Project Report On

Fake News Detection with Python

“A dissertation submitted in partial fulfillment of the requirements of Bachelor of Technology Degree in Computer Science and Engineering of the Maulana Abul Kalam Azad University of Technology for the year 2022-2023”

Submitted by

ROHIT KARMAKAR(26300119016)
ANKUSH CHOWDHURY(26300119021)
SOUVIK KUNDU(26300119032)

Under the guidance of

Shri. INDRAJIT DAWN
Assistant Professor
Dept of Computer Science & Engineering
Regent Education and Research Foundation

Department of Computer Science and Engineering

Regent Education and Research Foundation

(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)

Barrackpore - 700121, Barrackpore, WB

REGENT EDUCATION & RESEARCH FOUNDATION
GROUP OF INSTITUTIONS

Certificate of Approval

This is to certify that this report of B. Tech. Final Year project, entitled "Fake news detection with python" is a record of bona-fide work, carried out by Rohit Karmakar, Ankush Chowdhury, Souvik Kundu under my supervision and guidance.

In my opinion, the report in its present form is in partial fulfillment of all the requirements, as specified by the Regent Education and Research Foundation and as per regulations of the Maulana Abul Kalam Azad University of Technology. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for B. Tech. programme in Computer Science and Engineering in the year 2022-2023.

Guide / Supervisor

____
Mr. INDRAJIT DAWN SIR
Department of Computer Science and Engineering
Regent Education and Research Foundation

____
Examiner(s)

____
Head of the Department
Computer Science and Engineering
Regent Education and Research Foundation

Campus : Regent Education & Research Foundation Group of Institutions
Bara Kanthalia (Barrackpore), Post : Sewli Telinipara, P.S. : Titagarh, Kolkata - 700 121, Tel.: 033 2535-3051 / 3052, Fax : 033-2535-3052
Regd. Office : 88, Chowringhee Road, Kolkata - 700 020, E-mail : rerfkolkata@gmail.com, Website : www.rerf.co.in
City Office : 3rd Floor, 60B Chowringhee Road, Kolkata - 700 020, Tel : (+91 33) 2290 0112 / 13 / 14 , Fax No.: 033-2290-0115

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our guide Mr. Indrajit Dawn sir and the team members for their able co-operation and effort in our project.

Mr.____________________________
ROHIT KARMAKAR(26300119016)

Mr.____________________________
ANKUSH CHOWDHURY(26300119021)

Mr.____________________________
SOUVIK KUNDU (26300119032)

ABSTRACT

Fake news has become a pervasive problem in the digital age, with misinformation spreading rapidly through social media and other online platforms. The ability to detect fake news is critical for preserving the integrity of our information ecosystem and ensuring that people are making informed decisions based on accurate information.

Fake news detection is a complex problem that requires a combination of technical, social, and ethical considerations. Technical approaches involve developing algorithms that can automatically identify fake news based on a variety of features, such as the source of the information, the language used, and the presence of certain keywords or patterns. Social approaches involve educating people about how to identify and avoid fake news, as well as promoting media literacy and critical thinking skills. Ethical considerations involve balancing the need for accurate information with the right to free speech and avoiding the creation of a "truth police" that could be used to silence dissenting voices.

Despite the challenges involved, progress is being made in the field of fake news detection, with researchers and practitioners developing a range of tools and techniques to help identify and combat the spread of misinformation. These include machine learning algorithms, natural language processing techniques, and crowdsourced fact-checking platforms. Ultimately, the key to successful fake news detection lies in a multi-pronged approach that leverages the strengths of technical, social, and ethical strategies to create a more informed and trustworthy information ecosystem.

PROJECT SYNOPSIS

This is a Python code that performs classification of news articles as either true or fake using a Random Forest Classifier. Here is a summary of what the code does:

1. Imports necessary libraries such as pandas, numpy, re, and sklearn.
2. Reads two CSV files containing true and fake news data and assigns a class label (True or False) to each dataset.
3. Concatenates the two datasets into a single dataset.
4. Defines a filtering function to preprocess the news titles by converting them to lowercase, removing unwanted characters, URLs, punctuation, and numbers.
5. Applies the filtering function to the news titles in the merged dataset.
6. Splits the data into training and testing sets.
7. Uses the TfidfVectorizer to convert the text data into numerical form for model training.
8. Initializes a Random Forest Classifier and fits it to the training data.
9. Tests the trained model on the testing data and generates a classification report.
10. Defines a manual_testing function that takes user input as news articles, preprocesses them, and predicts their class using the trained model.
11. Prompts the user to enter two news articles and calls the manual_testing function for each input.
12. Prints the predicted class for each news article.

Please note that this code assumes the existence of two CSV files named "True.csv" and "Fake.csv" containing the true and fake news data, respectively. It also uses the RandomForestClassifier from the sklearn library for classification and the TfidfVectorizer for text-to-numerical conversion.

CONTENTS

1. INTRODUCTION

The rise of fake news during the 2016 U.S. Presidential Election highlighted not only the dangers of the effects of fake news but also the challenges presented when attempting to separate fake news from real news. Fake news may be a relatively new term but it is not necessarily a new phenomenon. Fake news has technically been around at least since the appearance and popularity of one-sided, partisan newspapers in the 19th century. However, advances in technology and the spread of news through different types of media have increased the spread of fake news today. As such, the effects of fake news have increased exponentially in the recent past and something must be done to prevent this from continuing in the future.

I have identified the three most prevalent motivations for writing fake news and chosen only one as the target for this project as a means to narrow the search in a meaningful way. The first motivation for writing fake news, which dates back to the 19th century one-sided party newspapers, is to influence public opinion. The second, which requires more recent advances in technology, is the use of fake headlines as clickbait to raise money. The third motivation for writing fake news, which is equally prominent yet arguably less dangerous, is satirical writing. While all three subsets of fake news, namely, clickbait, influential, and satire, share the common thread of being fictitious, their widespread effects are vastly different. As such, this paper will focus primarily on fake news as defined by politifact.com, "fabricated content that intentionally masquerades as news coverage of actual events." This definition excludes satire, which is intended to be humorous and not deceptive to readers. Most satirical articles come from sources like "The Onion", which specifically distinguish themselves as satire. Satire can already be classified, by machine learning techniques according to. Therefore, our goal is to move beyond these achievements and use machine learning to classify, at least as well as humans, more difficult discrepancies between real and fake news.

The dangerous effects of fake news, as previously defined, are made clear by events such as in which a man attacked a pizzeria due to a widespread fake news article. This story along with analysis from provide evidence that humans are not very good at detecting fake news, possibly not better than chance . As such, the question remains whether or not machines can do a better job. There are two methods by which machines could attempt to solve the fake news problem better than humans. The first is that machines are better at detecting and keeping track of statistics than humans, for example it is easier for a machine to detect that the majority of verbs used are "suggests" and "implies" versus, "states" and "proves." Additionally, machines may be more efficient in surveying a knowledge base to find all relevant articles and answering based on those many different sources. Either of these methods could prove useful in detecting fake news, but we decided to focus on how a machine can solve the fake news problem using supervised learning that extracts features of the language and content only within the source in question, without utilizing any fact checker or knowledge base. For many fake news detection techniques, a "fake" article published by a trustworthy author through a trustworthy source would not be caught. This approach would combat those "false negative" classifications of fake news. In essence, the task would be equivalent to what a human faces when reading a hard copy of a newspaper article, without internet access or outside knowledge of the subject (versus reading something online where he can simply look up relevant sources). The machine, like the human in the coffee shop, will have only access to the words in the article and must use strategies that do not rely on blacklists of authors and sources. The current project involves utilizing machine learning and natural language processing techniques to create a model that can expose documents that are, with high probability, fake news articles. Many of the current automated approaches to this problem are centered around a "blacklist" of authors and sources that are known producers of fake news. But, what about when the author is unknown or when fake news is published through a generally reliable source? In these cases it is necessary to rely simply on the content of the news article to make a decision on whether or not it is fake. By collecting examples of both real and fake news and training a model, it should be possible to classify fake news articles with a certain degree of accuracy. The goal of this project is to find the effectiveness and limitations of language-based techniques for detection of fake news through the use of machine

learning algorithms including but not limited to convolutional neural networks and recurrent neural networks. The outcome of this project should determine how much can be achieved in this task by analyzing patterns contained in the text and blind to outside information about the world.
This type of solution is not intended to be an end-to end solution for fake news classification. Like the “blacklist” approaches mentioned, there are cases in which it fails and some for which it succeeds. Instead of being an end-to-end solution, this project is intended to be one tool that could be used to aid humans who are trying to classify fake news. Alternatively, it could be one tool used in future applications that intelligently combine multiple tools to create an end-to-end solution to automating the process of fake news classification.

1.1 What is Fake news ?

Fake news refers to deliberately fabricated or misleading information presented as factual news. It involves the dissemination of false or distorted information with the intention to deceive or manipulate the public. Fake news can be created and spread through various mediums, including social media platforms, websites, and traditional media outlets.

The characteristics of fake news include the intentional distortion of facts, the manipulation of emotions, and the promotion of a particular agenda or viewpoint. Fake news often mimics the style and format of legitimate news sources to make it appear credible, making it challenging for people to distinguish between real and fake information.

Fake news can have significant consequences, as it can influence public opinion, shape political discourse, and impact social dynamics. It can create confusion, erode trust in media, and contribute to the spread of misinformation and disinformation. Recognizing and verifying the credibility of news sources, fact-checking information, and promoting media literacy are crucial in combating the spread of fake news.

Basically it is a type of yellow journalism, fake news encapsulates pieces of news that may be hoaxes and is generally spread through social media and other online media. This is often done to further or impose certain ideas and is often achieved with political agendas. Such news items may contain false and/or exaggerated claims, and may end up being viralized by algorithms, and users may end up in a filter bubble.

1.2 What is TFIDFVECTORIZER ?

TF (Term Frequency): The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
IDF (Inverse Document Frequency): Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.

TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer is a widely used technique in natural language processing (NLP) and information retrieval for transforming textual data into numerical representations that can be processed by machine learning algorithms. It aims to capture the importance of individual words or terms in a document within a larger collection of documents.

The TF-IDF vectorizer calculates a numerical value for each term in a document, representing its relevance and significance in the document and the entire corpus. The calculation involves two main components: term frequency (TF) and inverse document frequency (IDF).

Term Frequency (TF):
Term Frequency measures the frequency of a term within a document. It assigns a weight to each term, with higher weights indicating higher importance. TF can be calculated using various methods, such as raw term frequency (the number of times a term appears in a document) or logarithmic scaling to dampen the effect of overly frequent terms.

Inverse Document Frequency (IDF):

Inverse Document Frequency measures the significance of a term across the entire document collection. It aims to give higher weights to terms that appear in fewer documents, as they tend to be more informative. IDF is calculated by taking the logarithm of the ratio between the total number of documents and the number of documents containing a specific term.

The TF-IDF vectorizer combines the TF and IDF values to create a vector representation for each document. The resulting vector contains the TF-IDF values for all terms in the document. This vector can be used as input for various NLP tasks, such as text classification, information retrieval, clustering, and sentiment analysis.

The TF-IDF vectorizer has several advantages. It helps in capturing the importance of specific terms within a document and across the corpus, allowing for better discrimination between documents. It reduces the impact of common words that appear frequently in most documents (e.g., "the," "and"), which are less informative for distinguishing between texts. Additionally, it is a simple and computationally efficient method for text representation.

However, the TF-IDF vectorizer also has limitations. It does not consider the semantic relationships between terms, treating them as independent features. It may not be effective in capturing the context and meaning of phrases or sentences. Additionally, it requires a large amount of text data to estimate accurate IDF values for rare terms.

In summary, the TF-IDF vectorizer is a powerful tool for transforming text data into numerical representations. By considering both the local term frequency and the global document frequency, it provides a way to represent textual information in a more meaningful and informative manner, enabling various NLP tasks and analyses.

1.3 What is RANDOM FOREST CLASSIFIER ?

Random Forest Classifier is a popular supervised machine learning algorithm used for both classification and regression tasks. It belongs to the ensemble learning family, which combines the predictions of multiple individual models to make more accurate and robust predictions.

In the case of classification, Random Forest Classifier constructs a collection of decision trees during the training phase. Each decision tree is built on a randomly sampled subset of the training data and selects the best split at each node based on a selected criterion (such as Gini impurity or information gain). This randomness in sampling and feature selection helps to introduce diversity among the trees and prevent overfitting.

During the prediction phase, the Random Forest Classifier aggregates the predictions from each decision tree and outputs the majority vote (for classification problems) or the average prediction (for regression problems) as the final result. This ensemble-based approach tends to be more accurate and stable compared to a single decision tree.

Random Forest Classifier has several advantages. It is capable of handling both numerical and categorical features, making it versatile for various types of datasets. It can handle large datasets with a high number of features without overfitting. Random Forest also provides an estimate of feature importance, which can be useful for feature selection and understanding the underlying data relationships.

Furthermore, Random Forest Classifier is resistant to outliers and noise in the data, and it can handle missing values through appropriate techniques such as imputation. It is relatively fast to train and can be parallelized to leverage multiple processors or distributed computing.

However, Random Forest Classifier also has some limitations. The model's interpretability can be challenging due to the complexity of the ensemble of decision trees. It may not perform well on datasets with imbalanced class distributions, where the minority class can be easily dominated by the majority class. Additionally, Random Forests can be memory-intensive, especially when dealing with a large number of trees or high-dimensional datasets.

In summary, Random Forest Classifier is a versatile and powerful algorithm for classification tasks. It combines the predictions of multiple decision trees to make accurate predictions, handles various types of data, and provides insights into feature importance. It is widely used in practice for a range of applications, including but not limited to, image classification, fraud detection, and customer churn prediction. Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

A Random Forest Classifier is a machine learning algorithm that belongs to the ensemble learning family. It combines the predictions of multiple individual decision trees to make more accurate and robust predictions.

Here's a brief explanation of how a Random Forest Classifier works:

Data Preparation: The algorithm requires a labeled dataset as input, where each data point has a set of features and a corresponding target label. The features represent the characteristics of the data, while the target label indicates the class or category to which the data point belongs.

Building Individual Decision Trees: A Random Forest consists of multiple decision trees. Each tree is constructed using a random subset of the original dataset. At each node of the tree, a feature is selected based on its importance in predicting the target label. The tree recursively splits the data based on different feature values until it reaches leaf nodes containing the final predicted labels.

Randomness and Diversity: The Random Forest introduces randomness in two ways. First, the subsets of data used to build each tree are created through random sampling with replacement, a technique called bootstrap aggregating or "bagging." Second, at each node, only a random subset of features is considered for splitting. These random selections help to create diverse trees that capture different aspects of the data.

Voting and Consensus: Once all the individual trees are built, predictions are made by each tree for new, unseen data. In classification tasks, each tree "votes" for a particular class label. The final prediction of the Random Forest is determined by majority voting, i.e., the class that receives the most votes among all the trees.

Prediction and Evaluation: The Random Forest Classifier produces predictions for unseen data based on the consensus of the individual decision trees. The performance of the model can be evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Random Forests are known for their ability to handle large datasets with high-dimensional features and maintain good generalization performance. They are robust against overfitting and can provide insights into feature importance, making them widely used in various domains, including classification, regression, and feature selection tasks.

2. DATA SETS

The lack of manually labeled fake news datasets is certainly a bottleneck for advancing computationally intensive, text-based models that cover a wide array of topics. The dataset for the fake news challenge does not suit our purpose due to the fact that it contains the ground truth regarding the relationships between texts but not whether or not those texts are actually true or false statements. For our purpose, we need a set of news articles that is directly classified into categories of news types (i.e. real vs. fake or real vs parody vs. clickbait vs. propaganda). For more simple and common NLP classification tasks, such as sentiment analysis, there is an abundance of labeled data from a variety of sources including Twitter, Amazon Reviews, and IMDb Reviews. Unfortunately, the same is not true for finding labeled articles of fake and real news. This presents a challenge to researchers and data scientists who want to explore the topic by implementing supervised machine learning techniques. I have researched the available datasets for sentence-level classification and ways to combine datasets to create full sets with positive and negative examples for document-level classification.

Sentence Level :

Producing a new benchmark dataset for fake news detection that includes 12,800 manually labeled short statements on a variety of topics. These statements come from politifact.com, which provides heavy analysis of and links to the source documents for each of the statements. The labels for this data are not true and false but rather reflect the “sliding scale” of false news and have 6 intervals of labels. These labels, in order of ascending truthfulness, include ’pants-fire’, ’false’, barely true, ’half-true’, ’mostly-true’, and true. The creators of this database ran baselines such as Logistic Regression, Support Vector Machines, LSTM, CNN and an augmented CNN that used metadata. They reached 27% accuracy on this multiclass classification task with the CNN that involved metadata such as speaker and party related to the text.

Document Level :

There exists no dataset of similar quality to the Liar Dataset for documentlevel classification of fake news. As such, I had the option of using the headlines of documents as statements or creating a hybrid dataset of labeled fake and legitimate news articles. It is an informal and exploratory analysis carried out by combining two datasets that individually contain positive and negative fake news examples. Genes trains a model on a specific subset of both the Kaggle dataset and the data from NYT and the Guardian. In his experiment, the topics involved in training and testing are restricted to U.S News, Politics, Business and World news. However, he does not account for the difference in date range between the two datasets, which likely adds an additional layer of topic bias based on topics that are more or less popular during specific periods of time.

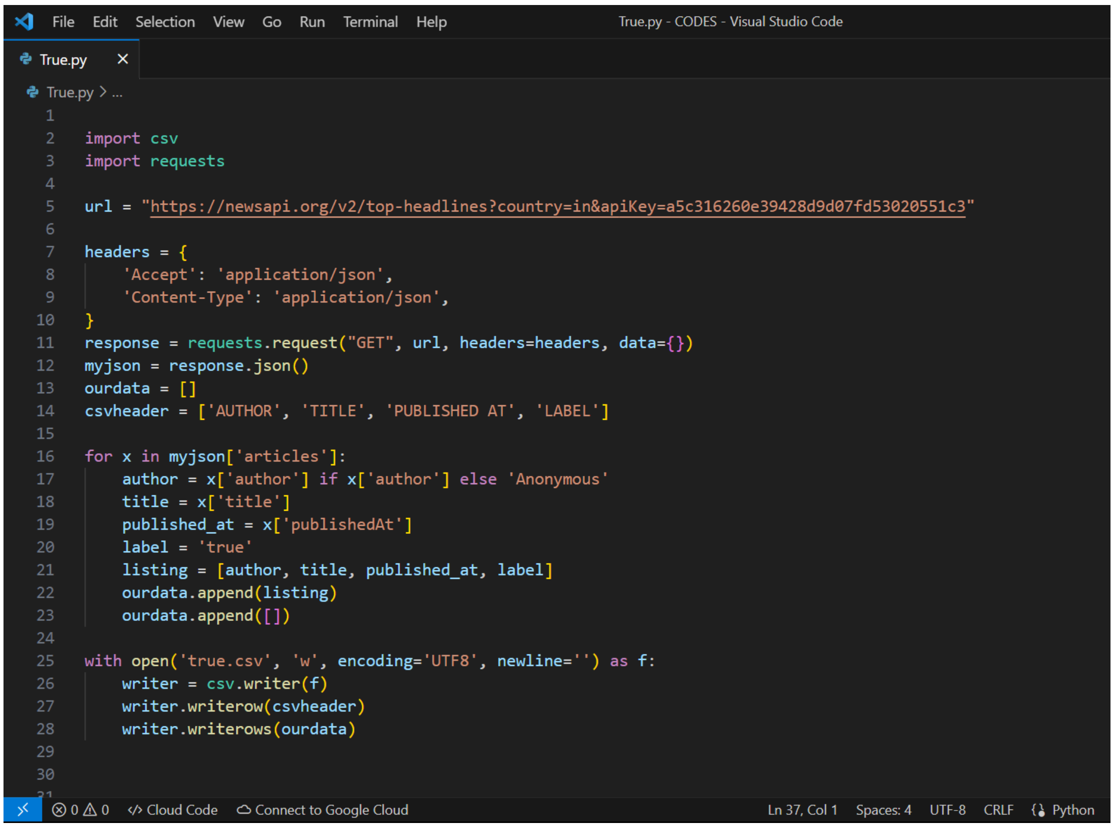
We have collected data in a manner similar to that of Genes, but more cautious in that we control for more bias in the sources and topics. Because the goal of our project was to find patterns in the language that are indicative of real or fake news, having source bias would be detrimental to our purpose. Including any source bias in our dataset, i.e. patterns that are specific to NYT, The Guardian, or any of the fake news websites, would allow the model to learn to associate sources with real/fake news labels. Learning to classify sources as fake or real news is an easy problem, but learning to classify specific types of language and language patterns as fake or real news is not. As such, we were very careful to remove as much of the source-specific patterns as possible to force our model to learn something more meaningful and generalizable.

