DECEPTION DETECTION

#Importing the Dependencies

import numpy as np

import pandas as pd

import re

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

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

import nltk

nltk.download('stopwords')

# printing the stopwords in English

print(stopwords.words('english'))

#Data Pre-processing

import pandas as pd # Import the pandas library with the alias 'pd'

news\_dataset = pd.read\_csv('/content/train.csv')

news\_dataset.shape

# print the first 10 rows of the dataframe

news\_dataset.head(10)

# counting the number of missing values in the dataset

news\_dataset.isnull().sum()

# replacing the null values with empty string

news\_dataset = news\_dataset.fillna('')

# merging the author name and news title

news\_dataset['content'] = news\_dataset['author']+' '+news\_dataset['title']

print(news\_dataset['content'])

# separating the data & label

X = news\_dataset.drop(columns='label', axis=1)

Y = news\_dataset['label']

print(X)

print(Y)

#Stemming

port\_stem = PorterStemmer()

#Transforming Data

def stemming(content):

    stemmed\_content = re.sub('[^a-zA-Z]',' ',content)

    stemmed\_content = stemmed\_content.lower()

    stemmed\_content = stemmed\_content.split() #splits sentence into words

    stemmed\_content = [port\_stem.stem(word) for word in stemmed\_content

                       if not word in stopwords.words('english')]

    stemmed\_content = ' '.join(stemmed\_content)

    return stemmed\_content

#Extracting data

news\_dataset['content'] = news\_dataset['content'].apply(stemming) #content is updated

print(news\_dataset['content'])

#separating the data and label

X = news\_dataset['content'].values #extracts content from columns

Y = news\_dataset['label'].values #extracts output(fake 1, real 0) from label column

print(X)

print(Y)

Y.shape

# converting the textual data to numerical data

vectorizer = TfidfVectorizer()

vectorizer.fit(X)

X = vectorizer.transform(X)

print(X)

#Splitting the dataset to training & test data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, stratify=Y, random\_state=2)

#Training the Model: Logistic Regression

model = LogisticRegression()

model.fit(X\_train, Y\_train)

#Evaluation

# accuracy score on the training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy score of the training data : ', training\_data\_accuracy)

# accuracy score on the test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)A

print('Accuracy score of the test data : ', test\_data\_accuracy)

#F1-score

from sklearn.metrics import f1\_score

# Accuracy

print('Accuracy score of the training data : ', training\_data\_accuracy)

print('Accuracy score of the test data : ', test\_data\_accuracy)

# F1 Score

train\_f1\_score = f1\_score(Y\_train, X\_train\_prediction)

print('F1 Score of the training data : ', train\_f1\_score)

test\_f1\_score = f1\_score(Y\_test, X\_test\_prediction)

print('F1 Score of the test data : ', test\_f1\_score)

X\_new = X\_test[]

prediction = model.predict(X\_new)

print(prediction)

if (prediction[0]==0):

  print('The news is Real')

else:

  print('The news is Fake')