MEDSYMPHONY

HARMONY CONDITIONS AND PREDICTION SYSTEM

A PROJECT REPORT SUBMITTED TO

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF SCIENCE IN APPLIED DATA SCIENCE

BY

RAJARAM S

REG NO. RA2332014010107

UNDER THE GUIDANCE OF

Dr. M.R. SUDHA, M.Sc, M.Phil, Ph.D

DEPARTMENT OF COMPUTER APPLICATIONS

FACULTY OF SCIENCE AND HUMANITIES

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur – 603 203

Chennai, Tamil Nadu

APRIL – 2025

BONAFIDE CERTIFICATE

This is to certify that the Project titled “**MEDSYMPHONY: HARMONY CONDITIONS AND PREDICTION SYSTEM”** is a bonafide work carried out by **Rajaram S (RA2332014010107)** under my supervision for the award of the **Degree in Master of Science in Applied Data Science.** To my knowledge the work reported herein is the original work done by these students.

Dr. M.R. Sudha,	**Dr. R. Jayashree,**
Assistant Professor	Associate Professor and Head,
Department of Computer Applications	Department of Computer Applications
(GUIDE)	

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION OF ASSOCIATION OF RESEARCH PROJECT WITH SUSTAINABLE DEVELOPMENT GOALS

This is to certify that the research project entitled "**MEDSYMPHONY: HARMONY CONDITIONS AND PREDICTION SYSTEM"** carried out by **Rajaram S** under the supervision of **Dr. M.R. Sudha** in partial fulfilment of the requirement for the award of Post Graduation program has been significantly or potentially associated with SDG Goal No **03 (THREE)** titled **GOOD HEALTH AND WELL-BEING.**

This study has clearly shown the extent to which its goals and objectives have been met in terms of filling the research gaps, identifying needs, resolving problems, and developing innovative solutions locally for achieving the above-mentioned SDG on a National and/or on an international level.

SIGNATURE OF THE STUDENT **SIGNATURE OF THE GUIDE**

SIGNATURE OF HEAD OF THE DEPARTMENT

ACKNOWLEDGEMENT

With profound gratitude to the ALMIGHTY, I take this chance to thank the people who helped me to complete this project.

I take this as a right opportunity to say THANKS to my parents who are there to stand with me always with the words “YOU CAN”.

I am thankful to **Dr. T. R. Paarivendhar**, Chancellor, and **Prof. A. Vinay Kumar,** Pro Vice-Chancellor (SBL), SRM Institute of Science and Technology, who gave me the platform to establish me to reach greater heights.

I earnestly thank **Dr. A. Duraisamy,** Dean, Faculty of Science and Humanities, SRM Institute of Science and Technology, who always encourage me to do novel things.

I earnestly thank **Dr. S. Albert Antony Raj**, Professor and Deputy Dean, College of Sciences, Faculty of Science and Humanities who always encourage me to do novel things.

I express my sincere thanks to **Dr. R. Jayashree**, Associate Professor and Head, Department of Computer Applications, Faculty of Science and Humanities for her valuable guidance and support to execute all incline in learning.

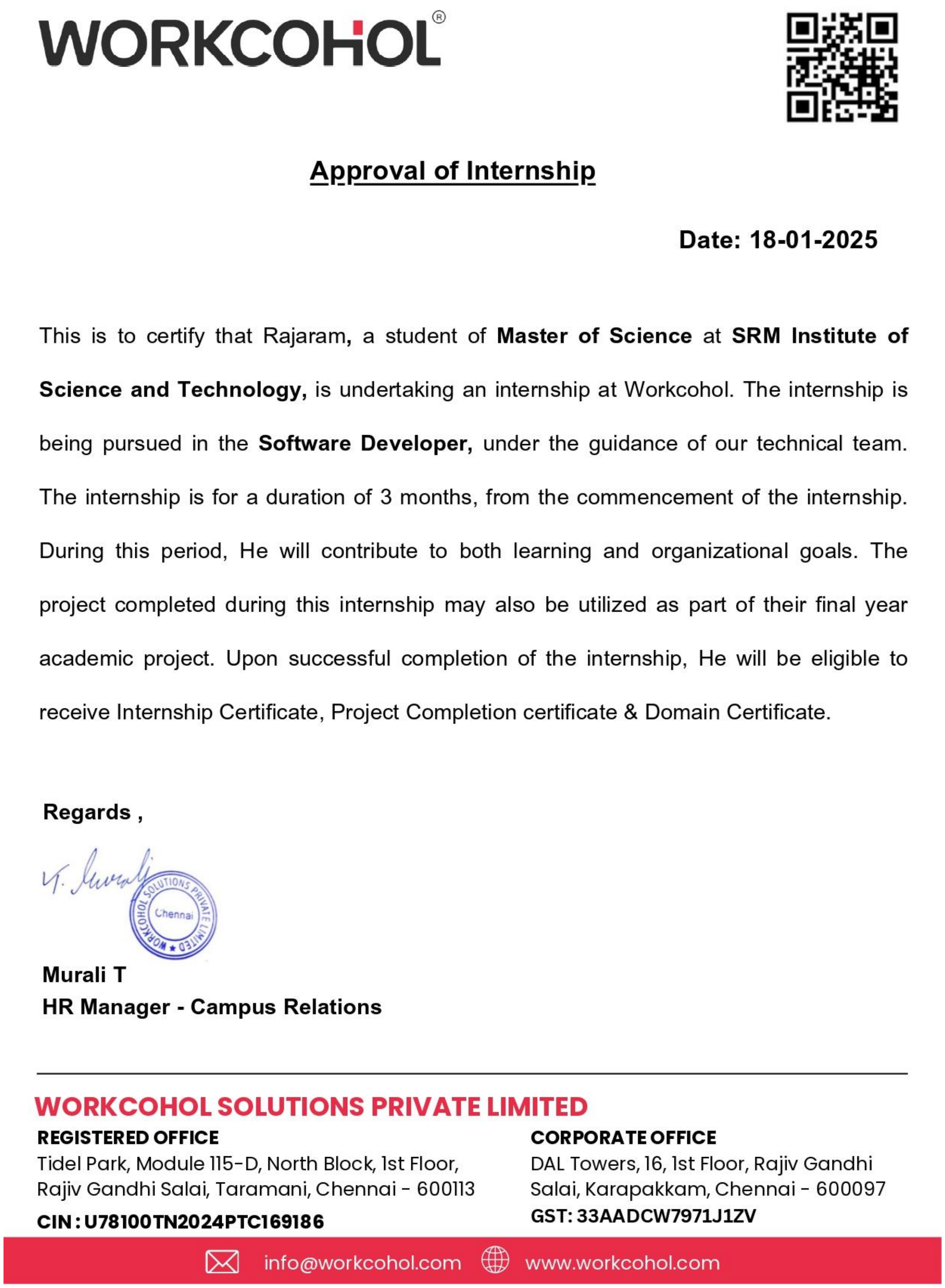
It is my delight to acknowledge **Dr. M.R. Sudha,** Assistant Professor, Department of Computer Applications for her help, support, encouragement, suggestions, and guidance, which have helped to establish me to greater heights throughout the development phases of the project.

I convey my gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

A great note of gratitude to friends and people who are known and unknown to me who helped in carrying out this project work a successful one.

Rajaram S (RA2332014010107)

COMPANY LETTER



WORKCOHOL®

Approval of Internship

Date: 18-01-2025

This is to certify that Rajaram, a student of **Master of Science** at **SRM Institute of Science and Technology,** is undertaking an internship at Workcohol. The internship is being pursued in the **Software Developer,** under the guidance of our technical team. The internship is for a duration of 3 months, from the commencement of the internship. During this period, He will contribute to both learning and organizational goals. The project completed during this internship may also be utilized as part of their final year academic project. Upon successful completion of the internship, He will be eligible to receive Internship Certificate, Project Completion certificate & Domain Certificate.

Regards ,

[illegible]

WORKCOHOL SOLUTIONS PRIVATE LIMITED ★ Chennai

Murali T
HR Manager - Campus Relations

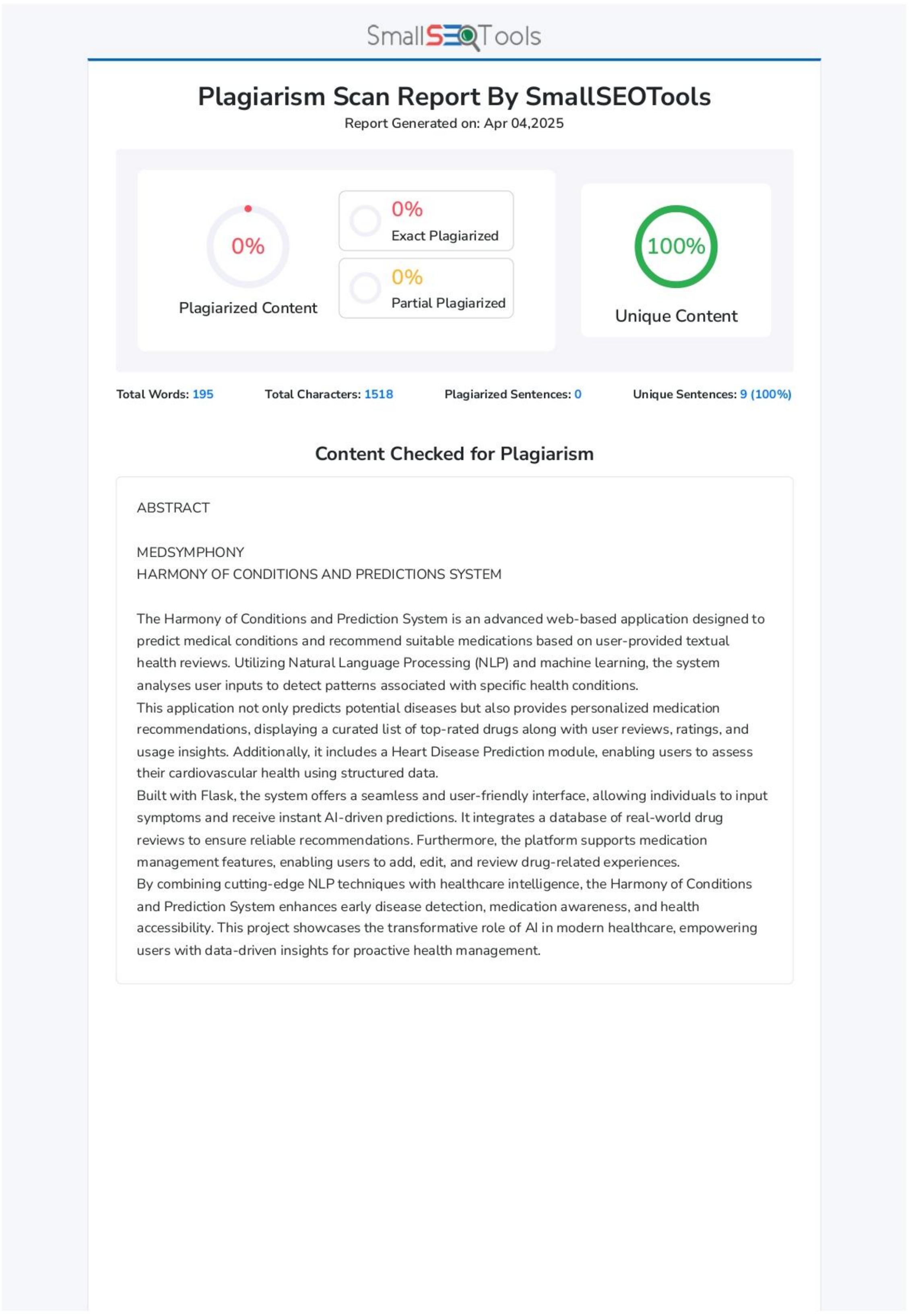
WORKCOHOL SOLUTIONS PRIVATE LIMITED

REGISTERED OFFICE
Tidel Park, Module 115-D, North Block, 1st Floor,
Rajiv Gandhi Salai, Taramani, Chennai - 600113
CIN : U78100TN2024PTC169186

