A PROJECT REPORT

on

MediBot

Project accomplished by:

Pranay Gupta (Roll No.: 10800220003)

Nidhi Kumari (Roll No.: 10800220071)

Koushik Sadhu (Roll No.: 10800220027)

Shuvam Kumar Choudhury (Roll No.: 10800220031)

Under the guidance of:

Mr. Sudip Kumar De

Assistant Professor

INFORMATION TECHNOLOGY

Asansol Engineering College

Asansol

AFFILIATED TO

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

June 2024

ASANSOL ENGINEERING COLLEGE

Kanyapur, Vivekananda Sarani, Asansol, Paschim Bardhaman, West Bengal - 713305
Phone: 225-3057, 225-2108 Telefax: (0341) 225-6334
E-mail: principal.aecwb@gmail.com Website: www.aecwb.edu.in

CERTIFICATE

Certified that this project report on **MediBot** is the bonafide work done by **"Pranay Gupta (10800220003), Nidhi Kumari (10800220071), Koushik Sadhu (10800220027), and Shuvam Kumar Choudhury (10800220031)"**, who carried out the project work under my supervision.

Mr. Sudip Kumar De	**Dr. Anup Kumar Mukhopadhyay**
Assistant Professor	HoD, Information Technology
Information Technology	

INFORMATION TECHNOLOGY

Asansol Engineering College

Asansol

ACKNOWLEDGEMENT

It is our great privilege to express our profound and sincere gratitude to our Project Supervisor, **Mr. Sudip Kumar De**, Assistant Professor in the Information Technology Department at Asansol Engineering College for providing us with invaluable and cooperative guidance at every stage of our project work under his supervision. His valuable advice and instructions in carrying out the present study have been a very rewarding and pleasurable experience that has greatly benefitted us throughout our work.

We would also like to pay our heartiest thanks and gratitude to **Dr. Anup Kumar Mukhopadhyay**, HoD and all the faculty members of the Information Technology Department, Asansol Engineering College for various suggestions being provided in attaining success in our work.

Finally, we would like to express our deep sense of gratitude to our parents for their constant motivation and support throughout our work.

Pranay Gupta (Roll No.: 10800220003)

Nidhi Kumari (Roll No.: 10800220071)

Koushik Sadhu (Roll No.: 10800220027)

Shuvam Kumar Choudhury (Roll No.: 10800220031)

Date: ___/___/______ **4th Year**

Place: Asansol **Information Technology**

CONTENT

Certificate …………………………………………….............……………….... ii

Acknowledgement ……………………………………….............…...…..... iii

Content ……………………………………………………………..............…….. iv

List of Tables …………………………………..............………………….. v

List of Figures ………………………………….............………………….. vi

Project Synopsis ……………………………………………..............……... vii

Introduction ……………………………………………………….................. 01

Project Details ……………………………………………………….................. 02 - 35

2.1 System Requirements ……………………………………………….... 02 - 03
2.2 Proposed Methodology …………………………………………….. 04 - 05
2.3 Definitions and Theories ………………………………………….... 06 - 07
2.4 Project Workflow ………………..…………………………………… 08 - 12
2.5 Outcomes of the Project ……..…………………………………………… 13 - 14
2.6 Code Implementation ……………………………………………… 15 - 31
2.7 Application Interface ……………………………………………….. 32 - 35

Conclusion & Future Scope …....…………………………...….............….... 36

References ……………..……………………………...……..……................. 37

LIST OF TABLES

Table No.	Table Name	Page No.
Table 1	Hardware Requirements	2
Table 2	Software Requirements	3

LIST OF FIGURES

Figure No.	Figure Name	Page No.
Figure 1	Workflow of Medicine Recommendation System	8
Figure 2	Workflow of Brain Tumour Detection	10
Figure 3	Reading Medicine Dataset	15
Figure 4	Overview of Dataframe Statistics and Structure	16
Figure 5	Data Cleaning and Initial Data Exploration	17
Figure 6	Medicine Composition Frequency Count	18
Figure 7	Composition Visualization Code	18
Figure 8	Top 10 Composition Frequencies Bar Plot	19
Figure 9	Plot Preparation Code for Top 10 Uses of Medicines	20
Figure 10	Top 10 Occurrences of Uses and Associated Medicines Bar Plot	21
Figure 11	TF-IDF Vectorization and Combined Matrix Formation	22
Figure 12	Cosine Similarity Matrix of Medications	22
Figure 13	Medicines Recommendation by Symptoms Function	23
Figure 14	Medicines Recommendation Based on User Input Symptoms	24
Figure 15	Data Preparation for Brain Tumor Classification	25
Figure 16	Visualizing Brain Tumor Data	26
Figure 17	Reshaped Image Data and Train-Test Split	27
Figure 18	Data Normalization for Train and Test Sets	27
Figure 19	Feature Selection (PCA)	28
Figure 20	Training and Testing Classification Models	29
Figure 21	Misclassification Analysis of SVM Model	30
Figure 22	Visualization of Model Predictions on Test Images	31
Figure 23	MediBot Homepage Interface	32
Figure 24	Medicine Recommendation System Interface	32
Figure 25	Medicine Recommendation for Fever	33
Figure 26	Brain Tumour Detection Interface	34
Figure 27	Selecting MRI Scan Image	34
Figure 28	Tumour Prediction - Tumour Detected	35
Figure 29	Tumour Prediction: No Tumour Detected	35

PROJECT SYNOPSIS

In the ever-evolving landscape of healthcare technology, the demand for accurate, timely diagnostic tools and personalized treatment recommendations remains paramount. The project, MediBot: Intelligent Medicine Recommendation System with Brain Tumor Detection, is an innovative endeavour aimed at revolutionizing personalized healthcare through the integration of artificial intelligence (AI) and advanced image processing techniques.

The primary aim of MediBot is to offer precise and personalized medication recommendations tailored to individual patient needs. Accurate and timely medication recommendations are crucial for managing various health conditions, ensuring that patients receive the most effective treatments with minimal side effects. In addition to this primary function, MediBot also includes a secondary, yet vital, feature for the early detection of brain tumors through MRI scan analysis. Brain tumors, often critical and life-threatening, require swift identification for effective medical intervention.

MediBot addresses critical problems in the healthcare sector. The need for efficient, accessible tools for personalized medicine recommendations and the early detection of brain tumors. Despite advancements in pharmacology and medical imaging, determining the most effective medications and interpreting complex MRI scans require specialized knowledge and time. MediBot acts as a catalyst, empowering healthcare professionals and individuals alike with a tool that enhances treatment efficacy and diagnostic accuracy, expediting the decision-making process.

The project's approach involves a strategic combination of advanced image processing, machine learning algorithms, and comprehensive pharmaceutical databases. Users input their medical histories and current health data through a user-friendly web application, which then analyses this information to recommend the most suitable medications. Additionally, users can upload MRI scan images, which the system processes to identify potential brain tumors.

1. INTRODUCTION

In the realm of healthcare technology, providing accurate medication recommendations and timely detection of critical conditions such as brain tumors remains a complex challenge. MediBot emerges as a beacon in healthcare, addressing this challenge head-on by seamlessly integrating artificial intelligence (AI) and advanced image processing techniques.

The primary aim of MediBot is to offer precise and personalized medication recommendations tailored to individual patient needs. Accurate and timely medication recommendations are crucial for managing various health conditions, ensuring that patients receive the most effective treatments with minimal side effects. Additionally, MediBot includes a secondary, yet vital, feature for the early detection of brain tumors through MRI scan analysis. Brain tumors, often critical and life-threatening, require swift identification for effective medical intervention.

MediBot simplifies these intricate processes through an intuitive web interface, empowering users to input their symptoms for personalized medication suggestions. Simultaneously, users can upload MRI scan images for swift and accurate analysis of brain tumors.

This project represents a significant advancement in personalized healthcare, where accurate medication recommendations and early detection of brain tumors play pivotal roles in treatment outcomes. By streamlining the interpretation of medical data and MRI scans, MediBot enhances the decision-making process and contributes to improved patient care.

With an emphasis on precision and reliability, the Python backend of MediBot employs state-of-the-art tools, including scikit-learn and pandas, to process patient data and images. MediBot is not merely a technological innovation; it is a collaborative effort dedicated to making a tangible impact on healthcare outcomes.

MediBot aligns with the vision of leveraging technology to enhance human well-being. This project introduction sets the stage for a deeper exploration of how MediBot combines cutting-edge technology with medical expertise to pave the way for a healthier future.

2. PROJECT DETAILS

Details of our project are outlined such as system requirements, proposed methodology, definitions and theories, project workflow, expected outcomes, code implementation, and the application interface.

2.1 SYSTEM REQUIREMENTS: The specific system requirements for MediBot will depend on the specific hardware and software components used in the system. Here are some general considerations that are required for this system.

2.1.1 Hardware: MediBot operates efficiently on standard hardware configurations commonly found in contemporary computing environments. In order to ensure the smooth development of our project, we have compiled a list of fundamental hardware requirements that are necessary to carry out the project as shown in Table 1.

Particulars	Specifications
Processor	4.10 GHz or Faster Processor
RAM	8 GB or above
Graphics	NVIDIA GeForce GTX 1050ti or above
Resolution	1920 X 1080 or Higher Resolution

Table 1: Hardware Requirements

2.1.2 Software: MediBot's software requirements encompass a range of technologies to ensure seamless functionality. In order to ensure the smooth development of our project, we have compiled a list of software requirements that are necessary to carry out the project shown in Table 2.

Development Stack	**Tools & Resources**
Operating System	Linux(preferred), Windows 10 and above
Front End	React JS, Tailwind CSS
Back End	Python, FastAPI
Libraries	NumPy, Pandas, Matplotlib, Scikit-learn, Pickle, Seaborn and Plotly
Code Editor	Jupyter Notebook, VS Code and Google Colab
Dataset	• https://www.kaggle.com/datasets/singhnavjot2062001/11000-medicine-details/download?datasetVersionNumber=84 • https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/download?datasetVersionNumber=1
Web Browser	Google Chrome, Firefox, Brave, Edge
Models	• Logistic Regression • Support Vector Machine (SVM) • TFIDF Vectorizer • Cosine Similarity

Table 2: Software Requirements

2.1.3 Network and Connectivity: MediBot's web interface demands a stable internet connection for uploading and processing MRI scan images. While the system is designed to be accessible on various network speeds, a reliable and reasonably fast connection is recommended to facilitate prompt image analysis and diagnosis.

2.1.4 Other Considerations: Considering the sensitive nature of medical data, security and privacy are paramount considerations. MediBot employs encryption protocols to safeguard user data during transmission and storage. Regular updates and maintenance procedures are integral to ensuring the system's optimal performance and security. Additionally, user education and training materials may be provided to facilitate seamless interaction with the platform, ensuring both healthcare professionals and individuals can leverage MediBot effectively.

2.2 PROPOSED METHODOLOGY: In developing MediBot for medicine recommendation and brain tumor detection, a systematic and well-defined methodology is crucial to ensure the successful integration of advanced technology into the healthcare domain. The proposed methodology encompasses key steps, each contributing to the overall effectiveness and reliability of the system.

2.2.1 Identify the needs and goals of the system:

Thorough research and collaboration with healthcare professionals will be conducted to identify the precise needs and goals of MediBot. Understanding the intricacies of medicine recommendation and brain tumor detection will guide the project's scope, ensuring alignment with the medical community's requirements for efficient and accurate diagnostic tools.

2.2.2 Select hardware and software components:

Based on the identified needs, a careful selection of hardware and software components will be executed. This involves choosing hardware configurations that balance performance with accessibility. Software technologies like React, FastAPI, and scikit-learn will be selected to create a robust, user-friendly, and technologically advanced platform suitable for medicine recommendation system and brain tumor detection.

2.2.3 Set up the hardware and software:

A meticulous setup process will follow, involving the configuration of hardware components to accommodate the computational demands of image processing. The software stack, comprising frontend and backend technologies, will be integrated cohesively to create a seamless and responsive user interface for uploading and processing MRI scan images and other relevant medical data.

2.2.4 Data Collection and Pre-processing:

Prior to testing, a comprehensive data collection and pre-processing phase will be implemented. Diverse datasets of MRI scans, patient medical histories, and other relevant medical data, meticulously curated to represent various scenarios for medicine recommendation and brain tumor detection, will be employed. Data pre-processing techniques, including normalization and augmentation, will be applied to enhance the model's adaptability and generalization capabilities.

2.2.5 Test and refine the system:

Extensive testing protocols will be implemented to validate the accuracy and reliability of MediBot's medicine recommendation and brain tumor detection capabilities. User feedback and real-world testing scenarios will be crucial in refining the system, addressing any discrepancies, and ensuring optimal performance under diverse conditions.

2.2.6 Implementation and Maintenance:

Following successful testing, the implementation phase will involve deploying MediBot for public use. Continuous monitoring and maintenance procedures will be established to address emerging issues, implement updates, and guarantee the sustained functionality and security of the system. Regular feedback loops with healthcare professionals and end-users will contribute to ongoing improvements and adaptations in response to evolving healthcare needs.

2.3 DEFINITIONS AND THEORIES:

2.3 DEFINITIONS AND THEORIES: The MediBot project is a ground-breaking healthcare initiative designed to address the critical challenge of recommending suitable medication and timely detection of brain tumors. It seamlessly integrates artificial intelligence (AI) and image processing technologies to provide a user-friendly solution accessible to both medical professionals and individuals.

2.3.1 Objective and Scope:

The primary objective of the MediBot project is to simplify the complex task of recommending suitable medication and identifying brain tumors in the realm of healthcare technology. The project focuses on creating an intuitive web interface that empowers users to upload medical data and MRI scan images for swift and accurate analysis.

2.3.2 Medical Significance:

This project represents a crucial advancement in medical diagnostics, emphasizing the importance of early detection in determining treatment outcomes for brain tumors and medication recommendations. By streamlining the interpretation of MRI scans, MediBot enhances the decision-making process, ultimately contributing to improved patient care.

2.3.3 Technological Components:

The Python backend of MediBot employs state-of-the-art tools, including scikit-learn and pandas, to ensure precision and reliability in processing uploaded medical data and MRI scan images. These technologies play a vital role in the accurate analysis and interpretation of medical data.

2.3.4 Collaborative Innovation:

MediBot is not merely a technological innovation but a collaborative effort that brings together technological expertise and medical knowledge. This collaboration aims to make a tangible impact on healthcare outcomes, particularly in the medication recommendations and early detection of brain tumors.

2.3.5 Human-Centric Approach:

The project aligns with the vision of leveraging technology to enhance human well-being. By combining cutting-edge technology with medical expertise, MediBot is dedicated to paving the way for a healthier future.

2.3.6 Integration of AI and Image Processing:

The core theoretical underpinning of the project involves the integration of artificial intelligence (AI) algorithms and advanced image processing techniques. This synergy allows for the development of a robust system capable of accurately recommending medication and analyzing brain tumors from a MRI scan images.

2.3.7 Role of Early Detection:

The project is grounded in the theory that early detection of brain tumors significantly impacts treatment outcomes. By providing a platform for quick and accurate analysis, MediBot aims to facilitate early intervention, thereby improving the prognosis for individuals with these conditions.

2.3.8 Python Backend Technology Stack:

The use of state-of-the-art tools such as scikit-learn and pandas in the Python backend is based on the theory that these technologies enhance the processing capabilities of the system. They contribute to the precision and reliability of the analysis, ensuring accurate results for medical professionals and users.

2.3.9 User-Friendly Interface:

The emphasis on a user-friendly web interface is rooted in the theory that accessibility is crucial for both medical professionals and individuals. A streamlined and intuitive interface facilitates easy upload of symptoms and MRI scan images, making the diagnostic process more accessible and efficient.

2.3.10 Holistic Impact on Healthcare:

The overarching theory guiding the project is that by combining technological innovation with medical expertise, MediBot can have a holistic impact on healthcare outcomes. This project aims to contribute to a paradigm shift in how medication recommendation system and brain tumors are approached, ultimately leading to better patient care and well-being.

2.4 PROJECT WORKFLOW: A work flow diagram is a visual representation of the flow of data within a system. The MediBot project integrates two main components: A Medicine Recommendation System and Brain Tumor Detection.

