

**GOVERNMENT COLLEGE OF TECHNOLOGY**

**COIMBATORE**

(An Autonomous Institution Affiliated to Anna University)

**COLLEGE CODE-7177**

**PROJECT TITLE: FAKE NEWS DETECTION USING NLP**

**TEAM MEMBERS:**

**SUBASH S -71772114146**

**ATCHAYA A -71772114104**

**ARUNTHATHI R -71772114103**

**RITHISHA P -71772114L01**

**PROBLEM STATEMENT:**

It involves developing algorithms and models to accurately identify and classify misinformation in news articles or social media posts. The goal is to create a system that can distinguish between reliable and fake news sources based on various factors such as credibility, source reputation, fact-checking, and content analysis. The challenge lies in designing effective algorithms that can analyse large amounts of data and detect patterns or indicators of fake news with high accuracy.

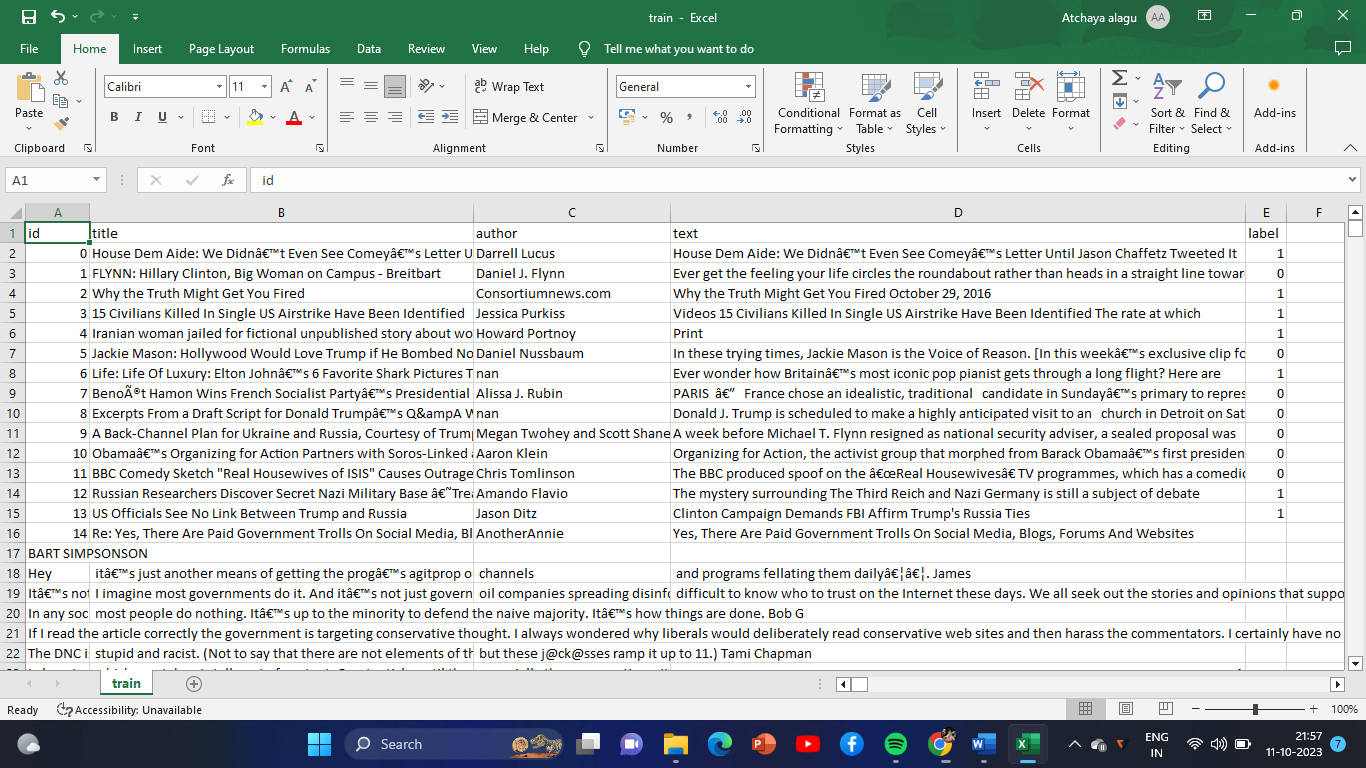
**PROBLEM SOLUTION :**

* To solve the problem of fake news detection, various approaches can be taken.
* One solution involves using natural language processing (NLP) techniques to analyse the content of news articles and identify patterns that indicate misinformation.
* This can include analysing the language used, fact-checking claims, and identifying biased or misleading information.
* Additionally, machine learning algorithms can be trained on labelled datasets to classify news articles as either reliable or fake based on features such as source credibility, writing style, and social media engagement.
* Regular updates and improvements to the algorithms are crucial to keep up with evolving techniques used by those spreading fake news.
* To tackle the problem of fake news detection is by leveraging the power of crowdsourcing.
* Platforms can engage users to report and flag suspicious or misleading content, which can then be reviewed by fact-checkers or community moderators.
* This collaborative approach helps in identifying and verifying the accuracy of news articles through collective efforts.
* Additionally, educating users about media literacy and critical thinking skills can empower them to identify and question sources of fake news, thereby reducing its impact.