We admit that there are certainly instances of fake news in the New York Times and probably instances of real news in the Kaggle dataset because it is based on a list of unreliable websites. However, because these instances are the exception and not the rule, we expect that the model will learn from the majority of articles that are consistent with the label of the source. Additionally, we are not trying to train a model to learn facts but rather learn deliveries. To be more clear, the deliveries and reporting mechanisms found in fake news articles within New York Times should still possess characteristics more commonly found in real news, although they will contain fictitious factual information.

1.1 Importing True.csv file :

An acceptable approach would be to use the APIs from reliable sources like New York Times and The Guardian. The NYT API provides similar information to that of the kaggle dataset, including both text and images that are found in the document. The Kaggle Dataset also provides the source of each article, which is trivial for the APIs of specific newspaper sources. We pulled articles from both of these sources in the same range of dates that the fake news was restricted to (October 26 , 2016 to November 25, 2016). This is important because of the specificity of the current events at that time - information that would not likely be present in news outside of this timeframe. There were just over 9,000 Guardian articles and just over 2,000 New York Times articles. Unlike the Kaggle dataset, which had 244 different websites as sources, our real news dataset only has two different source: The New York Times and The Guardian. Due to this difference, we found that extra effort was required to ensure that we removed any source-specific patterns so that the model would not simply learn to identify how an article from the New York Times is written or how an article from The Guardian is written. Instead, we wanted our model to learn more meaningful language patterns that are similar to real news reporting, regardless of the source.

This code fetches the top headlines from the News API for India, extracts specific information from the response, and saves it to a CSV file named 'true.csv'. Each article is represented as a row in the CSV file, and the extracted information includes the author, title, published date, and label.

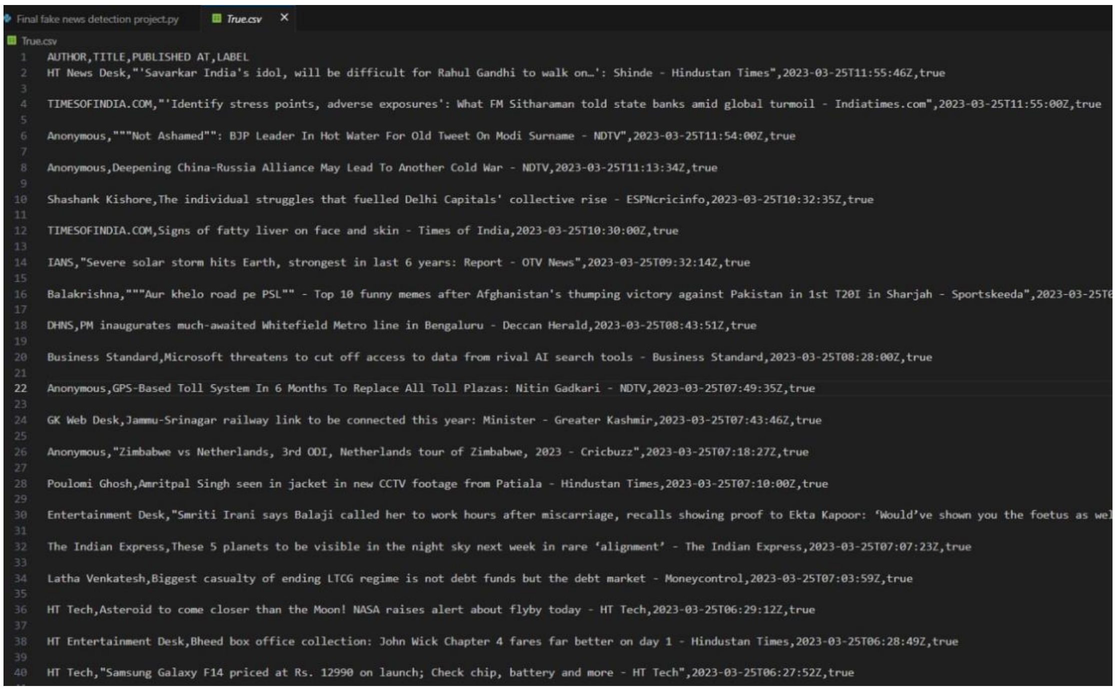

File Edit Selection View Go Run Terminal Help
True.py - CODES - Visual Studio Code
True.py
True.py > ...
⊗ 0 ⚠ 0 Cloud Code Connect to Google Cloud
Ln 37, Col 1 Spaces: 4 UTF-8 CRLF Python

```python

import csv
import requests

url = "https://newsapi.org/v2/top-headlines?country=in&apiKey=a5c316260e39428d9d07fd53020551c3"

headers = {
    'Accept': 'application/json',
    'Content-Type': 'application/json',
}
response = requests.request("GET", url, headers=headers, data={})
myjson = response.json()
ourdata = []
csvheader = ['AUTHOR', 'TITLE', 'PUBLISHED AT', 'LABEL']

for x in myjson['articles']:
    author = x['author'] if x['author'] else 'Anonymous'
    title = x['title']
    published_at = x['publishedAt']
    label = 'true'
    listing = [author, title, published_at, label]
    ourdata.append(listing)
    ourdata.append([])

with open('true.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(csvheader)
    writer.writerows(ourdata)
```

Fig. 1 : True news program to make “true.csv” file

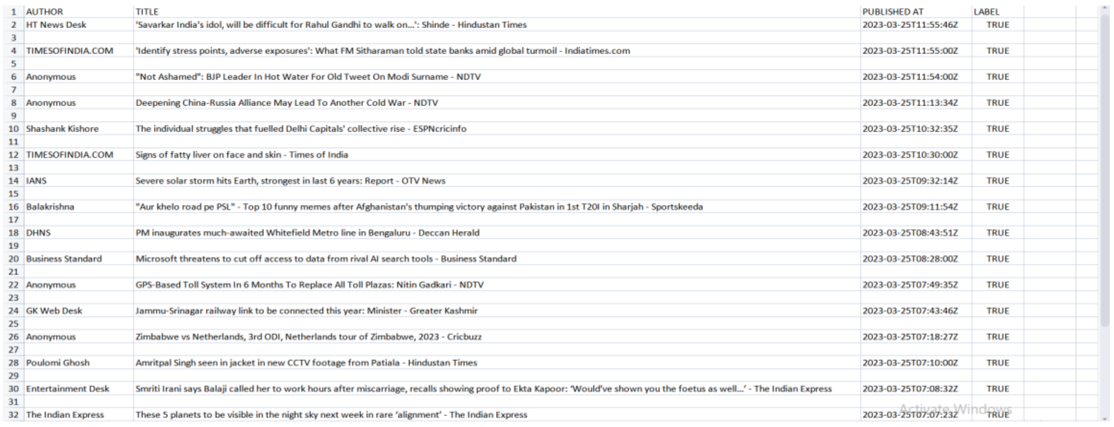
The step by step explanation of the true.csv program are mentioned below :