2.4.1 Work Flow of Medicine Recommendation System:

The workflow of the medicine recommendation system in the MediBot project begins with importing and analyzing the medicine dataset. The data is cleaned and preprocessed to ensure accuracy. Text features are converted into numerical form using TF-IDF vectorization. The dataset is then split into training and testing sets. Machine learning models are built and validated to measure performance. The best model is selected and fine-tuned. Finally, a prediction function recommends medicines based on user-provided symptoms, using the trained model for accurate recommendations. The workflow diagram illustrating these processes is provided below:

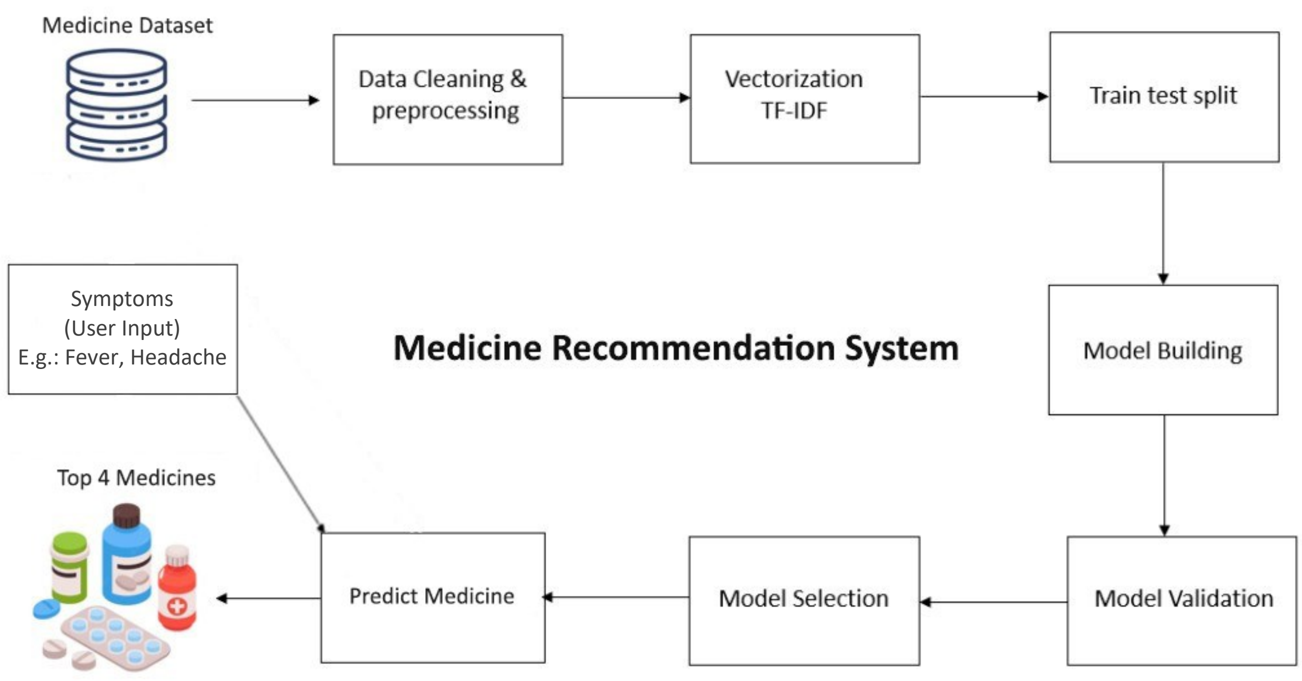


Figure 1: Workflow of Medicine Recommendation System

- **Medicine Dataset:** The process starts by importing and parsing the medicine dataset using the pandas library. We collected the Medical_Details.csv dataset from Kaggle, which contains approximately 12,000 records of medicine data. This step facilitates detailed exploratory data analysis (EDA) to understand the dataset's schema, data types, and initial statistical summaries. The dataset link is: https://www.kaggle.com/datasets/singhnavjot2062001/11000-medicine-details/download?datasetVersionNumber=84

- **Data Cleaning and Preprocessing:** Data cleaning involves removing duplicate entries (drop_duplicates) and identifying missing values. This ensures data integrity and prepares the dataset for subsequent preprocessing steps, such as text normalization and feature extraction.

- **Vectorization TF-IDF:** Once the data is cleaned, textual features such as medicine uses, compositions, and side effects are transformed into numerical representations. This is achieved using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique from sklearn.feature_extraction.text. TF-IDF calculates the importance of each word in relation to its frequency in the dataset while adjusting for the word's frequency across all documents, thus creating a meaningful numerical representation of the text data.

- **Train Test Split:** The next step involves splitting the dataset into independent training and testing subsets using the train_test_split method. This ensures that the model is trained on one portion of the data and evaluated on unseen data, preventing overfitting and providing an unbiased assessment of model performance.

- **Model Building:** Machine learning models are then constructed using the TF-IDF transformed matrices (tfidf_matrix_uses, tfidf_matrix_composition, tfidf_matrix_side_effects). For this project, cosine similarity is used, which measures the cosine of the angle between two vectors. This metric assesses the similarity between symptom vectors and medicine vectors, helping in the recommendation process.

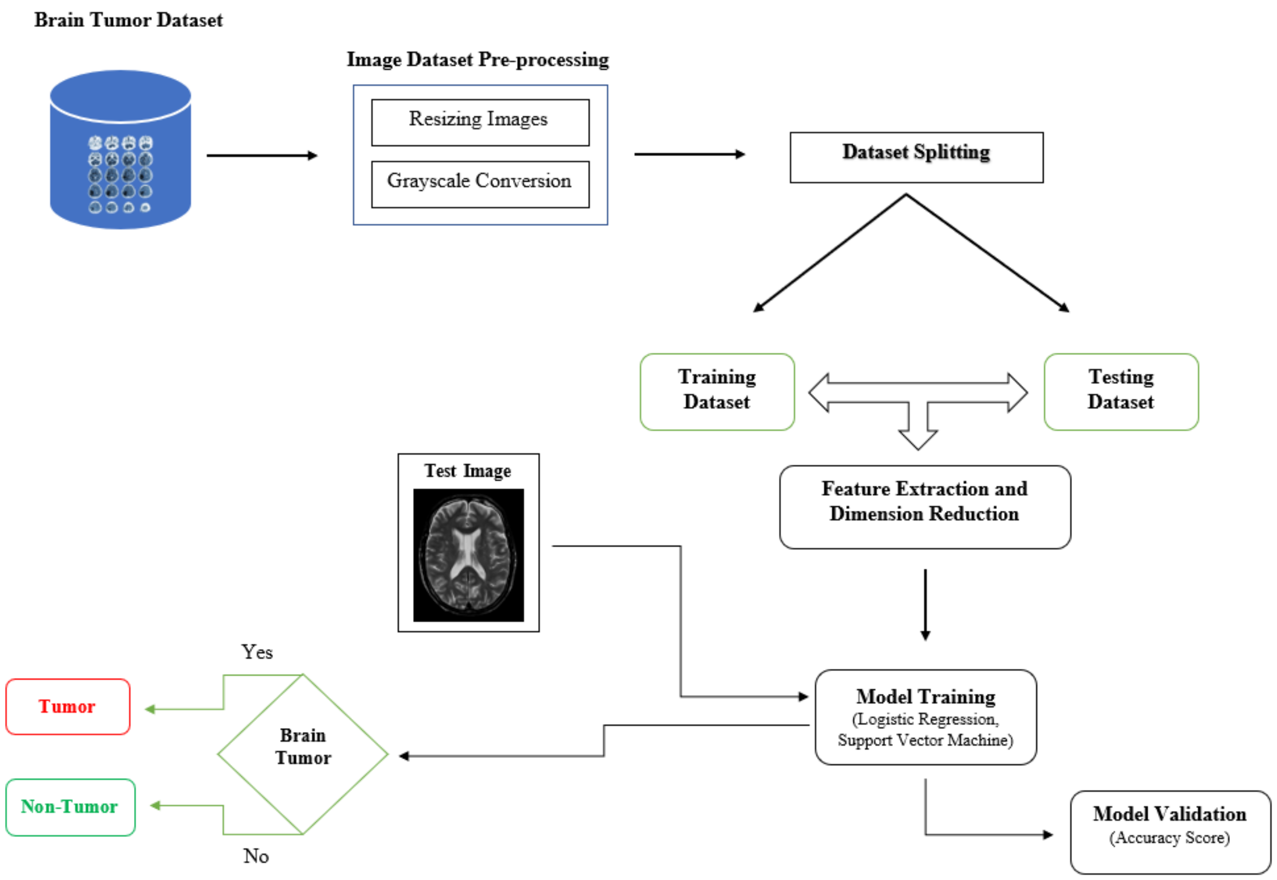
- **Model Validation:** Model validation is crucial to ensure the reliability of the predictions. It involves evaluating performance metrics such as precision, recall, and F1-score. Cross-validation techniques, like k-fold cross-validation, partition the data into multiple subsets. The model is trained on different combinations of these subsets, and the results are averaged to ensure robustness.

- **Model Selection:** Selecting the optimal model is based on rigorous validation results and criteria such as accuracy and computational efficiency. Hyperparameter tuning, performed through techniques like grid search, fine-tunes model parameters to enhance predictive performance.

- **Prediction of Medical Condition and Drug Prediction:** Finally, the implemented function predicts and recommends medicines based on user-provided symptoms. This function utilizes the pre-trained TF-IDF vectorizer stored in a serialized format (tfidf_vectorizer.pkl). This enables efficient vectorization of new symptom inputs and subsequent cosine similarity calculations for medicine recommendation. The function effectively recommends the top four medicines that best match the given symptoms based on the trained model.

2.4.2 Work Flow of Brain Tumour Detection:

The brain tumor detection module in MediBot enables users to upload MRI images for automated evaluation. Using advanced machine learning techniques, the system preprocesses uploaded images, extracts relevant features, and employs trained models to accurately detect tumors. This workflow ensures systematic processing and reliable diagnostic outcomes, supporting healthcare professionals in timely clinical decision-making. The workflow diagram illustrating these processes is provided below:



Brain Tumor Dataset
Image Dataset Pre-processing
Resizing Images
Grayscale Conversion
Dataset Splitting
Training Dataset
Testing Dataset
Feature Extraction and Dimension Reduction
Test Image
Model Training (Logistic Regression, Support Vector Machine)
Model Validation (Accuracy Score)
Brain Tumor
Yes
No
Tumor
Non-Tumor

Figure 2: Workflow of Brain Tumor Detection

- **Data Collection and Preparation:** The workflow begins with collecting medical images containing brain scans categorized into two classes: images without tumors and images with tumors. These images are prepared by standardizing their format and enhancing their suitability for analysis. We collected the dataset from Kaggle, which contains 7,023 MRI images of human brains. The dataset link is: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/download?datasetVersionNumber=1

- **Image Preprocessing:** Once collected, the images undergo preprocessing to standardize their format and enhance their suitability for analysis. This involves resizing all images to a uniform size and converting them to grayscale. These steps are crucial for simplifying subsequent processing and ensuring consistency in image analysis.

- **Dataset Splitting:** To evaluate the effectiveness of the brain tumor detection models, the dataset is divided into training and testing sets. This division allows the models to be trained on a subset of data and tested on unseen data, ensuring that their performance can be accurately assessed and validated.

- **Feature Extraction and Dimension Reduction:** Feature extraction techniques are applied to extract relevant features from the preprocessed images. Principal Component Analysis (PCA) is used to reduce the dimensionality of the feature space while retaining the most important information. This step helps in improving computational efficiency and enhancing the model's ability to generalize from the training data to unseen images.

- **Model Training:** Machine learning models, such as Logistic Regression and Support Vector Machines (SVM), are trained on the extracted features from the training dataset. These models learn to classify whether an image contains a brain tumor based on the patterns and characteristics identified during training. Model training involves adjusting model parameters to optimize performance and accuracy.

- **Model Validation:** Once trained, the models are evaluated using the testing dataset to assess their ability to accurately classify new, unseen images. Performance metrics such as accuracy, precision, recall, and F1-score are calculated to measure the effectiveness and reliability of each model in detecting brain tumors.

- **Prediction and Visualization:** Using the best-performing model, predictions are made on new medical images to determine the presence or absence of brain tumors. These predictions are visualized alongside the corresponding actual images to provide a clear demonstration of the model's diagnostic capabilities. This visualization aids in understanding how the model interprets and categorizes medical images for clinical decision-making. Once validated, the trained model is deployed within the MediBot system. Users upload MRI images, and the system automatically evaluates these images to brain tumors.

2.5 OUTCOMES OF THE PROJECT: The outcomes highlight the project's achievements in recommending medicines and early detection of brain tumors, improving diagnostic accuracy, streamlining the analysis of medical data, and enhancing healthcare delivery. These outcomes demonstrate the system's potential to support medical professionals in making informed decisions. Specific outcomes include:

2.5.1 Medicine Recommendation and Early Detection of Brain Tumours:

The primary goal of the project is to recommend medicines for particular symptoms and the early detection of brain tumors. By leveraging AI and data-processing the system provides tailored medication recommendations based on the provided symptoms. Additionally, the system enhances the ability to identify brain tumors at an early stage when intervention and treatment options are most effective.

2.5.2 Improved Diagnostic Accuracy:

The integration of advanced technologies, including AI algorithms, contributes to improved diagnostic accuracy. The system aims to provide more precise and reliable results in the interpretation of medical data and MRI scan images, assisting medical professionals in making informed decisions for brain tumors and medication recommendations.

2.5.3 Efficient Analysis of Medical Data and MRI Scans:

The project streamlines the interpretation of medical data and MRI scans through a user-friendly web interface. This leads to more efficient analysis, reducing the time required for medical professionals to examine images and make diagnostic assessments for brain tumors and medication recommendations.

2.5.4 Enhanced Decision-Making Process:

By automating aspects of the analysis process, the system supports medical professionals in the decision-making process for medication recommendations and brain tumors. It provides valuable insights, potentially leading to quicker and more accurate treatment plans based on the identified conditions.

2.5.5 Accessible Healthcare Technology:

The user-friendly interface of the system makes healthcare technology more accessible to a broader audience. Individuals can upload their medical data and MRI scan images, fostering a collaborative approach to healthcare where both medical professionals and individuals actively participate in the diagnostic process for brain tumors and medication recommendations.

2.5.6 Contribution to Medical Diagnostics:

The project represents a significant advancement in medical diagnostics, particularly in the field of neurology. The successful implementation of the system contributes to the ongoing evolution of diagnostic tools and methodologies, setting a precedent for the integration of AI in healthcare for and medication recommendations and brain tumor detection.

2.5.7 Potential for Increased Treatment Success:

Early detection, coupled with improved diagnostic accuracy, has the potential to increase the success of brain tumor treatments. Identifying these conditions in their early stages allows for more targeted and effective interventions, potentially leading to better patient outcomes.

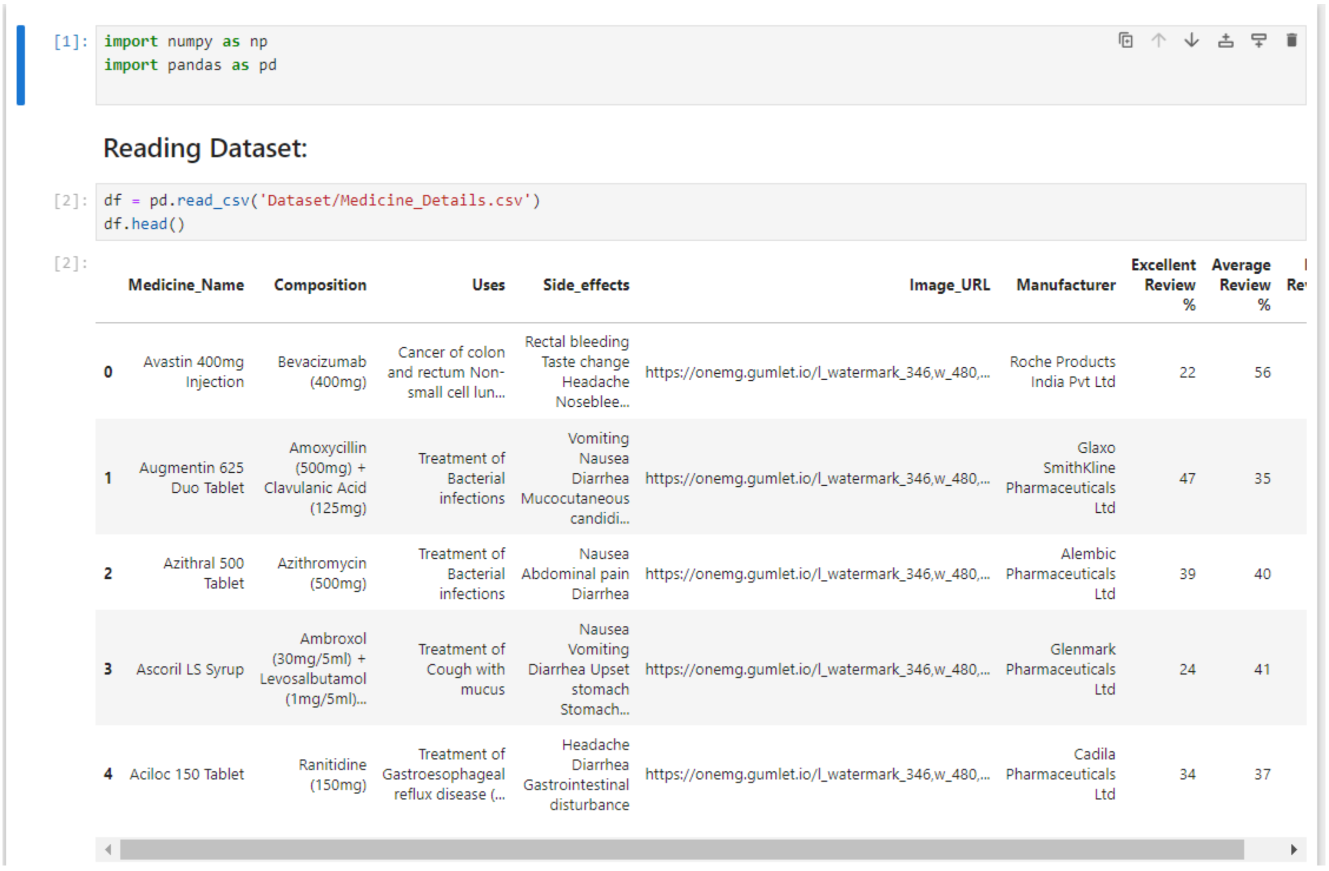
2.5.8 Holistic Impact on Healthcare Outcomes:

The collaborative effort behind the project, combining technological innovation with medical expertise, aims to have a holistic impact on healthcare outcomes. The project aligns with the broader vision of leveraging technology to enhance human well-being and contribute to a healthier future.