**DATASET:**

The dataset is taken from

<https://www.kaggle.com/c/fake-news/data>



**SOFTWARE USED:**

Python – 3.X

**LIBRARIES USED:**

* + - Pandas
    - Scikit – learn
    - Nltk (Natural Language Toolkit)

**IMPORTING LIBRARIES:**

import pandas as pd

import numpy as np

import nltk

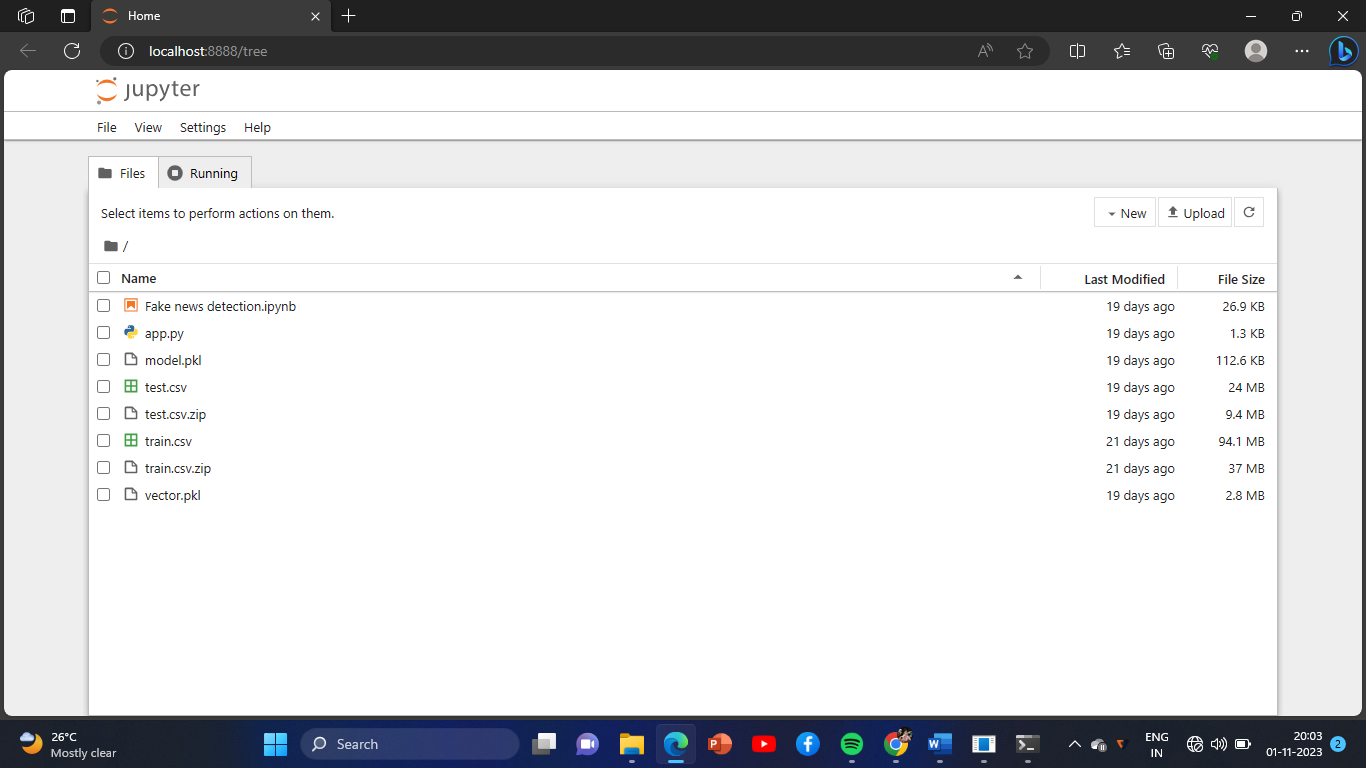
from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

