**Social Media Analytics**

*A project report submitted to ICT Academy of Kerala*

*in partial fulfillment of the requirements*

*for the certification of*

**CERTIFIED SPECIALIST**

**IN**

**DATA SCIENCE & ANALYTICS**

submitted by

**Shaima Jinsy P A**

**Musthafa Neduvanchery**

**Haneena V P**

**Adarsh B**



**ICT ACADEMY OF KERALA**

**THIRUVANANTHAPURAM, KERALA, INDIA**

**Dec 2023**

# List of Figures

| Figure 1 | Implementation Plan | 10 |
| --- | --- | --- |
| Figure 2 | Before Filling Missing Values | 14 |
| Figure 3 | After Filling Missing Values | 14 |
| Figure 4 | Dropping Duplicate Values | 15 |
| Figure 5 | Data Transformation | 15 |
| Figure 6 | Dataframe Information | 15 |
| Figure 7 | Target Value Count Plot | 16 |
| Figure 8 | Target Value Count Pie Chart | 17 |
| Figure 9 | Data Serialization | 21 |
| Figure 10 | Snapshots of Flask Web application Coding | 25 |
| Figure 11 | Loading necessary libraries and trained model | 25 |
| Figure12 | Snapshots of Flask application after deployment-Initial Page | 27 |
| Figure13 | Snapshots of Flask application after deployment-Final Page | 27 |

# List of Abbreviations

| 1. | EDA | Exploratory Data Analysis |
| --- | --- | --- |
| 2. | DF | Dataframe |
| 3. | TF-IDF | Term Frequency - Inverse Document Frequency |
| 4. | SVM | Support Vector Machine |
| 5. | RF | Random Forest Classifier |
| 6. | NB | Naive Bayes |

**Table of Contents**

|  | Abstract | 5 |
| --- | --- | --- |
| 1. | Problem Definition | 6 |
| 2. | Introduction | 7 |
| 3. | Literature Survey | 9 |
| 4. | Implementation Plan | 10 |
| 5. | Dataset Preparation | 13 |
| 6. | Model Training And Web App Deployment | 20 |
| 7. | Result | 27 |
| 8. | Conclusion | 28 |
|  | *Reference* | 29 |

# Abstract

The growing impact of fake news on social media and communication necessitates improved detection methods. An important objective in enhancing the reliability of information in online social networks is to promptly identify fake news. However, limited resources have hindered progress in this area. Therefore, this project aims to investigate and implement a platform for accurately detecting fake news. The proposed dataset, named *'fake\_news\_dataset.csv'*, is compiled from various publicly available datasets on US domestic politics and international affairs. We have combined similar columns and merged relevant features into a single dataset. By utilizing diverse machine learning algorithms and ensemble methods, the project aims to evaluate their performance and develop a web application for fact-checking fake news. The absence of labeled benchmark datasets has constrained statistical approaches, underscoring the significance of this endeavor. Ultimately, this project addresses the critical need to combat fake news and contributes to the trustworthiness of information in online social networks.

# 1. Problem Definition

## 1.1 Overview

The proliferation of fake news and misinformation has become a significant societal issue, impacting public discourse, decision-making, and trust in media sources. To address this challenge, the project aims to develop a fact-checking platform that utilizes a classification model to distinguish between real and fake news articles. The platform will provide users with a reliable tool to verify the authenticity of news content and combat the spread of misinformation.

## **1.2 Problem Statement**

The primary problem addressed by this project is the need for an effective mechanism to identify and categorize news articles as real or fake. The proliferation of misinformation and disinformation in the digital age has led to widespread confusion and distrust in media sources. As a result, there is a critical need for a fact-checking platform that can accurately assess the credibility of news content.

Specifically, the problems to be addressed include:

* Identifying and selecting a suitable dataset containing labeled news articles for model training.
* Developing preprocessing procedures to clean and tailor the dataset, addressing issues such as text normalization and handling imbalanced classes.
* Creating a robust classification model capable of accurately predicting whether a given news article is real or fake.
* Establishing a user-friendly web hosting platform using Python Flask to deploy the classification model and provide a practical tool for fact-checking.

By addressing these challenges, the project aims to contribute to the fight against misinformation by providing a reliable and accessible solution for verifying the authenticity of news articles.

# 2. Introduction

In this project, we aim to develop a fake news classification model to be used in a fact-checking platform. The project will involve the following steps:

1. Selecting a Dataset and Performing Exploratory Data Analysis (EDA)
2. Performing Extensive Preprocessing Procedures to Clean and Tailor the Dataset
3. Creating a Classification Model to Predict Fake News
4. Establishing a Python Flask Web Hosting for the Fact-Checking Platform

Selecting a Dataset and Performing EDA For this project, we will select a suitable dataset containing textual information, such as news articles, along with their labels indicating whether they are real or fake. The EDA process will involve analyzing the structure of the dataset, identifying any missing or inconsistent data, and gaining insights into the characteristics of real and fake news articles.

Performing Extensive Preprocessing Procedures during this phase, we will implement various preprocessing techniques such as text cleaning, tokenization, stop-word removal, and lemmatization to prepare the textual data for model training. Additionally, we will handle any imbalanced classes and address any outliers or noise in the dataset.

Creating a Classification Model We will develop a classification model, such as a Logistic Regression or Random Forest Classifier. The model will be trained on the preprocessed dataset to predict whether a given news article is real or fake.

Establishing a Python Flask Web Hosting After developing the classification model, we will establish a web hosting platform using Python Flask. This platform will serve as a fact-checking tool where users can input news articles' titles, and the model will classify them as real or fake. The platform will provide a user-friendly interface for individuals to verify the authenticity of news content.

By completing these steps, we aim to contribute to the efforts in combating disinformation and promoting media literacy by providing a reliable tool for identifying fake news.

# 3. Literature Survey

The development of a fact-checking platform and a fake news classification model is a topic of significant interest in the field of natural language processing, machine learning, and media studies. There are multidimensional aspects of fake news detection ranging from using chatbots for spread of misinformation to use of clickbaits for the rumor spreading . There are many clickbaits available in social media networks including facebook which enhance sharing and liking Proceedings of posts which in turn spreads falsified information. Lot of work has been done to detect falsified information.. The literature survey for this project encompasses the following key areas:

## 3.1 Fake News Detection and Classification Datasets

The custom dataset created using various sources such as Mendeley, GitHub, and Kaggle provides a diverse collection of labeled news articles, offering an opportunity to explore different types and contexts of fake news. These datasets have been used by researchers and practitioners to develop and evaluate fake news detection models, providing valuable insights into the complexities of misinformation.

## 3.2 Exploratory Data Analysis (EDA) with Seaborn and Matplotlib

The utilization of Seaborn and Matplotlib for data visualization and exploratory data analysis has been well-documented in the literature. Studies have demonstrated the effectiveness of these libraries in visualizing textual data characteristics, word distributions, and class imbalances, providing essential insights for model development

## 3.3 Data Preprocessing with Python Libraries

Leveraging Python libraries such as Pandas and NumPy for data preprocessing has been a common practice in the literature. Researchers have utilized these libraries to handle missing values, perform text normalization, and address data quality issues, laying the groundwork for effective model training and evaluation.