2.6 CODE IMPLEMENTATION: The detailed overview of the code implementation used in both the Medicine Recommendation System and Brain Tumor Detection within MediBot is discussed below.

2.6.1 Medicine Recommendation System Code:

In this figure 3, we import the necessary libraries, load a CSV file containing medicine details into a Dataframe, and display the first few rows of the dataset.



```
[1]: import numpy as np
     import pandas as pd
```

Reading Dataset:

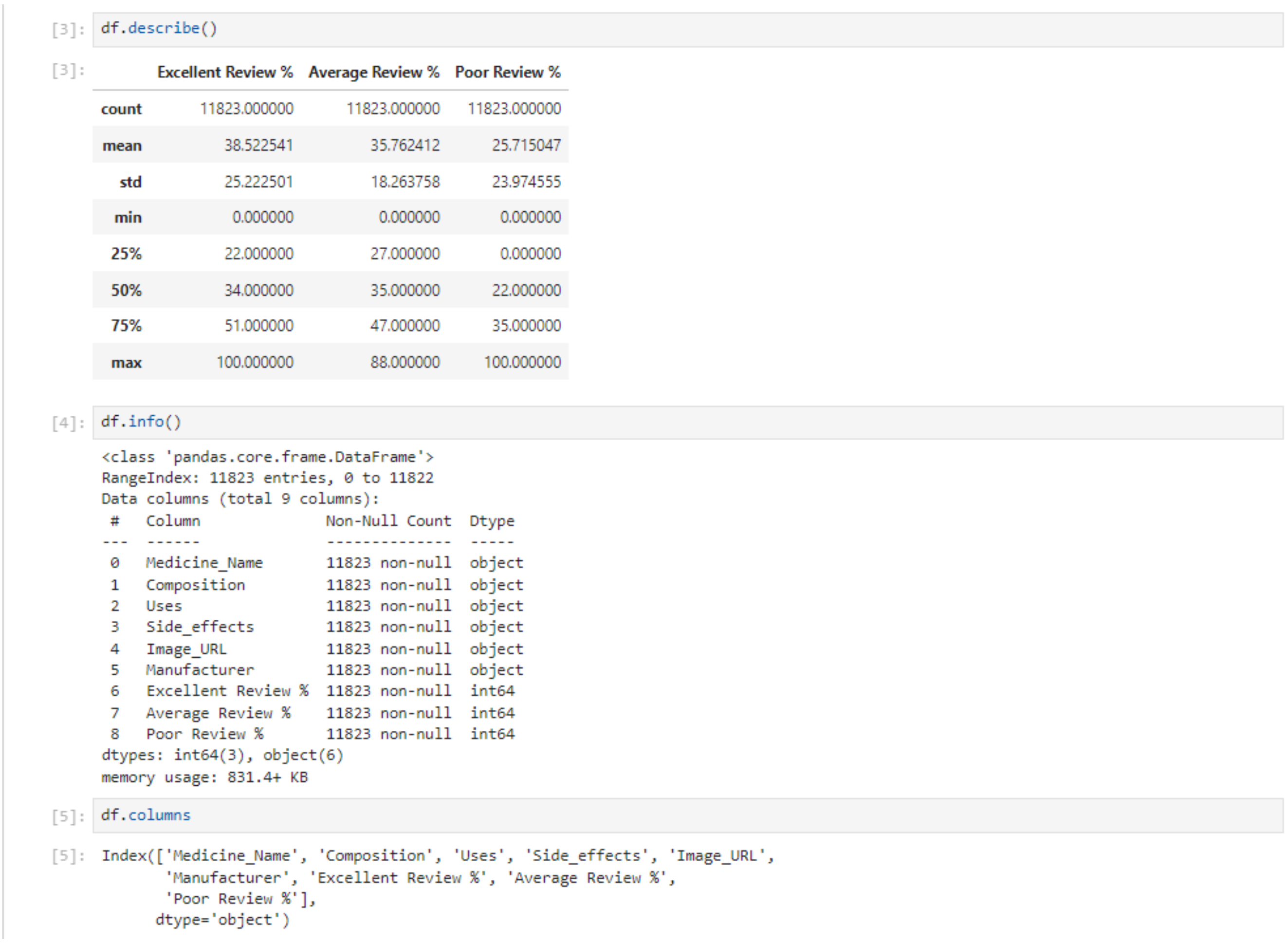
```
[2]: df = pd.read_csv('Dataset/Medicine_Details.csv')
     df.head()
```

[2]:

	Medicine_Name	Composition	Uses	Side_effects	Image_URL	Manufacturer	Excellent Review %	Average Review %	Re
0	Avastin 400mg Injection	Bevacizumab (400mg)	Cancer of colon and rectum Non-small cell lun...	Rectal bleeding Taste change Headache Noseblee...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Roche Products India Pvt Ltd	22	56	
1	Augmentin 625 Duo Tablet	Amoxycillin (500mg) + Clavulanic Acid (125mg)	Treatment of Bacterial infections	Vomiting Nausea Diarrhea Mucocutaneous candidi...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Glaxo SmithKline Pharmaceuticals Ltd	47	35	
2	Azithral 500 Tablet	Azithromycin (500mg)	Treatment of Bacterial infections	Nausea Abdominal pain Diarrhea	https://onemg.gumlet.io/l_watermark_346,w_480,...	Alembic Pharmaceuticals Ltd	39	40	
3	Ascoril LS Syrup	Ambroxol (30mg/5ml) + Levosalbutamol (1mg/5ml)...	Treatment of Cough with mucus	Nausea Vomiting Diarrhea Upset stomach Stomach...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Glenmark Pharmaceuticals Ltd	24	41	
4	Aciloc 150 Tablet	Ranitidine (150mg)	Treatment of Gastroesophageal reflux disease (...	Headache Diarrhea Gastrointestinal disturbance	https://onemg.gumlet.io/l_watermark_346,w_480,...	Cadila Pharmaceuticals Ltd	34	37	

Figure 3: Reading Medicine Dataset

In the figure 4, an overview of DataFrame statistics (df.describe()), structure (df.info()), and column names (df.columns) provides insights into data characteristics essential for Medicine Recommendation System.



```
[3]: df.describe()
```

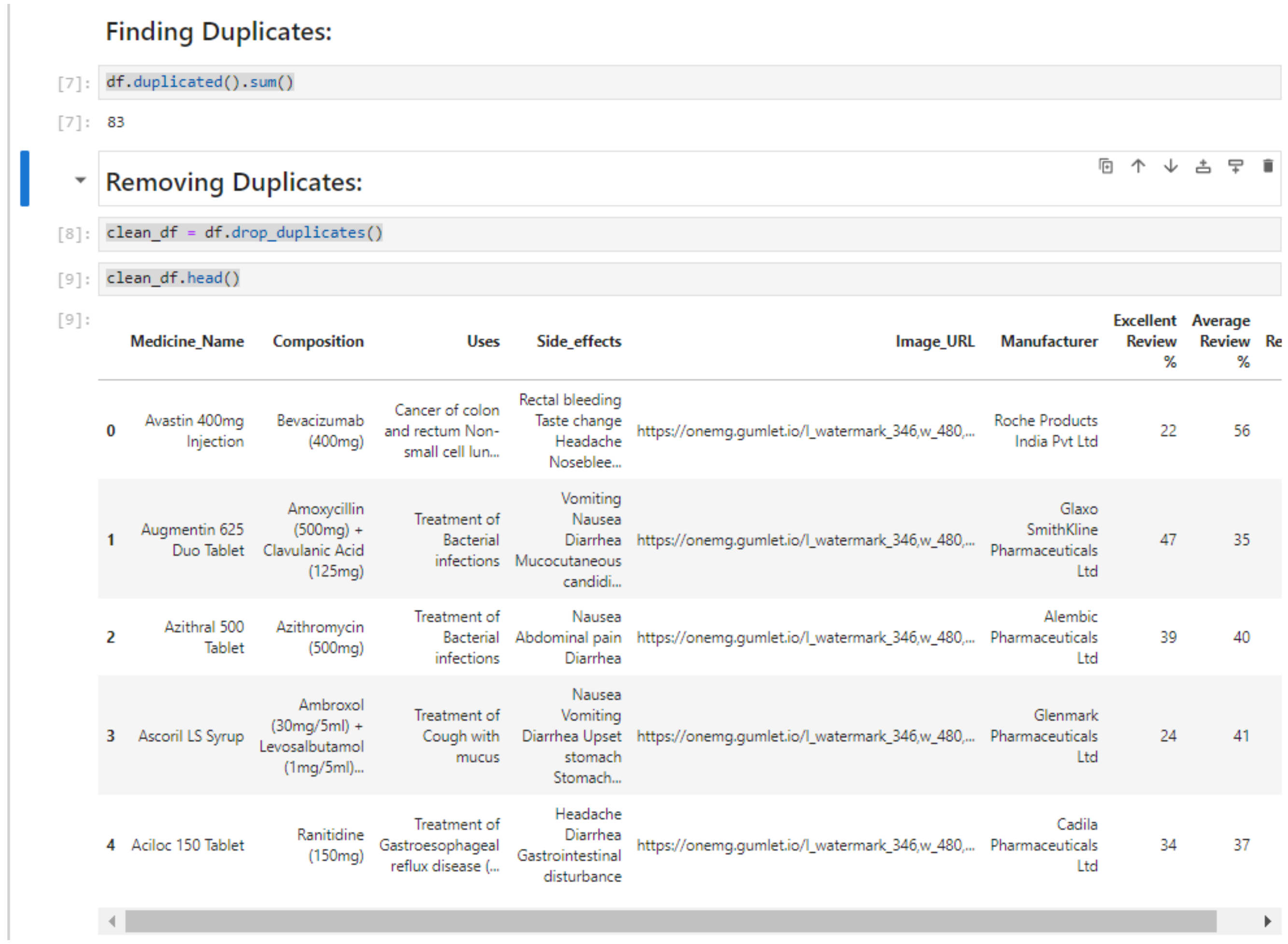
[3]:

	Excellent Review %	Average Review %	Poor Review %
count	11823.000000	11823.000000	11823.000000
mean	38.522541	35.762412	25.715047
std	25.222501	18.263758	23.974555
min	0.000000	0.000000	0.000000
25%	22.000000	27.000000	0.000000
50%	34.000000	35.000000	22.000000
75%	51.000000	47.000000	35.000000
max	100.000000	88.000000	100.000000

```
[4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11823 entries, 0 to 11822
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Medicine_Name       11823 non-null  object
 1   Composition         11823 non-null  object
 2   Uses                11823 non-null  object
 3   Side_effects        11823 non-null  object
 4   Image_URL           11823 non-null  object
 5   Manufacturer        11823 non-null  object
 6   Excellent Review %  11823 non-null  int64 
 7   Average Review %    11823 non-null  int64 
 8   Poor Review %       11823 non-null  int64 
dtypes: int64(3), object(6)
memory usage: 831.4+ KB

[5]: df.columns

[5]: Index(['Medicine_Name', 'Composition', 'Uses', 'Side_effects', 'Image_URL',
        'Manufacturer', 'Excellent Review %', 'Average Review %',
        'Poor Review %'],
       dtype='object')
```

Figure 4: Overview of Dataframe Statistics and Structure

In the figure 5, the code snippet demonstrates the identification and removal of duplicate entries in the dataset (df.duplicated().sum() and clean_df = df.drop_duplicates()). The resulting clean_df.head() displays the initial records of the cleaned dataset, crucial for ensuring data integrity and accuracy in Medicine Recommendation System.



Finding Duplicates:

```
[7]: df.duplicated().sum()
[7]: 83
```

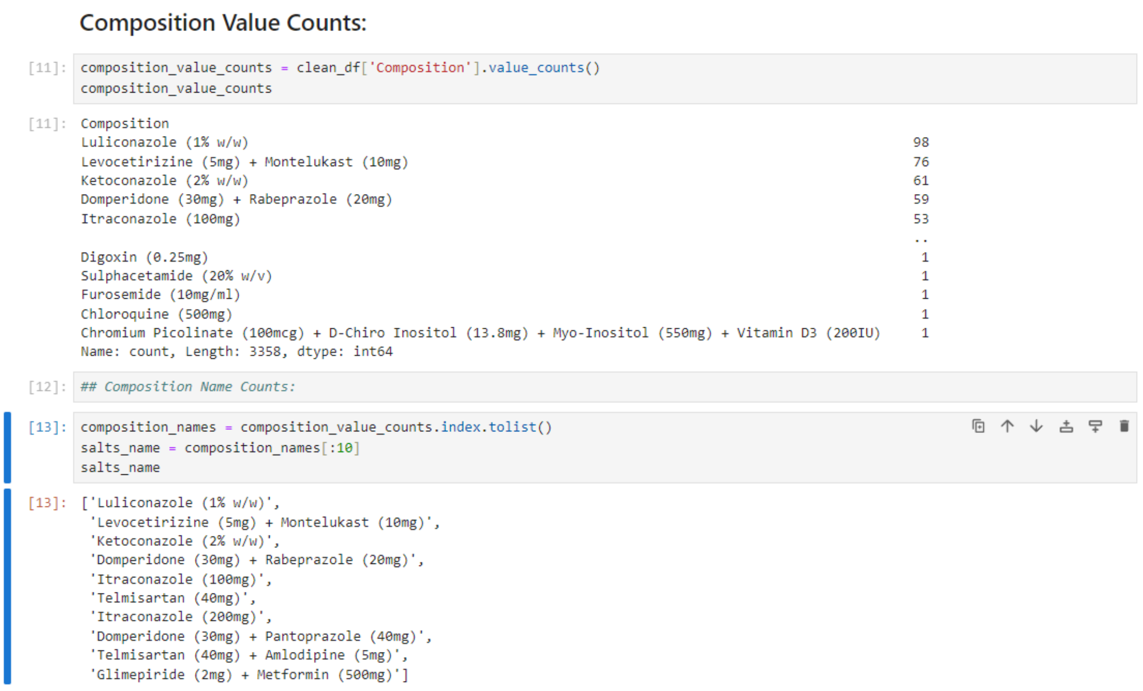
Removing Duplicates:

```
[8]: clean_df = df.drop_duplicates()
[9]: clean_df.head()
[9]:
```

	Medicine_Name	Composition	Uses	Side_effects	Image_URL	Manufacturer	Excellent Review %	Average Review %	Re
0	Avastin 400mg Injection	Bevacizumab (400mg)	Cancer of colon and rectum Non-small cell lun...	Rectal bleeding Taste change Headache Noseblee...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Roche Products India Pvt Ltd	22	56	
1	Augmentin 625 Duo Tablet	Amoxycillin (500mg) + Clavulanic Acid (125mg)	Treatment of Bacterial infections	Vomiting Nausea Diarrhea Mucocutaneous candidi...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Glaxo SmithKline Pharmaceuticals Ltd	47	35	
2	Azithral 500 Tablet	Azithromycin (500mg)	Treatment of Bacterial infections	Nausea Abdominal pain Diarrhea	https://onemg.gumlet.io/l_watermark_346,w_480,...	Alembic Pharmaceuticals Ltd	39	40	
3	Ascoril LS Syrup	Ambroxol (30mg/5ml) + Levosalbutamol (1mg/5ml)...	Treatment of Cough with mucus	Nausea Vomiting Diarrhea Upset stomach Stomach...	https://onemg.gumlet.io/l_watermark_346,w_480,...	Glenmark Pharmaceuticals Ltd	24	41	
4	Aciloc 150 Tablet	Ranitidine (150mg)	Treatment of Gastroesophageal reflux disease (...	Headache Diarrhea Gastrointestinal disturbance	https://onemg.gumlet.io/l_watermark_346,w_480,...	Cadila Pharmaceuticals Ltd	34	37	

Figure 5: Data Cleaning and Initial Data Exploration

In the figure 6, the code snippet calculates the frequency of each composition in the cleaned dataset, extracts the composition names, and identifies the top 10 most frequently occurring compositions.