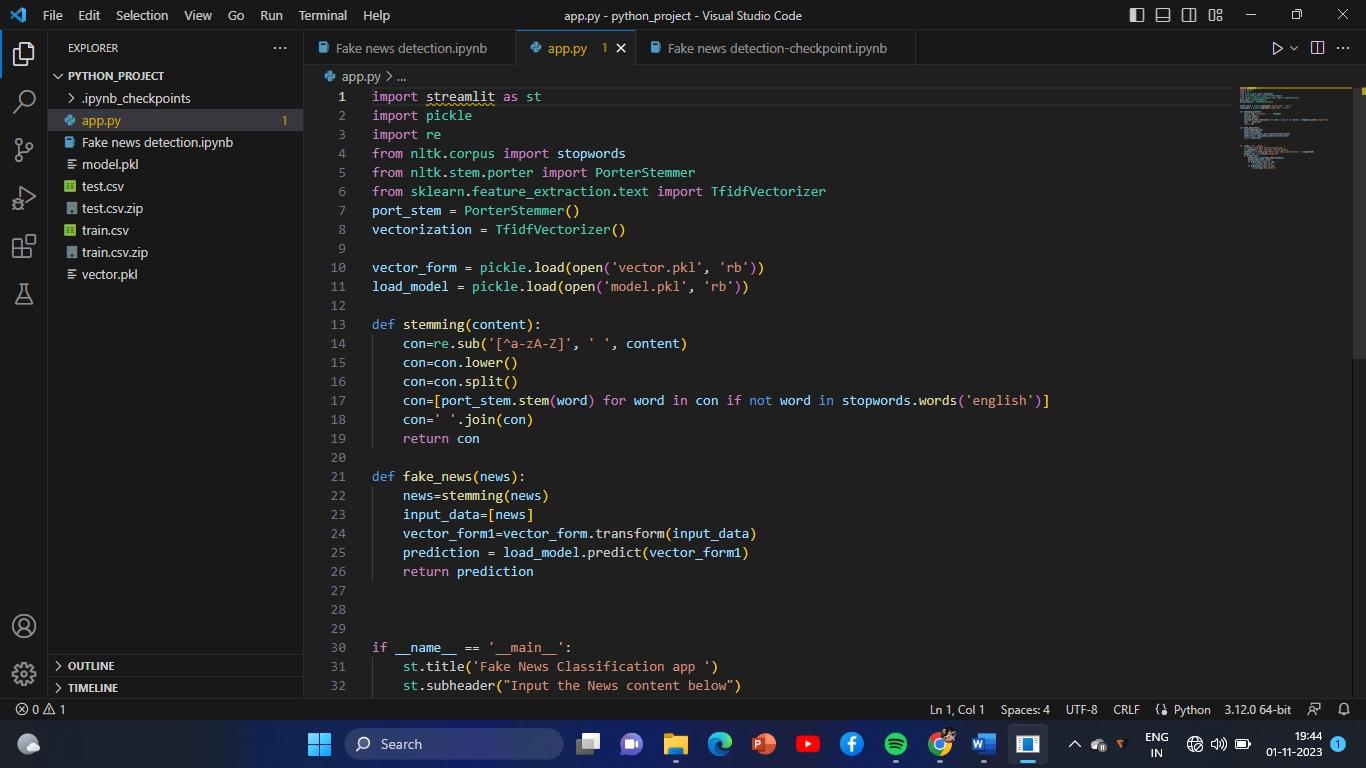
from sklearn.feature\_extraction.text import TfidfVectorizer