CORPORATE OFFICE
DAL Towers, 16, 1st Floor, Rajiv Gandhi
Salai, Karapakkam, Chennai - 600097
GST: 33AADCW7971J1ZV

info@workcohol.com www.workcohol.com

PLAGIARISM CERTIFICATE



SmallSEOTools

Plagiarism Scan Report By SmallSEOTools

Report Generated on: Apr 04,2025

0% Plagiarized Content

0% Exact Plagiarized

0% Partial Plagiarized

100% Unique Content

Total Words: 195 Total Characters: 1518 Plagiarized Sentences: 0 Unique Sentences: 9 (100%)

Content Checked for Plagiarism

ABSTRACT

MEDSYMPHONY
HARMONY OF CONDITIONS AND PREDICTIONS SYSTEM

The Harmony of Conditions and Prediction System is an advanced web-based application designed to predict medical conditions and recommend suitable medications based on user-provided textual health reviews. Utilizing Natural Language Processing (NLP) and machine learning, the system analyses user inputs to detect patterns associated with specific health conditions.
This application not only predicts potential diseases but also provides personalized medication recommendations, displaying a curated list of top-rated drugs along with user reviews, ratings, and usage insights. Additionally, it includes a Heart Disease Prediction module, enabling users to assess their cardiovascular health using structured data.
Built with Flask, the system offers a seamless and user-friendly interface, allowing individuals to input symptoms and receive instant AI-driven predictions. It integrates a database of real-world drug reviews to ensure reliable recommendations. Furthermore, the platform supports medication management features, enabling users to add, edit, and review drug-related experiences.
By combining cutting-edge NLP techniques with healthcare intelligence, the Harmony of Conditions and Prediction System enhances early disease detection, medication awareness, and health accessibility. This project showcases the transformative role of AI in modern healthcare, empowering users with data-driven insights for proactive health management.

TABLE OF CONTENTS

CHAPTER NO	TOPIC	PAGE NO
	ABSTRACT	
CHAPTER 1	**INTRODUCTION**	**01**
CHAPTER 2	**SOFTWARE REQUIREMENT ANALYSIS** 2.1 Hardware Requirements 2.2 Needs in Software 2.3 Overview of the Software and Its Functionalities	**05**
CHAPTER 3	**DATA PREPROCESSING** 3.1 Data preparation 3.2 Data Attributes 3.3 Key Steps in Data Preparation	**07**
CHAPTER 4	**EXISTING METHOD** 4.1 NLP- Based Automated Text Evaluation System 4.2 Limitations of Existing System 4.3 Need for a new Approach	**08**
CHAPTER 5	**MODULE DESCRIPTION** 5.1 User Interface (Frontend) Module 5.2 Backend (Flack Server) Module 5.3 NLP- Based Prediction Module 5.4 Database Management Module 5.5 Data Processing & Machine Learning Module 5.6 Styling & Animation Module 5.7 Security & Validation Module	**10**
CHAPTER 6	**OBJECTIVES**	**13**
CHAPTER 7	**PROPOSED METHOD** 7.1 Data collection & preprocessing 7.2 Feature engineering & model training 7.3 Disease prediction module 7.4 Medication recommendation module 7.5 Web application development 7.6 Database management 7.7 System testing & deployment	**14**

CHAPTER 8	**UML DIAGRAM** 8.1 Use case Diagram 8.2 Activity Diagram 8.3 Sequence Diagram 8.4 Class Diagram	**17**
CHAPTER 9	**EXPLORATORY DATA ANALYSIS (EDA)** 9.1 Exploratory Data Analysis 9.2 Key Findings From EDA	**21**
CHAPTER 10	**SOURCE CODE** 10.1 Source Code 10.2 Heart Disease Prediction 10.3 Harmony Conditions and Prediction 10.4 Output	**24**
CHAPTER 11	**CONCLUSION**	**83**
CHAPTER 12	**FUTURE WORK**	**85**
CHAPTER 13	**REFERENCES**	**87**

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
8.1	Use Case Diagram	17
8.2	Activity Diagram	18
8.3	Sequence Diagram	19
8.4	Class Diagram	20
10.2.1	Plotting The Distribution Plot	50
10.2.2	Correlation Matrix	51
10.2.3	Bar Plot Using Colours	54
10.2.4	ROC- Heart Disease Prediction	57
10.2.5	Performance Evaluation- Heart Disease Prediction	60
10.3.1	Confusion Matrix Using Passive Aggressive Classifier	65
10.3.2	Confusion Matrix Using TFIDF	67
10.3.3	Confusion Matrix Using TFIDF-Bigrams	69
10.3.4	Confusion Matrix Using TFIDF- Trigram	71
10.4.1	Welcome Page	80
10.4.2	Enter Your Condition	80
10.4.3	Prediction And Medications	81
10.4.4	Database With CRUD	81
10.4.5	Heart Disease Predication	82
10.4.6	Heart Disease Predication- Result	82

ABSTRACT

MEDSYMPHONY

HARMONY OF CONDITIONS AND PREDICTIONS SYSTEM

The Harmony of Conditions and Prediction System is an advanced web-based application designed to predict medical conditions and recommend suitable medications based on user-provided textual health reviews. Utilizing Natural Language Processing (NLP) and machine learning, the system analyses user inputs to detect patterns associated with specific health conditions.

This application not only predicts potential diseases but also provides personalized medication recommendations, displaying a curated list of top-rated drugs along with user reviews, ratings, and usage insights. Additionally, it includes a Heart Disease Prediction module, enabling users to assess their cardiovascular health using structured data.

Built with Flask, the system offers a seamless and user-friendly interface, allowing individuals to input symptoms and receive instant AI-driven predictions. It integrates a database of real-world drug reviews to ensure reliable recommendations. Furthermore, the platform supports medication management features, enabling users to add, edit, and review drug-related experiences.

By combining cutting-edge NLP techniques with healthcare intelligence, the Harmony of Conditions and Prediction System enhances early disease detection, medication awareness, and health accessibility. This project showcases the transformative role of AI in modern healthcare, empowering users with data-driven insights for proactive health management.

1. INTRODUCTION

The integration of artificial intelligence (AI) in healthcare has transformed how medical conditions are diagnosed and managed. With the increasing availability of medical data, drug reviews, and patient experiences, AI-driven solutions can analyze large datasets to provide valuable insights into disease prediction and medication effectiveness. The Harmony of Conditions and Prediction System is a web-based application that utilizes machine learning and natural language processing (NLP) to assist users in identifying potential medical conditions based on textual inputs and recommending suitable medications.

The Harmony of Conditions and Prediction System exemplifies the power of AI in modern healthcare. By integrating machine learning, NLP, and real-world drug reviews, the system provides data-driven health insights, medication awareness, and early disease detection. This innovative platform helps users make informed decisions about their health, making medical guidance more accessible and personalized.

1.1. PURPOSE AND SIGNIFICANCE

Medical conditions can manifest through various symptoms, and patients often struggle to identify the underlying illness or find the right medication. Traditional diagnosis methods require expert medical consultations, which may not always be accessible. Similarly, while online drug reviews exist, manually searching for reliable medication recommendations can be time-consuming. This system bridges the gap by providing AI-driven disease predictions and personalized medication suggestions, improving healthcare accessibility and assisting users in making informed health decisions.

By analyzing user-provided symptom descriptions, the system can predict potential conditions using a trained classification model. Additionally, it evaluates patient-submitted drug reviews, helping users understand which medications are most effective for a given condition. The application also includes a heart disease prediction module, which analyzes structured health data to determine the likelihood of heart-related conditions.

1.2. HOW IT WORKS

- **DISEASE PREDICTION**
  - Users enter their symptoms or condition descriptions in text format.
  - The system processes the text using NLP techniques (TF-IDF vectorization).
  - A pre-trained machine learning model analyzes the input and predicts the most likely condition.
  - Users receive a list of the top recommended medications for the predicted condition.
- **DRUG REVIEW ANALYSIS**
  - The system uses a database of drug reviews to evaluate medication effectiveness.
  - It filters highly rated drugs based on user reviews, ratings, and usefulness count.
  - Users can compare different medications, their effects, and precautions before making a decision.
- **HEART DISEASE PREDICTION**
  - Users enter structured health parameters such as age, cholesterol levels, blood pressure, and more.
  - The system applies a machine learning model trained on heart disease datasets.
  - The model determines the risk of heart disease and provides preventive recommendations.

1.3. TECHNOLOGY IMPLEMENTATION

The system is built using a **robust technology stack** to ensure smooth performance and accurate predictions.

- **FRONTEND**
  - Developed using HTML, CSS, and JavaScript for an interactive and responsive UI.
  - Implements Bootstrap and modern UI frameworks for a seamless user experience.

- **BACKEND**
  - Powered by Flask (Python) to handle form submissions, requests, and AI model inference.
  - Connects with a relational database (SQLite3) to store and manage user reviews and prediction results.
- **DATABASE**
  - Utilizes SQLite3 to efficiently store and retrieve drug reviews, user inputs, and disease prediction records.
- **MACHINE LEARNING & AI**
  - Employs Natural Language Processing (NLP) techniques, including TF-IDF vectorization, to analyze text-based symptom descriptions.
  - Uses classification models trained on medical condition datasets for accurate disease prediction.
  - Incorporates a heart disease prediction model trained on structured patient health data.
- **DATA SOURCES**
  - The system leverages **medical datasets** containing information about diseases, symptoms, and treatments.
  - It integrates **drug review datasets** sourced from real patient experiences.

1.4. KEY BENEFITS AND IMPACT

- ✓ **Early Disease Detection:** Helps users identify potential conditions before consulting a doctor.
- ✓ **Personalized Medication Recommendations:** Provides users with highly rated drugs based on patient feedback.
- ✓ **Healthcare Accessibility:** Assists individuals in remote areas where medical consultations are limited.

- ✓ **AI-Powered Insights:** Uses data-driven predictions to enhance health awareness and decision-making.
- ✓ **User-Friendly Interface:** A simple and interactive platform for entering symptoms and getting predictions instantly.

2. SOFTWARE REQUIREMENT ANALYSIS

To develop a robust and efficient machine learning model for cryptocurrency price prediction, the project requires specific hardware and software resources. This section outlines the necessary tools and environments to ensure smooth implementation and performance.

2.1. HARDWARE REQUIREMENTS

- **Operating System:** Windows 11 or higher, macOS, or Linux
- **Processor:** Intel Core i3 or higher, or equivalent AMD processor.
- **RAM:** Minimum 8 GB (Suggested: 16 GB or higher for handling larger datasets professionally)
- **Storage:** The installation of necessary software programs and the storage of datasets and outputs require at least 100 GB of free disk space.
- **Visual Aids (Optional):** Model training can be greatly accelerated with a dedicated GPU (NVIDIA GTX series or higher), especially with quality deep learning knowledge.