- I. Imports the necessary libraries: csv and requests.
- II. Defines the URL for the News API.
- III. Sets the headers for the HTTP request.
- IV. Sends a GET request to the specified URL using the requests.request() function and saves the response.
- V. Converts the response to JSON format using the .json() method.
- VI. Initializes an empty list called ourdata to store the extracted article information.
- VII. Defines the header row for the CSV file as csv header.
- VIII. Iterates over each article in the JSON response.
- IX. Creates a our data with blank space.
- X. Extracts the author, title, published date, and assigns the label as 'true'.
- XI. Creates a new list called listing with the extracted information.
- XII. Appends the listing to the ourdata list.
- XIII. Appends an empty list to ourdata (for creating a blank row in the CSV file between each article).
- XIV. Opens a new CSV file called 'true.csv' in write mode.
- XV. Creates a CSV writer object using the csv.writer() function.
- XVI. Writes the csv header as the first row in the CSV file using the writerow() method.
- XVII. Writes all the data in the ourdata list to the CSV file using the writerows() method.
- XVIII. Closes the CSV file.



```
AUTHOR,TITLE,PUBLISHED AT,LABEL
HT News Desk,"'Savarkar India's idol, will be difficult for Rahul Gandhi to walk on…': Shinde - Hindustan Times",2023-03-25T11:55:46Z,true

TIMESOFINDIA.COM,"'Identify stress points, adverse exposures': What FM Sitharaman told state banks amid global turmoil - Indiatimes.com",2023-03-25T11:55:00Z,true

Anonymous,"""Not Ashamed"": BJP Leader In Hot Water For Old Tweet On Modi Surname - NDTV",2023-03-25T11:54:00Z,true

Anonymous,Deepening China-Russia Alliance May Lead To Another Cold War - NDTV,2023-03-25T11:13:34Z,true

Shashank Kishore,The individual struggles that fuelled Delhi Capitals' collective rise - ESPNcricinfo,2023-03-25T10:32:35Z,true

TIMESOFINDIA.COM,Signs of fatty liver on face and skin - Times of India,2023-03-25T10:30:00Z,true

IANS,"Severe solar storm hits Earth, strongest in last 6 years: Report - OTV News",2023-03-25T09:32:14Z,true

Balakrishna,"""Aur khelo road pe PSL"" - Top 10 funny memes after Afghanistan's thumping victory against Pakistan in 1st T20I in Sharjah - Sportskeeda",2023-03-25T

DHNS,PM inaugurates much-awaited Whitefield Metro line in Bengaluru - Deccan Herald,2023-03-25T08:43:51Z,true

Business Standard,Microsoft threatens to cut off access to data from rival AI search tools - Business Standard,2023-03-25T08:28:00Z,true

Anonymous,GPS-Based Toll System In 6 Months To Replace All Toll Plazas: Nitin Gadkari - NDTV,2023-03-25T07:49:35Z,true

GK Web Desk,Jammu-Srinagar railway link to be connected this year: Minister - Greater Kashmir,2023-03-25T07:43:46Z,true

Anonymous,"Zimbabwe vs Netherlands, 3rd ODI, Netherlands tour of Zimbabwe, 2023 - Cricbuzz",2023-03-25T07:18:27Z,true

Poulomi Ghosh,Amritpal Singh seen in jacket in new CCTV footage from Patiala - Hindustan Times,2023-03-25T07:10:00Z,true

Entertainment Desk,"Smriti Irani says Balaji called her to work hours after miscarriage, recalls showing proof to Ekta Kapoor: 'Would've shown you the foetus as wel

The Indian Express,These 5 planets to be visible in the night sky next week in rare 'alignment' - The Indian Express,2023-03-25T07:07:23Z,true

Latha Venkatesh,Biggest casualty of ending LTCG regime is not debt funds but the debt market - Moneycontrol,2023-03-25T07:03:59Z,true

HT Tech,Asteroid to come closer than the Moon! NASA raises alert about flyby today - HT Tech,2023-03-25T06:29:12Z,true

HT Entertainment Desk,Bheed box office collection: John Wick Chapter 4 fares far better on day 1 - Hindustan Times,2023-03-25T06:28:49Z,true

HT Tech,"Samsung Galaxy F14 priced at Rs. 12990 on launch; Check chip, battery and more - HT Tech",2023-03-25T06:27:52Z,true
```

Fig. 2 : true csv

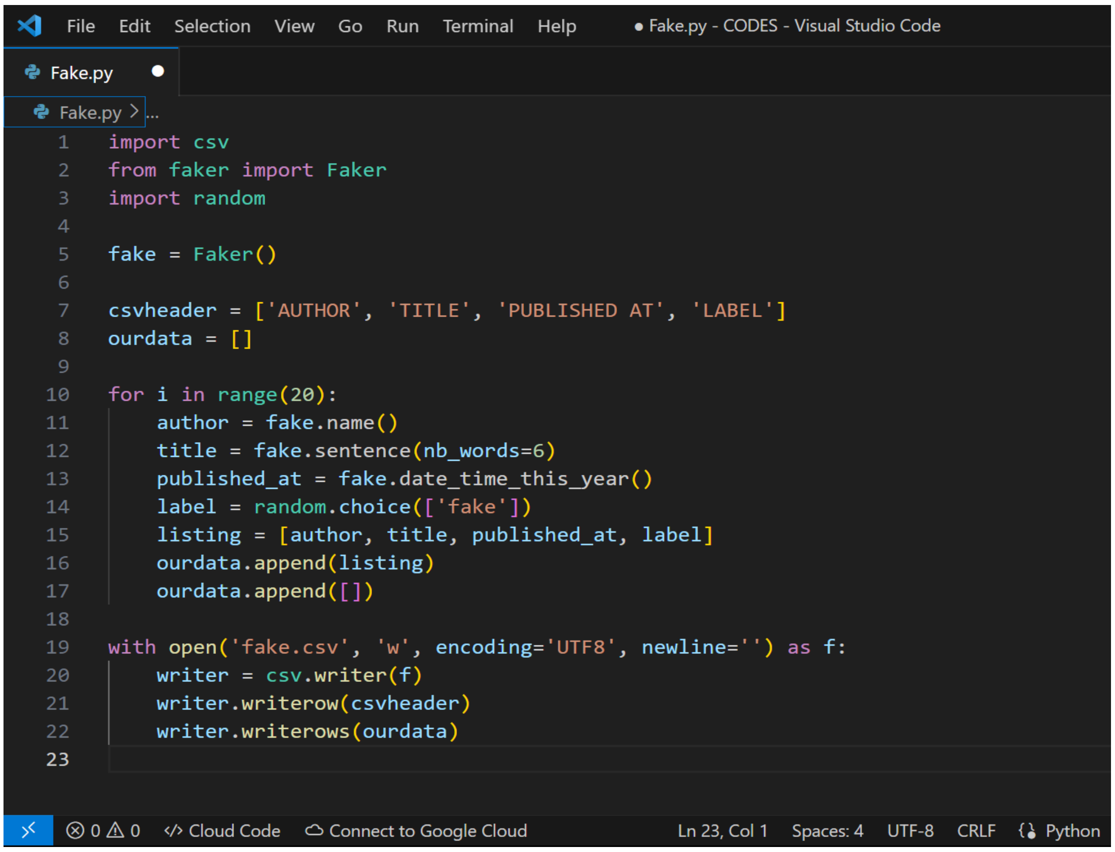


	AUTHOR	TITLE	PUBLISHED AT	LABEL
2	HT News Desk	'Savarkar India's idol, will be difficult for Rahul Gandhi to walk on...': Shinde - Hindustan Times	2023-03-25T11:55:46Z	TRUE
3				
4	TIMESOFINDIA.COM	'Identify stress points, adverse exposures': What FM Sitharaman told state banks amid global turmoil - Indiatimes.com	2023-03-25T11:55:00Z	TRUE
5				
6	Anonymous	"Not Ashamed": BJP Leader In Hot Water For Old Tweet On Modi Surname - NDTV	2023-03-25T11:54:00Z	TRUE
7				
8	Anonymous	Deepening China-Russia Alliance May Lead To Another Cold War - NDTV	2023-03-25T11:13:34Z	TRUE
9				
10	Shashank Kishore	The individual struggles that fuelled Delhi Capitals' collective rise - ESPNcricinfo	2023-03-25T10:32:35Z	TRUE
11				
12	TIMESOFINDIA.COM	Signs of fatty liver on face and skin - Times of India	2023-03-25T10:30:00Z	TRUE
13				
14	IANS	Severe solar storm hits Earth, strongest in last 6 years: Report - OTV News	2023-03-25T09:32:14Z	TRUE
15				
16	Balakrishna	"Aur khelo road pe PSL" - Top 10 funny memes after Afghanistan's thumping victory against Pakistan in 1st T20I in Sharjah - Sportskeeda	2023-03-25T09:11:54Z	TRUE
17				
18	DHNS	PM inaugurates much-awaited Whitefield Metro line in Bengaluru - Deccan Herald	2023-03-25T08:43:51Z	TRUE
19				
20	Business Standard	Microsoft threatens to cut off access to data from rival AI search tools - Business Standard	2023-03-25T08:28:00Z	TRUE
21				
22	Anonymous	GPS-Based Toll System In 6 Months To Replace All Toll Plazas: Nitin Gadkari - NDTV	2023-03-25T07:49:35Z	TRUE
23				
24	GK Web Desk	Jammu-Srinagar railway link to be connected this year: Minister - Greater Kashmir	2023-03-25T07:43:46Z	TRUE
25				
26	Anonymous	Zimbabwe vs Netherlands, 3rd ODI, Netherlands tour of Zimbabwe, 2023 - Cricbuzz	2023-03-25T07:18:27Z	TRUE
27				
28	Poulomi Ghosh	Amritpal Singh seen in jacket in new CCTV footage from Patiala - Hindustan Times	2023-03-25T07:10:00Z	TRUE
29				
30	Entertainment Desk	Smriti Irani says Balaji called her to work hours after miscarriage, recalls showing proof to Ekta Kapoor: 'Would've shown you the foetus as well...' - The Indian Express	2023-03-25T07:08:32Z	TRUE
31				
32	The Indian Express	These 5 planets to be visible in the night sky next week in rare 'alignment' - The Indian Express	2023-03-25T07:07:23Z	TRUE

Fig. 3 : True.csv File

1.2 Importing Fake.csv file :

This code uses the Faker library to generate 40 rows of fake data with fake author names, titles, published dates, and labels. It then saves this data to a CSV file named 'fake.csv', with each row representing a set of fake data and a blank row separating each set.



```python
import csv
from faker import Faker
import random

fake = Faker()

csvheader = ['AUTHOR', 'TITLE', 'PUBLISHED AT', 'LABEL']
ourdata = []

for i in range(20):
    author = fake.name()
    title = fake.sentence(nb_words=6)
    published_at = fake.date_time_this_year()
    label = random.choice(['fake'])
    listing = [author, title, published_at, label]
    ourdata.append(listing)
    ourdata.append([])

with open('fake.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(csvheader)
    writer.writerows(ourdata)

```

Fig. 4 : Fake news program to make Fake.csv file

This code generates fake data using the Faker library and writes it to a CSV file. Here are the steps involved:

I. Imports the required libraries: csv for working with CSV files and Faker for generating fake data.

II. Defines the header row for the CSV file as csv header.

III. Initializes an empty list called ourdata to store the generated fake data.

IV. Sets up a loop to generate 20 rows of fake data.

V. Inside the loop, generates a fake author name using fake.name().

VI. Generates a fake title consisting of 6 words using fake.sentence(nb_words=6).

VII. Generates a fake published date and time within the current year using fake.date_time_this_year().

VIII. Assigns a random label from the choices ['fake'] using random.choice(['fake']).

IX. Creates a new list called listing with the generated fake data.

X. Appends the listing to the ourdata list.

XI. Appends an empty list to ourdata (for creating a blank row in the CSV file between each set of fake data).

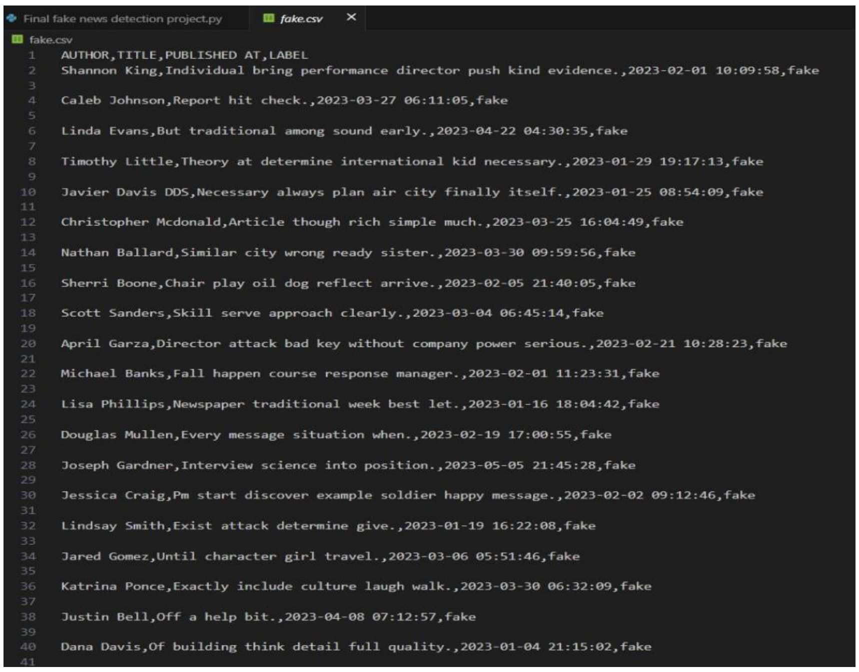
XII. Opens a new CSV file called 'fake.csv' in write mode.

XIII. Creates a CSV writer object using the csv.writer() function.

XIV. Writes the csv header as the first row in the CSV file using the writerow() method.

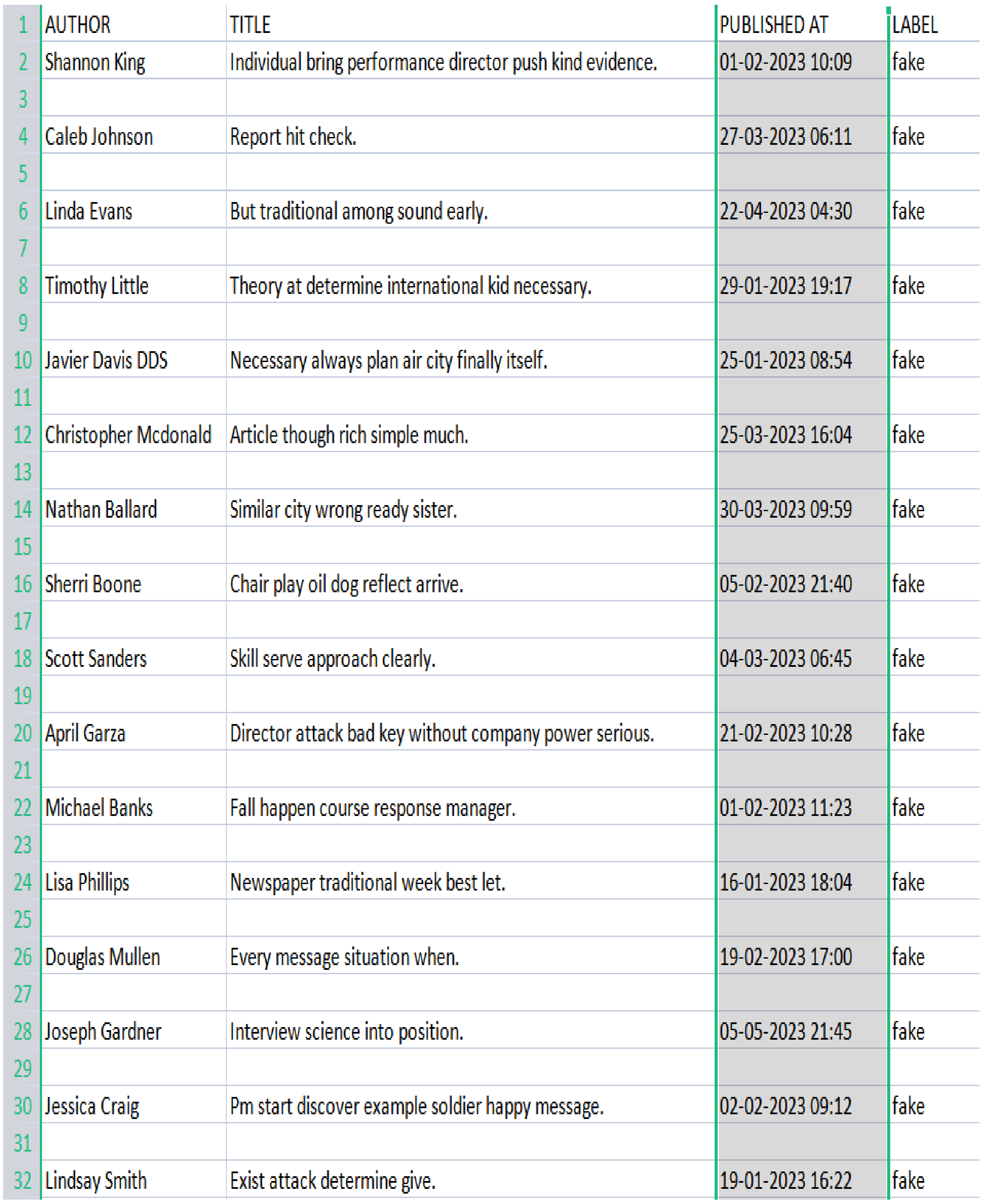
XV. Writes all the data in the ourdata list to the CSV file using the writerows() method.

XVI. Closes the CSV file.



```
AUTHOR,TITLE,PUBLISHED AT,LABEL
Shannon King,Individual bring performance director push kind evidence.,2023-02-01 10:09:58,fake

Caleb Johnson,Report hit check.,2023-03-27 06:11:05,fake

Linda Evans,But traditional among sound early.,2023-04-22 04:30:35,fake

Timothy Little,Theory at determine international kid necessary.,2023-01-29 19:17:13,fake

Javier Davis DDS,Necessary always plan air city finally itself.,2023-01-25 08:54:09,fake

Christopher Mcdonald,Article though rich simple much.,2023-03-25 16:04:49,fake

Nathan Ballard,Similar city wrong ready sister.,2023-03-30 09:59:56,fake

Sherri Boone,Chair play oil dog reflect arrive.,2023-02-05 21:40:05,fake

Scott Sanders,Skill serve approach clearly.,2023-03-04 06:45:14,fake

April Garza,Director attack bad key without company power serious.,2023-02-21 10:28:23,fake

Michael Banks,Fall happen course response manager.,2023-02-01 11:23:31,fake

Lisa Phillips,Newspaper traditional week best let.,2023-01-16 18:04:42,fake

Douglas Mullen,Every message situation when.,2023-02-19 17:00:55,fake

Joseph Gardner,Interview science into position.,2023-05-05 21:45:28,fake

Jessica Craig,Pm start discover example soldier happy message.,2023-02-02 09:12:46,fake

Lindsay Smith,Exist attack determine give.,2023-01-19 16:22:08,fake

Jared Gomez,Until character girl travel.,2023-03-06 05:51:46,fake

Katrina Ponce,Exactly include culture laugh walk.,2023-03-30 06:32:09,fake

Justin Bell,Off a help bit.,2023-04-08 07:12:57,fake

Dana Davis,Of building think detail full quality.,2023-01-04 21:15:02,fake

```

Fig. 5 : fake csv



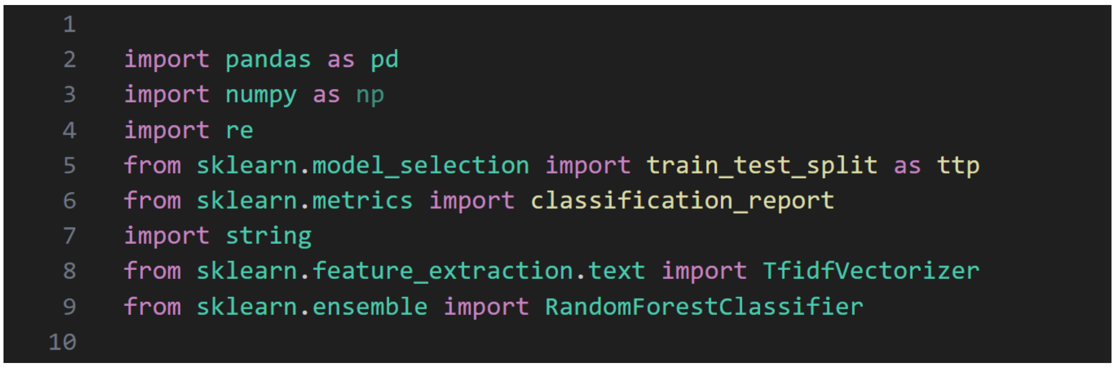
1	AUTHOR	TITLE	PUBLISHED AT	LABEL
2	Shannon King	Individual bring performance director push kind evidence.	01-02-2023 10:09	fake
3				
4	Caleb Johnson	Report hit check.	27-03-2023 06:11	fake
5				
6	Linda Evans	But traditional among sound early.	22-04-2023 04:30	fake
7				
8	Timothy Little	Theory at determine international kid necessary.	29-01-2023 19:17	fake
9				
10	Javier Davis DDS	Necessary always plan air city finally itself.	25-01-2023 08:54	fake
11				
12	Christopher Mcdonald	Article though rich simple much.	25-03-2023 16:04	fake
13				
14	Nathan Ballard	Similar city wrong ready sister.	30-03-2023 09:59	fake
15				
16	Sherri Boone	Chair play oil dog reflect arrive.	05-02-2023 21:40	fake
17				
18	Scott Sanders	Skill serve approach clearly.	04-03-2023 06:45	fake
19				
20	April Garza	Director attack bad key without company power serious.	21-02-2023 10:28	fake
21				
22	Michael Banks	Fall happen course response manager.	01-02-2023 11:23	fake
23				
24	Lisa Phillips	Newspaper traditional week best let.	16-01-2023 18:04	fake
25				
26	Douglas Mullen	Every message situation when.	19-02-2023 17:00	fake
27				
28	Joseph Gardner	Interview science into position.	05-05-2023 21:45	fake
29				
30	Jessica Craig	Pm start discover example soldier happy message.	02-02-2023 09:12	fake
31				
32	Lindsay Smith	Exist attack determine give.	19-01-2023 16:22	fake