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

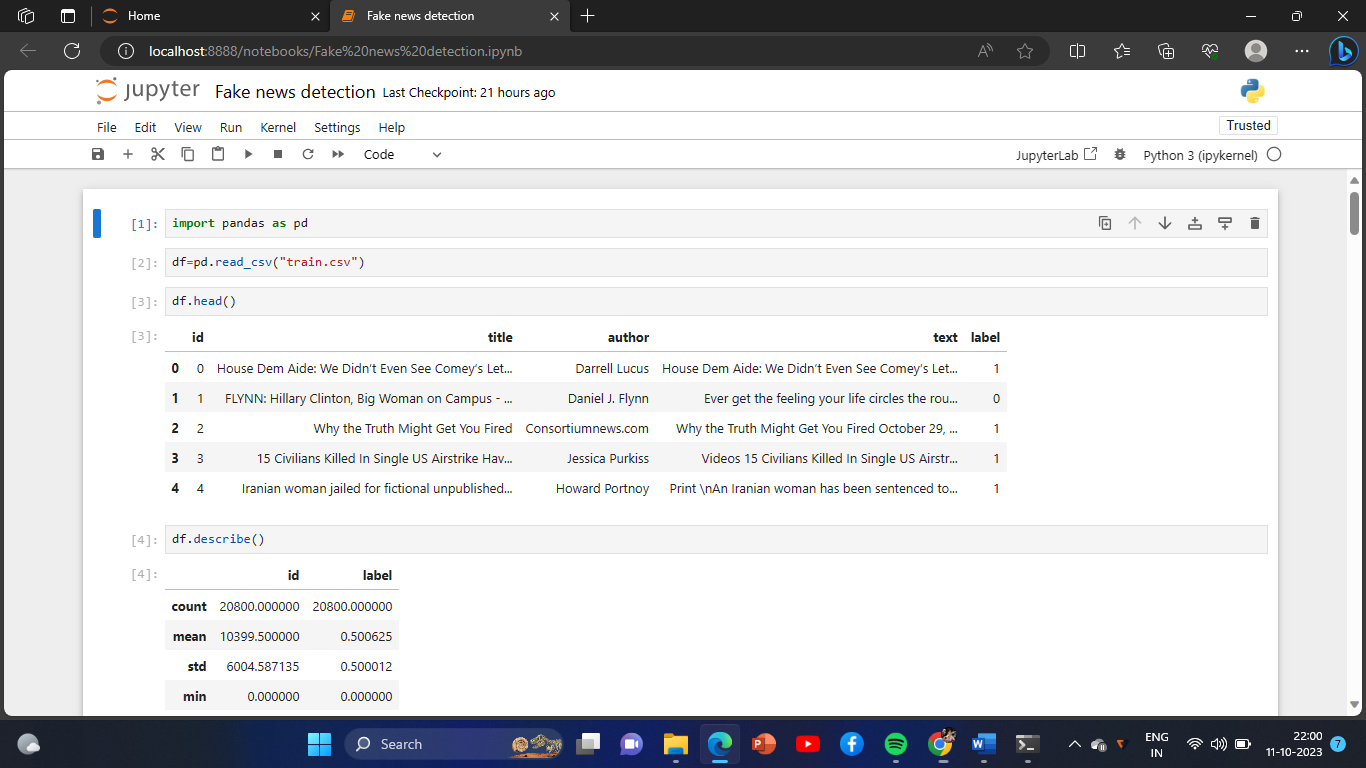
**JUPYTER NOTEBOOK**

****

**VISUAL STUDIO CODE:**



**JUPYTER NOTEBOOK COMMAND EXECUTION:**



Additionally we have included the workbook of jupyter notebook execution commands in github repository.

In this word document we have included pictures of the command execution in jupyter notebook,builting in visual studio code,uploading the dataset format.

**HARDWARE REQUIRED:**

* System - Pentium-IV
* Speed - 2.4GHZ
* Hard disk - 40GB
* Monitor - 15VGA colour
* RAM - 512MB

**DEFINITION OF PYTHON:**

Python is a high-level, interpreted, interactive and object- oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

# INTERACTIVE MODE PROGRAMMING:

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.

>>>

Type the following text at the Python prompt and press the Enter :

>>>print"Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");** However in Python version 2.4.3, this produces the following result :

Hello, Python!

# SCRIPT MODE PROGRAMMING:

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

print"Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows :

$ python test.py

This produces the following result :

, Python!

# FLASK FRAMEWORK:

Flask is a web application framework written in Python.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods :

|  |  |
| --- | --- |
| **Sr.No** | **Methods & Description** |
| 1 | **GET**    Sends data in unencrypted form to the server. Most common method. |
| 2 | **HEAD**    Same as GET, but without response body |
| 3 | **POST**    Used to send HTML form data to server. Data received by POST method is not cached by server. |
| 4 | **PUT**    Replaces all current representations of the target resource with the uploaded content. |

|  |  |
| --- | --- |
| 5 | **DELETE**    Removes all current representations of the target resource given by a URL |

By default, the Flask route responds to the **GET**requests. However,this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit" value="submit"/></p>

</form>

</body>

</html>

Now enter the following script in Python shell.

Now enter the following script in python shell.

from flask importFlask, redirect,url\_for, request

app=Flask( name )

@app.route('/success/<name>') def success(name): return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login(): ifrequest.method=='POST': user=request.form['nm']

return redirect(url\_for('success',name= user)) else:

user=request.args.get('nm')

return redirect(url\_for('success',name= user)) if name ==' main ':

app.run(debug =True)

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**

**LOGIN PAGE:**



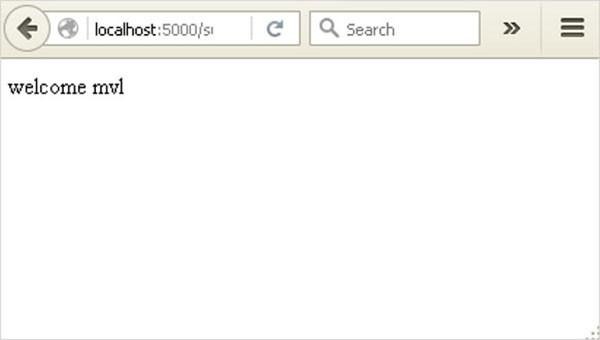
Form data is POSTed to the URL in action clause of form tag.