2.2. NEEDS IN SOFTWARE

2.2.1. PYTHON VERSION & IDES REQUIRED

- Python 3.8 or later
- **Jupyter Notebook:** For analysis and interactive coding
- **PyCharm or Visual Studio Code:** For project management and code editing

2.2.2. PACKAGES AND LIBRARIES

- **Pandas:** For managing and processing tabular data
- **NumPy:** For array operations and numerical calculations
- **Matplotlib:** For data visualization (static, interactive, animated)
- **Seaborn:** For visualizing statistical data
- **Scikit-learn:** For preliminary modeling, evaluation, and preprocessing
- **Flask:** A lightweight web framework for building web applications and APIs.

2.2.3. VERSION CONTROL SYSTEM

Git: For project version management, especially in team settings.

2.2.4. DATA MANAGEMENT AND STORAGE

- ❖ **SQLite3:** Used for lightweight, local database storage, suitable for managing structured data efficiently in this project.
- ❖ **MySQL or PostgreSQL:** Alternatives for larger applications requiring durable, scalable data storage
- ❖ **CSV files:** For local dataset storage

2.3. OVERVIEW OF THE SOFTWARE AND ITS FUNCTIONALITIES

2.3.1. THE PYTHON PROGRAMMING LANGUAGE

Python was selected for this project because of its huge community support, ease of use, and abundance of data science and machine learning tools.

Code, images, and text can all be seamlessly integrated with Jupyter Notebook, making it perfect for data exploration, modeling, and result presentation.

2.3.2. GIT & VERSION CONTROL

Git guarantees that many project iterations may be efficiently managed, facilitating improved cooperation and change monitoring across the development cycle.

2.3.3. SQLITE3

Lightweight Database Management: SQLite3 was chosen for this project due to its simplicity, efficiency, and ease of integration with Python. As a serverless, self-contained database engine, SQLite3 allows for local data storage without requiring complex configurations. Its lightweight nature makes it an excellent choice for applications that do not require high concurrency or distributed database management.

Additionally, SQLite3's compatibility with Python libraries enables smooth data handling, while its reliability ensures efficient querying and storage of structured data for the disease prediction system.

3. DATA PREPROCESSING

3.1 DATA PREPARATION

Data preparation is a crucial step in transforming raw medical data into a structured format suitable for machine learning models. In this project, historical medical data and drug review datasets were collected, including details such as patient symptoms, disease labels, drug names, review texts, and sentiment scores. The dataset included the following key attributes:

3.2 DATA ATTRIBUTES

❖ **Patient Symptoms:** Textual descriptions of symptoms provided by users.
❖ **Disease Label:** The corresponding disease classification based on symptoms.
❖ **Drug Name:** The name of the prescribed medication.
❖ **Review Text:** User-submitted feedback on drug effectiveness.
❖ **Rating:** Numeric rating (1-10) given by users for medications.
❖ **Sentiment Score:** The sentiment analysis result (Positive, Neutral, Negative) based on the review text.

3.3 KEY STEPS IN DATA PREPARATION

❖ **Text Cleaning & Tokenization:** The raw textual data was cleaned by removing special characters, stopwords, and redundant words. Tokenization was applied to split text into meaningful words.
❖ **Addressing Missing Values:** The dataset was examined for missing values, and any incomplete entries were handled using imputation techniques or removal if necessary.
❖ **Feature Extraction:** Text-based features were extracted using TF-IDF (Term Frequency-Inverse Document Frequency) and Word Embeddings (Word2Vec, GloVe) for meaningful representation in machine learning models.
❖ **Feature Normalization:** Numerical features such as user ratings and sentiment scores were normalized to maintain uniformity across different scales and prevent biases in predictions.

This preprocessing ensures that the machine learning models receive high-quality, structured data for disease prediction and drug review sentiment analysis.

4. EXISTING METHOD

Current advancements in Natural Language Processing (NLP) and machine learning have led to various AI-driven applications in different domains, including education, automated evaluation, and healthcare. Several existing systems utilize NLP to analyze text, detect patterns, and provide automated decision-making. However, most of these systems focus on subjective answer evaluation rather than medical condition prediction and medication recommendations.

4.1. NLP-BASED AUTOMATED TEXT EVALUATION SYSTEM

Various studies have explored the use of NLP for subjective text analysis in fields like education and automated grading. These approaches focus on analyzing written responses, extracting key information, and evaluating the semantic relevance of the provided text.

4.1.1. AUTOMATED SUBJECTIVE ANSWER EVALUATION USING NLP

- This system utilizes NLP techniques to assess student responses by analyzing their language and reasoning skills.
- Key features include grammar checks, keyword extraction, and semantic similarity measurements to ensure accurate grading.
- Although useful in text analysis, this approach is not designed for medical condition prediction or drug recommendation.

4.1.2. AUTOMATIC SUBJECTIVE ANSWER EVALUATION

- This model applies automated grammar checks and scans for essential keywords in student responses.
- It employs similarity measures to compare text against predefined reference answers, reducing human intervention and bias.
- While effective for education-based assessments, it lacks the ability to interpret medical text inputs and suggest treatments.

4.1.3. AUTOMATING DESCRIPTIVE ANSWER GRADING USING REFERENCE-BASED MODELS

- This research explores reference-based models that compare student answers with ideal reference responses.
- It leverages advanced NLP techniques to evaluate the relevance and accuracy of responses.
- Though beneficial for grading descriptive answers, this system does not extend to healthcare applications, where contextual understanding of symptoms and medication effectiveness is required.

4.2. LIMITATIONS OF EXISTING SYSTEM

While NLP-based automated evaluation has shown success in text analysis, these systems have significant limitations when applied to medical condition prediction and drug review analysis:

- ❖ **Lack of Medical Context Understanding** – Existing models focus on evaluating grammar and content relevance but do not process medical terminology, symptoms, or drug effectiveness.
- ❖ **No Health Condition Prediction** – These systems analyze written text but do not classify medical conditions based on symptom descriptions.
- ❖ **No Drug Recommendation System** – They do not analyze medication reviews to suggest effective drugs for a specific condition.
- ❖ **No Personalized Healthcare Insights** – Unlike disease prediction models, these systems lack real-time AI-driven recommendations for users.

4.3. NEED FOR A NEW APPROACH

The Harmony of Conditions and Prediction System overcomes these limitations by integrating:

- AI-driven disease prediction based on textual symptom descriptions.
- NLP-based drug review analysis to recommend top-rated medications.
- A heart disease prediction model that analyzes structured health data.
- A user-friendly platform built using Flask and SQLite3 for efficient data management.

5. MODULE DESCRIPTION

5.1. USER INTERFACE (FRONTEND) MODULE

- **Home Page:** Displays an overview of the project, navigation links, and disease prediction input.
- **Review Submission Page:** Users can submit their health condition or medication review for analysis.
- **Prediction Result Page:** Displays the predicted medical condition and top medication recommendations based on user input.
- **Dataset Viewing Page:** Allows users to browse and search drug reviews stored in the system.
- **Add/Edit Review Page:** Enables users to add new reviews or update existing ones related to medications.
- **Search Functionality:** Users can search for medications based on medical conditions.
- **Heart Disease Prediction Page:** Allows users to enter health parameters for heart disease risk assessment.

5.2. BACKEND (FLASK SERVER) MODULE

- **Route Management:** Handles navigation between pages and processes user inputs.
- **Prediction Processing:** Calls the NLP-based ML model to predict the medical condition based on text input.
- **Medication Recommendation:** Extracts the top-rated drugs from the dataset based on reviews and effectiveness.
- **Heart Disease Prediction:** Processes structured health parameters to predict the risk of heart disease.
- **Database Interaction:** Manages storing, retrieving, updating, and deleting drug reviews.

- **Session Management:** Maintains user session data (if implemented).

5.3. NLP-BASED PREDICTION MODULE

- **Text Preprocessing:** Cleans user input by removing HTML tags, stopwords, and performing lemmatization.
- **Vectorization:** Converts textual input into numerical form using TF-IDF vectorization.
- **Model Prediction:** Uses a pre-trained machine learning model to classify the condition based on user input.
- **Drug Recommendation:** Identifies and recommends top-rated medications based on condition ratings and user reviews.

5.4. DATABASE MANAGEMENT MODULE

- **SQLite Database:** Stores data related to drug reviews, conditions, and medications.
- **CRUD Operations:** Supports adding, updating, deleting, and retrieving records efficiently.
- **Condition-Based Search:** Retrieves relevant drug reviews and medication data based on user queries.

5.5. DATA PROCESSING & MACHINE LEARNING MODULE

- **Dataset Integration:** Uses drugsComTrain.csv and drugsComTest_raw.tsv for training and validation.
- **Feature Engineering:** Extracts meaningful insights from drug review data to enhance prediction accuracy.
- **Model Training:** Trains and saves the machine learning model using joblib.
- **Model Deployment:** Loads and utilizes the trained ML model for real-time predictions.

- **Heart Disease Prediction Model:** Uses structured patient health data to assess the likelihood of heart disease.

5.6. STYLING & ANIMATION MODULE

- **CSS Styling:** Uses styles.css for a modern and clean UI design.
- **Bootstrap Integration:** Ensures a responsive and mobile-friendly design.
- **Animations:** Implements smooth UI transitions and effects using animate.css for better user engagement.

5.7. SECURITY & VALIDATION MODULE

- **Input Validation:** Prevents invalid or harmful input by enforcing proper formatting.
- **Database Security:** Protects against SQL injection and unauthorized access.
- **Session Handling:** Manages user sessions and authentication (if implemented).

6. OBJECTIVES

The Harmony of Conditions and Prediction System aims to enhance healthcare accessibility and decision-making by leveraging Natural Language Processing (NLP) and machine learning. The key objectives of this system are:

- ❖ **ACCURATE DISEASE PREDICTION:** The system analyzes user-submitted text inputs (such as symptoms or condition descriptions) using NLP techniques. It processes the text, extracts key medical terms, and applies machine learning models to predict the most likely medical condition associated with the input. This assists users in gaining early insights into potential health concerns.
- ❖ **PERSONALIZED MEDICATION RECOMMENDATIONS:** Once a medical condition is predicted, the system recommends top-rated medications based on a database of patient drug reviews. By considering factors such as effectiveness ratings, patient feedback, and useful vote counts, it provides users with data-driven medication suggestions, helping them make informed choices about treatment options.
- ❖ **EFFICIENT DRUG REVIEW MANAGEMENT:** The system includes a drug review management module that allows users to add, edit, and browse medication reviews. This feature helps individuals share their experiences with medications, aiding others in understanding effectiveness, side effects, and overall satisfaction.
- ❖ **USER-FRIENDLY INTERFACE:** Designed with a simple and intuitive UI, the system ensures that users can easily input their symptoms, receive condition predictions, and view drug recommendations. The interface includes features like search functionality, dataset browsing, and heart disease risk assessment, making healthcare insights accessible to all users.
- ❖ **DATA-DRIVEN DECISION MAKING:** The system relies on machine learning models trained on real-world medical datasets, including drug reviews and condition reports. This enables it to provide accurate, AI-powered predictions that enhance healthcare decision-making for users, bridging the gap between medical AI and patient needs.