## 3.4 Classification Models in Python

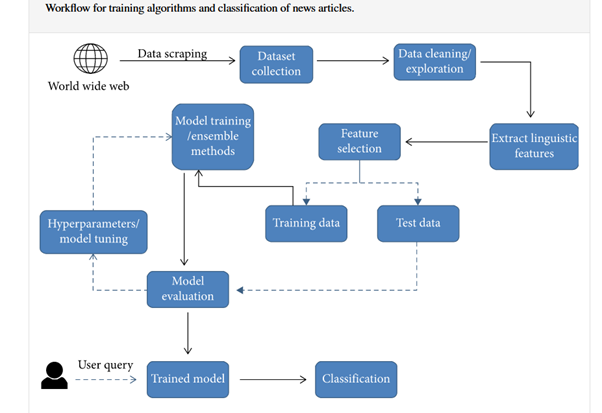
Research papers and practical guides have extensively discussed the implementation of classification models, including logistic regression, random forest classifier, and others, using Python-based machine learning libraries. These resources have highlighted the strengths and limitations of various algorithms for fake news classification tasks.

## 3.5 Python Flask Web Application Development

The development of web applications using Python Flask for deploying machine learning models has gained significant attention in the literature. Scholars and practitioners have outlined methodologies for integrating classification models into web hosting platforms, enabling the creation of user-friendly applications for fact-checking and misinformation detection.

By synthesizing the findings from these key areas, the project aims to leverage the insights from the literature and the practical application of the custom dataset to develop an effective fact-checking platform with a robust fake news classification model. The literature survey serves as a foundation for integrating best practices and insights from prior research into the project's methodology and approach.

# 4. Implementation Plan

****

As shown in the above diagram which indicate the workflow of our project and various intermediate stages. Initially we have to scrape data from various publicly available datasets which can meet our requirements.

Data cleaning is a crucial step in the implementation plan, as it directly impacts the quality and reliability of the subsequent analysis and model training. The purpose of Data cleaning is to get our data ready to analyze and visualize. During this phase, the collected data undergoes comprehensive preprocessing to address various issues such as missing values, outliers, and inconsistencies. Techniques such as imputation, outlier detection, and normalization are employed to ensure that the dataset is in a suitable form for further processing. Additionally, standardizing the data format and structure is essential to facilitate seamless integration with the model and to enable accurate feature extraction and selection. By conducting thorough data cleaning, we aim to enhance the overall robustness and integrity of the dataset, laying a solid foundation for subsequent stages of the implementation plan.

The process of extracting linguistic features involves leveraging available techniques to derive meaningful insights and patterns from textual data. This encompasses tasks such as tokenization, lemmatization, and the extraction of syntactic and semantic structures to capture the inherent linguistic characteristics within the dataset. Furthermore, feature selection plays a pivotal role in identifying the most relevant attributes that contribute to the predictive power of the model. By employing methods such as statistical tests, dimensionality reduction, and domain knowledge, we aim to isolate the most discriminative features while mitigating the impact of noise and redundant information. The synergy between linguistic feature extraction and feature selection is crucial in shaping the input space of the model, ensuring that it encapsulates the most pertinent linguistic properties while discarding superfluous attributes. This holistic approach not only facilitates the construction of a more refined and efficient model but also enhances its interpretability and generalization capabilities, ultimately contributing to the overall efficacy of the web application.

The division of the dataset into training and testing subsets is a fundamental aspect of the implementation plan. This process is pivotal in assessing the model's performance and generalization to unseen data. By allocating a portion of the data for training, the model can learn the underlying patterns and relationships within the dataset. The remaining portion, designated for testing, serves as an independent evaluation set to gauge the model's predictive accuracy and robustness. Striking a balance between the training and testing data is essential to prevent overfitting or underfitting, ensuring that the model achieves a harmonious equilibrium between learning from the training data and exhibiting proficiency on unseen instances. Through this meticulous partitioning, the model's ability to effectively capture and generalize patterns within the data can be systematically evaluated, laying the groundwork for the subsequent phases of model training and evaluation within the implementation plan.

The phase of model training constitutes a critical stage in the implementation plan, wherein the selected machine learning algorithm is exposed to the training data to learn the underlying patterns and relationships. Subsequently, the model's performance is rigorously evaluated using the designated test data, allowing for an assessment of its predictive accuracy and generalization capabilities. Concurrently, hyper-parameter tuning is undertaken to fine-tune the model's configuration, leveraging techniques such as grid search or randomized search to identify the optimal hyper-parameter values that enhance the model's performance. This iterative process of hyper-parameter optimization seeks to maximize the model's predictive prowess while mitigating the risk of overfitting. Upon achieving an optimized configuration, the final model is derived, encapsulating the refined parameters and hyper-parameter settings. This culmination represents the culmination of an intricate process, culminating in the deployment of a robust and finely-tuned model, poised to underpin the functionality of the web application with its adept predictive capabilities.

The development and deployment of the web application using Python Flask represent pivotal phases in the implementation plan. Leveraging the robust capabilities of Flask, a micro web framework, the development process involves the creation of a seamless and interactive interface to showcase the model's predictive functionality. Through Python's extensive libraries and Flask's intuitive structure, the web application is tailored to effortlessly interact with users, offering a user-friendly experience. Upon completion, the deployment phase ensures the accessibility of the web application to a wider audience, enabling users to leverage the predictive model's insights and functionalities. The deployment process may encompass considerations such as server configuration, scalability, and security measures to safeguard the application and its underlying model. By meticulously orchestrating the development and deployment of the web application, the implementation plan culminates in the realization of a dynamic and accessible platform, poised to deliver the model's predictive capabilities to its intended audience.

# 5. Dataset Preparation

The custom dataset created using various sources such as *Mendeley*, *GitHub*, and *Kaggle* provides a diverse collection of labeled news articles, offering an opportunity to explore different types and contexts of fake news. Combining data values from various datasets which have similar subject relevancy and relatable features may come with more potential tasks in the cleaning process.