Fig. 6 : fake news file preview

3. Steps for detecting fake news with Python

Following below steps we are able to make Fake news detection -

1.1 Make necessary imports:

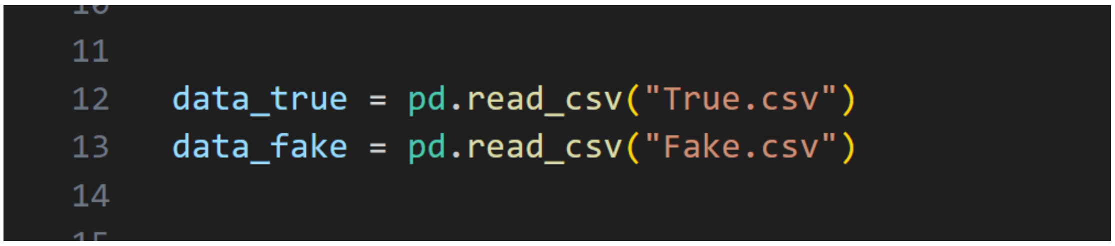


```python

import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split as ttp
from sklearn.metrics import classification_report
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier

```

Fig. 7 : imports

- train_test_split function from the sklearn.model_selection module. This function is used to split a dataset into training and testing subsets, which is a common practice in machine learning to evaluate model performance.
- classification_report function from the sklearn.metrics module. This function is used to generate a classification metrics report, which provides various evaluation metrics for a classification model.
- TfidfVectorizer class from the sklearn.feature_extraction.text module. TfidfVectorizer is a feature extraction technique commonly used in natural language processing (NLP) tasks. It transforms text data into numerical feature vectors based on the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm.
- RandomForestClassifier class from the sklearn.ensemble module. RandomForestClassifier is a machine learning model that belongs to the ensemble learning method and is based on the random forest algorithm. It is commonly used for classification tasks.

1.2 Read the data :

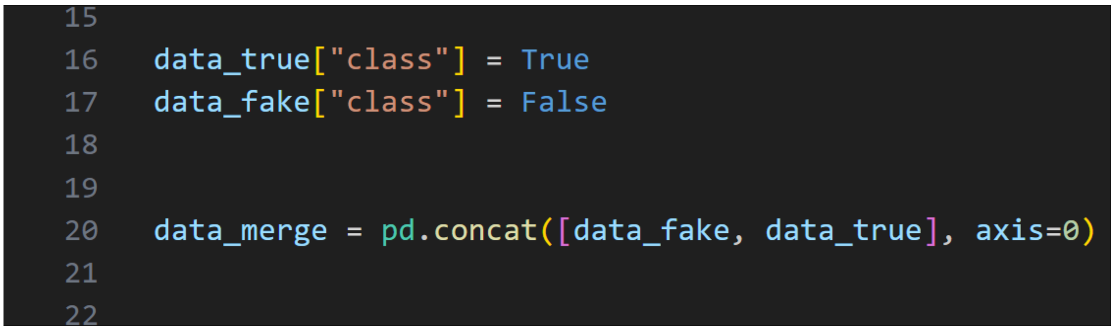


```
data_true = pd.read_csv("True.csv")
data_fake = pd.read_csv("Fake.csv")
```

Fig. 8 : csv file reading

In this code snippet, we are using the pandas library to read two CSV files named "True.csv" and "Fake.csv" into separate DataFrames. The pd.read_csv() function is used to read the CSV files and store the data into the respective DataFrames.

1.3 Adding labels and merging datasets :



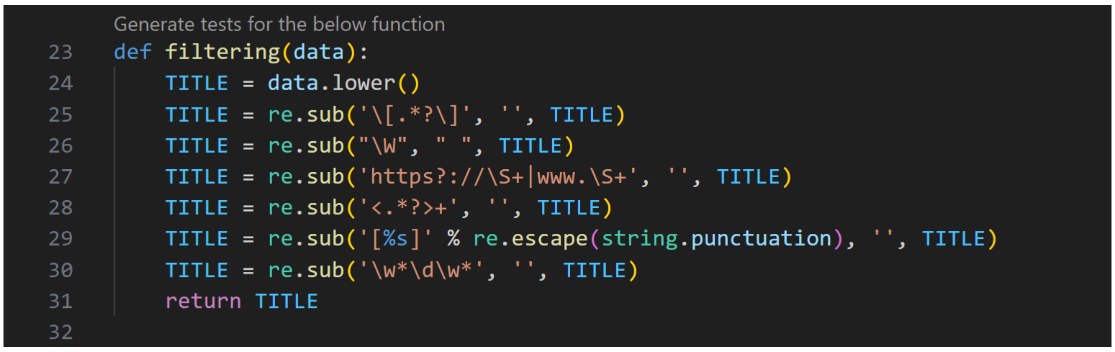
```
data_true["class"] = True
data_fake["class"] = False


data_merge = pd.concat([data_fake, data_true], axis=0)
```

Fig. 9 : labels and merging

In this code snippet, we are adding a new column named "class" to the "data_true" and "data_fake" DataFrames. We assign the value True to the "class" column in the "data_true" DataFrame and the value False to the "class" column in the "data_fake" DataFrame.

In this code snippet, we are concatenating the "data_fake" and "data_true" DataFrames along the row axis (axis=0) using the pd.concat() function from the pandas library. The resulting DataFrame, data_merge, will contain all the rows from "data_fake" followed by all the rows from "data_true".

1.4 Defining the filtering function :



Generate tests for the below function

```python
def filtering(data):
    TITLE = data.lower()
    TITLE = re.sub('\[.*?\]', '', TITLE)
    TITLE = re.sub("\W", " ", TITLE)
    TITLE = re.sub('https?://\S+|www.\S+', '', TITLE)
    TITLE = re.sub('<.*?>+', '', TITLE)
    TITLE = re.sub('[%s]' % re.escape(string.punctuation), '', TITLE)
    TITLE = re.sub('\w*\d\w*', '', TITLE)
    return TITLE
```

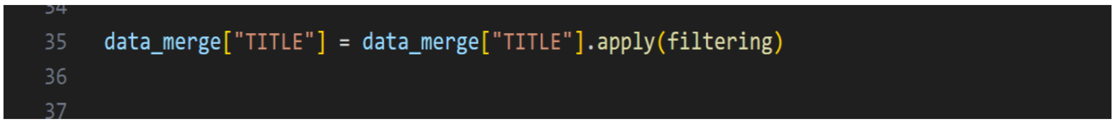
Fig. 10 : filtering function

In this code snippet, we have a function named filtering that takes a string data as input. The function performs several text filtering operations on the input string:

- It converts the input string to lowercase using lower().
- It removes square brackets and the arbitrary characters inside them using re.sub('\[.*?\]', '', title).
- It replaces every non-alphanumeric character with a space using re.sub("\W", " ", title).
- It removes URLs starting with "http" or "https" and "www." using re.sub('https?://\S+|www.\S+', '', title).
- It removes HTML tags like <p>, <div>, etc. using re.sub('<.*?>+', '', title).
- It replaces punctuation characters with blanks using re.sub('[%s]' % re.escape(string.punctuation), '', title).
- It removes digits and attached letters using re.sub('\w*\d\w*', '', title).

The filtered string is then returned by the function.

1.5 Filtering the title of news article :

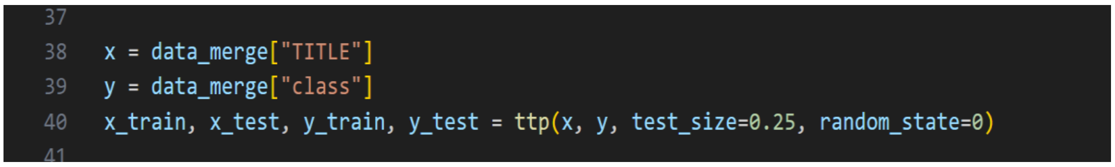


```python
data_merge["TITLE"] = data_merge["TITLE"].apply(filtering)
```

Fig. 11 : filtering news article

- Filtering the data in "TITLE" column using filter function defined above.