7. PROPOSED METHOD

The Harmony of Conditions and Prediction System follows a structured approach to predict medical conditions and recommend medications using machine learning and NLP. The proposed method consists of the following steps:

7.1. DATA COLLECTION & PREPROCESSING

- Collect drug review datasets (e.g., *drugsComTrain.csv*) containing patient reviews, medical conditions, drug names, ratings, and usefulness scores.
- Perform data cleaning using NLP techniques by removing HTML tags, stopwords, special characters, and unnecessary text.
- Apply lemmatization to standardize words and improve text analysis accuracy.

7.2. FEATURE ENGINEERING & MODEL TRAINING

- Convert text data into numerical format using TF-IDF vectorization, which helps in extracting important keywords and patterns.
- Train a Machine Learning (ML) model (such as Logistic Regression, Random Forest, or Deep Learning) to classify medical conditions based on patient reviews.
- Save the trained model using joblib for real-time disease prediction.

7.3. DISEASE PREDICTION MODULE

- When a user enters a health condition or symptom description, preprocess the text using the trained TF-IDF vectorizer.
- The processed text is passed into the pre-trained ML model, which predicts the most probable medical condition based on patterns in user input.

7.4. MEDICATION RECOMMENDATION MODULE

- Extract top-rated medications from the dataset for the predicted condition.
- Filter medications with high ratings (≥9) and high usefulness scores (>100 reviews).
- Display the top 3 recommended drugs along with relevant details such as dosage guidelines, patient feedback, and precautions.

7.5. WEB APPLICATION DEVELOPMENT

- Develop a Flask-based web application with the following features:
  - Home Page – Users can enter their health conditions or symptoms.
  - Prediction Result Page – Displays the predicted condition and recommended medications.
  - Drug Review Page – Users can browse, add, and edit drug reviews.
  - Search Feature – Allows users to filter and search for drug reviews based on condition.
- Use Bootstrap & CSS for a modern, responsive UI.

7.6. DATABASE MANAGEMENT

- Implement an SQLite database to store:
  - Patient drug reviews
  - Medication details
  - User-submitted information
- Provide CRUD (Create, Read, Update, Delete) operations for managing drug reviews and user data.

7.7. SYSTEM TESTING & DEPLOYMENT

- Test the system using real-world drug review data to ensure prediction accuracy and reliability.
- Deploy the Flask web application on a cloud platform (e.g., Heroku or AWS) to make it accessible to users online.

By implementing this systematic approach, the Harmony of Conditions and Prediction System offers data-driven disease prediction, personalized medication recommendations, and efficient drug review management, contributing to better healthcare decision-making.

8. UML DIAGRAM

8.1. USE CASE DIAGRAM

This use case diagram represents the key functionalities of MEDSYMPHONY, a disease prediction system. Users interact with the system by inputting symptoms, after which the system processes the data and predicts possible diseases. The system provides recommendations based on the diagnosis and presents the results to the user. It ensures efficient data processing and accurate disease prediction.

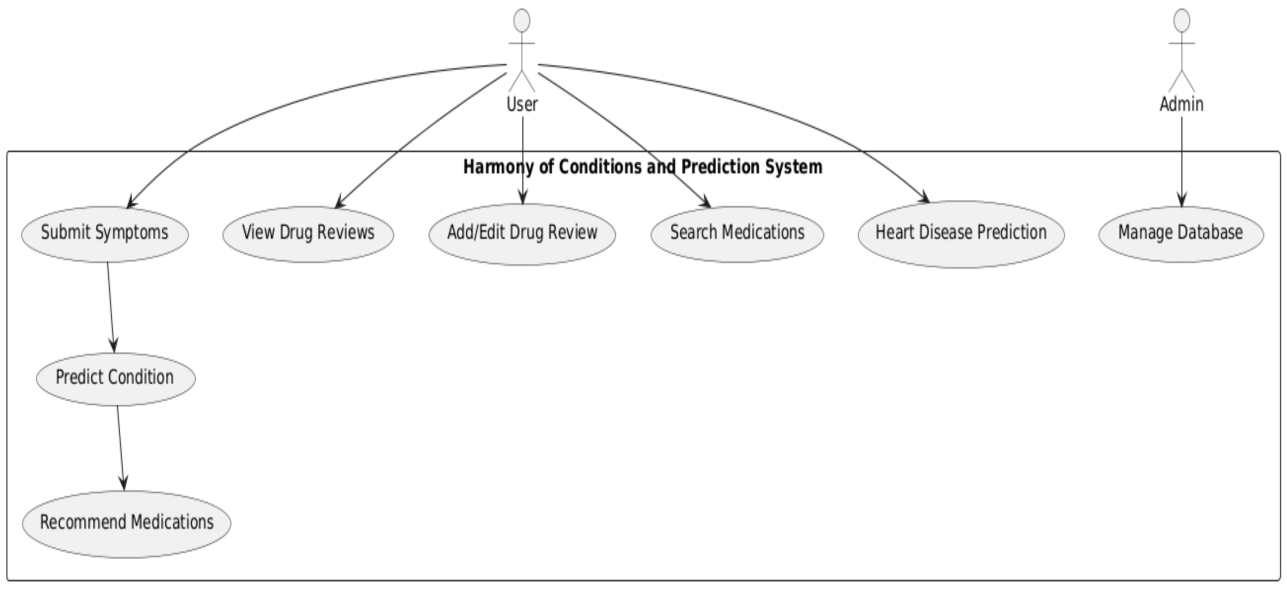
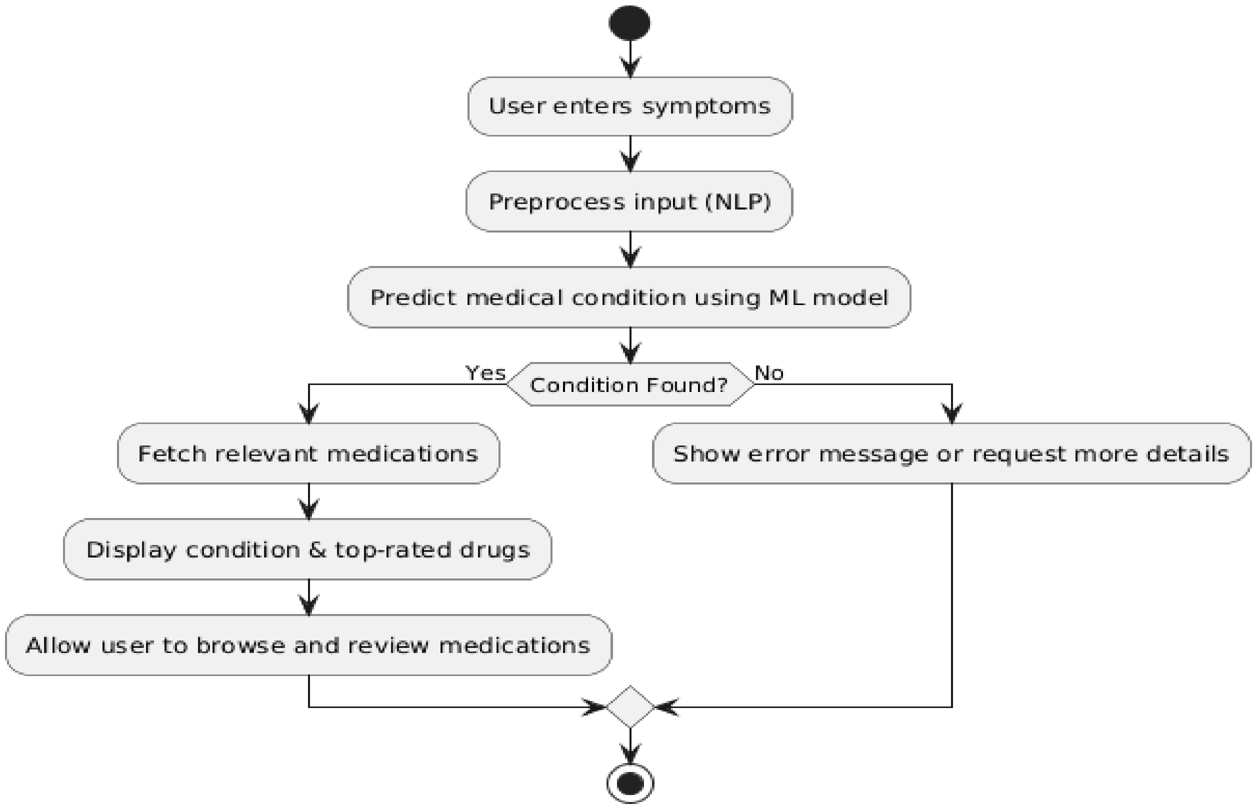


Figure 8.1 Use Case Diagram

8.2. ACTIVITY DIAGRAM

This flowchart outlines the disease prediction process in MEDSYMPHONY, starting with users entering symptoms. The system preprocesses the input, applies machine learning models, and generates a disease prediction. The process concludes with displaying the prediction and recommendations to the user.



User enters symptoms
Preprocess input (NLP)
Predict medical condition using ML model
Yes
Condition Found?
No
Fetch relevant medications
Show error message or request more details
Display condition & top-rated drugs
Allow user to browse and review medications

Figure 8.2 Activity Diagram

8.3. SEQUENCE DIAGRAM

This diagram illustrates the disease prediction process in MEDSYMPHONY, where the user inputs symptoms, and the system analyzes the data using machine learning algorithms. The system retrieves relevant medical data, predicts the disease, provides recommendations, and displays the results to the user. The process ends after displaying the prediction.

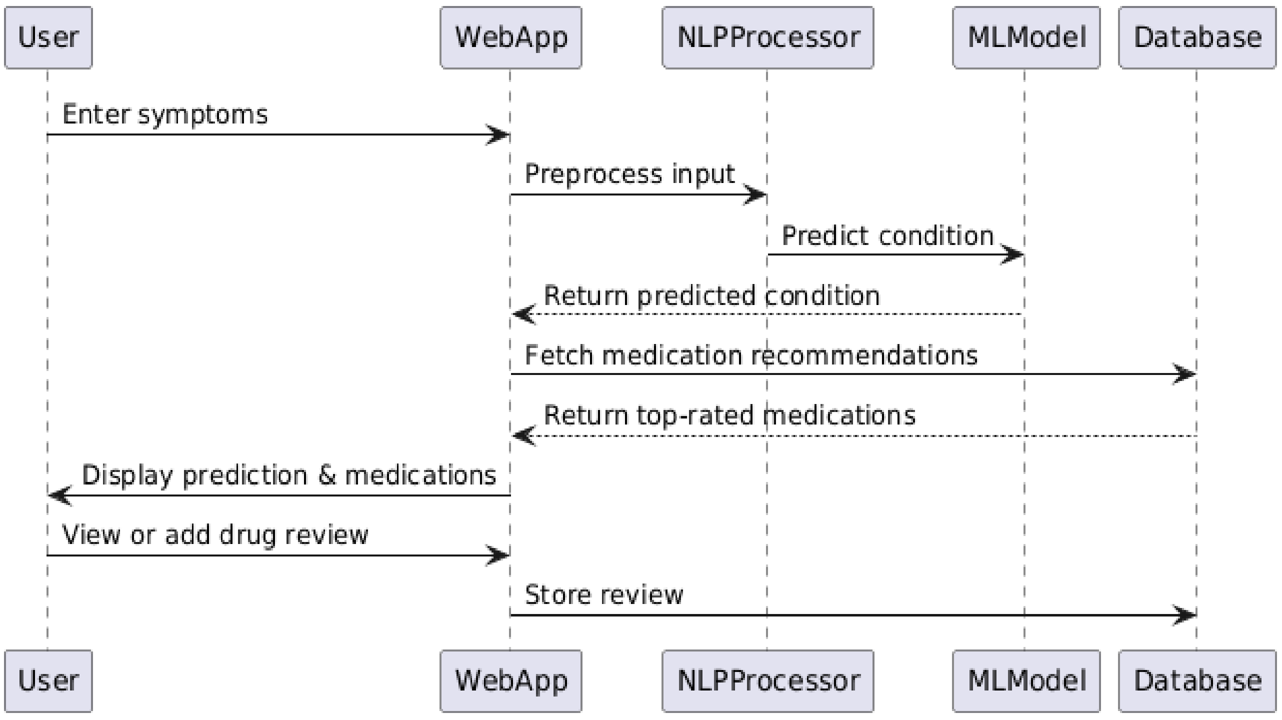


Figure 8.3 Sequence Diagram

8.4. CLASS DIAGRAM

This class diagram represents MEDSYMPHONY, including entities such as User, Symptom, Disease, Prediction Model, and Database. It defines relationships between these entities, showing how symptoms are analyzed, processed through predictive models, and linked to possible diseases. The system efficiently manages data storage and retrieval for accurate predictions.