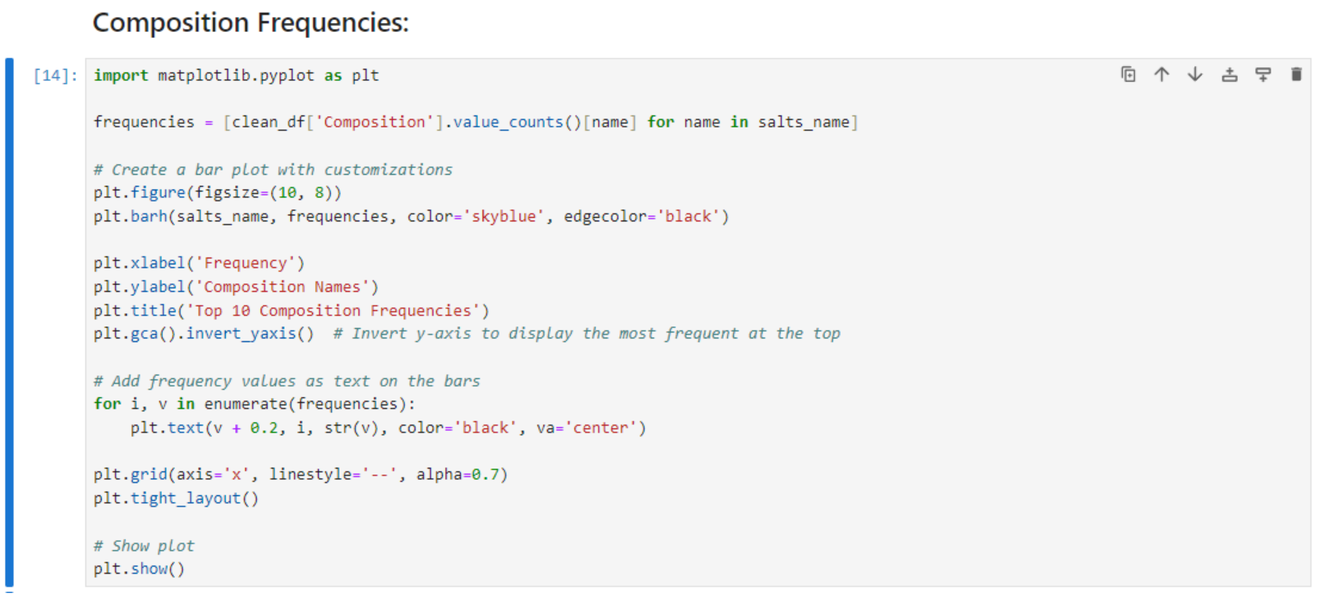


Composition Value Counts:

```
[11]: composition_value_counts = clean_df['Composition'].value_counts()
      composition_value_counts

[11]: Composition
      Luliconazole (1% w/w)                                                                                  98
      Levocetirizine (5mg) + Montelukast (10mg)                                                              76
      Ketoconazole (2% w/w)                                                                                  61
      Domperidone (30mg) + Rabeprazole (20mg)                                                                59
      Itraconazole (100mg)                                                                                   53
                                                                                                             ..
      Digoxin (0.25mg)                                                                                        1
      Sulphacetamide (20% w/v)                                                                                1
      Furosemide (10mg/ml)                                                                                    1
      Chloroquine (500mg)                                                                                     1
      Chromium Picolinate (100mcg) + D-Chiro Inositol (13.8mg) + Myo-Inositol (550mg) + Vitamin D3 (200IU)    1
      Name: count, Length: 3358, dtype: int64

[12]: ## Composition Name Counts:

[13]: composition_names = composition_value_counts.index.tolist()
      salts_name = composition_names[:10]
      salts_name

[13]: ['Luliconazole (1% w/w)',
       'Levocetirizine (5mg) + Montelukast (10mg)',
       'Ketoconazole (2% w/w)',
       'Domperidone (30mg) + Rabeprazole (20mg)',
       'Itraconazole (100mg)',
       'Telmisartan (40mg)',
       'Itraconazole (200mg)',
       'Domperidone (30mg) + Pantoprazole (40mg)',
       'Telmisartan (40mg) + Amlodipine (5mg)',
       'Glimepiride (2mg) + Metformin (500mg)']
```

Figure 6: Medicine Composition Frequency Count

In the figure 7, the code snippet utilizes Matplotlib to create a horizontal bar plot depicting the top 10 most frequent compositions in the cleaned medication dataset. Each bar represents a composition name with its corresponding frequency displayed on the bar.



Composition Frequencies:

```
[14]: import matplotlib.pyplot as plt

      frequencies = [clean_df['Composition'].value_counts()[name] for name in salts_name]

      # Create a bar plot with customizations
      plt.figure(figsize=(10, 8))
      plt.barh(salts_name, frequencies, color='skyblue', edgecolor='black')

      plt.xlabel('Frequency')
      plt.ylabel('Composition Names')
      plt.title('Top 10 Composition Frequencies')
      plt.gca().invert_yaxis()  # Invert y-axis to display the most frequent at the top

      # Add frequency values as text on the bars
      for i, v in enumerate(frequencies):
          plt.text(v + 0.2, i, str(v), color='black', va='center')

      plt.grid(axis='x', linestyle='--', alpha=0.7)
      plt.tight_layout()

      # Show plot
      plt.show()
```

Figure 7: Composition Visualization Code

The bar plot in Figure 8 illustrates the frequencies of the top 10 most common compositions in the cleaned medication dataset. Each bar represents a composition name, with its frequency displayed on the bar, offering a visual representation of the predominant components utilized in medications.

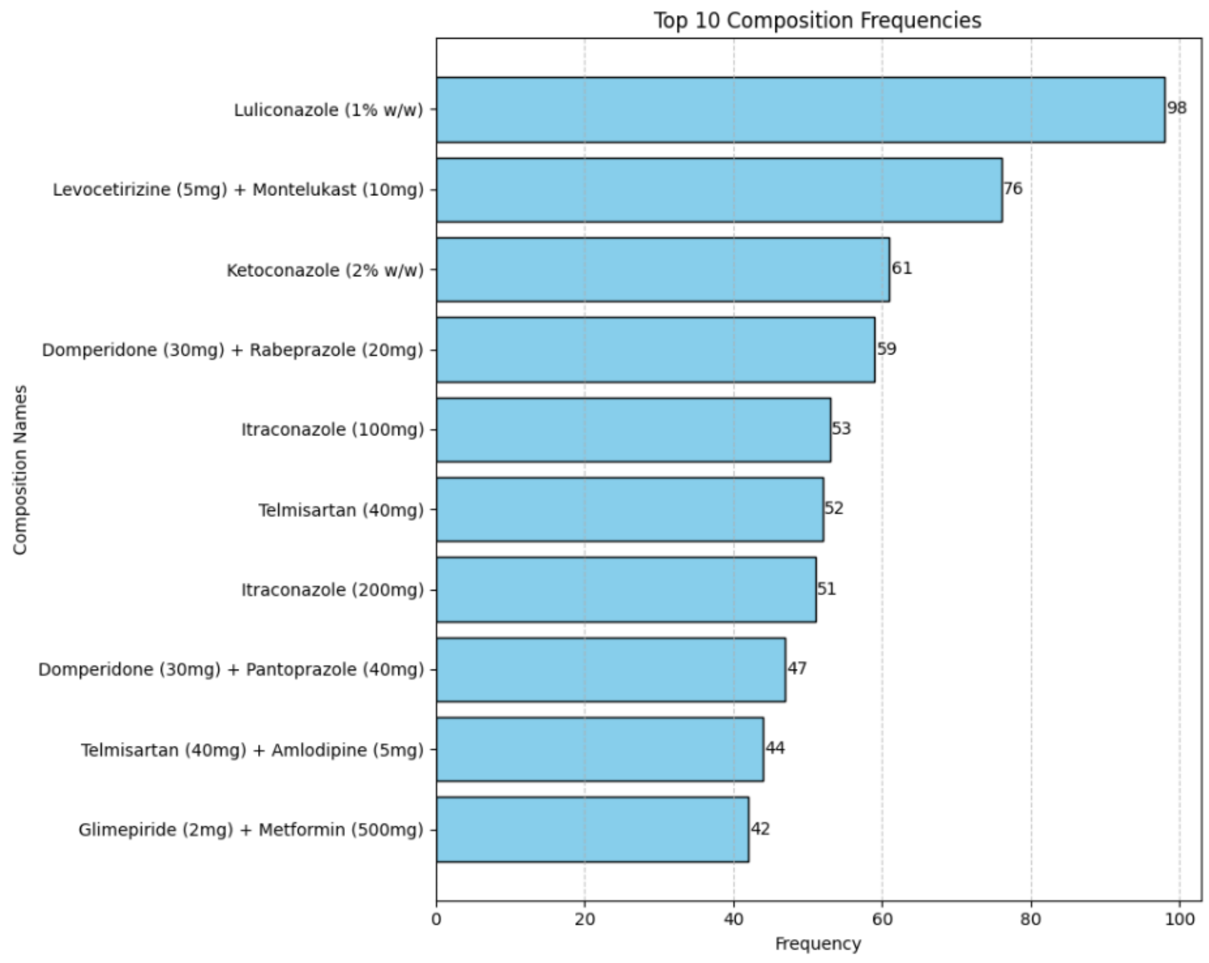
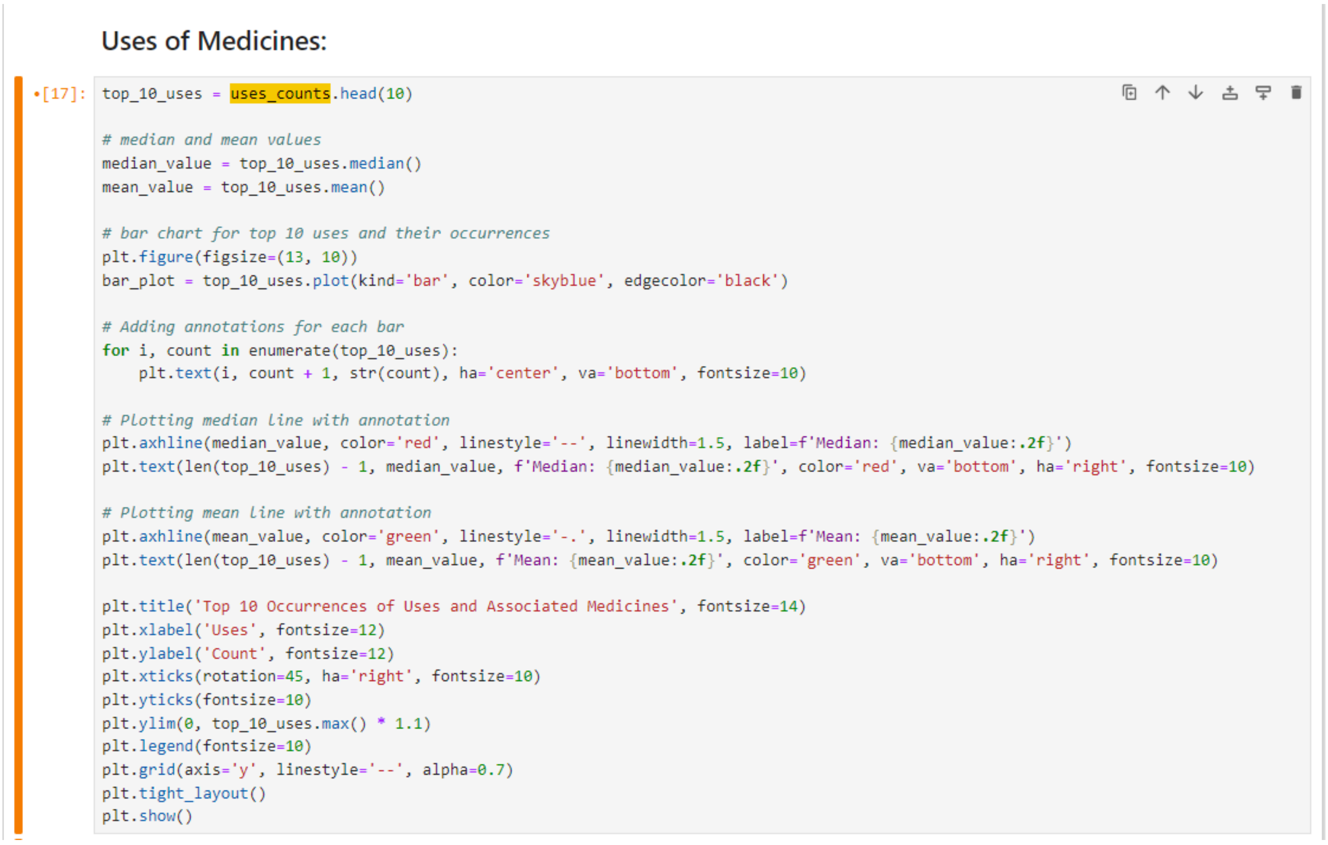


Figure 8: Top 10 Composition Frequencies Bar Plot

The code snippet in figure 9 prepares to visualize the top 10 occurrences of uses and their associated medicines. It calculates the median and mean values for these occurrences and sets up annotations and plot settings for the subsequent bar plot.



Uses of Medicines:

```
[17]: top_10_uses = uses_counts.head(10)

# median and mean values
median_value = top_10_uses.median()
mean_value = top_10_uses.mean()

# bar chart for top 10 uses and their occurrences
plt.figure(figsize=(13, 10))
bar_plot = top_10_uses.plot(kind='bar', color='skyblue', edgecolor='black')

# Adding annotations for each bar
for i, count in enumerate(top_10_uses):
    plt.text(i, count + 1, str(count), ha='center', va='bottom', fontsize=10)

# Plotting median line with annotation
plt.axhline(median_value, color='red', linestyle='--', linewidth=1.5, label=f'Median: {median_value:.2f}')
plt.text(len(top_10_uses) - 1, median_value, f'Median: {median_value:.2f}', color='red', va='bottom', ha='right', fontsize=10)

# Plotting mean line with annotation
plt.axhline(mean_value, color='green', linestyle='-.', linewidth=1.5, label=f'Mean: {mean_value:.2f}')
plt.text(len(top_10_uses) - 1, mean_value, f'Mean: {mean_value:.2f}', color='green', va='bottom', ha='right', fontsize=10)

plt.title('Top 10 Occurrences of Uses and Associated Medicines', fontsize=14)
plt.xlabel('Uses', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.ylim(0, top_10_uses.max() * 1.1)
plt.legend(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Figure 9: Plot Preparation Code for Top 10 Uses of Medicines

Figure 10 presents a bar plot illustrating the top 10 occurrences of uses and their associated medicines. Each bar represents a specific use, with its count annotated on top. The plot includes median and mean lines (red dashed and green dash-dot, respectively) along with their annotations, offering insights into the distribution of use frequencies across medications.

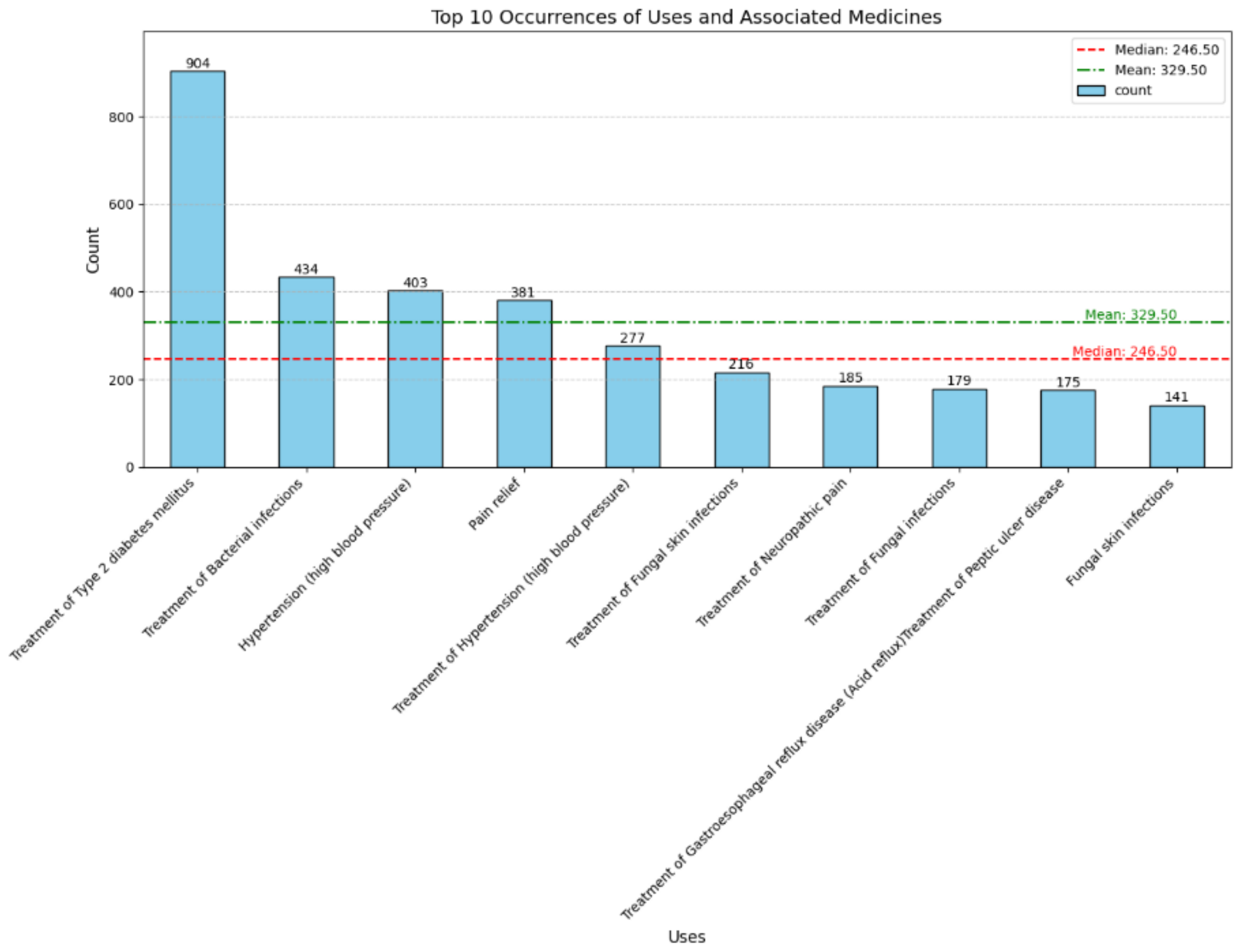
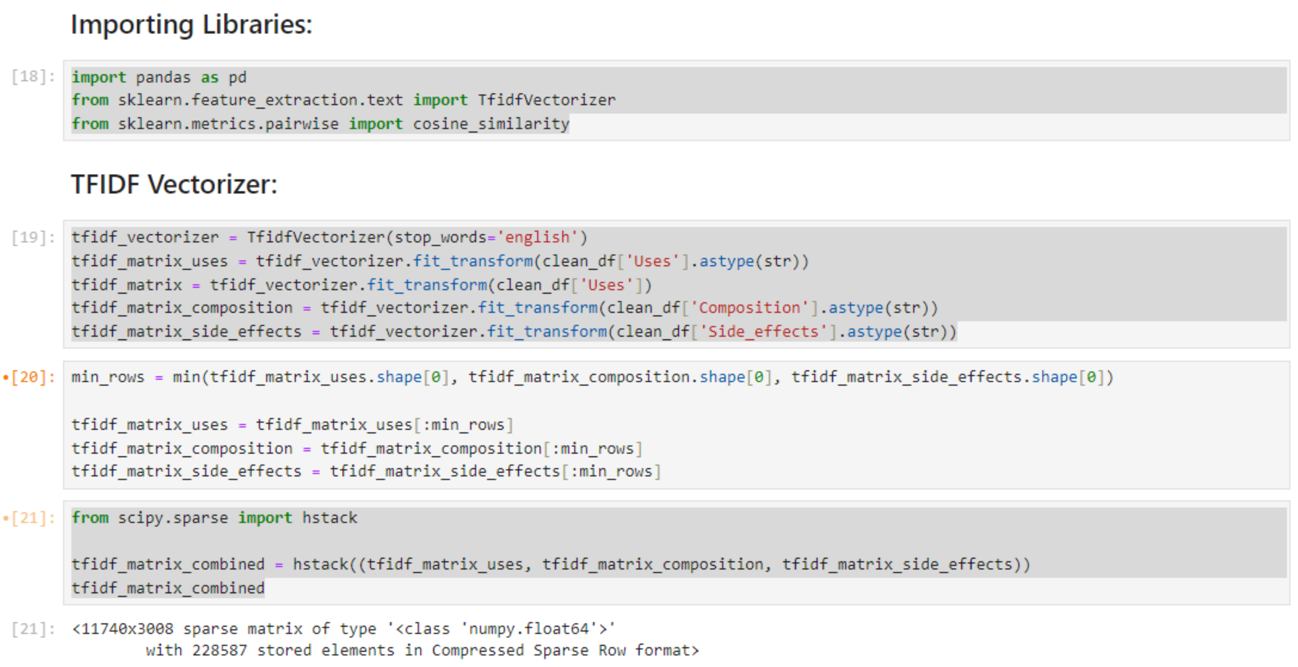


Figure 10: Top 10 Occurrences of Uses and Associated Medicines Bar Plot

Figure 11 showcases the TF-IDF vectorization process using TfidfVectorizer from scikit-learn on the 'Uses', 'Composition', and 'Side_effects' columns in the Medicine dataset. The resulting TF-IDF matrices are combined into tfidf_matrix_combined using hstack from scipy.sparse, providing a unified representation of medication descriptions and components essential for similarity analysis.



Importing Libraries:

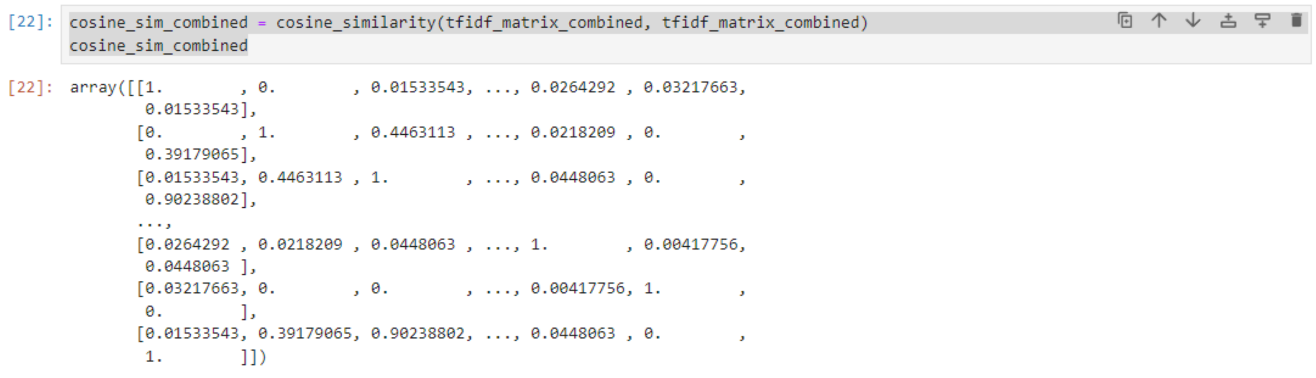
```
[18]: import pandas as pd
      from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.metrics.pairwise import cosine_similarity
```

TFIDF Vectorizer:

```
[19]: tfidf_vectorizer = TfidfVectorizer(stop_words='english')
      tfidf_matrix_uses = tfidf_vectorizer.fit_transform(clean_df['Uses'].astype(str))
      tfidf_matrix = tfidf_vectorizer.fit_transform(clean_df['Uses'])
      tfidf_matrix_composition = tfidf_vectorizer.fit_transform(clean_df['Composition'].astype(str))
      tfidf_matrix_side_effects = tfidf_vectorizer.fit_transform(clean_df['Side_effects'].astype(str))

•[20]: min_rows = min(tfidf_matrix_uses.shape[0], tfidf_matrix_composition.shape[0], tfidf_matrix_side_effects.shape[0])

       tfidf_matrix_uses = tfidf_matrix_uses[:min_rows]
       tfidf_matrix_composition = tfidf_matrix_composition[:min_rows]
       tfidf_matrix_side_effects = tfidf_matrix_side_effects[:min_rows]

•[21]: from scipy.sparse import hstack

       tfidf_matrix_combined = hstack((tfidf_matrix_uses, tfidf_matrix_composition, tfidf_matrix_side_effects))
       tfidf_matrix_combined

[21]: <11740x3008 sparse matrix of type '<class 'numpy.float64'>'
              with 228587 stored elements in Compressed Sparse Row format>
```

Figure 11: TF-IDF Vectorization and Combined Matrix Formation

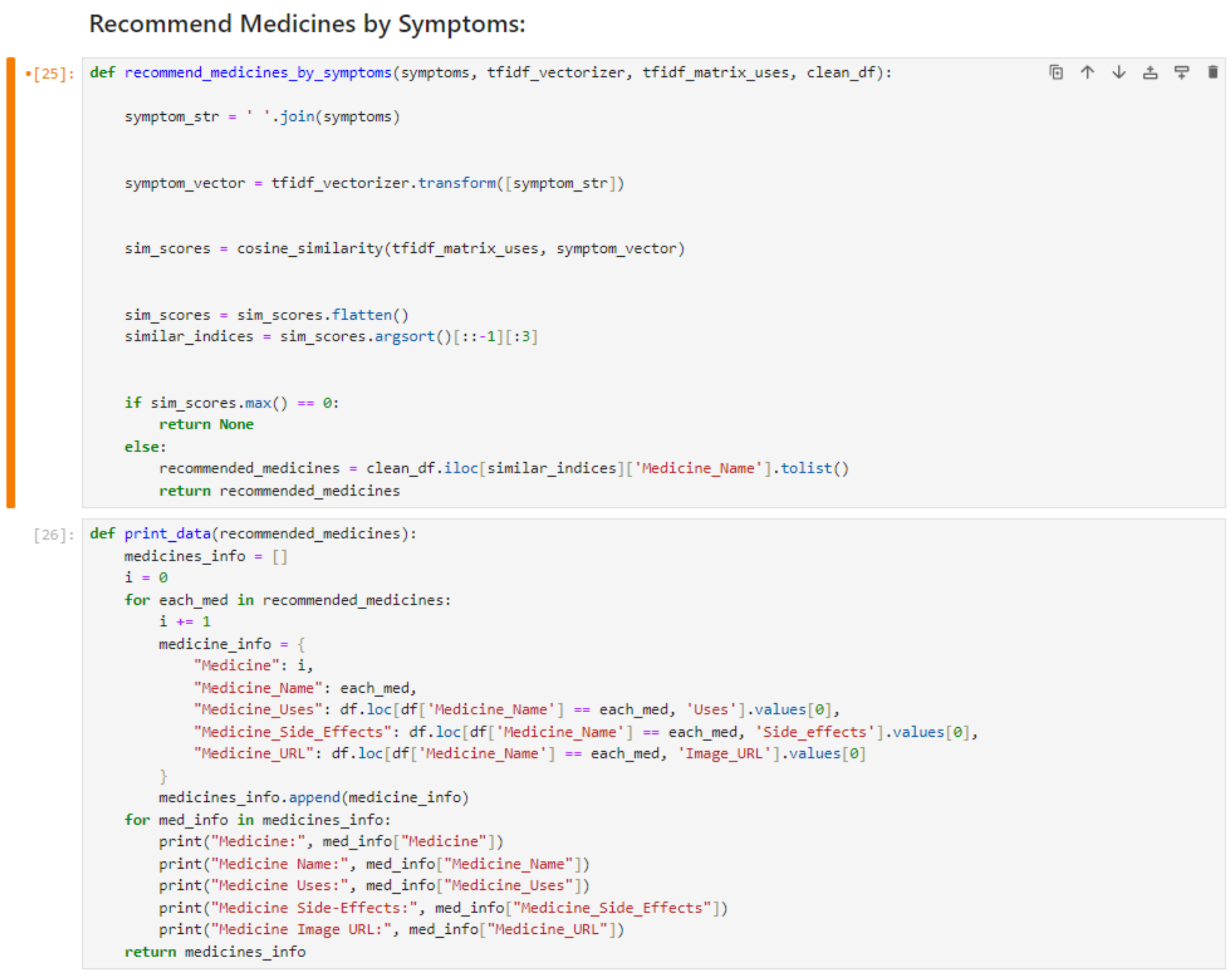
In the figure 12, the cosine similarity matrix is displayed. It is calculated from the combined TF-IDF matrix of medication descriptions and components in the medicine dataset. This matrix quantitatively measures the similarity between medications based on their textual features, providing insights into medication similarity.



```
[22]: cosine_sim_combined = cosine_similarity(tfidf_matrix_combined, tfidf_matrix_combined)
      cosine_sim_combined

[22]: array([[1.        , 0.        , 0.01533543, ..., 0.0264292 , 0.03217663,
              0.01533543],
             [0.        , 1.        , 0.4463113 , ..., 0.0218209 , 0.        ,
              0.39179065],
             [0.01533543, 0.4463113 , 1.        , ..., 0.0448063 , 0.        ,
              0.90238802],
             ...,
             [0.0264292 , 0.0218209 , 0.0448063 , ..., 1.        , 0.00417756,
              0.0448063 ],
             [0.03217663, 0.        , 0.        , ..., 0.00417756, 1.        ,
              0.        ],
             [0.01533543, 0.39179065, 0.90238802, ..., 0.0448063 , 0.        ,
              1.        ]])
```

Figure 12: Cosine Similarity Matrix of Medications

In the figure 13, the code snippet defines a function recommend_medicines_by_symptoms that recommends medicines based on input symptoms. It utilizes TF-IDF vectorization and cosine similarity calculations on the 'Uses' column of the cleaned dataset. The function returns a list of recommended medicines based on similarity scores. Additionally, the print_data function formats and prints information about recommended medicines including their uses, side effects, and image URLs, providing detailed insights for medication recommendation.



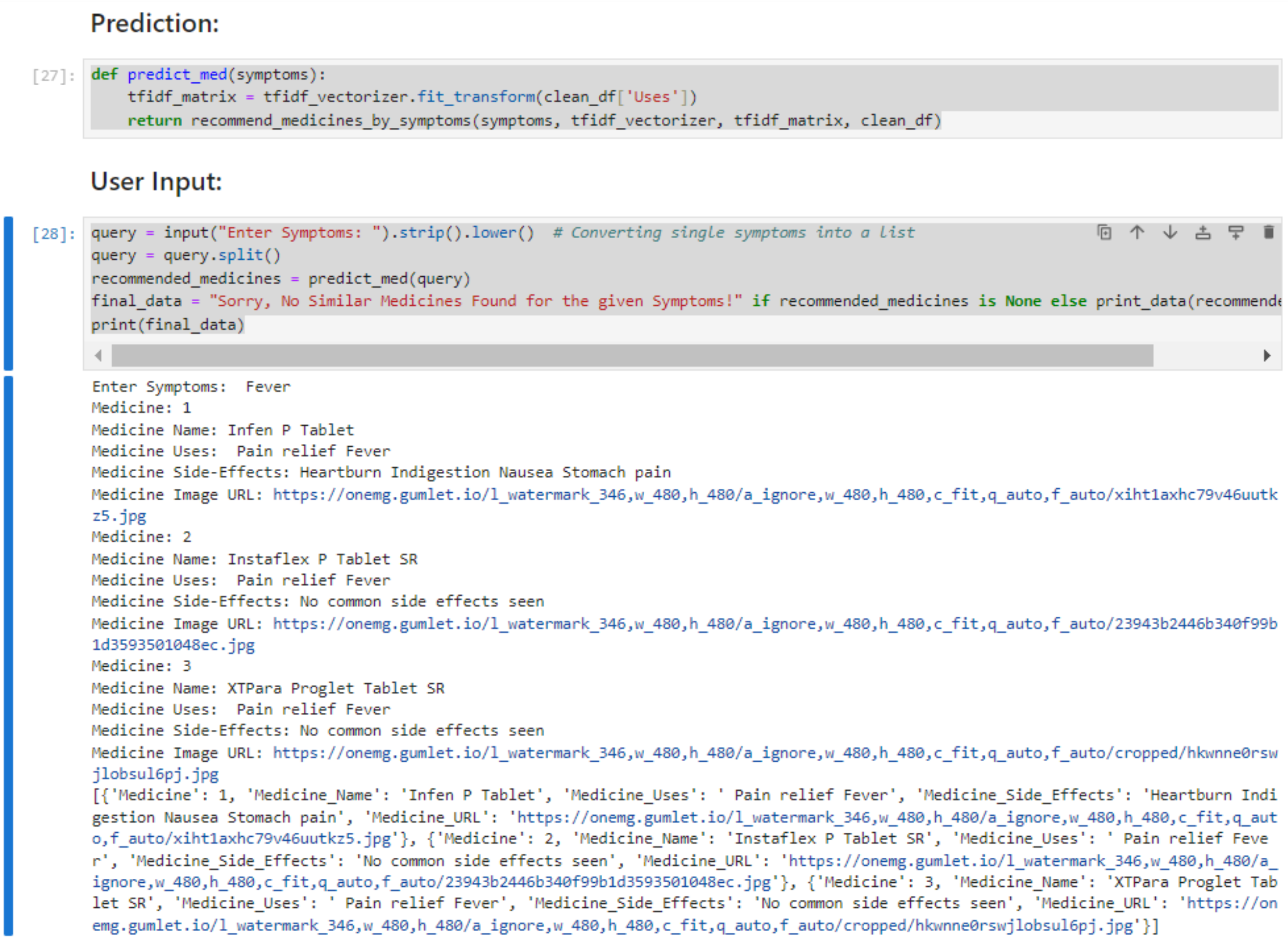
Recommend Medicines by Symptoms:

```
•[25]: def recommend_medicines_by_symptoms(symptoms, tfidf_vectorizer, tfidf_matrix_uses, clean_df):

           symptom_str = ' '.join(symptoms)


           symptom_vector = tfidf_vectorizer.transform([symptom_str])


           sim_scores = cosine_similarity(tfidf_matrix_uses, symptom_vector)


           sim_scores = sim_scores.flatten()
           similar_indices = sim_scores.argsort()[::-1][:3]


           if sim_scores.max() == 0:
               return None
           else:
               recommended_medicines = clean_df.iloc[similar_indices]['Medicine_Name'].tolist()
               return recommended_medicines
```

```
[26]: def print_data(recommended_medicines):
          medicines_info = []
          i = 0
          for each_med in recommended_medicines:
              i += 1
              medicine_info = {
                  "Medicine": i,
                  "Medicine_Name": each_med,
                  "Medicine_Uses": df.loc[df['Medicine_Name'] == each_med, 'Uses'].values[0],
                  "Medicine_Side_Effects": df.loc[df['Medicine_Name'] == each_med, 'Side_effects'].values[0],
                  "Medicine_URL": df.loc[df['Medicine_Name'] == each_med, 'Image_URL'].values[0]
              }
              medicines_info.append(medicine_info)
          for med_info in medicines_info:
              print("Medicine:", med_info["Medicine"])
              print("Medicine Name:", med_info["Medicine_Name"])
              print("Medicine Uses:", med_info["Medicine_Uses"])
              print("Medicine Side-Effects:", med_info["Medicine_Side_Effects"])
              print("Medicine Image URL:", med_info["Medicine_URL"])
          return medicines_info
```

Figure 13: Medicines Recommendation by Symptoms Function

In the figure 14, the code snippet defines a function predict_med that incorporates TF-IDF vectorization on the 'Uses' column of the cleaned dataset. This function utilizes the recommend_medicines_by_symptoms function to predict and recommend medicines based on user-input symptoms. The subsequent code captures user input for symptoms, calls predict_med to generate recommended medicines, and prints detailed information about the recommended medicines if any matches are found.



Prediction:

```
[27]: def predict_med(symptoms):
          tfidf_matrix = tfidf_vectorizer.fit_transform(clean_df['Uses'])
          return recommend_medicines_by_symptoms(symptoms, tfidf_vectorizer, tfidf_matrix, clean_df)
```

User Input:

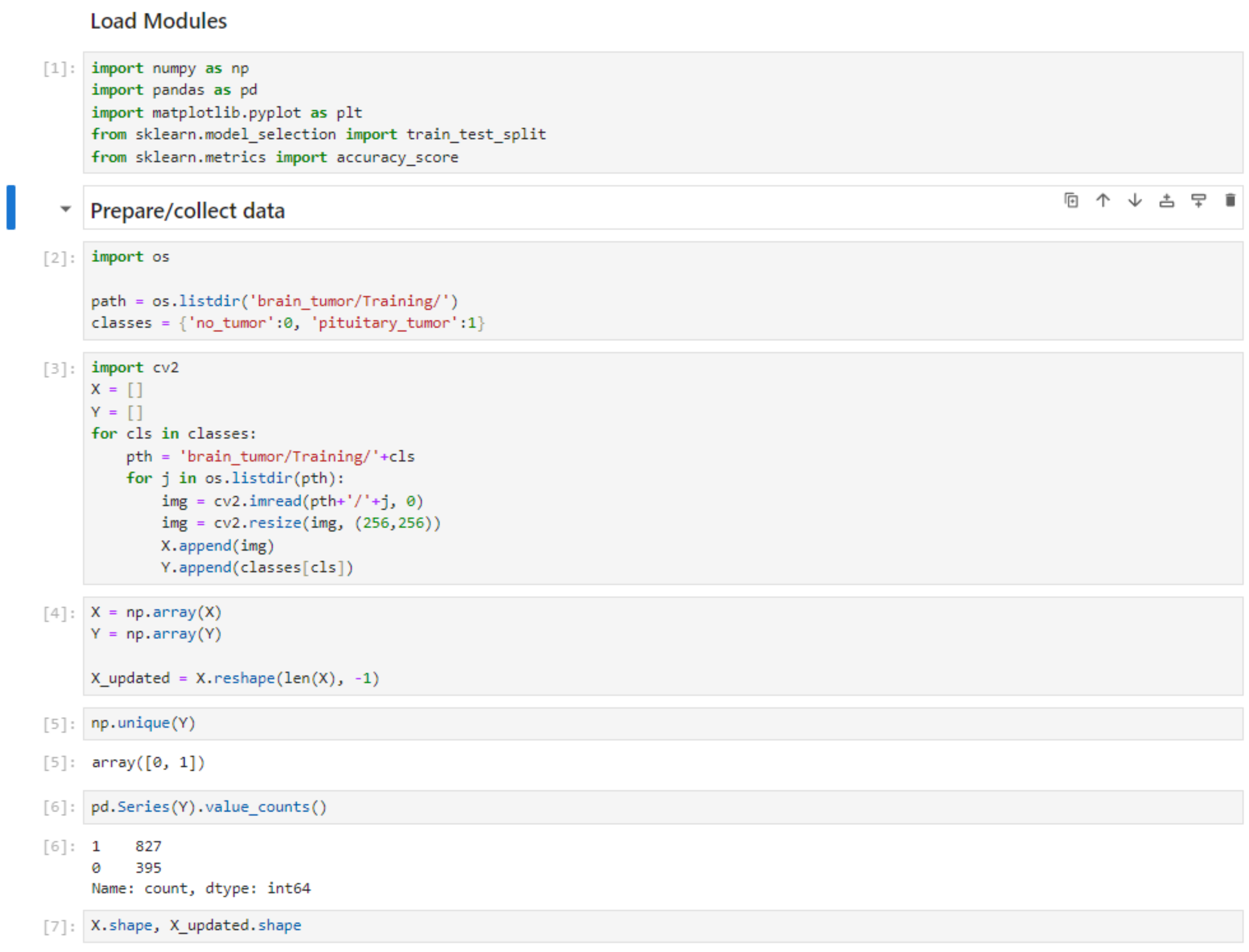
```
[28]: query = input("Enter Symptoms: ").strip().lower()  # Converting single symptoms into a list
      query = query.split()
      recommended_medicines = predict_med(query)
      final_data = "Sorry, No Similar Medicines Found for the given Symptoms!" if recommended_medicines is None else print_data(recommende
      print(final_data)
```

```
Enter Symptoms:  Fever
Medicine: 1
Medicine Name: Infen P Tablet
Medicine Uses:  Pain relief Fever
Medicine Side-Effects: Heartburn Indigestion Nausea Stomach pain
Medicine Image URL: https://onemg.gumlet.io/l_watermark_346,w_480,h_480/a_ignore,w_480,h_480,c_fit,q_auto,f_auto/xiht1axhc79v46uutk
z5.jpg
Medicine: 2
Medicine Name: Instaflex P Tablet SR
Medicine Uses:  Pain relief Fever
Medicine Side-Effects: No common side effects seen
Medicine Image URL: https://onemg.gumlet.io/l_watermark_346,w_480,h_480/a_ignore,w_480,h_480,c_fit,q_auto,f_auto/23943b2446b340f99b
1d3593501048ec.jpg
Medicine: 3
Medicine Name: XTPara Proglet Tablet SR
Medicine Uses:  Pain relief Fever
Medicine Side-Effects: No common side effects seen
Medicine Image URL: https://onemg.gumlet.io/l_watermark_346,w_480,h_480/a_ignore,w_480,h_480,c_fit,q_auto,f_auto/cropped/hkwnne0rsw
jlobsul6pj.jpg
[{'Medicine': 1, 'Medicine_Name': 'Infen P Tablet', 'Medicine_Uses': ' Pain relief Fever', 'Medicine_Side_Effects': 'Heartburn Indi
gestion Nausea Stomach pain', 'Medicine_URL': 'https://onemg.gumlet.io/l_watermark_346,w_480,h_480/a_ignore,w_480,h_480,c_fit,q_aut
o,f_auto/xiht1axhc79v46uutkz5.jpg'}, {'Medicine': 2, 'Medicine_Name': 'Instaflex P Tablet SR', 'Medicine_Uses': ' Pain relief Feve
r', 'Medicine_Side_Effects': 'No common side effects seen', 'Medicine_URL': 'https://onemg.gumlet.io/l_watermark_346,w_480,h_480/a_
ignore,w_480,h_480,c_fit,q_auto,f_auto/23943b2446b340f99b1d3593501048ec.jpg'}, {'Medicine': 3, 'Medicine_Name': 'XTPara Proglet Tab
let SR', 'Medicine_Uses': ' Pain relief Fever', 'Medicine_Side_Effects': 'No common side effects seen', 'Medicine_URL': 'https://on
emg.gumlet.io/l_watermark_346,w_480,h_480/a_ignore,w_480,h_480,c_fit,q_auto,f_auto/cropped/hkwnne0rswjlobsul6pj.jpg'}]
```

Figure 14: Medicines Recommendation Based on User Input Symptoms

2.6.2 Brain Tumor Detection Code:

In the figure 15, the code snippet initializes data preparation for brain tumor classification. It imports necessary libraries such as NumPy, pandas, Matplotlib, and OpenCV (cv2). The os library is utilized to list directories containing training data categorized into classes ('no_tumor' and 'pituitary_tumor'). Images are loaded using OpenCV, resized to 256x256 pixels, and stored in X, while corresponding labels ('no_tumor' as 0 and 'pituitary_tumor' as 1) are stored in Y. The shape and dimensions of the loaded images and their flattened versions (X_updated) are examined to ensure data integrity and consistency for subsequent machine learning tasks. This setup forms the foundational steps for training a brain tumor classification model.



Load Modules

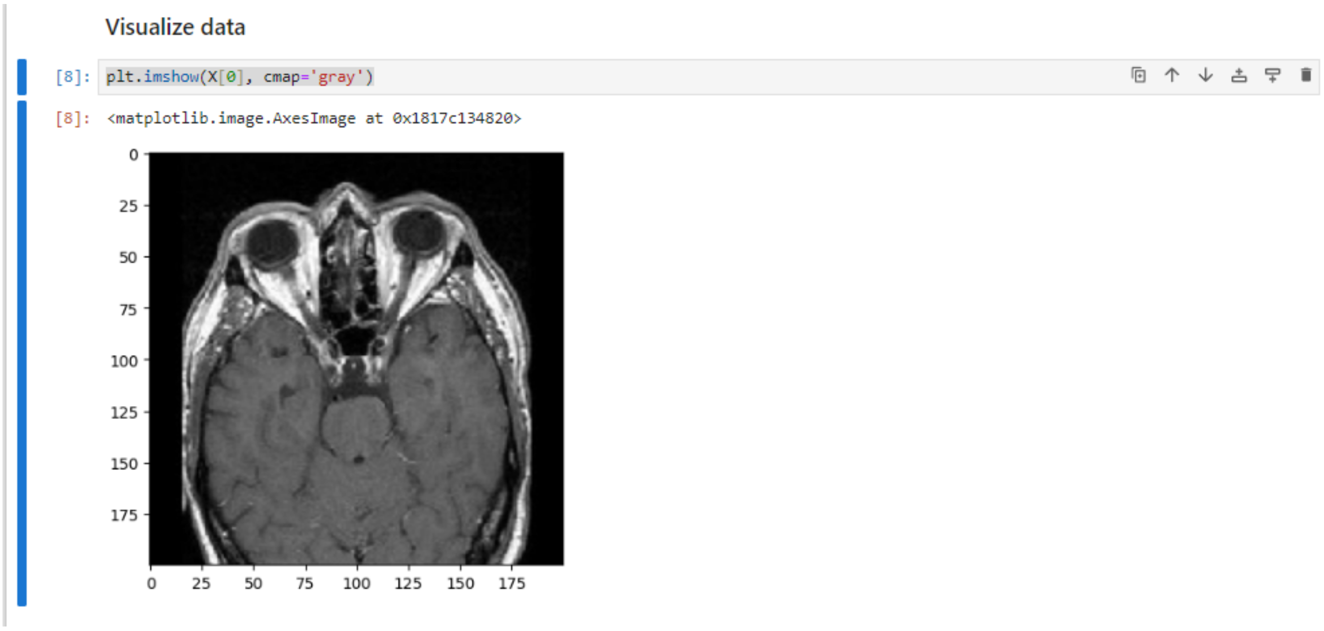
```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score
```

Prepare/collect data

```
[2]: import os

     path = os.listdir('brain_tumor/Training/')
     classes = {'no_tumor':0, 'pituitary_tumor':1}

[3]: import cv2
     X = []
     Y = []
     for cls in classes:
         pth = 'brain_tumor/Training/'+cls
         for j in os.listdir(pth):
             img = cv2.imread(pth+'/'+j, 0)
             img = cv2.resize(img, (256,256))
             X.append(img)
             Y.append(classes[cls])

[4]: X = np.array(X)
     Y = np.array(Y)

     X_updated = X.reshape(len(X), -1)

[5]: np.unique(Y)

[5]: array([0, 1])

[6]: pd.Series(Y).value_counts()

[6]: 1    827
     0    395
     Name: count, dtype: int64

[7]: X.shape, X_updated.shape
```

Figure 15: Data Preparation for Brain Tumor Classification

In the figure 16, plt.imshow displays the first image (X[0]) from the loaded dataset of brain tumor images. The image is visualized using a grayscale colormap (cmap='gray'), showcasing the initial data input for brain tumor classification within the 'MediBot' application. This step aids in visualizing and verifying the quality and content of the image data used for subsequent processing and analysis.

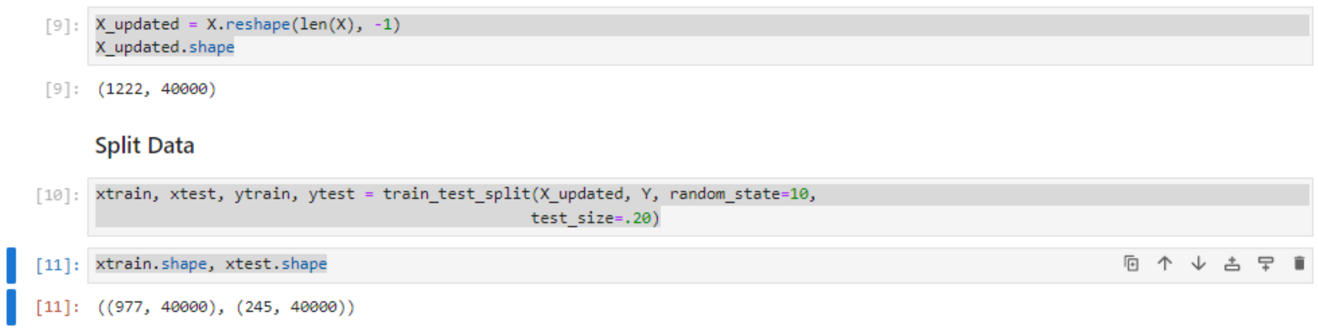


Visualize data

```
[8]: plt.imshow(X[0], cmap='gray')
[8]: <matplotlib.image.AxesImage at 0x1817c134820>
```

Figure 16: Visualizing Brain Tumor Data

In the figure 17, X_updated is reshaped to flatten each image in X while preserving the number of samples. This transformation results in X_updated having a shape of (number_of_samples, flattened_image_size). Subsequently, the code snippet performs a train-test split using train_test_split from scikit-learn. This step divides the data into training (xtrain, ytrain) and testing (xtest, ytest) sets with a test size of 20%, ensuring randomized partitioning based on a random seed. The shapes of xtrain and xtest are then displayed to verify the dimensions of the training and testing datasets respectively, facilitating effective model training and evaluation for brain tumor classification.



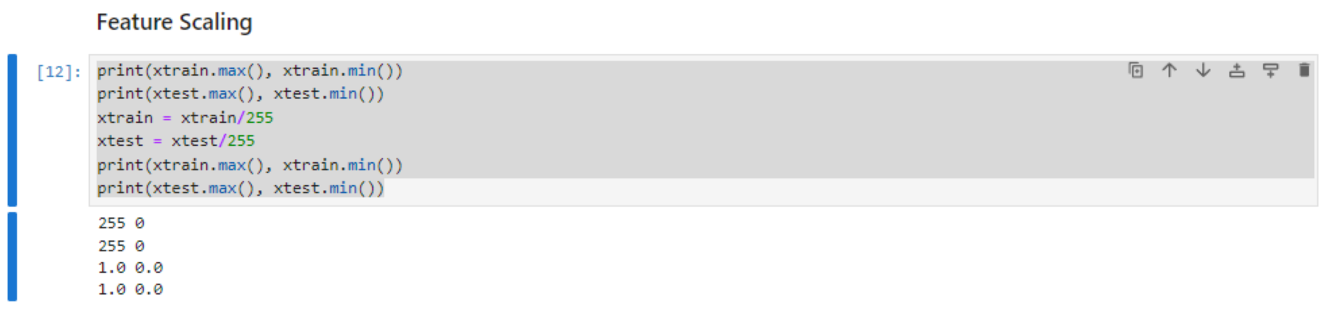
```
[9]: X_updated = X.reshape(len(X), -1)
     X_updated.shape

[9]: (1222, 40000)
```

Split Data

```
[10]: xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, random_state=10,
                                                      test_size=.20)

[11]: xtrain.shape, xtest.shape

[11]: ((977, 40000), (245, 40000))
```

Figure 17: Reshaped Image Data and Train-Test Split

In the figure 18, the code snippet prints the maximum and minimum pixel values of the training (xtrain) and testing (xtest) datasets before normalization. It then normalizes the pixel values by dividing them by 255, which scales the pixel intensities from the original range [0, 255] to [0, 1]. After normalization, the code snippet prints the new maximum and minimum pixel values of the normalized training and testing datasets. This preprocessing step ensures consistent data scaling across the datasets, enhancing the effectiveness of training and evaluating the brain tumor classification model within the 'MediBot' application.

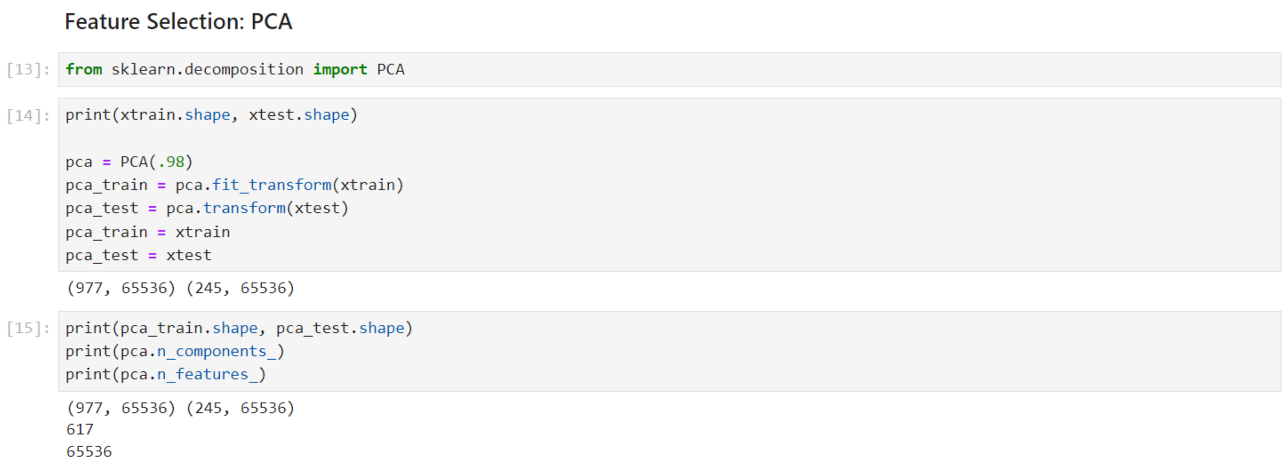


Feature Scaling