1.6 Splitting the train and test data set :

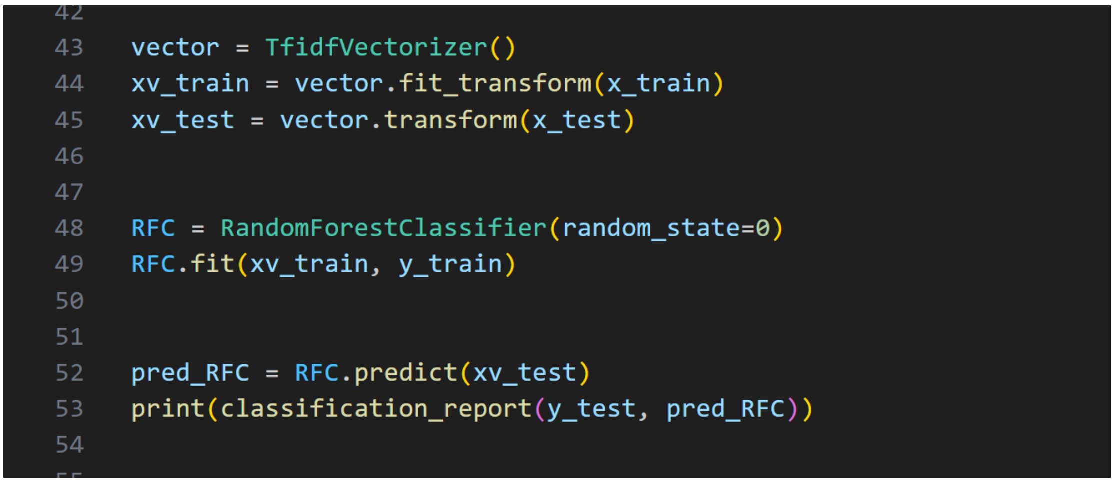


```python
x = data_merge["TITLE"]
y = data_merge["class"]
x_train, x_test, y_train, y_test = ttp(x, y, test_size=0.25, random_state=0)
```

Fig. 12 : train test data set

It takes the input features (x) and the corresponding labels (y) and splits them into four sets: x_train (training features), x_test (testing features), y_train (training labels), and y_test (testing labels). The test_size parameter determines the proportion of the dataset that should be allocated for testing (in this case, 25%), and random_state is set to a specific value to ensure reproducibility of the split.

1.7 Train test and RFC classifier :



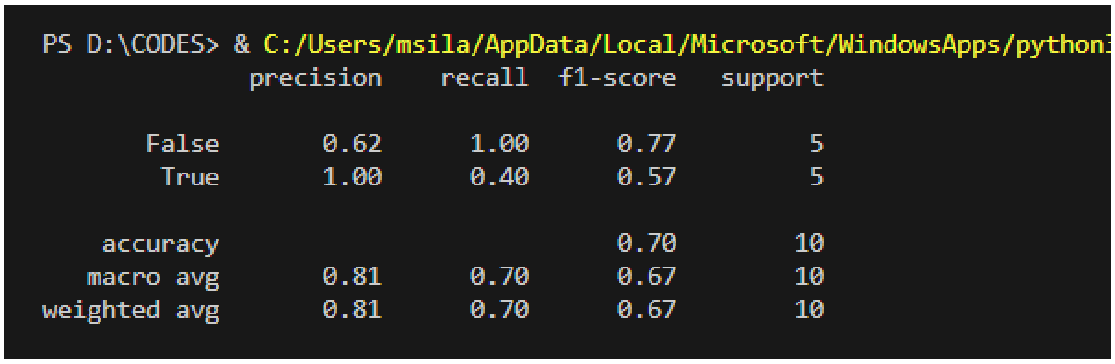
```python
vector = TfidfVectorizer()
xv_train = vector.fit_transform(x_train)
xv_test = vector.transform(x_test)


RFC = RandomForestClassifier(random_state=0)
RFC.fit(xv_train, y_train)


pred_RFC = RFC.predict(xv_test)
print(classification_report(y_test, pred_RFC))
```

Fig. 13 : RFC classifier

This code performing the following tasks:

1. Creating a TfidfVectorizer object, which is used to convert text data into a numerical representation using the TF-IDF (Term Frequency-Inverse Document Frequency) technique.
2. Using the fit_transform method of the TfidfVectorizer object to transform the training text data (x_train) into a numerical matrix (xv_train).
3. Using the transform method of the TfidfVectorizer object to transform the test text data (x_test) into a numerical matrix (xv_test), using the same vocabulary learned from the training data.
4. Creating a RandomForestClassifier object and fitting it to the training data (xv_train and y_train).
5. Using the trained RandomForestClassifier model to predict the class labels for the test data (xv_test) and storing the predictions in pred_RFC.
6. Printing a classification report that provides various metrics (such as precision, recall, F1-score, and support) by comparing the predicted labels (pred_RFC) with the true labels (y_test).

1.8 Output of testing :



```
PS D:\CODES> & C:/Users/msila/AppData/Local/Microsoft/WindowsApps/python3
              precision    recall  f1-score   support

       False       0.62      1.00      0.77         5
        True       1.00      0.40      0.57         5

    accuracy                           0.70        10
   macro avg       0.81      0.70      0.67        10
weighted avg       0.81      0.70      0.67        10
```

Fig. 14 : result of testing

The given information appears to be a classification report for a machine learning model's performance on a dataset. The report includes metrics such as precision, recall, and F1-score, which are commonly used to evaluate the performance of a classification model.

In this case, the model's performance is assessed on two classes: False and True. Let's break down the metrics:

Precision: It measures the proportion of correctly predicted instances of a class out of all instances predicted as that class. For the False class, the precision is 0.62, indicating that 62% of the instances predicted as False were actually True. For the True class, the precision is 1.00, meaning that all instances predicted as True were indeed True.

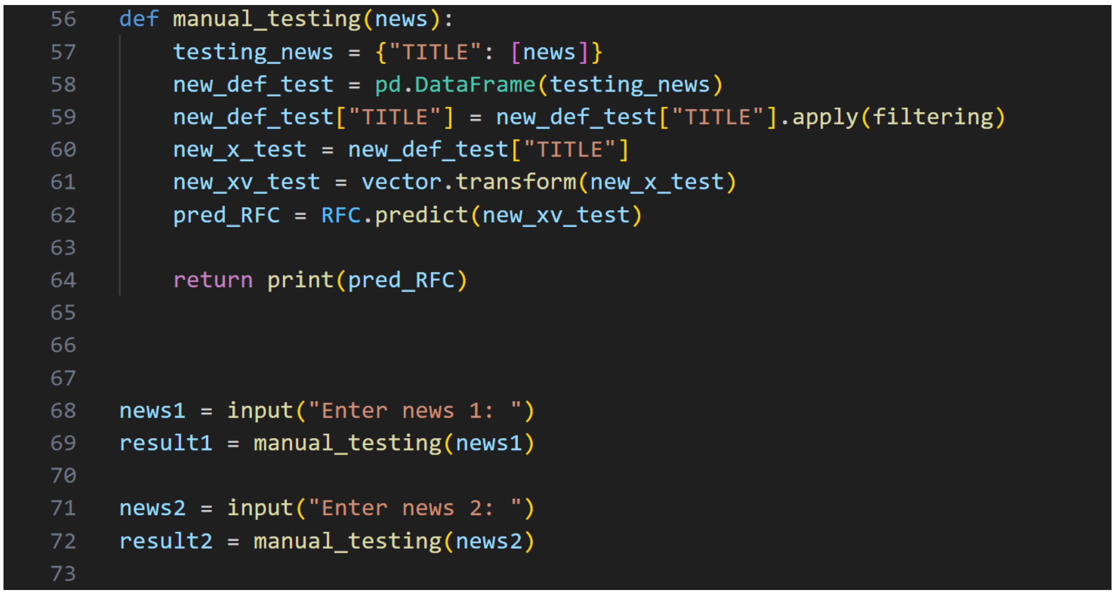
Recall: It calculates the proportion of correctly predicted instances of a class out of all instances that actually belong to that class. For the False class, the recall is 1.00, indicating that all instances that were actually False were correctly identified. However, for the True class, the recall is 0.40, indicating that only 40% of the True instances were correctly classified.

F1-score: It is the harmonic mean of precision and recall, providing a balanced measure of the model's accuracy. For the False class, the F1-score is 0.77, representing a balanced performance between precision and recall. For the True class, the F1-score is 0.57, indicating a lower accuracy due to the lower recall value.

Support: It represents the number of instances belonging to each class in the dataset. In this case, there are 5 instances for each class.

Overall, the accuracy of the model is 0.70, meaning it correctly predicted the class for 70% of the instances in the dataset. The macro average of precision, recall, and F1-score is calculated by taking the average across classes, giving more weight to the smaller class (True). The weighted average takes into account class imbalance as well.

1.9 Manual testing :



```python
def manual_testing(news):
    testing_news = {"TITLE": [news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["TITLE"] = new_def_test["TITLE"].apply(filtering)
    new_x_test = new_def_test["TITLE"]
    new_xv_test = vector.transform(new_x_test)
    pred_RFC = RFC.predict(new_xv_test)

    return print(pred_RFC)



news1 = input("Enter news 1: ")
result1 = manual_testing(news1)

news2 = input("Enter news 2: ")
result2 = manual_testing(news2)
```

Fig. 15 : testing final output

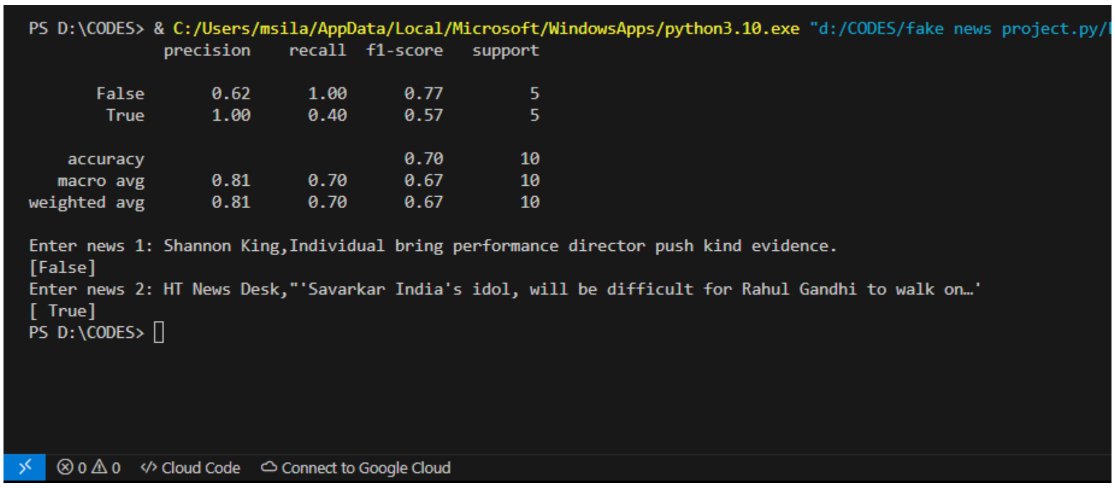
The code provided defines a function called manual_testing that allows users to manually test the trained model by providing news inputs. Here's a breakdown of what the code does:

1. The function manual_testing takes a news input (news) as a parameter.
2. It creates a dictionary testing_news with the news input under the key "TITLE".
3. The dictionary is converted into a DataFrame new_def_test using pd.DataFrame(testing_news), assuming pd represents the pandas library.
4. The filtering function is applied to the "TITLE" column of the DataFrame using the apply method. This suggests that there is a custom function filtering defined elsewhere in the code.
5. The processed news input is extracted from the DataFrame as new_x_test.
6. The transform method of the TfidfVectorizer object (vector) is used to transform the processed news input (new_x_test) into a numerical matrix (new_xv_test).
7. The trained RandomForestClassifier model (RFC) is used to predict the class label for the transformed news input (new_xv_test), and the predictions are stored in pred_RFC.