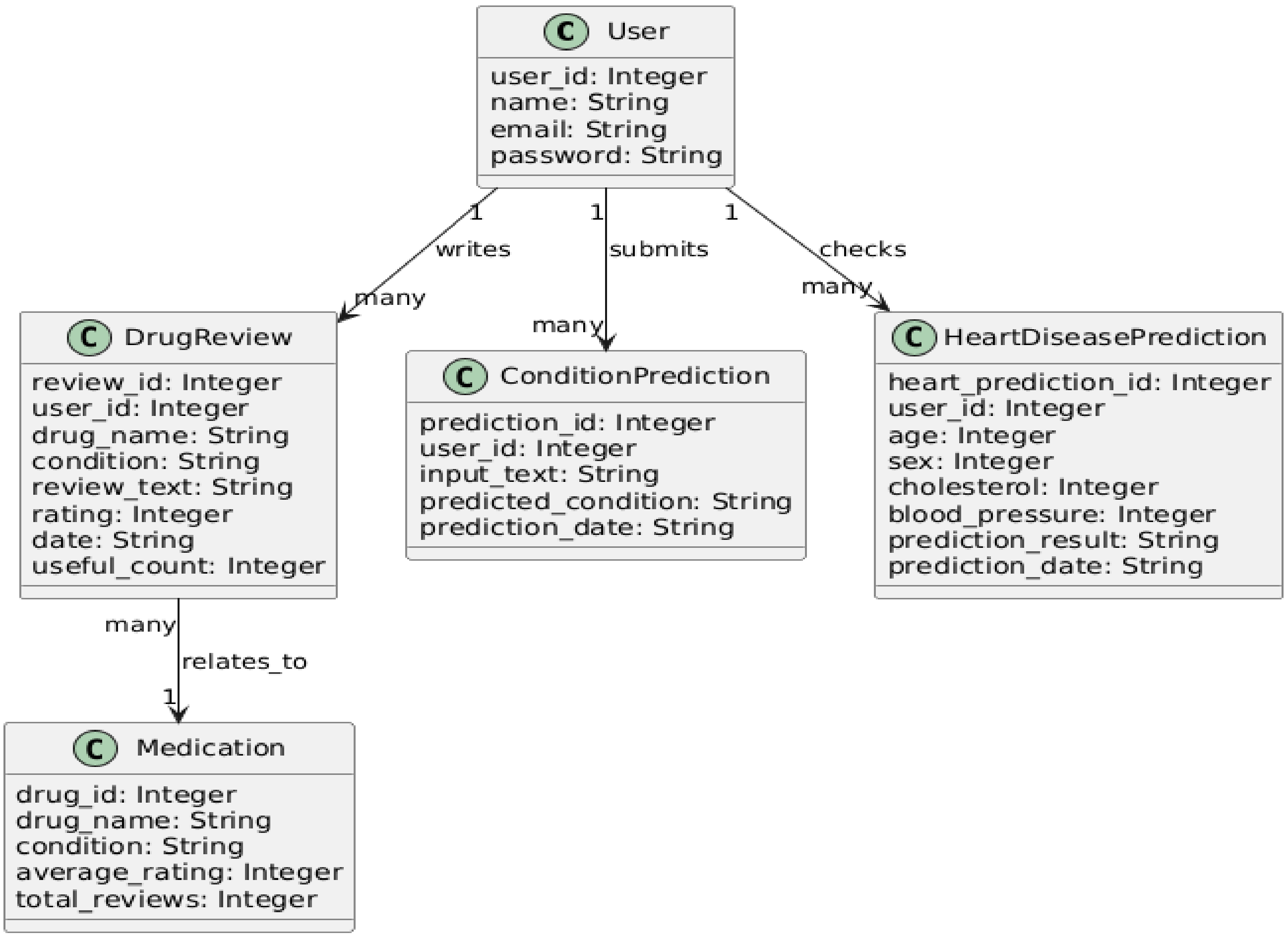


Figure 8.4 Class Diagram

9. EXPLORATORY DATA ANALYSIS (EDA)

9.1 EXPLORATORY DATA ANALYSIS (EDA):

Exploratory Data Analysis (EDA) is essential for understanding the structure and distribution of data before training predictive models. In the Harmony of Conditions and Prediction System, EDA was conducted on:

1. User-submitted symptom data – Free-text descriptions of health conditions.
2. Drug review dataset – Includes drug names, associated conditions, patient reviews, ratings, and usefulness votes.

EDA WAS USED TO IDENTIFY:

- Common symptom patterns and their relationship to predicted conditions.
- Trends in medication effectiveness based on user feedback.
- Outliers and anomalies in drug review ratings.
- Review usefulness impact on drug recommendations.
- Data imbalances and preprocessing techniques required for better model performance.

Visualizations such as bar charts, histograms, word clouds, scatter plots, and heatmaps were used to analyze the data and improve disease prediction accuracy.

9.2. KEY FINDINGS FROM EDA:

9.2.1. CORRELATION BETWEEN SYMPTOMS AND PREDICTED CONDITIONS

- Certain symptom combinations frequently led to specific condition predictions.
    - "Chronic pain" and "fatigue" → often associated with arthritis, fibromyalgia.
    - "Shortness of breath" and "chest pain" → linked to cardiovascular diseases.
    - "Restlessness" and "lack of focus" → mapped to ADHD.
- Word Cloud Analysis revealed that some symptoms were more common in patient inputs, helping refine the NLP model.

9.2.2. TRENDS IN DRUG REVIEWS AND MEDICATION EFFECTIVENESS

- ❖ Medications for well-documented conditions (e.g., diabetes, hypertension) had high ratings (8-10) due to standardized treatments.
- ❖ Mental health medications showed higher rating variability, indicating that patient response to these drugs was more individualized.
- ❖ Opioid-based pain medications had polarized reviews, with some patients rating them highly for pain relief and others negatively due to dependency risks.
- ❖ Heatmap visualizations helped identify the most commonly prescribed and highest-rated medications for various conditions.

9.2.3. DETECTING ANOMALIES IN DRUG REVIEWS

- ❖ Outliers were found where some patients reported extremely positive or negative experiences for the same medication.
- ❖ Negative outliers often stemmed from severe side effects, incorrect dosages, or misuse.
- ❖ Sentiment analysis and z-score outlier detection were used to filter misleading patterns in medication recommendations.

9.2.4. RATING DISTRIBUTION AND REVIEW PATTERNS

- ❖ Histogram analysis showed that most drugs had ratings between 6 and 8, with fewer extreme ratings (1 or 10).
- ❖ Longer reviews were often associated with negative feedback, as patients tended to describe side effects in more detail.
- ❖ Shorter reviews were mostly positive, often stating quick relief from symptoms.

9.2.5. IMPACT OF REVIEW USEFULNESS ON MEDICATION PERCEPTION

- ❖ Reviews marked as “highly useful” contained detailed dosage experiences, long-term effects, and patient recommendations.
- ❖ Scatter plots showed a positive correlation between high-usefulness votes and well-rated medications.
- ❖ Reviews with low usefulness scores tended to have incomplete or generic feedback.

9.2.6. IMPORTANCE OF DATA PREPROCESSING FOR MODEL PERFORMANCE

- ❖ Stopword removal, lemmatization, and TF-IDF vectorization significantly improved NLP-based predictions.
- ❖ The dataset had imbalanced condition distributions, which required oversampling techniques (such as SMOTE) to ensure fair representation of all conditions.
- ❖ Noise removal and normalization techniques improved the accuracy of both condition predictions and drug recommendations.

10. SOURCE CODE

10.1. SOURCE CODE

app.py

```
from flask import Flask, render_template, request, redirect, session, url_for

import os

import joblib

import pandas as pd

import re

import numpy as np

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

from bs4 import BeautifulSoup

import nltk

from db import Database

db = Database()

try:

    stop = stopwords.words('english')

except LookupError:

    nltk.download('stopwords')

    stop = stopwords.words('english')

try:

    lemmatizer = WordNetLemmatizer()
    nltk.data.find('corpora/wordnet')
except LookupError:
    nltk.download('wordnet')
app = Flask(__name__)
MODEL_PATH = 'model/passmodel.pkl'
TOKENIZER_PATH = 'model/tfidfvectorizer.pkl'
DATA_PATH = 'data/drugsComTrain.csv'
vectorizer = joblib.load(TOKENIZER_PATH)
model = joblib.load(MODEL_PATH)
rawtext = ""
@app.route('/', methods=["GET", "POST"])
def predict():
    if request.method == 'POST':
        raw_text = request.form['rawtext']
        if raw_text != "":
            clean_text = cleanText(raw_text)
            clean_lst = [clean_text]
            tfidf_vect = vectorizer.transform(clean_lst)
            prediction = model.predict(tfidf_vect)
            predicted_cond = prediction[0]
            df = pd.read_csv(DATA_PATH)
            top_drugs = top_drugs_extractor(predicted_cond, df)
            return render_template('home.html', rawtext=raw_text, result=predicted_cond,
top_drugs=top_drugs)
        else:
            raw_text = "There is no text to select"
    return render_template('home.html', rawtext=rawtext)
def cleanText(raw_review):
    review_text = BeautifulSoup(raw_review, 'html.parser').get_text()
    letters_only = re.sub('[^a-zA-Z]', ' ', review_text)
    words = letters_only.lower().split()
    meaningful_words = [w for w in words if not w in stop]
    lemmitize_words = [lemmatizer.lemmatize(w) for w in meaningful_words]
    return ' '.join(lemmitize_words)
def top_drugs_extractor(condition, df):
    df_top = df[(df['rating'] >= 9) & (df['usefulCount'] >= 100)].sort_values(by=['rating',
'usefulCount'], ascending=[False, False])
    drug_lst = df_top[df_top['condition'] == condition]['drugName'].head(3).tolist()
    return drug_lst
@app.route('/index')
def index():
    condition = request.args.get('condition', '')
    if condition:
        reviews = db.search_by_condition(condition)
    else:
        reviews = db.fetch()
    return render_template('index.html', reviews=reviews)
@app.route('/add', methods=['GET', 'POST'])
def add_review():
    if request.method == 'POST':
        patientID = request.form['patientID']
        drugName = request.form['drugName']
        condition = request.form['condition']
        review = request.form['review']
        rating = request.form['rating']
        date = request.form['date']
        usefulCount = request.form['usefulCount']
        db.insert(patientID, drugName, condition, review, rating, date, usefulCount)
        return redirect(url_for('index'))
    return render_template('add.html')
@app.route('/edit/<string:patientID>', methods=['GET', 'POST'])
def edit_review(patientID):
    review = db.get_review(patientID)
    if request.method == 'POST':
        drugName = request.form['drugName']
        condition = request.form['condition']
        review_text = request.form['review']
        rating = request.form['rating']
        date = request.form['date']
        usefulCount = request.form['usefulCount']
        db.update(patientID, drugName, condition, review_text, rating, date, usefulCount)
        return redirect(url_for('index'))
    return render_template('edit.html', review=review)
@app.route('/delete/<string:patientID>')
def delete_review(patientID):
    db.remove(patientID)
    return redirect(url_for('index'))
@app.route('/home')
def home():
    return redirect(url_for('predict'))
heart_model_path = "heart.pkl"
scaler, heart_model = joblib.load(heart_model_path)
@app.route('/heart', methods=['GET', 'POST'])
def heart():
    if request.method == 'POST':
        data = [float(x) for x in request.form.values()]
        features = np.array(data).reshape(1, -1)
        features_scaled = scaler.transform(features)
        prediction = heart_model.predict(features_scaled)
        output = "Heart Disease Detected" if prediction[0] == 1 else "No Heart Disease"
        return render_template("heart.html", prediction_text=output)
    return render_template("heart.html")

@app.route('/contact')

def contact():

    return render_template('contact.html')

if __name__ == "__main__":

    app.run(debug=False)
```

