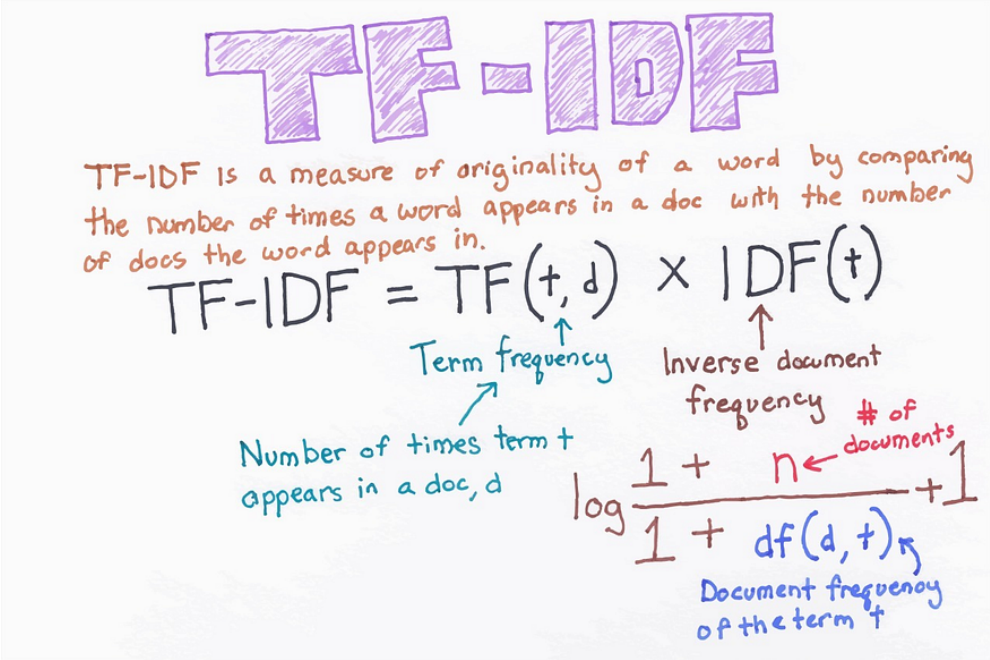
Next step is to name our input and output features

```
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,



Source: https://becominghuman.ai/word-vectorizing-and-statistical-meaning-of-tf-idf-d45f3142be63

To understand it in depth, visit ***this link***.

```
X = TfidfVectorizer().fit_transform(X)
```

```
print(X)
```

The output of this code should like this,

```
  (0, 15686)    0.28485063562728646
  (0, 2483)     0.3676519686797209
  (0, 7692)     0.24785219520671603
  (0, 8630)     0.29212514087043684
  (0, 2959)     0.2468450128533713
```

Now that we have the X in our desired form, we can move onto the next step.

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.

Training the Model

Fitting the model to our dataset

```
model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

Now that we have trained it, let's check the accuracy of our training set predictions,

```
X_train_prediction = model.predict(X_train)
```

```
training_accuracy = accuracy_score(X_train_prediction, y_train)
```

```
print(training_accuracy)
```

```
0.9865985576923076
```

Training accuracy score

So I got about 98.66%, which is pretty good. Similarly for the test dataset.

```
X_test_prediction = model.predict(X_test)
```

```
testing_accuracy = accuracy_score(X_test_prediction, y_test)
```

```
print(testing_accuracy)
```

```
0.9790865384615385
```

Test accuracy score

So test accuracy is also pretty good.

(Note: The score may differ for you if you make any changes to this code)

With this we have successfully trained our ML model!

Building a system

Finally to make this model useful we need to make a system. Taking a sample out of the test-set (I took the first sample),

```
X_sample = X_test[0]
```

Checking our prediction for this sample,

```
prediction = model.predict(X_sample)

if prediction == 0:

   print('The NEWS is Real!')

else:

   print('The NEWS is Fake!')
```

```
The NEWS is Fake!
```

With this we have built a system as well. Now if you want to take it a step further, try inputing a textual sample and predict using that. You can now give yourself a pat on the back as you now know how to detect a Fake News article using Logistic Regression only!!!