## 5.1 Data Preprocessing

For machine learning , it’s necessary to convert raw data into a clean data set, which means we must convert the data set to numeric data. We do this by encoding all the categorical labels to column vectors with binary values. Missing values, or NaNs (not a number) in the data set is handled by either dropping the missing rows or filling them up with a mean or interpolated values. The below mentioned preprocessing steps which we completed for our dataset for preprocessing

### 5.1.1 Drop Columns That Aren’t Useful

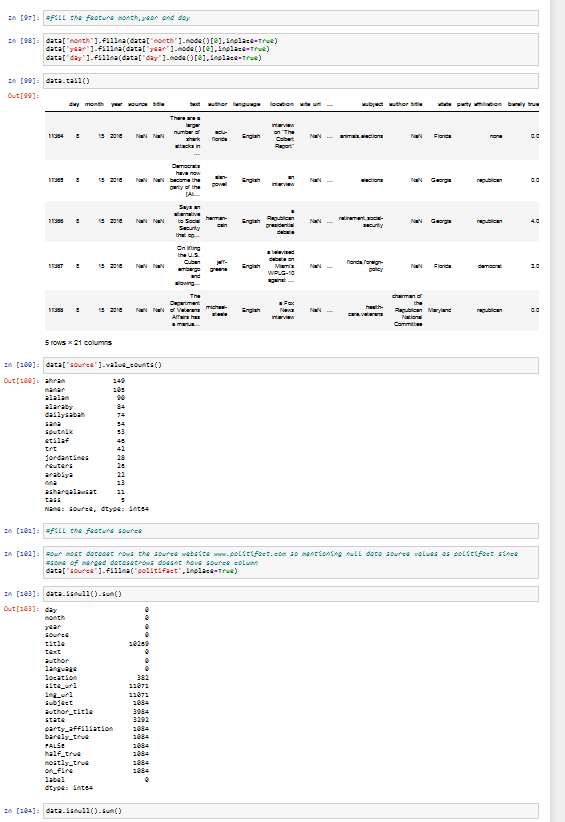
Drop some of the columns which won't contribute much to our machine learning model. Here we dropped the column ‘id’

### 5.1.2 Handling missing data

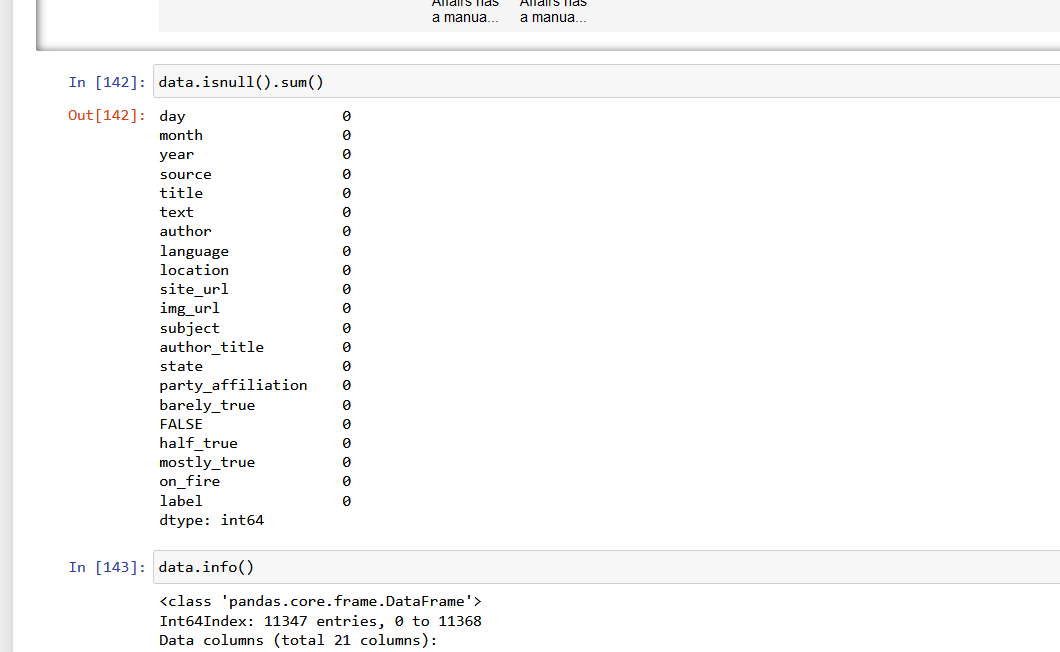
Dealing with missing data is a common and inherent issue in data collection, especially when working with large dataset**s.** If missing values have been found, there are particularly two ways to resolve this issue:

* Either Remove the entire row that contains a missing value. However, removing the entire row can generate a possibility of losing some important data. This approach is useful if the dataset is very large
* Or Estimate the value by taking the mean, median or mode.

Here in our dataset, we mainly filled the missed data by taking either mean or mode. Details shown below



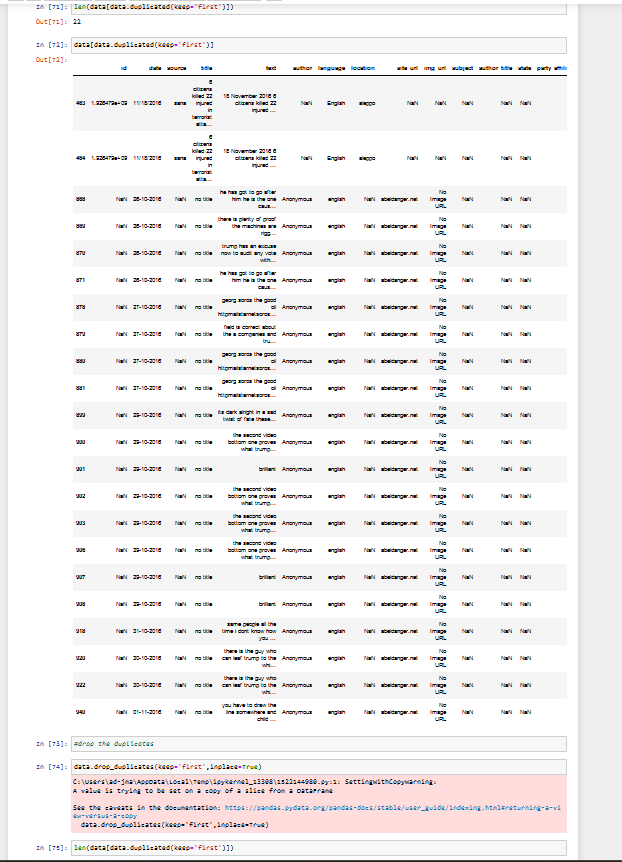
After filling missing values in the dataset:



### 5.1.3 Dropping dataset rows

Once we have identified duplicates in the dataset, it is time to remove them. To delete duplicates, we use a function *drop\_duplicates* in Pandas which is shown below:

Further the for better performance of our dataset, we reduced row counts to almost half of the original dataset. So the current dataset has a rows count of 6335 rows.