db.py

```
import sqlite3

class Database:

    def __init__(self, db="harmony_medications.db"):

        self.con = sqlite3.connect(db, check_same_thread=False)

        self.cur = self.con.cursor()

        self.create_table()

    def create_table(self):

        self.cur.execute('''

        CREATE TABLE IF NOT EXISTS drug_reviews (

            patientID TEXT PRIMARY KEY,

            drugName TEXT,

            condition TEXT,

            review TEXT,

            rating INTEGER,

            date TEXT,
            usefulCount INTEGER
        )''')
        self.con.commit()
    def insert(self, patientID, drugName, condition, review, rating, date, usefulCount):
        self.cur.execute("INSERT INTO drug_reviews (patientID, drugName, condition, review,
rating, date, usefulCount) VALUES (?, ?, ?, ?, ?, ?, ?)",
                         (patientID, drugName, condition, review, rating, date, usefulCount))
        self.con.commit()
    def fetch(self):
        self.cur.execute("SELECT * FROM drug_reviews LIMIT 10")
        return self.cur.fetchall()
    def get_review(self, patientID):
        self.cur.execute("SELECT * FROM drug_reviews WHERE patientID=?", (patientID,))
        return self.cur.fetchone()
    def update(self, patientID, drugName, condition, review, rating, date, usefulCount):
        self.cur.execute("UPDATE drug_reviews SET drugName=?, condition=?, review=?,
rating=?, date=?, usefulCount=? WHERE patientID=?",
                         (drugName, condition, review, rating, date, usefulCount, patientID))
        self.con.commit()
    def remove(self, patientID):
        self.cur.execute("DELETE FROM drug_reviews WHERE patientID=?", (patientID,))
        self.con.commit()
    def search_by_condition(self, condition):
        self.cur.execute("SELECT * FROM drug_reviews WHERE condition LIKE ?", ('%' +
condition + '%',))

        return self.cur.fetchall()
```

Add.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Add Drug Review</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body>

    <div class="container">

        <h2 class="text-center text-primary mt-4 text-black">

            <b>ADD HARMONY OF CONDITIONS AND PREDICTIONS REVIEW</b>

        </h2>

        <div class="card p-4 shadow-lg mt-3">

            <form action="{{ url_for('add_review') }}" method="POST">

                <div class="mb-3">

                    <label for="patientID" class="form-label">Patient ID</label>
```

```
          <input type="text" class="form-control" id="patientID" name="patientID"
required>
        </div>
        <div class="mb-3">
          <label for="drugName" class="form-label">Drug Name</label>
          <input type="text" class="form-control" id="drugName" name="drugName"
required>
        </div>
        <div class="mb-3">
          <label for="condition" class="form-label">Condition</label>
          <input type="text" class="form-control" id="condition" name="condition"
required>
        </div>
        <div class="mb-3">
          <label for="review" class="form-label">Review</label>
          <textarea class="form-control" id="review" name="review" rows="3"
required></textarea>
        </div>
        <div class="mb-3">
          <label for="rating" class="form-label">Rating (1-10)</label>
          <input type="number" class="form-control" id="rating" name="rating" min="1"
max="10" required>
        </div>
        <div class="mb-3">
```

```
                <label for="date" class="form-label">Date</label>

                <input type="date" class="form-control" id="date" name="date" required>

            </div>

            <div class="mb-3">

                <label for="usefulCount" class="form-label">Useful Count</label>

                <input          type="number"          class="form-control"          id="usefulCount"
name="usefulCount" required>

            </div>

            <div class="d-flex justify-content-between">

                <button type="submit" class="btn btn-primary">Submit</button>

                <a href="{{ url_for('index') }}" class="btn btn-secondary">Cancel</a>

            </div>

        </form>

    </div>    </div>

</body>

</html>
```

Edit.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Drug Review</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
            <h2 class="text-center text-primary mt-4 text-black"><b>EDIT HARMONY OF CONDITIONS AND PREDICTIONS REVIEW</b></h2><br>
        <div class="card p-4 shadow-lg">
            <form action="{{ url_for('edit_review', patientID=review[0]) }}" method="POST">
                <center>
                <div class="mb-3">
                    <label for="patientID" class="form-label">Patient ID</label>
                        <input type="text" class="form-control" id="patientID" name="patientID" value="{{ review[0] }}" required readonly>
                </div>
                <div class="mb-3">
                    <label for="drugName" class="form-label">Drug Name</label>
                        <input type="text" class="form-control" id="drugName" name="drugName" value="{{ review[1] }}" required>
                </div>
                <div class="mb-3">
                    <label for="condition" class="form-label">Condition</label>
                        <input type="text" class="form-control" id="condition" name="condition" value="{{ review[2] }}" required>
                </div>
                <div class="mb-3">
                    <label for="review" class="form-label">Review</label>
                    <textarea class="form-control" id="review" name="review" rows="3" required>{{
review[3] }}</textarea>
                </div>
                <div class="mb-3">
                    <label for="rating" class="form-label">Rating (1-10)</label>
                    <input type="number" class="form-control" id="rating" name="rating" min="1"
max="10" value="{{ review[4] }}" required>
                </div>
                <div class="mb-3">
                    <label for="date" class="form-label">Date</label>
                        <input type="date" class="form-control" id="date" name="date" value="{{
review[5] }}" required>
                </div>
                <div class="mb-3">
                    <label for="usefulCount" class="form-label">Useful Count</label>
                                    <input type="number" class="form-control" id="usefulCount"
name="usefulCount" value="{{ review[6] }}" required>
                </div>
                <button type="submit" class="btn btn-primary">Update</button>
                <a href="{{ url_for('index') }}" class="btn btn-secondary">Cancel</a>
            </center>
```

```
            </form>
        </div>
    </div>
</body>
</html>
```

Heart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Heart Disease Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='heart.css') }}">
</head>
<body>
    <a href="{{ url_for('home') }}" class="logout-btn">Logout</a>
    <div class="container">
        <h1>Heart Disease Prediction</h1>
        <form action="/predict" method="post">
            <input type="text" name="age" placeholder="Age" required>
            <input type="text" name="sex" placeholder="Sex (0: Female, 1: Male)" required>
            <input type="text" name="cp" placeholder="Chest Pain Type" required>
            <input type="text" name="trestbps" placeholder="Resting Blood Pressure" required>

            <input type="text" name="chol" placeholder="Cholesterol Level" required>

            <input type="text" name="fbs" placeholder="Fasting Blood Sugar (0 or 1)" required>

            <input type="text" name="restecg" placeholder="Resting ECG Results" required>

           <input type="text" name="thalach" placeholder="Max Heart Rate Achieved" required>

             <input type="text" name="exang" placeholder="Exercise Induced Angina (0 or 1)"
required>

            <input type="text" name="oldpeak" placeholder="ST Depression Induced" required>

            <input type="text" name="slope" placeholder="Slope of Peak Exercise" required>

           <input type="text" name="ca" placeholder="Number of Major Vessels (0-3)" required>

            <input type="text" name="thal" placeholder="Thalassemia (0-3)" required>

            <button type="submit">Predict</button>

        </form>

        {% if prediction_text %}

            <h2>{{ prediction_text }}</h2>

        {% endif %}

    </div>

</body>

</html>
```

Home.html

```
<!DOCTYPE html>

<html lang="en">
```

```
<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>MEDSYMPHONY</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">

</head>

<body>

    <div class="container">

        <div class="navbar">

            <ul>

                <li><a href="{{ url_for('home') }}">HOME</a></li>

                <li><a href="{{ url_for('index') }}">VIEW DATASETS</a></li>

                <li class="dropdown">

                    <a class="reservation" href="">Different Types of Disease Prediction</a>

                    <div class="dropdown-content">

                        <center>

                            <a href="{{ url_for('heart') }}">Heart Disease Prediction</a>

                        </center>

                    </div>

                </li>

            </ul>

        </div>

           <form method="POST" action="/">
```

```
            <label for="rawtext" class="animate__animated animate__fadeIn" style=" font-weight:bold;">Enter your Condition:</label><br>

        <br>

                <textarea id="rawtext" name="rawtext" rows="6" cols="50" required class="animate__animated animate__fadeIn">{{ rawtext }}</textarea>

        <br><br>

         <button type="submit" class="animate__animated animate__fadeInUp" style=" font-weight:bold;">Predict</button>

      </form>

      {% if result %}

         <h2 class="animate__animated animate__fadeInUp" style="color: white;">Prediction Result:</h2>

           <p class="animate__animated animate__fadeInUp" style=" font-weight:bold; font-size:20px; background:#ffffff42; border-radius:15px; padding:12px; width:fit-content; margin:auto; color:cyan;">Condition: <b style="font-size:26px; color:white;">{{ result }}</b></p>

           <p class="animate__animated animate__fadeInUp" style=" font-weight:bold; font-size:20px; color: white;">Top Recommended Drugs:</p>