[**http://localhost/login**i](http://localhost/logini)s mapped to the **login()** function. Since the server has received data by **POST** method, value of „nm‟ parameter obtained from the form data is obtained by :

user = request.form['nm']

It is passed to **„/success‟** URL as variable part. The browser displays a **welcome** message in the window.

**DASHBOARD:**

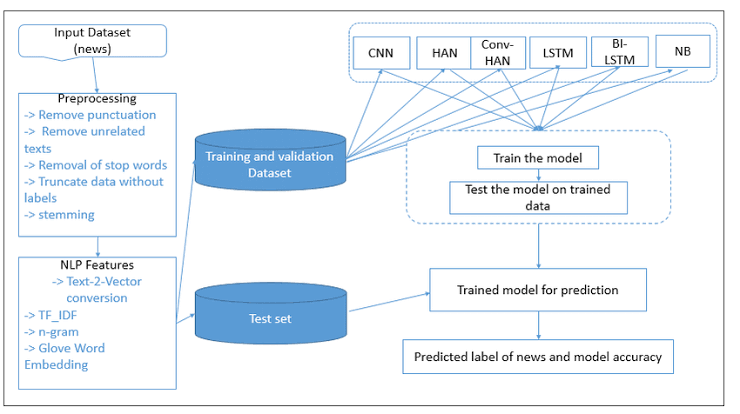


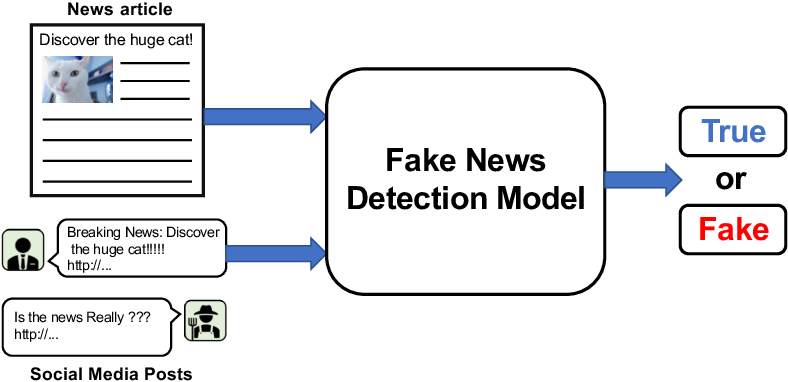
Change the method parameter to **„GET‟** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of „nm‟ parameter is now obtained by :

User = request.args.get(„nm‟)

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to „nm‟ parameter is passed on to “/success‟ URL as before.

**PROCESS:**





**PROGRAM:**

port\_stem = PorterStemmer()

vectorization = TfidfVectorizer()

vector\_form = pickle.load(open('vector.pkl', 'rb'))

load\_model = pickle.load(open('model.pkl', 'rb'))

def stemming(content):

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

con=con.lower()

con=con.split()

con=[port\_stem.stem(word) for word in con if not word in stopwords.words('english')]

con=' '.join(con)

return con

def fake\_news(news):

news=stemming(news)

input\_data=[news]

vector\_form1=vector\_form.transform(input\_data)

prediction = load\_model.predict(vector\_form1)

return prediction

if \_\_name\_\_ == '\_\_main\_\_':

st.title('Fake News Classification app ')

st.subheader("Input the News content below")

sentence = st.text\_area("Enter your news content here", "",height=200)

predict\_btt = st.button("predict")

if predict\_btt:

prediction\_class=fake\_news(sentence)

print(prediction\_class)

if prediction\_class == [0]:

st.success('Reliable')

if prediction\_class == [1]:

st.warning('Unreliable')

**UP-TO-DATE:**

* Get data from BuzzFeed news and politifact.
* Extend the dataset using data from social media.
* Detect communities around fake news posts and real news posts.
* Analyse our results to detect unusual user behaviour.

**CONCLUSION:**

After finishing all these process run this file and make api server with local host then paste the news . The compilation process will check whether the entered news is reliable or not.