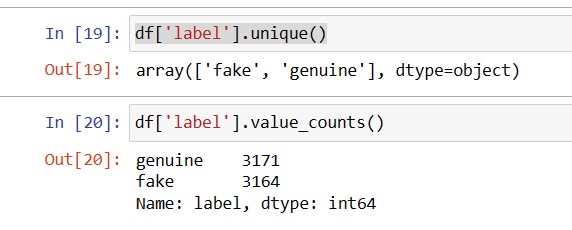




### 5.1.3 Data Transformation

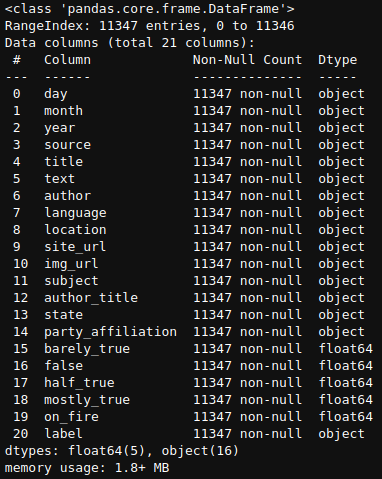
Data transformation involves converting the data from one form to another to make it more suitable for analysis. Techniques such as normalization, scaling, or encoding can be used to transform the data. Here we scaled our target variable to desired values by reducing the target feature columns values and classifying it into basically two unique values as shown below:



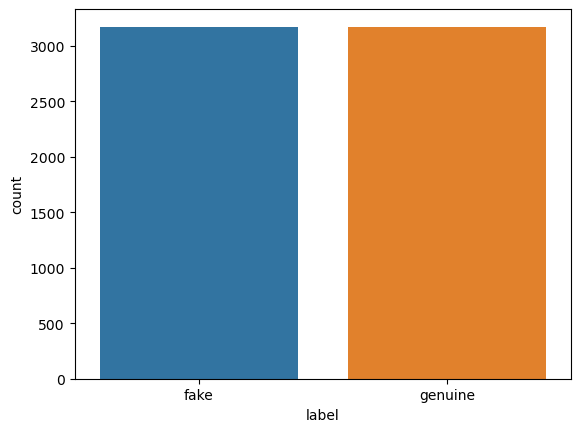


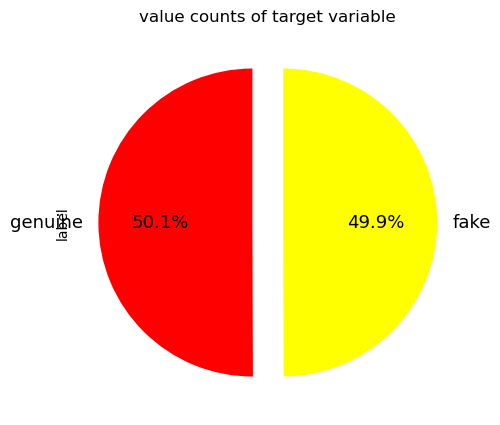
For better model analysis, we are using text feature column .

## 5.2 Exploratory Data Analysis

****

The cleaned dataset contains 21 columns including both feature sets and a target column, and the dataset contains more than 6000 rows. The columns are *day, month, year, source, title, text, author, language, location, site\_url, img\_url, subject, author\_title, state, party\_affiliation, barely\_true, false, half\_true, mostly\_true, on\_fire* and *label*.





The target column of our dataset contains only two unique values, which are *genuine* and *fake.* The plot shows the unique values in the target column plotted against their individual counts using seaborn’s countplot. And data values in the target columns are  *fake* and *genuine*; and their counts are 3164 and 3171 respectively, and the percentage values are 49.9 and 50.1 respectively.

## 5.3 Dataframe Splitting

We have to split the dataset into train and test data prior to Tokenization which may avoid data leakage and thereby give better performance.



## 5.4 Text Tokenization

In the dataset, the column called *text* is very important, which is also a very integral part of our project. As our project is based on text analysis and their prediction. The technique we are going to use in this project is *Term Frequency - Inverse Document Frequency (TF-IDF)*, in order to get better results, it is necessary to remove any noisefrom the textual data prior to the tokenization.

## 5.4.1 Noise Removal

The process of noise removal from text strings is a crucial step in preparing the dataset for fake news classification. Python, particularly the Pandas library, offers powerful tools for effectively handling textual data. The noise removal process involves several techniques, including removing numbers, cleaning special characters, and spell checking. This section outlines the methodology for noise removal using Python Pandas and associated techniques.

To eliminate numerical characters from the text strings, the isnumeric() method provided by Python can be leveraged. This method allows for the identification of numeric strings within the dataset, enabling their removal to enhance the quality of the textual data. The implementation involves iterating through the title and excluding strings that consist solely of numeric characters.

cleaned\_data['text'] = cleaned\_data['text'].apply(lambda x: ' '.join(word for word in x.split() if not word.isnumeric()))

Regular expressions (regex) can be employed to remove special characters and non-alphanumeric symbols from the title strings. By utilizing the pattern [^\w\s], which matches any non-alphanumeric or non-whitespace characters, the unwanted noise in the title data can be effectively filtered out, leading to cleaner and more coherent title content.

cleaned\_data['text'] = cleaned\_data['text'].apply(lambda x: re.sub(r'[^\w\s]', '', x))

By incorporating these techniques into the data preprocessing pipeline using Python Pandas, the noise removal process significantly enhances the quality and interpretability of the title data, contributing to the robustness of the subsequent fake news classification model.

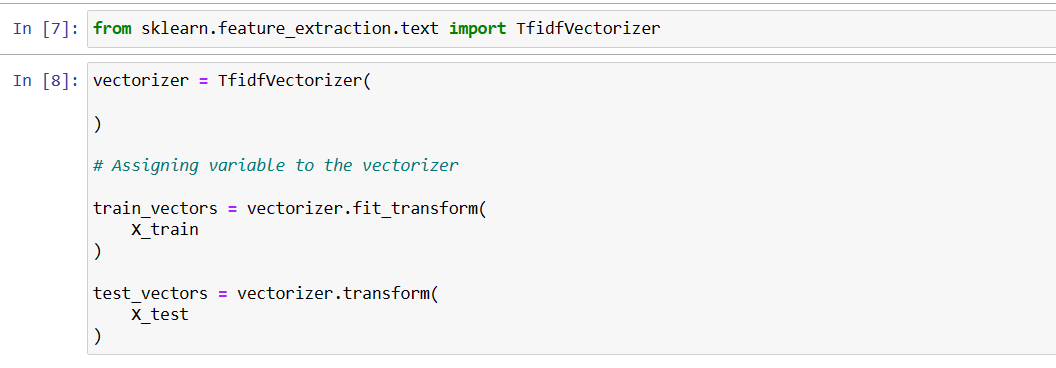
### 5.4.2 Term Frequency - Inverse Document Frequency

TF-IDF (Term Frequency-Inverse Document Frequency) technique plays a pivotal role in text data preprocessing and feature extraction. TF-IDF is widely used in natural language processing and information retrieval to quantify the importance of terms in a document relative to a collection of documents. Including a discussion of TF-IDF in the project report is essential to demonstrate the utilization of advanced text processing techniques.

* **Term Frequency (TF):** It measures the frequency of a term within a document. It is calculated as the number of times a term appears in a document divided by the total number of terms in the document. Higher TF values indicate the importance of a term within a specific document.
* **Inverse Document Frequency (IDF):** It evaluates the significance of a term across the entire document collection. IDF is computed as the logarithm of the total number of documents divided by the number of documents containing the term. Terms with low document frequency and high IDF scores are considered more informative.
* **TF-IDF Calculation:** The TF-IDF score of a term in a document is obtained by multiplying its TF by its IDF. This results in a measure that highlights terms that are both frequent within a specific document and rare across the entire document collection.