        <ul class="animate__animated animate__fadeInUp unordered_list">

           {% for drug in top_drugs %}

              <li class="lists">{{ drug }}</li>

           {% endfor %}

        </ul><br>

        {# CONDITION-SPECIFIC INFORMATION BLOCKS BELOW #}

        {% if result == "Birth Control" %}
        <div class="condition-info animate__animated animate__fadeInUp">
          <div class="info"><h3>Information about Birth Control:</h3>
           <p>Birth control methods include hormonal contraceptives, barrier methods, and
more. Consult with a healthcare professional to find the best option for you.</p>
        </div><div class="precautions"><h3>Precautions and Measures:</h3>
          <ul>
            <li>Consistent and correct use of the chosen contraceptive method.</li>
            <li>Regular check-ups with a healthcare provider.</li>
          </ul></div>
        </div>
      {% elif result == "Depression" %}
        <div class="condition-info animate__animated animate__fadeInUp">
          <div class="info"><h3>Information about Depression:</h3>
           <p>Depression is a mental health disorder characterized by persistent feelings of
sadness and a lack of interest or pleasure in daily activities.</p>
          </div><div class="precautions"><h3>Precautions and Measures:</h3>
          <ul>
            <li>Build a support system.</li>
            <li>Engage in regular physical activity.</li>
            <li>Practice self-care.</li>
            <li>Establish a daily routine.</li>
            <li>Avoid excessive alcohol and substance use.</li>
            <li>Consider medication if prescribed.</li>
```

```
                    <li>Attend therapy sessions consistently.</li>

                    <li>Monitor and challenge negative thoughts.</li>

                    <li>Educate yourself and loved ones about depression.</li>

                </ul></div>

            </div>

        {% elif result == "Pain" %}

            <div class="condition-info animate__animated animate__fadeInUp">

                <div class="info"><h3>Information about Pain:</h3>

                    <p>Pain can be caused by various factors. Consult with a healthcare professional
to determine the cause and appropriate treatment.</p>

            </div><div class="precautions"><h3>Precautions and Measures:</h3>

                <ul>

                    <li>Follow the recommended treatment plan.</li>

                    <li>Practice stress-reducing techniques.</li>

                </ul></div>

            </div>

        {% elif result == "Anxiety" %}

            <div class="condition-info animate__animated animate__fadeInUp">

                <div class="info"><h3>Information about Anxiety:</h3>

                    <p>Anxiety is a mental health condition characterized by excessive worry or
fear.</p>

            </div><div class="precautions"><h3>Precautions and Measures:</h3>

                <ul>
                    <li>Practice relaxation techniques and mindfulness.</li>
                    <li>Consider therapy or counseling.</li>
                </ul></div>
            </div>
        {% elif result == "Bipolar Disorder" %}
            <div class="condition-info animate__animated animate__fadeInUp">
                <div class="info"><h3>Information about Bipolar Disorder:</h3>
                    <p>Bipolar disorder is a mental health condition characterized by extreme mood
swings.</p>
            </div><div class="precautions"><h3>Precautions and Measures:</h3>
                <ul>
                    <li>Adhere to prescribed medication regimens.</li>
                    <li>Regular follow-ups with mental health professionals.</li>
                </ul></div>
            </div>
        {% elif result == "ADHD" %}
            <div class="condition-info animate__animated animate__fadeInUp">
                <div class="info"><h3>Information about ADHD:</h3>
               <p>ADHD is a neurodevelopmental disorder marked by inattention, hyperactivity,
and impulsivity.</p>
            </div><div class="precautions"><h3>Precautions and Measures:</h3>
                <ul>
                    <li>Establish routines and structure.</li>
                    <li>Break tasks into smaller steps.</li>
                    <li>Use visual aids and reminders.</li>
                    <li>Encourage regular physical activity.</li>
                    <li>Explore behavioral therapy.</li>
                    <li>Consider medication if prescribed.</li>
                    <li>Provide a quiet and organized workspace.</li>
                    <li>Encourage healthy sleep habits.</li>
                    <li>Teach and reinforce social skills.</li>
                    <li>Involve family and support networks.</li>
                    <li>Stay informed and seek guidance.</li>
                </ul></div>
            </div>
        {% elif result == "Diabetes, Type 2" %}
            <div class="condition-info animate__animated animate__fadeInUp">
                <div class="info"><h3>Information about Diabetes, Type 2:</h3>
                 <p>A chronic condition affecting how your body processes sugar due to insulin
resistance or deficiency.</p>
            </div><div class="precautions"><h3>Precautions and Measures:</h3>
                <ul>
                    <li>Adopt a balanced diet.</li>
                    <li>Maintain a healthy weight.</li>
                    <li>Engage in regular physical activity.</li>
                    <li>Monitor blood sugar levels.</li>
                    <li>Take medications as prescribed.</li>
                    <li>Manage stress and stay hydrated.</li>
                    <li>Quit smoking.</li>
                    <li>Regular check-ups and foot care.</li>
                    <li>Limit alcohol and sugar intake.</li>
                </ul></div>
            </div>
        {% elif result == "High Blood Pressure" %}
            <div class="condition-info animate__animated animate__fadeInUp">
                <div class="info"><h3>Information about High Blood Pressure:</h3>
                    <p>A condition where blood pressure in the arteries is consistently elevated,
increasing risk of heart disease.</p>
            </div><div class="precautions"><h3>Precautions and Measures:</h3>
                <ul>
                    <li>Eat a heart-healthy diet.</li>
                    <li>Maintain a healthy weight.</li>
                    <li>Exercise regularly.</li>
                    <li>Limit alcohol and caffeine.</li>
                    <li>Quit smoking and manage stress.</li>
                    <li>Monitor blood pressure regularly.</li>
                    <li>Take medications as prescribed.</li>
                    <li>Ensure quality sleep.</li>
                </ul> </div> </div>
            {% endif %}
        {% endif %}
    </div>
    <div class="content">
        <h3><b>Welcome to MedSymphony</b></h3>
        <h1>HARMONY OF CONDITIONS AND PREDICTIONS</h1>
         <p>MedSymphony specializes in advanced hormonal health solutions, offering precise
diagnosis and
            effective medication for balanced well-being.</p>
    </div>
</body>
</html>
```

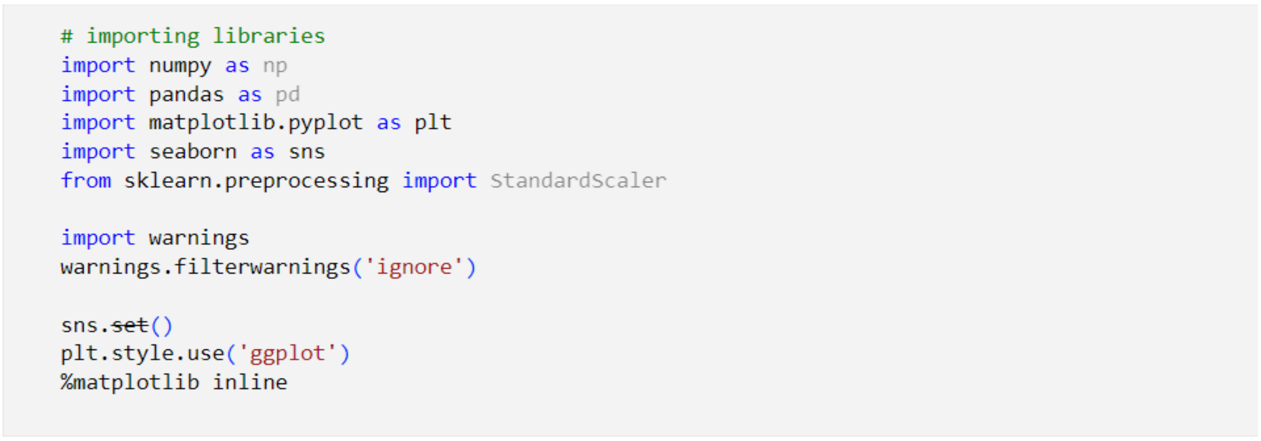
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Harmony of Conditions and Medications</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
```

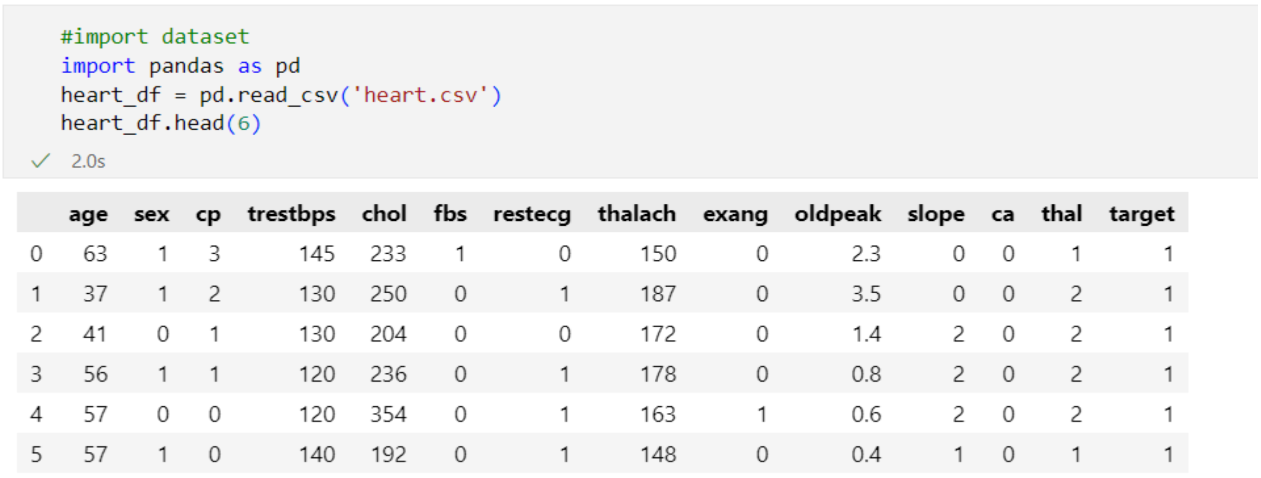
```
  <div class="container">
            <h2 class="text-center text-primary mt-4 text-black"><b>HARMONY OF
CONDITIONS AND PREDICTIONS</b></h2>
    <div class="d-flex justify-content-between align-items-center mt-4">
      <a href="{{ url_for('predict') }}" class="btn btn-danger">Logout</a>
    </div>
    <form action="{{ url_for('index') }}" method="GET" class="mb-3">
      <div class="input-group">
        <input type="text" class="form-control" name="condition" placeholder="Search by
Condition..." value="{{ request.args.get('condition', '') }}">
        <button type="submit" class="btn btn-primary">Search</button>
      </div>
    </form>
      <a href="{{ url_for('add_review') }}" class="btn btn-success mb-3">Add New
Review</a>
    <table class="table table-striped table-bordered">
      <thead>
        <tr><center><b>
          <th>Patient ID</th>
          <th>Drug Name</th>
          <th>Condition</th>
          <th>Review</th>
          <th>Rating</th>
          <th>Date</th>
          <th>Useful Count</th>

          <th>Actions</th></b></center>

        </tr>

      </thead>

      <tbody>

        {% for review in reviews %}

        <tr>

          <td>{{ review[0] }}</td>

          <td>{{ review[1] }}</td>

          <td>{{ review[2] }}</td>

          <td>{{ review[3] }}</td>

          <td>{{ review[4] }}</td>

          <td>{{ review[5] }}</td>

          <td>{{ review[6] }}</td>

          <td>

              <a href="{{ url_for('edit_review', patientID=review[0]) }}" class="btn btn-
  warning btn-sm">Edit</a>

             <a href="{{ url_for('delete_review', patientID=review[0]) }}" class="btn btn-
  danger btn-sm" onclick="return confirm('Are you sure?')">Delete</a>

          </td>

        </tr>

        {% endfor %}

      </tbody>
```

```
        </table>

    </div>

</body>

</html>
```