```
[12]: print(xtrain.max(), xtrain.min())
      print(xtest.max(), xtest.min())
      xtrain = xtrain/255
      xtest = xtest/255
      print(xtrain.max(), xtrain.min())
      print(xtest.max(), xtest.min())

      255 0
      255 0
      1.0 0.0
      1.0 0.0
```

Figure 18: Data Normalization for Train and Test Sets

In the figure 19, the code snippet demonstrates the application of Principal Component Analysis (PCA) using scikit-learn (from sklearn.decomposition import PCA). Initially, it prints the shapes of the training (xtrain) and testing (xtest) datasets to confirm their dimensions. PCA is then instantiated with PCA(.98), indicating that PCA should retain 98% of the variance in the original data. The fit_transform method is applied to xtrain to compute the principal components and transform the training data (pca_train). Similarly, the transform method is used on xtest to transform the testing data (pca_test) based on the PCA model fitted to the training data.


Feature Selection: PCA

```
[13]: from sklearn.decomposition import PCA
```

```
[14]: print(xtrain.shape, xtest.shape)

      pca = PCA(.98)
      pca_train = pca.fit_transform(xtrain)
      pca_test = pca.transform(xtest)
      pca_train = xtrain
      pca_test = xtest

      (977, 65536) (245, 65536)
```

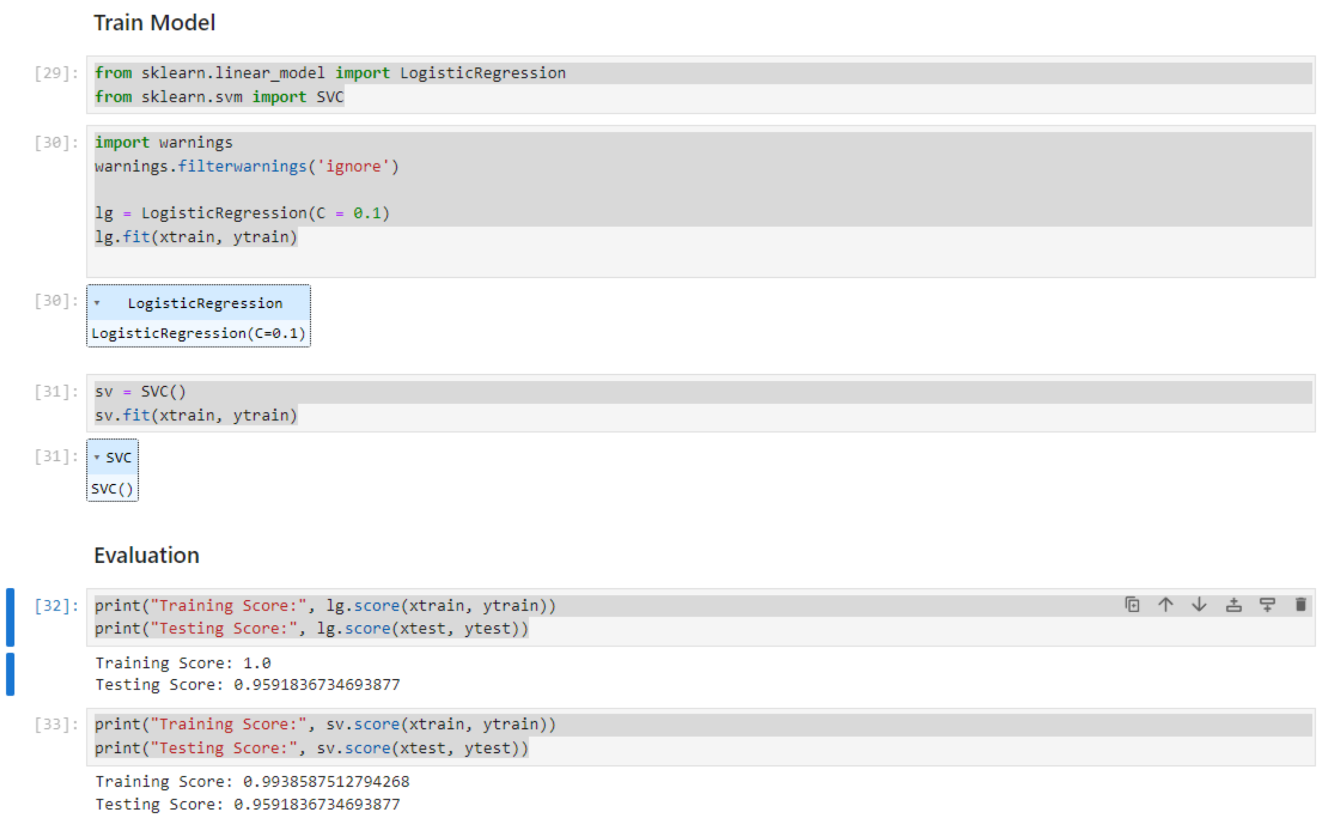
```
[15]: print(pca_train.shape, pca_test.shape)
      print(pca.n_components_)
      print(pca.n_features_)

      (977, 65536) (245, 65536)
      617
      65536
```

Figure 19: Feature Selection (PCA)

In the figure 20, the code snippet demonstrates the training and evaluation of classification models using Logistic Regression (LogisticRegression) and Support Vector Classifier (SVC) from scikit-learn (from sklearn.linear_model import LogisticRegression and from sklearn.svm import SVC). Two models are instantiated and trained:

- **Logistic Regression Model:** Initialized with LogisticRegression(C=0.1), it is trained on the normalized and possibly PCA-transformed training data (xtrain, ytrain).
- **Support Vector Classifier (SVC):** Initialized with default parameters (SVC()), it is also trained on the same training data.

It then prints the training and testing scores for both models. For the Logistic Regression model, it prints the training and testing scores using lg.score(xtrain, ytrain) and lg.score(xtest, ytest), respectively. For the Support Vector Classifier (SVC), it prints the training and testing scores using sv.score(xtrain, ytrain) and sv.score(xtest, ytest). These scores quantify the accuracy of each model in predicting brain tumor classes ('no_tumor' and 'pituitary_tumor') based on the normalized and possibly PCA-transformed features.



Train Model

```
[29]: from sklearn.linear_model import LogisticRegression
      from sklearn.svm import SVC
```

```
[30]: import warnings
      warnings.filterwarnings('ignore')

      lg = LogisticRegression(C = 0.1)
      lg.fit(xtrain, ytrain)
```

```
[30]: ▾ LogisticRegression
      LogisticRegression(C=0.1)
```

```
[31]: sv = SVC()
      sv.fit(xtrain, ytrain)
```

```
[31]: ▾ SVC
      SVC()
```

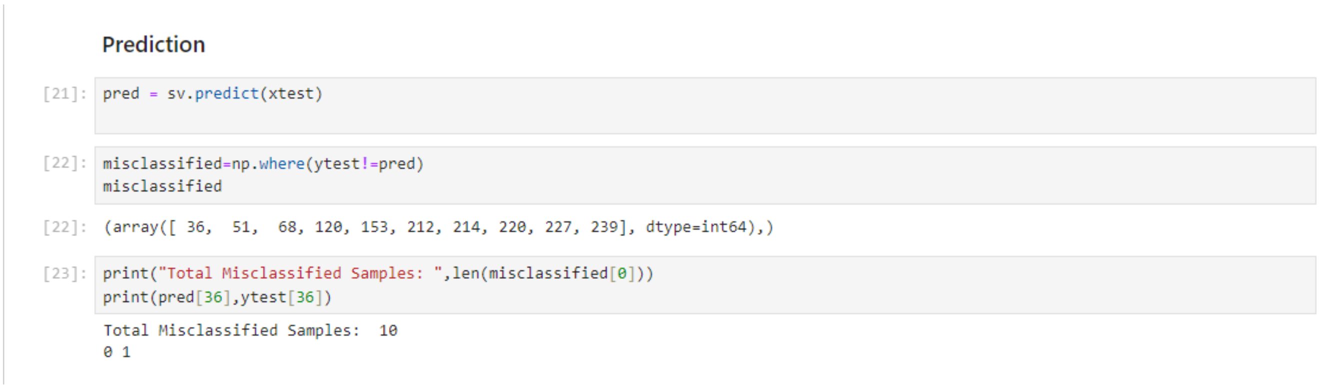
Evaluation

```
[32]: print("Training Score:", lg.score(xtrain, ytrain))
      print("Testing Score:", lg.score(xtest, ytest))

      Training Score: 1.0
      Testing Score: 0.9591836734693877
```

```
[33]: print("Training Score:", sv.score(xtrain, ytrain))
      print("Testing Score:", sv.score(xtest, ytest))

      Training Score: 0.9938587512794268
      Testing Score: 0.9591836734693877
```

Figure 20: Training and Testing Classification Models

In the figure 21, the code snippet evaluates the performance of the Support Vector Machine (SVM) model by identifying and analyzing misclassified samples. After predicting the test set labels (xtest) using the trained SVM model (pred = sv.predict(xtest)), it identifies misclassified samples using np.where(ytest != pred), where the actual labels (ytest) differ from the predicted labels (pred). The snippet prints the total number of misclassified samples (len(misclassified[0])), providing insight into the model's prediction errors. Additionally, it prints the predicted and actual labels for a specific sample (pred[36] and ytest[36]), offering a detailed look at individual misclassification cases. This analysis helps in understanding the SVM model's performance and guiding further improvements in the brain tumor classification module.

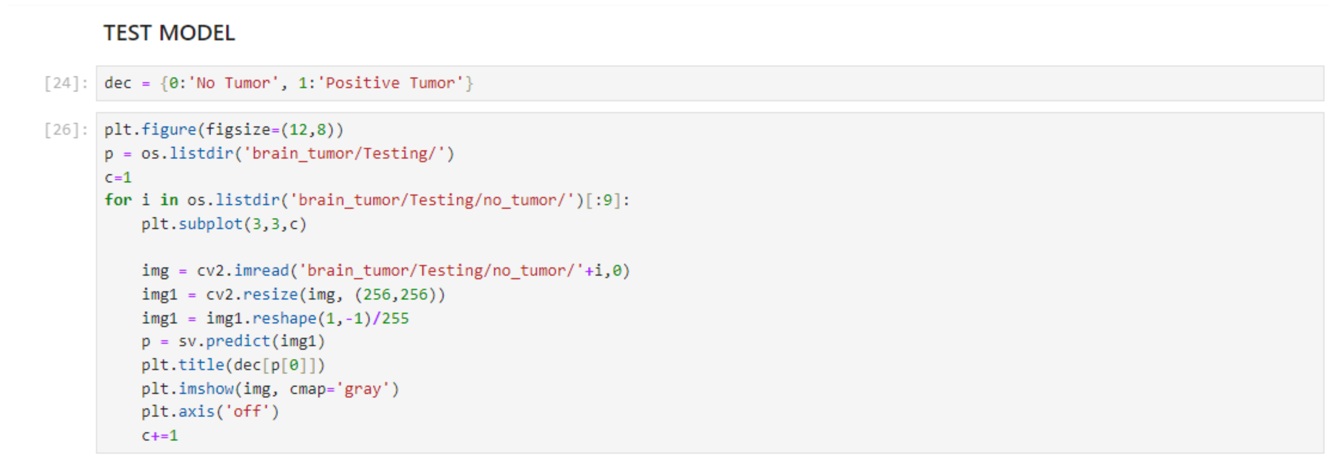


Prediction

```
[21]: pred = sv.predict(xtest)

[22]: misclassified=np.where(ytest!=pred)
      misclassified

[22]: (array([ 36,  51,  68, 120, 153, 212, 214, 220, 227, 239], dtype=int64),)

[23]: print("Total Misclassified Samples: ",len(misclassified[0]))
      print(pred[36],ytest[36])

      Total Misclassified Samples:  10
      0 1
```

Figure 21: Misclassification Analysis of SVM Model

In the figure 22, the code snippet visualizes the SVM model's predictions on test images labeled as 'No Tumor'. Each subplot displays a test image with the predicted label ('No Tumor' or 'Positive Tumor') as the title. This visualization helps assess the model's performance on unseen test data within the 'MediBot' application's brain tumor detection module.



TEST MODEL

```
[24]: dec = {0:'No Tumor', 1:'Positive Tumor'}
```

```
[26]: plt.figure(figsize=(12,8))
      p = os.listdir('brain_tumor/Testing/')
      c=1
      for i in os.listdir('brain_tumor/Testing/no_tumor/')[:9]:
          plt.subplot(3,3,c)