In our project, TF-IDF can be applied to the textual data as part of the feature extraction process, which involves transforming the raw text into a numerical representation that can be utilized by machine learning algorithms. The reasons to use TF-IDF in this project are:

* *Feature Representation:* TF-IDF provides a method for converting unstructured text data into a structured numerical format, facilitating the training of machine learning models.
* *Feature Importance:* It highlights the significance of terms within individual documents and the broader document collection, aiding in the identification of relevant features for classification tasks.
* *Dimensionality Reduction:* TF-IDF can contribute to reducing the dimensionality of the feature space by emphasizing informative terms and downplaying common terms.



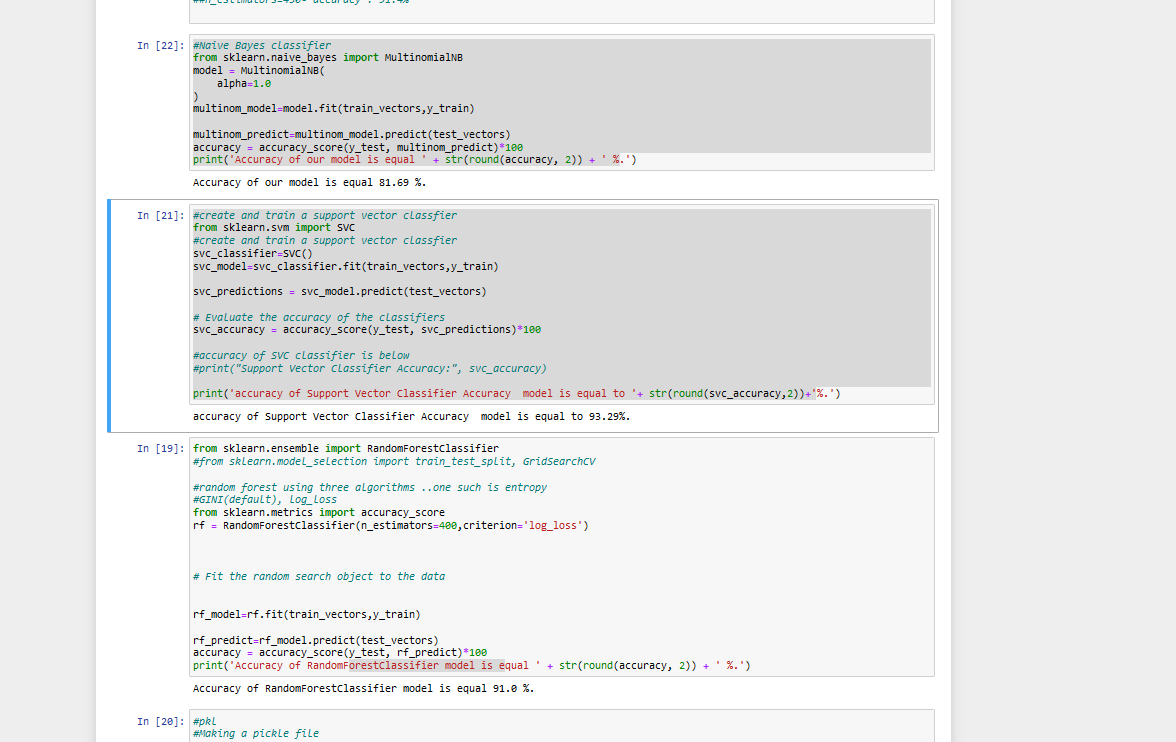
# 6. Model Training And Deployment

After the completion of dataset cleaning and preprocessing, then next immediate to follow model training and getting a good accuracy score. And a successfully evaluated dataset can be used in the web application development using data serialization, which follows deploying the web application in a suitable platform.

## 6.1 Model Training

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from. Here in our dataset we performed various machine learning algorithms which include SVM(Support Vector Machine), Naive Bayes and Random Forest Classifiers.

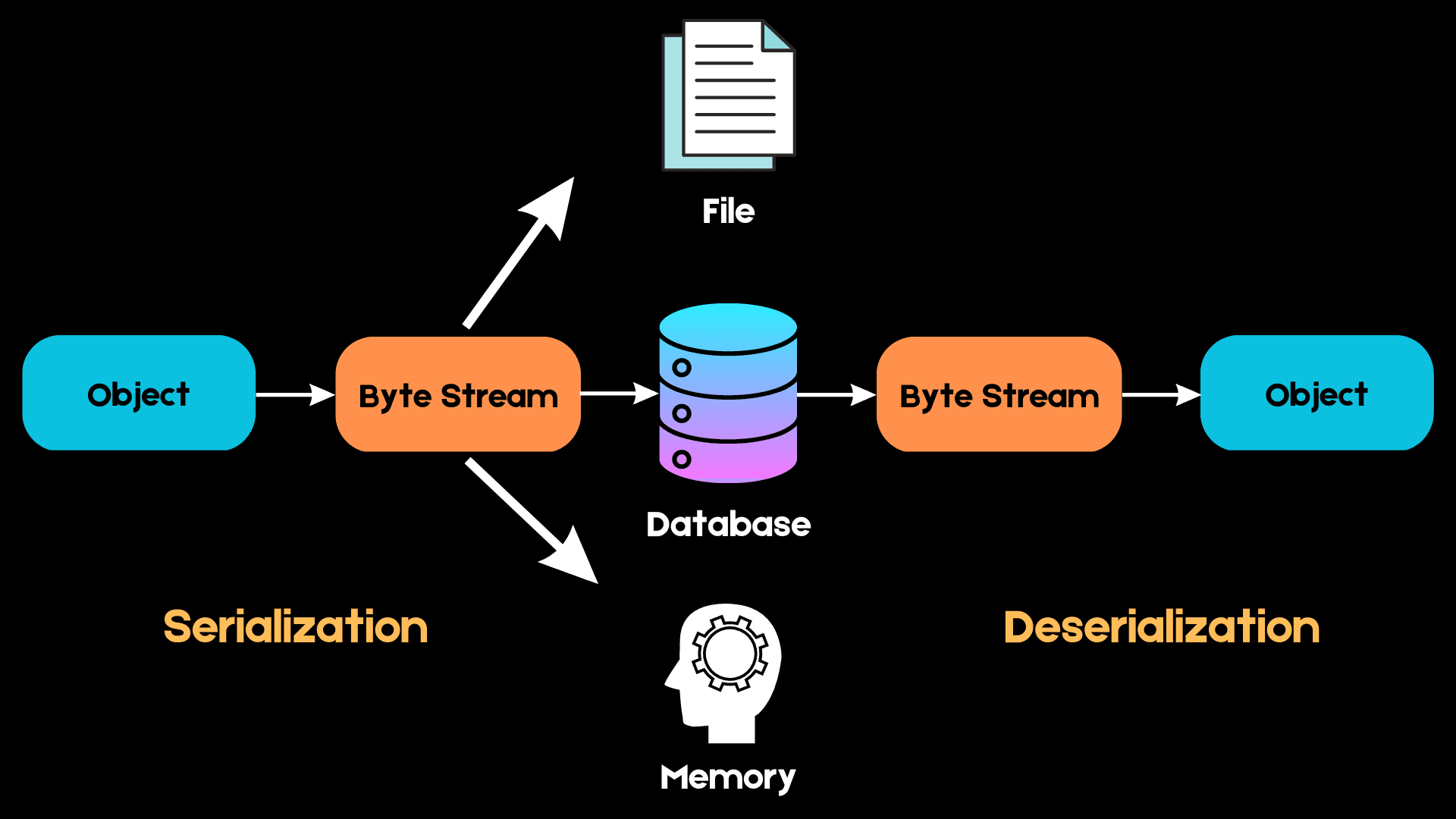
Below shows training the accuracy code comparison for SVM, NB and RF classifiers.