Finally, the function prints the predicted class label (pred_RFC).

The code also demonstrates the usage of the manual_testing function by taking user inputs for news and calling the function. However, there are a couple of issues with the code. First, the filtering function is referenced but not provided in the code snippet. Second, the function manual_testing already prints the predicted class label, so assigning the function call to result1 and result2 will assign None to those variables.

1.10 Manual testing result :



```
PS D:\CODES> & C:/Users/msila/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/CODES/fake news project.py/
              precision    recall  f1-score   support

       False       0.62      1.00      0.77         5
        True       1.00      0.40      0.57         5

    accuracy                           0.70        10
   macro avg       0.81      0.70      0.67        10
weighted avg       0.81      0.70      0.67        10

Enter news 1: Shannon King,Individual bring performance director push kind evidence.
[False]
Enter news 2: HT News Desk,"'Savarkar India's idol, will be difficult for Rahul Gandhi to walk on…'
[ True]
PS D:\CODES> 
```

Fig. 16 : manual testing of the news

DISCUSSION

1.1 Tracking Important Neurons :

As seen in the results that compare the most important trigrams in classifying fake versus real news (see Table 9.4, there are some obvious differences in the types of words that the CNN model looks for. The words in our results are interesting for two reasons. The first reason is that they were among the most common 1000 words (including stop words) in the accumulated trigrams supporting a category.

As such, they were repeatedly found to be important words in the classification decision. The second reason they are interesting is because their presence in the table means that they were exclusively found in the most common 1000 words for one category, and not the other. This should be very telling if they were so common in one categorys important trigrams but hardly appeared in the other categorys most important trigrams. Like the results, I will break down by part of speech the discussion of the differences between the words found to most strongly support a "Fake" classification vs a "Real" classification.

In the Noun category, it seems that there are some differences in topics. While we eliminated proper nouns for this reason, words like "museum", "music", "opera", and "novelist" are certainly instances of topics that would likely not solicit fake news.

However, there are definitely some nouns that show stylistic differences in writing. It seems as though the real news trigrams include many professional sources such as "economist", "experts", "historian", "scientist" and "teacher". The real news trigrams also contain more analytical words like "benefit", "cost", "figures", "formula", "survey", "table." Contrastively, the fake news nouns include many informal/exaggeration words such as "aliens", "asylum", "blacks", "catastrophe", "corruption", "conspiracy", "enemy", "gods", "greed", "liar", "traitors". Generalization words like "anyone", "anything", "everybody", "everything", "somebody." Offensive words like "bullshit", "coward", "cult", "idiot", "shit", "thugs".

In the Verb category, we observed very conclusive words such as causing, conclude, confirmed, promised, prove, and revealed. This observation insinuates to us that a pattern of fake news may be the tendency to jump to conclusions and make unsupported claims. To complement this observation, we found significantly less conclusive verbs in the real word trigrams such as affected, expected, qualified, suggested. This may highlight the fact that real news is intended to report the facts that are actually there and not draw causation from correlations while fake news will often make unbacked claims. In addition to conclusiveness, we found colloquialisms like “gotta” and “hate” in the important fake news trigrams.

The adjectives in real news, like the verbs, are not very “exciting.” Some examples include conservative, professional, and original. On the other hand, the adjectives that support a fake news classification are much more extreme and amusing. Some examples of extreme adjectives include “evil”, “false”, “fascist”, “illegitimate”, “incredible”, “sexual”, “unconstitutional”, and even better, “whopping”. In the adverb category, we had similar findings that indicate that the model picked up on the exaggerated conclusiveness and generalizations in fake news and the honest, more soft conclusions found in real news. Examples of cautiously inconclusive adverbs in the important real news trigrams include “largely”, “often”, “partly”, “possibly”, “slightly”, “sometimes”. These words show a tendency for real news authors to not overstate. Likewise, we find highly conclusive adverbs in the important fake news trigrams, including “completely”, “entirely”, “grossly”, “obviously”, “never”, “precisely”, “totally”, and “truly”. Even in the numbers category, the numbers presented in real news, “five” and “six” are much smaller than “millions” and “thousands” found in fake news, representing a possible tendency to overstate in fake news.

1.2 Topic Dependency

The accuracy does decrease across the board when we separate training and testing datasets by the inclusion or exclusion of a popular “topic” word, as seen in Figure. However, the accuracies are still high enough to believe that although the original model may have relied somewhat on these topic words for classification, a model retrained without the topic is able to pick up new important trigrams to pay attention to.

This supports the conclusion that the more we control our dataset, the more likely it is that the model will pick up on the types of stylistic language patterns that we had hoped to be able to detect with this machine learning tool. Some of the variance in how much the topic words removal from the training set affected the accuracy may come from how many samples in each category, and overall, included that word. Because this number varied, the training/test split changed from trial to trial as did the vocabulary size of the training set that the model was able to learn embeddings for.

1.3 Cleaning

From step 1 to step 2, the overall accuracy went up, but the vocab size was cut by 75 %, as demonstrated by Figure 16. In general, decreasing the vocabulary size would restrict what the model can learn. In this case, however, it seems to direct the models attention to more important and meaningful words (i.e. words found in the English dictionary). While the overall accuracy increased, the accuracy of fake news detection decreased, as we can see This shows that the number of false positives increased, probably due to fake news articles that were easily detectable through their inclusion of non-real English words. However, the number of false negatives decreased very significantly, meaning that the model likely had to "look closer" at some real news articles instead of immediately classifying them as fake due to their inclusion of non-real English words. The accuracy decreased for overall accuracy, real news accuracy and fake news accuracy. While many projects attempt to create the most accurate neural net possible, our goal was to create a dataset and model that would show the potential of machine learning to be useful in language pattern detection for the purpose of fake news classification. So, our goal was to create a more challenging circumstance for the neural net, an almost adversarial technique. By removing the pattern seen at the end of every article from The Guardian, we decreased the models accuracies but also increased the "helpfulness" of its trigrams. We would expect the number of trigrams that affect this classification to be high. As a human journalist wouldnt advise classifying an article based on one set of three words, but rather by taking a holistic view of the language in the article. With the removal of the pattern in the Guardian, we found an increase in the spread of neuron weights, as indicated by the standard deviations.

1.4 Describing Neurons :

These results, as demonstrated in Tables that in the final model, the neurons were looking for somewhat-distinct and cohesive patterns. In our original trials, there were many neurons that detected the "this content" pattern, and not many that seemed cohesive or sensible otherwise. After cleaning, it seems as though there are more diverse patterns being detected and the neural net is learning patterns of similar "types" of words that likely have similar embeddings and existence in news articles. It seems as though some of them have two patterns which may mean that we couldve used more filters to separate the distinct patterns. An increase in our dataset size would also help the cohesiveness because with such a small dataset size, it is likely that there are not enough articles that have common patterns.

4. CONCLUSION

We learned to detect fake news with Python. We took a political dataset, implemented a TfidfVectorizer, initialized a RandomForestClassifier, and fit our model. We ended up obtaining an accuracy of 70% in magnitude And we tested the result manually that the news is true or false.

5. FUTURE WORK

Through the work done in this project, we have shown that machine learning certainly does have the capacity to pick up on sometimes subtle language patterns that may be difficult for humans to pick up on. The next steps involved in this project come in three different aspects. The first of aspect that could be improved in this project is augmenting and increasing the size of the dataset. We feel that more data would be beneficial in ridding the model of any bias based on specific patterns in the source. There is also question as to weather or not the size of our dataset is sufficient.

The second aspect in which this project could be expanded is by comparing it to humans performing the same task. Comparing the accuracies would be beneficial in deciding whether or not the dataset is representative of how difficult the task of separating fake from real news is. If humans are more accurate than the model, it may mean that we need to choose more deceptive fake news examples. Because we acknowledge that this is only one tool in a toolbox that would really be required for an end-to-end system for classifying fake news, we expect that its accuracy will never reach perfect. However, it may be beneficial as a stand-alone application if its accuracy is already higher than human accuracy at the same task. In addition to comparing the accuracy to human accuracy, it would also be interesting to compare
the phrases/trigrams that a human would point out if asked what they based their classification decision on. Then, we could quantify how similar these patterns are to those that humans find indicative of fake and real news.

Finally, as we have mentioned throughout, this application is only one that would be necessary in a larger toolbox that could function as a highly accurate fakenews classifier. Other tools that would need to be built may include a fact detector and a stance detector. In order to combine all of these "routines," there would need to be some type of model that combines all of the tools and learns how to weight each of them in its final decision.

6. REFERENCES

- https://scikit-learn.org/
- https://pandas.pydata.org/
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html