          img = cv2.imread('brain_tumor/Testing/no_tumor/'+i,0)
          img1 = cv2.resize(img, (256,256))
          img1 = img1.reshape(1,-1)/255
          p = sv.predict(img1)
          plt.title(dec[p[0]])
          plt.imshow(img, cmap='gray')
          plt.axis('off')
          c+=1
```

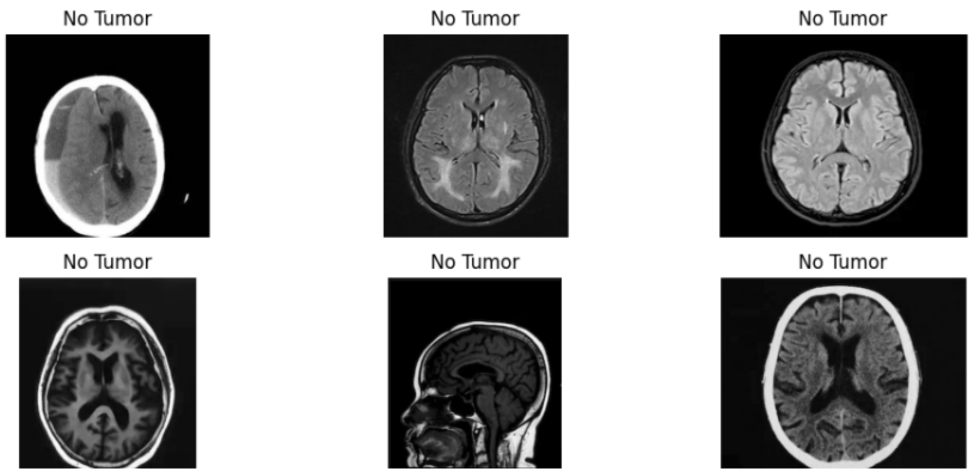
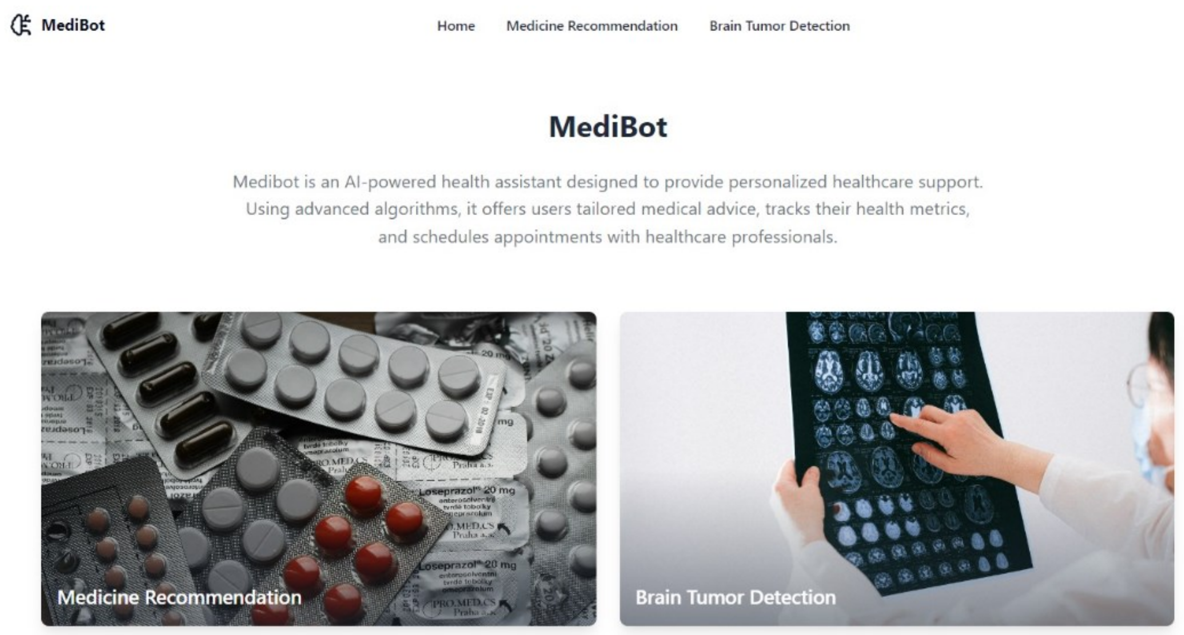


Figure 22: Visualization of Model Predictions on Test Images

2.7 APPLICATION INTERFACE: The MediBot application interface, developed using ReactJS and Tailwind CSS, offers a user-friendly experience for accessing healthcare functionalities such as personalized medicine recommendations and brain tumor detection. It ensures easy navigation and enhances accessibility to essential medical services.

In the figure 23, the MediBot home page interface is showcased, highlighting its two primary sections: the Medicine Recommendation System and the Brain Tumor Detection module. This layout ensures users can effortlessly navigate to either feature, facilitating personalized medication suggestions and MRI image uploads for accurate tumor diagnosis.



Medibot is an AI-powered health assistant designed to provide personalized healthcare support. Using advanced algorithms, it offers users tailored medical advice, tracks their health metrics, and schedules appointments with healthcare professionals.

Figure 23: MediBot Homepage Interface

In the figure 24, the Medicine Recommendation System UI is displayed, where users can input their symptoms. This interface enables the MediBot application to provide personalized medication recommendations based on the entered symptoms.

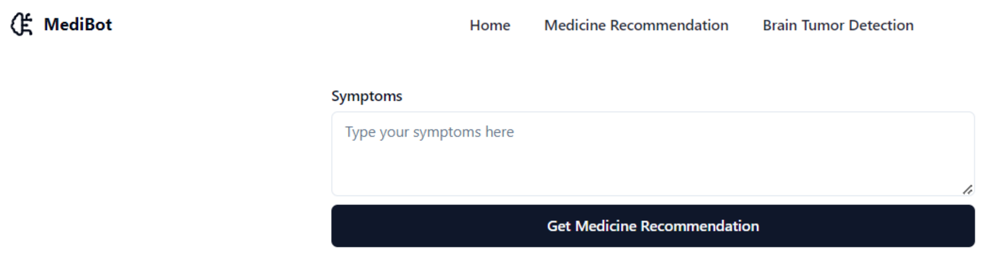
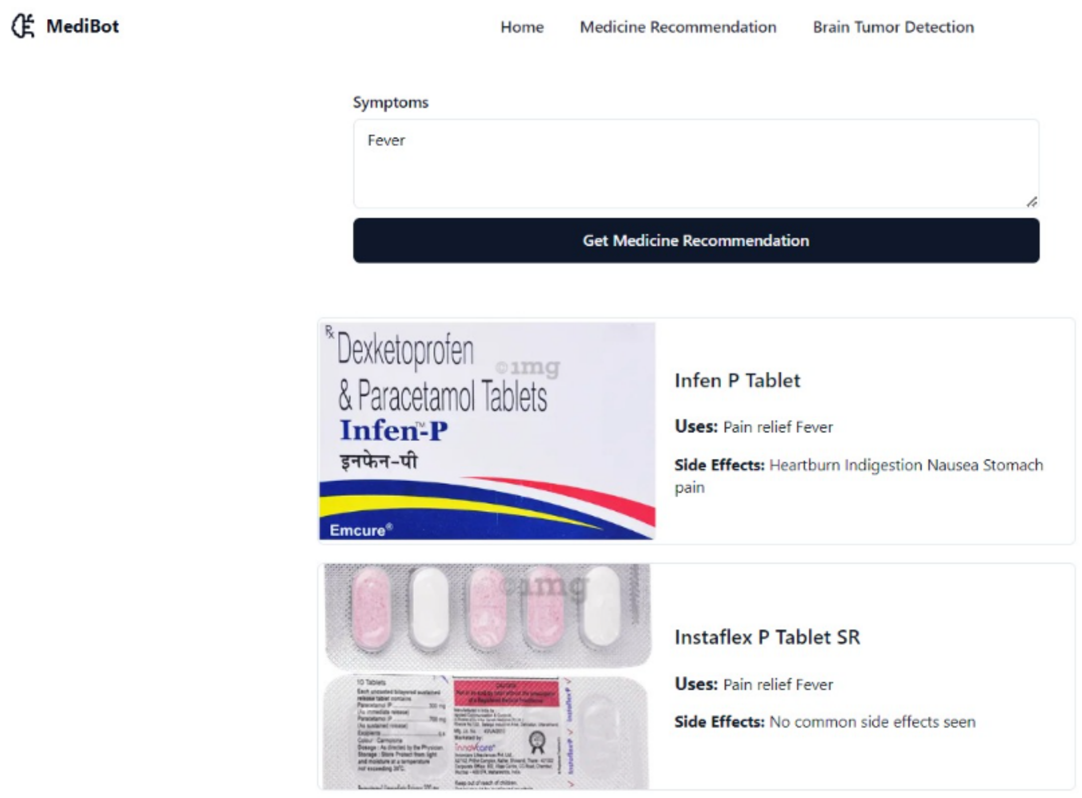


Figure 24: Medicine Recommendation System Interface

In the figure 25, the Medicine Recommendation System interface is depicted. Upon entering symptoms such as fever, users receive four medicine recommendations tailored to the provided symptoms, enhancing the user experience with personalized healthcare suggestions.



Symptoms

Fever

Infen P Tablet

Uses: Pain relief Fever

Side Effects: Heartburn Indigestion Nausea Stomach pain

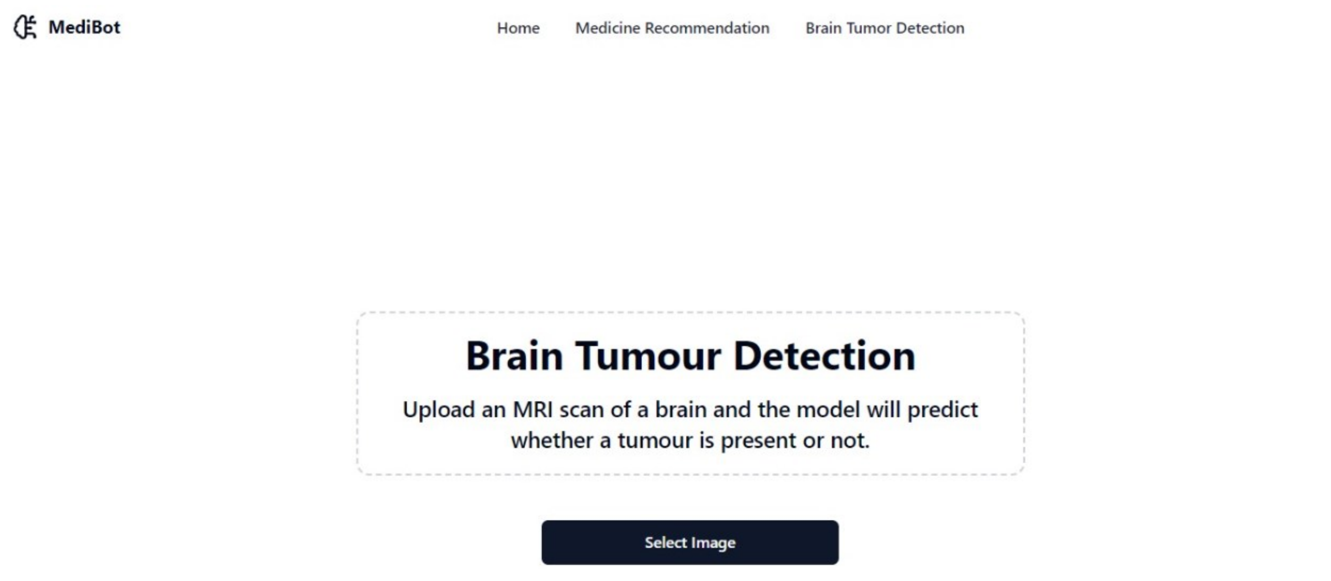
Instaflex P Tablet SR

Uses: Pain relief Fever

Side Effects: No common side effects seen

Figure 25: Medicine Recommendation for Fever

In the figure 26, the Brain Tumor Detection UI is displayed, designed to accept MRI images as input. This interface facilitates the upload and analysis of MRI scans within the MediBot application, aiding in accurate detection and diagnosis of brain tumors.



Brain Tumour Detection

Upload an MRI scan of a brain and the model will predict whether a tumour is present or not.

Figure 26: Brain Tumour Detection Interface

The figure 27 illustrates the Brain Tumor Detection UI where users initiate image selection by clicking "Select Image," prompting a dialog box for MRI image selection. This intuitive interface in MediBot streamlines the process of uploading MRI scans for accurate brain tumor detection and diagnosis.

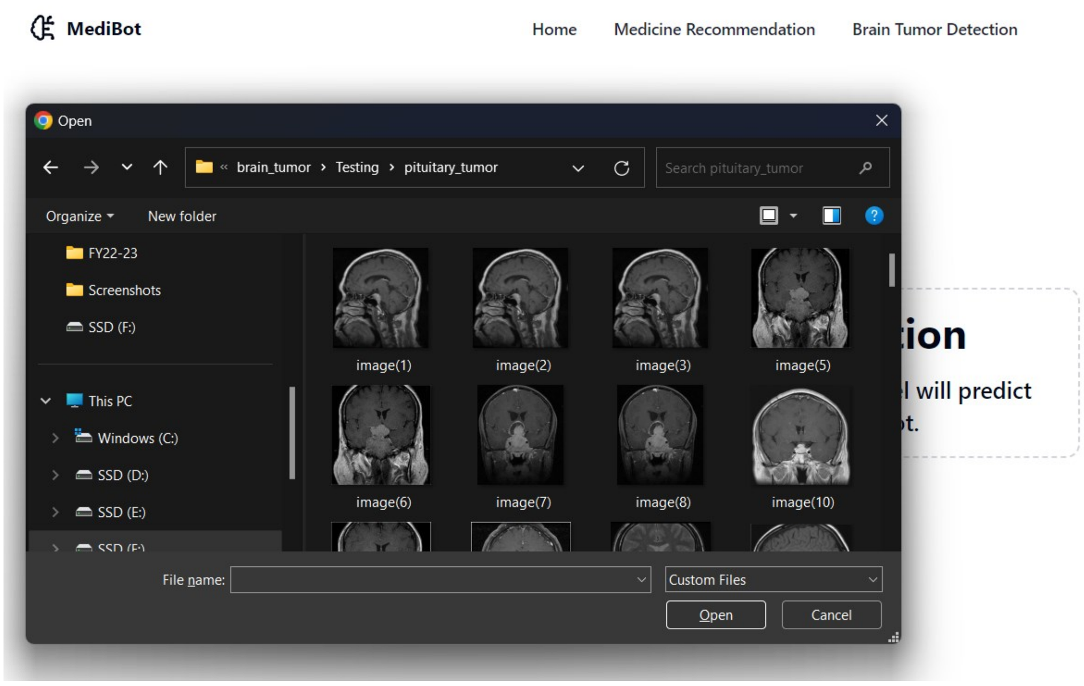


Figure 27: Selecting MRI Scan Image

The figure 28 showcases the Brain Tumor Detection interface where users submit a selected MRI image. Upon detection of a tumor in the MRI scan, the interface displays "Tumor Detected".

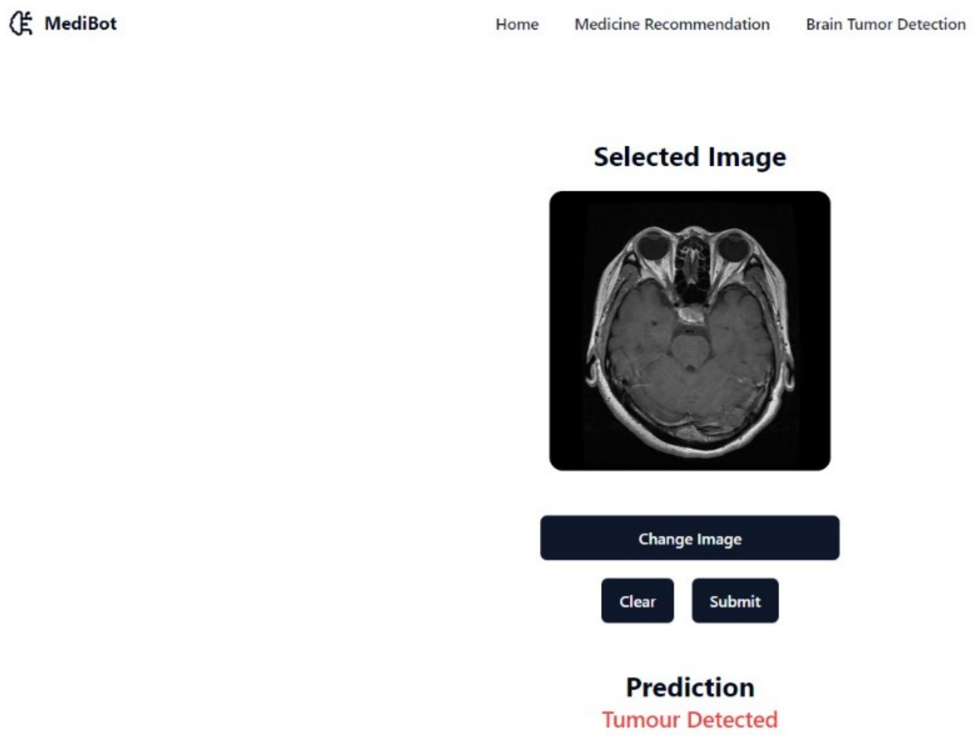


Figure 28: Tumour Prediction - Tumour Detected

The figure 29 depicts the Brain Tumor Detection interface where users submit a selected MRI image. If the scan indicates no tumor, the interface prominently displays "No Tumor Detected".

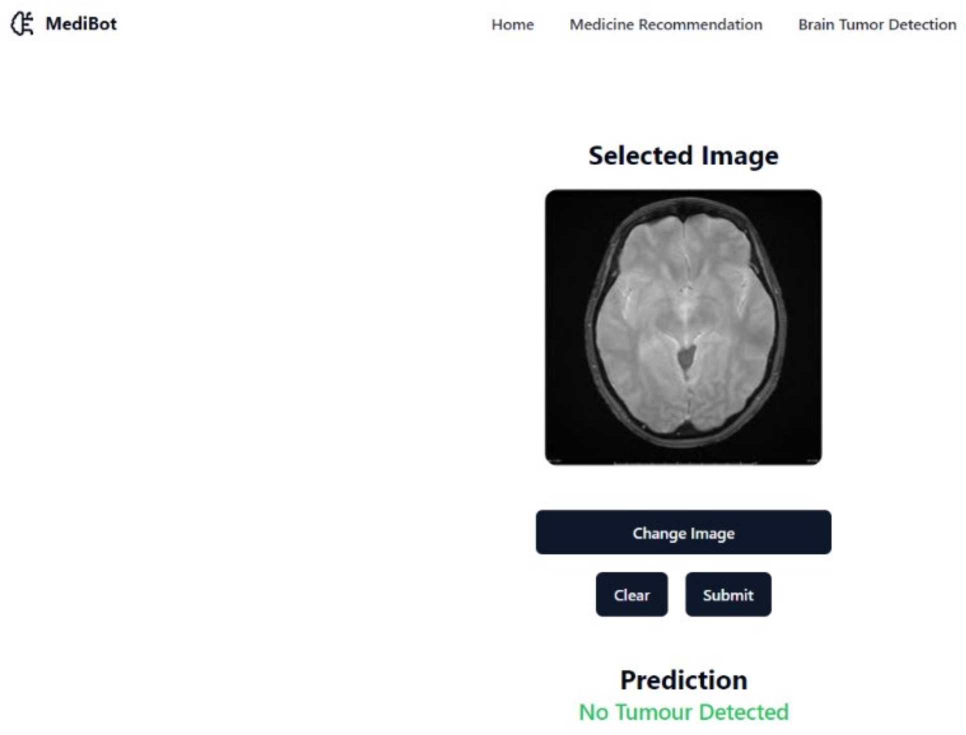


Figure 29: Tumour Prediction - No Tumour Detected

3. CONCLUSION AND FUTURE SCOPE

MediBot integrates personalized medicine recommendations system and brain tumor detection, facilitated through a frontend interface built with ReactJS and Tailwind. The homepage serves as a central hub, offering dedicated tabs for the medicine recommendation system and brain tumor detection, ensuring intuitive navigation and seamless access to critical healthcare services.

In the Medicine Recommendation tab, we employ advanced natural language processing techniques using TFIDF Vectorizer and Pandas. This enables personalized medication advice based on patient symptoms.

The Brain Tumour Detection tab utilizes SVM and Logistic Regression models from scikit-learn, combined with OpenCV for image processing, ensuring robust diagnostic capabilities.

FUTURE SCOPE:

MediBot has established a strong foundation poised for significant future developments in personalized medicine recommendation and advanced medical image analysis. This innovative AI-driven healthcare solution stands as a testament to collaborative efforts and technical expertise, aiming to revolutionize medical diagnostics. Looking ahead, key objectives include expanding diagnostic capabilities to encompass a wider range of medical conditions, engaging closely with medical professionals to refine and validate AI models, and continually enhancing user interface and recommendation accuracy through iterative user feedback. These strategic initiatives position MediBot to evolve into a sophisticated, adaptable solution with global relevance. Its ongoing contributions hold immense potential to drive healthcare technology forward, ultimately enhancing patient care and fostering improved health outcomes worldwide.

4. REFERENCES

1. https://www.youtube.com

2. https://www.kaggle.com

3. https://www.scikit-learn.org

4. https://www.docs.opencv.org

5. Drug recommendation system using machine learning based on TF-IDF feature extraction process by K.Sowmya and Saragadam Sridhar

6. An MRI brain tumour detection using logistic regression and support vector machine models by Srinivasarao Gajula.