We have got an accuracy score for SVM as 93.29% , for Rf as 91% and for NB it's 81.69%. Based on the score , we decided to go with the SVM classifier for model training.

## 6.2 Data Serialization using pickle

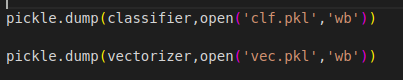
Serialization is essential in order to use a trained model in a web application, which directly converts a python coded model into a byte stream file that is more convenient using different programmes or platforms. Here is simple diagram showing how serialization works:



The next time we want to access the same data structure, this sequence of bytes must be converted back into the high-level object in a process known as **deserialization**.

We can use formats such as JSON, XML, HDF5, and Pickle for serialization. In this project we are using a pickle library for the serialization process. The reason to choose pickle library is that is native to the python environment and also one of the most capable serialization algorithms available out there.

Here we use the *dump* method from the pickle library by referring to the trained model and save as a pickle file with an ‘*.pkl’* extension. And in our project we have to pickle the Tf Idf vectorizing file into a pickle file other than the classifier model.



### 6.2.1 Advantages of using Pickle to serialize objects

* Unlike serialization formats like JSON, which cannot handle tuples and datetime objects, Pickle can serialize almost every commonly used built-in Python data type. It also retains the exact state of the object which JSON cannot do.
* Pickle is also a good choice when storing recursive structures since it only writes an object once.
* Pickle allows for flexibility when deserializing objects. We can easily save different variables into a Pickle file and load them back in a different Python session, recovering your data exactly the way it was without having to edit your code.

### 6.2.2 Disadvantages of using Pickle

* Pickle is unsafe because it can execute malicious Python callables to construct objects. When deserializing an object, Pickle cannot tell the difference between a malicious callable and a non-malicious one. Due to this, users can end up executing arbitrary code during deserialization.
* Pickle is a Python-specific module, and we may struggle to deserialize pickled objects when using a different language.
* According to multiple benchmarks, Pickle appears to be slower and produces larger serialized values than formats such as JSON and ApacheThrift.

## 6.3 Web Page Designing

The web page design of our Flask web app deployment is tailored to provide a seamless and intuitive user experience. The design elements and features are aimed at facilitating easy interaction and information presentation. Below is an overview of the key components of our web page design.

The content section of our web page allows users to input their news article into a text area or upload a text file. Upon submission, the system processes the input and provides relevant information based on the content. Our web page includes easy-to-use navigation elements to enhance the user experience. Users can submit their article, clear the input area, and upload a text file using the provided buttons and file input field.

We have incorporated visual elements such as the company logo and icons to improve the aesthetics and overall appeal of the web page. The use of icons for buttons and file input enhances the interactivity and visual appeal.

Upon submission of an article, relevant information is displayed in a visually appealing manner. The system shows the category of the content, word count, and how many words are matched to the model. This information is displayed using a combination of text, marks, and well-organized sections. The design of our web page aims to create a user-friendly environment where users can seamlessly interact with the application and obtain meaningful insights from their input.

This structured description provides an overview of the web page design and its key elements, allowing readers to understand the approach taken to create an engaging and functional user interface for the Flask web app deployment.

In our Flask web app deployment, the web page design is complemented by custom CSS styling to enhance the visual appeal and user interaction. JavaScript has been utilized in the project to enhance the interactivity and functionality of the web page.

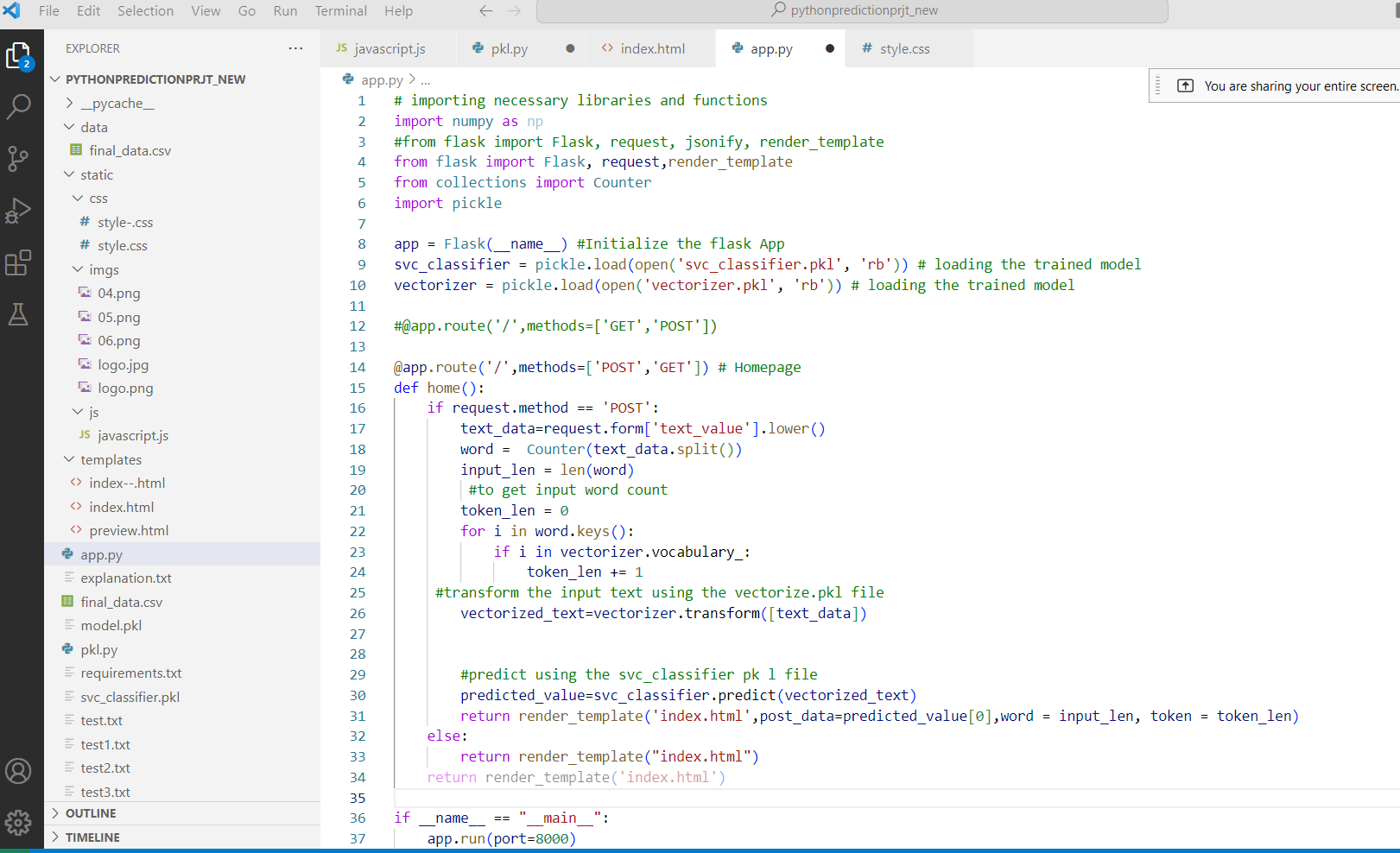
## 6.3 Flask Web Application

In the project, a Flask application has been developed to handle the classification of news text using a pre-trained model that was mentioned earlier. The following steps outline the creation and functionality of the Flask application:

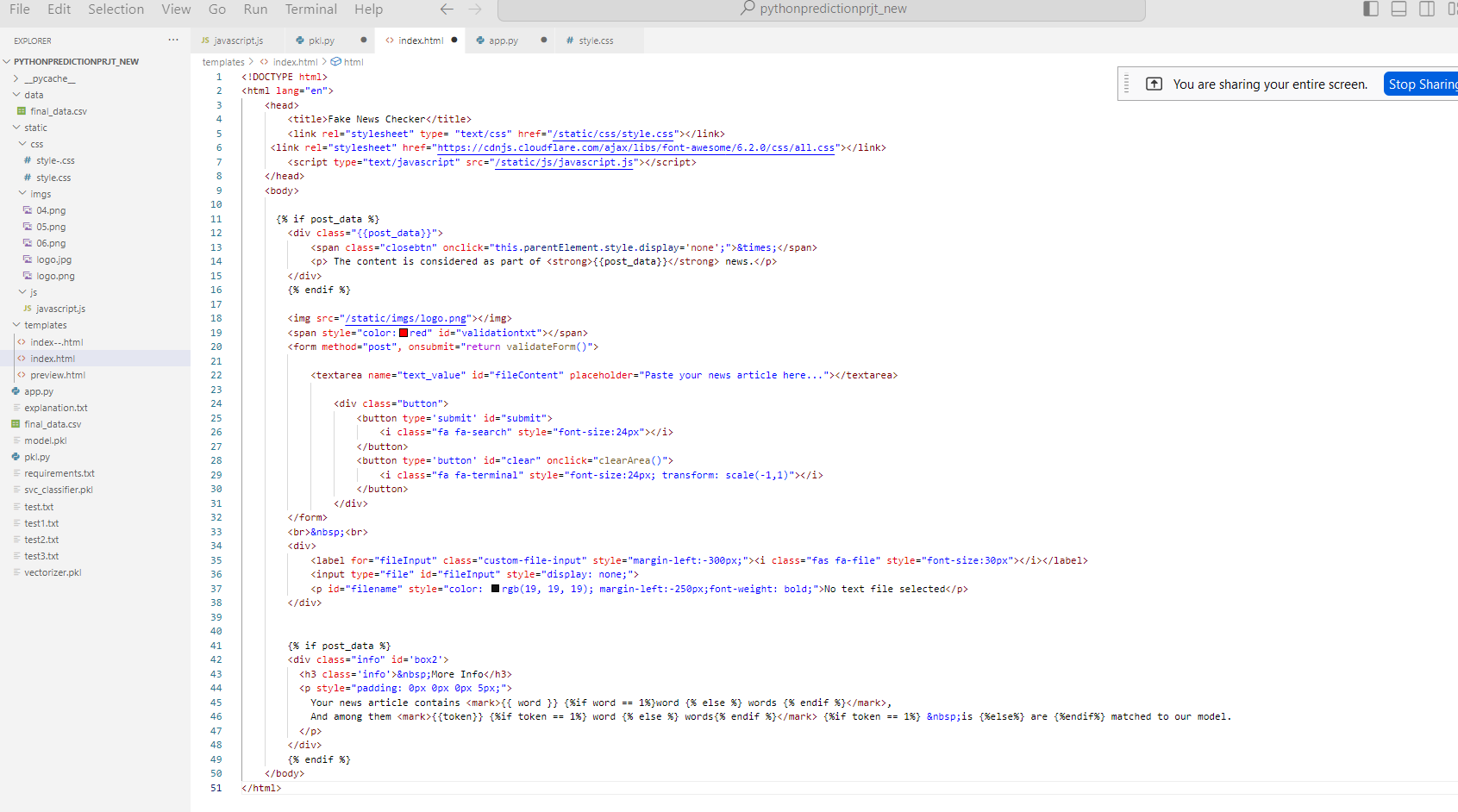
* **Importing Libraries and Modules:** The necessary libraries including Flask, request, render\_template, pickle, and Counter are imported at the beginning of the script.
* **Initializing the Flask App:** The Flask application is initiated using Flask(name), creating an instance of the Flask class.
* **Loading the Classifier and Vectorizer:** The pre-trained classifier and vectorizer are loaded using pickle.load to deserialize the saved model files.
* **Defining the Routes and Views:** The ‘/’ route is defined to handle both GET and POST requests. On POST request, the input news text is processed, transformed using the vectorizer, and then classified using the loaded classifier. The results along with relevant statistics (word count and token count) are passed to the ‘*home.html’* template for display.
* **Running the Flask App:** The main block contains app.run(port=8000) to run the Flask app on port 8000.

By incorporating Flask in the project, the team has developed a robust web application that leverages a pre-trained model for news text classification, providing users with real-time insights and analysis. On the local host it is giving the expected result.

**6.3.1 Snapshots of Flask Web application Coding :**

****

**Loading necessary libraries and trained model**

****

**index.html web page code (main page)**

## 6.4 Deploying the Flask Web Application

In order to deploy our application we need a website hosting service provider. First priority was the service which should be free since it is a project model. And we choose to host in *pythonanywhere.com*, which is lightweight and easy to configure apart from providing limited free service.

Within the PythonAnywhere dashboard, a new web app was created, and the Flask web framework was selected as the preferred framework for our application. Additionally, the appropriate Python version was specified to align with our project requirements.

The web app was configured by setting up a virtual environment and specifying the project path, ensuring that all necessary dependencies and project files were appropriately organized within the PythonAnywhere environment.

Critical Flask application files, including the Python script, HTML templates, and other essential components, were uploaded to the PythonAnywhere account. This step ensured that the entire application, along with its resources, was seamlessly integrated into the hosting environment.

Following the configuration and file upload process, the web app was initiated from the PythonAnywhere dashboard, marking the commencement of our Flask application's availability over the internet.

**Web Url Links to our python web application after deployment**:

Link1: <https://pyexpert.pythonanywhere.com/>(Deployed by Musthafa Neduvanchery)

Link2: <https://haneenavp.pythonanywhere.com/> (Deployed by Haneena VP)

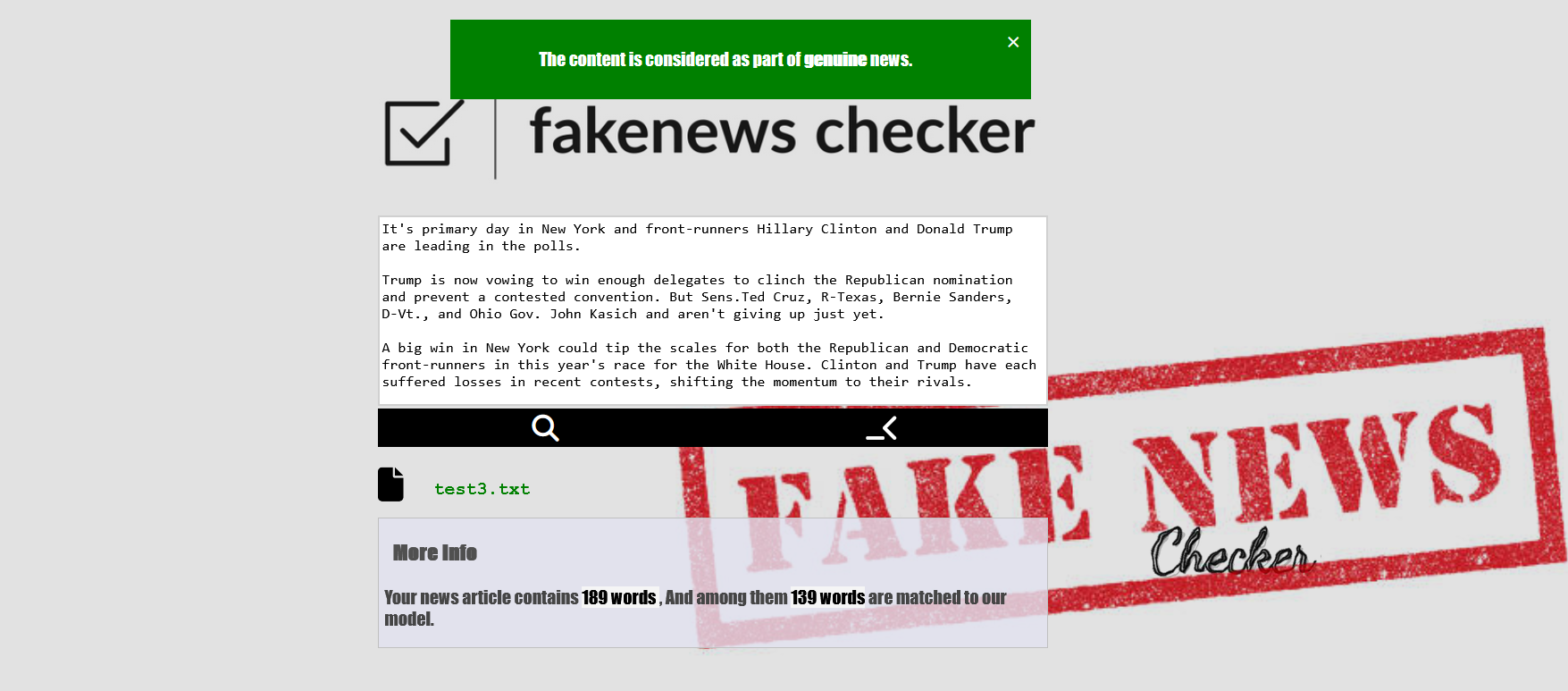
Link3: <https://pythonprjtprediction.pythonanywhere.com/> (Deployed by Adarsh Babu)

Link4:<https://projectfakenewspred.pythonanywhere.com/> (Deployed by Shaimajinsy)

**6.4.1 Snapshots of Flask application after deployment :**

****

**Initial page**

****

**Final page after submitting a news content**

# 7. Result

Upon successful deployment, the hosted Flask application became accessible via a unique domain provided by PythonAnywhere. Users could interact with the application through a standard web browser, enabling seamless access to the news article classification system.

The successful hosting of our Flask application on PythonAnywhere has extended the reach of our news article classification system, making it readily available to users across the internet. This hosting solution aligns with our project's objective of ensuring widespread accessibility and usability.

# 8. Conclusion

Our project focused on the development and deployment of a news article classification system using machine learning and web technologies. Through the collaborative efforts of our team, we successfully achieved the following key outcomes:

1. System Development:

We designed and implemented a robust news article classification system that leverages machine learning techniques to analyze and categorize news content based on predefined categories. The system's intuitive user interface allows seamless interaction, making it accessible to a wide audience.

1. Technology Integration:

By integrating machine learning models with a Flask web application, we created a cohesive platform that empowers users to submit news articles for classification, receive real-time results, and contribute to a dynamic dataset for continuous model improvement.

1. Deployment on PythonAnywhere:

The hosting of our Flask application on PythonAnywhere further extended the reach of our news article classification system, ensuring its availability and accessibility to users across the internet. This deployment marked a significant milestone in making our system widely available.

1. User Engagement and Impact:

Throughout the development and deployment phases, we prioritized user engagement and usability, aiming to provide a valuable tool for individuals seeking efficient news categorization. The successful deployment and accessibility of our system align with our vision of creating a practical and impactful solution.

1. Future Prospects:

As we conclude this phase of the project, we recognize the potential for future enhancements and iterations. This includes expanding the dataset, refining the machine learning models, and incorporating user feedback to further improve the accuracy and relevance of the classification system.

In conclusion, our project has realized its primary objectives of developing, deploying, and making accessible a news article classification system that harnesses the power of machine learning and web technologies. The successful collaboration, technical innovation, and commitment to user-centric design have culminated in a valuable solution with the potential for continued evolution and positive impact.

# References

**Dataset:**

1. <https://data.mendeley.com/datasets/945z9xkc8d/1>
2. [https://github.com/topics/fake-news-detection](https://github.com/topics/fake-news-detectipn)
3. <https://kaggle.com/code/abhishek09/fake-news-dataset-beginner/input>
4. <https://kaggle.com/datasets/mohamadalhasan/a-fake-news-dataset-around-the-syrian-war>

**Python Libraries**

1. <https://www.numpy.org>
2. <https://www.pandas.pydata.org/>
3. <https://www.matplotlib.org/stable/>
4. <https://www.matplotlib.org/stable/tutorials/pyplot.html>
5. <https://www.seaborn.pydata.org/>
6. <https://www.flask.palletsprojects.com/en/3.0.x/>

**Other Urls**

1. <https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/>
2. <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089>