10.2. HEART DISEASE PREDICTION



```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')

sns.~~set~~()
plt.style.use('ggplot')
%matplotlib inline
```



```
#import dataset
import pandas as pd
heart_df = pd.read_csv('heart.csv')
heart_df.head(6)
```

✓ 2.0s

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

```python
# information about the dataset
heart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```python
#description about dataset
heart_df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```python
#Plotting the distribution plot.
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(20,25))
plotnumber=1

for column in heart_df:
    if plotnumber<14:
        ax=plt.subplot(4,4,plotnumber)
        sns.distplot(heart_df[column])
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Values',fontsize=20)
    plotnumber+=1
plt.show()
```

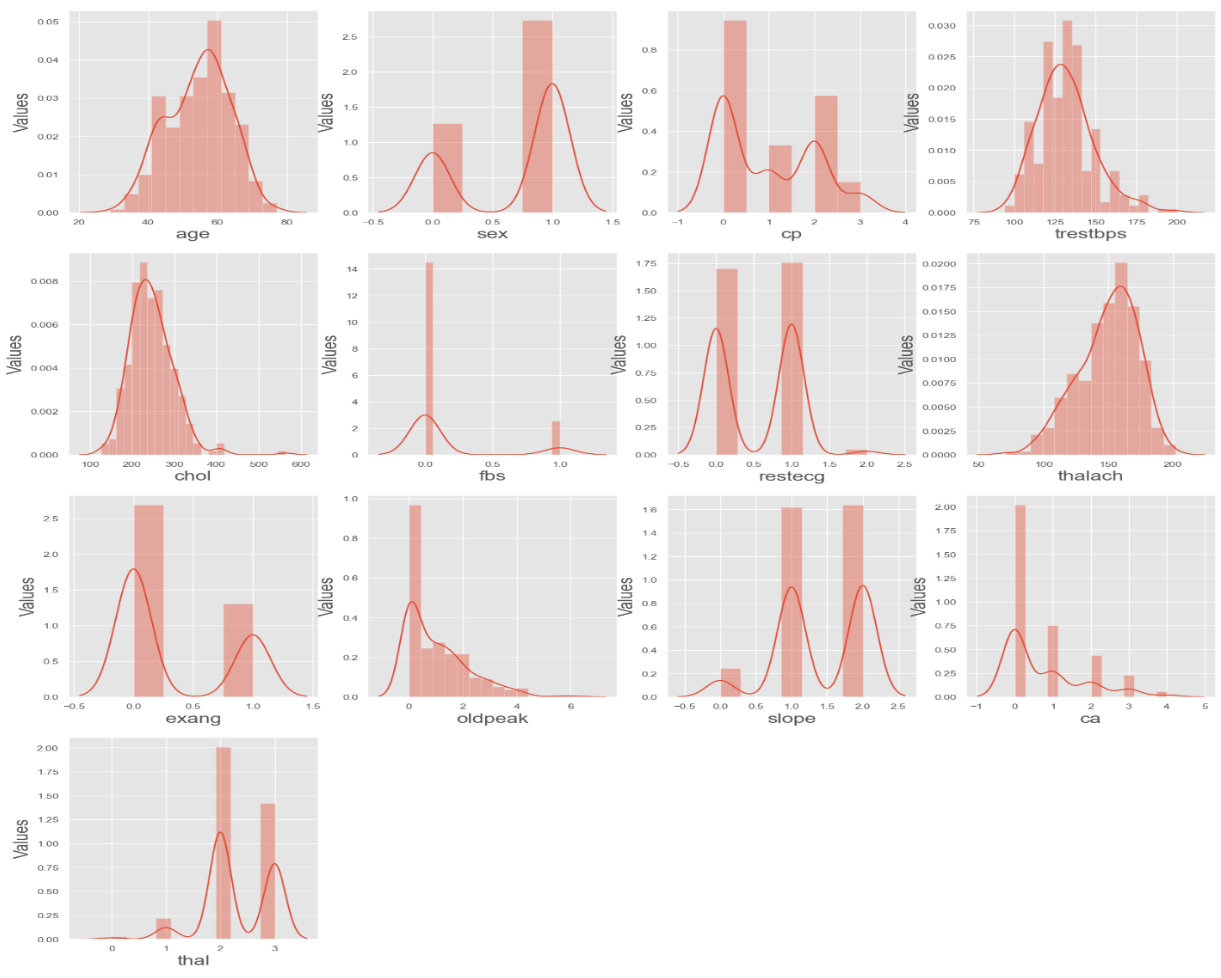


Figure 10.2.1. Plotting the Distribution Plot

```
#Correlation matrix
import numpy as np
plt.figure(figsize = (16, 8))

corr = heart_df.corr()
mask = np.triu(np.ones_like(corr, dtype = bool))
sns.heatmap(corr, mask = mask, annot = True, fmt = '.2g', linewidths = 1)
plt.show()
```

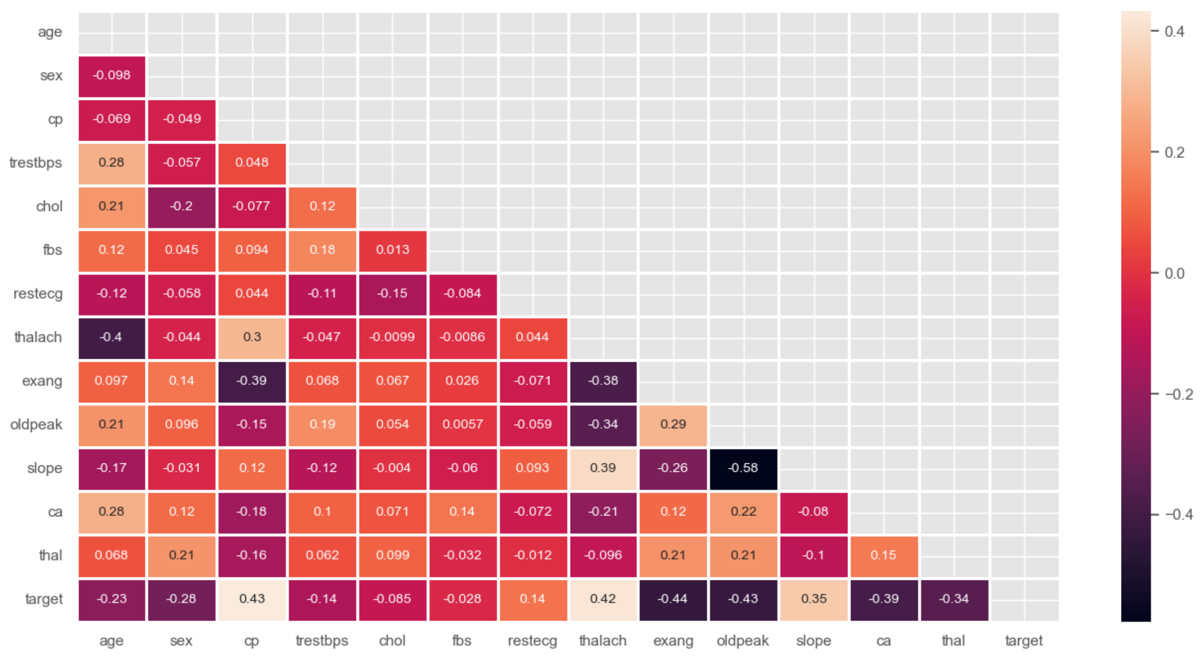


Figure 10.2.2. Correlation Matrix

```python
accuracies={}

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
lr = LogisticRegression(penalty='l2')
lr.fit(x_train,y_train)

y_pred = lr.predict(x_test)

acc=accuracy_score(y_test,y_pred)
accuracies['LR']=acc*100
print("Training accuracy score of the model is:",accuracy_score(y_train, lr.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred)*100,"%")
```

```
Training accuracy score of the model is: 85.37735849056604 %
Testing accuracy score of the model is: 80.21978021978022 %
```

```python
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred))

print("Classification Report",classification_report(y_test,y_pred))
```

```
Confusion matrix of the model [[32 12]
 [ 6 41]]
Classification Report               precision    recall  f1-score   support

           0       0.84      0.73      0.78        44
           1       0.77      0.87      0.82        47

    accuracy                           0.80        91
   macro avg       0.81      0.80      0.80        91
weighted avg       0.81      0.80      0.80        91
```

```
from sklearn.svm import SVC

svc = SVC(probability=True)
svc.fit(x_train, y_train)

y_pred2 = svc.predict(x_test)

acc2=accuracy_score(y_test,y_pred2)
accuracies['SVM']=acc2*100

print("Training accuracy score of the model is:",accuracy_score(y_train, svc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred2)*100,"%")
```

```
Training accuracy score of the model is: 55.660377358490564 %
Testing accuracy score of the model is: 51.64835164835166 %
```

```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred2))

print("Classification Report",classification_report(y_test,y_pred2))
```

```
Confusion matrix of the model [[ 0 44]
 [ 0 47]]
Classification Report               precision    recall  f1-score   support

           0       0.00      0.00      0.00        44
           1       0.52      1.00      0.68        47

    accuracy                           0.52        91
   macro avg       0.26      0.50      0.34        91
weighted avg       0.27      0.52      0.35        91
```

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)

y_pred3 = dtc.predict(x_test)

acc3=accuracy_score(y_test,y_pred3)
accuracies['DT']=acc3*100

print("Training accuracy score of the model is:",accuracy_score(y_train, dtc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred3)*100,"%")
```

```
Training accuracy score of the model is: 100.0 %
Testing accuracy score of the model is: 71.42857142857143 %
```

```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred3))

print("Classification Report",classification_report(y_test,y_pred3))
```

```
Confusion matrix of the model [[33 11]
 [15 32]]
Classification Report               precision    recall  f1-score   support

           0       0.69      0.75      0.72        44
           1       0.74      0.68      0.71        47

    accuracy                           0.71        91
   macro avg       0.72      0.72      0.71        91
weighted avg       0.72      0.71      0.71        91
```

```
dtc2 = DecisionTreeClassifier(criterion= 'entropy', max_depth= 12, min_samples_leaf= 1, min_samples_split= 2, splitter= 'random')
dtc2.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=12, splitter='random')
```

```
y_pred4 = dtc2.predict(x_test)
acc4=accuracy_score(y_test,y_pred4)
accuracies['DT2']=acc4*100

print("Training accuracy score of the model is:",accuracy_score(y_train, dtc2.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred4)*100,"%")
```

```
Training accuracy score of the model is: 99.52830188679245 %
Testing accuracy score of the model is: 73.62637362637363 %
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=8)

knn.fit(x_train,y_train)

y_pred1 = knn.predict(x_test)

acc1=accuracy_score(y_test,y_pred1)
accuracies['KNN']=acc1*100

print("Training accuracy score of the model is:",accuracy_score(y_train, knn.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred1)*100,"%")
```

```
Training accuracy score of the model is: 85.84905660377359 %
Testing accuracy score of the model is: 75.82417582417582 %
```

```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred1))

print("Classification Report",classification_report(y_test,y_pred1))
```

```
Confusion matrix of the model [[29 15]
 [ 7 40]]
Classification Report               precision    recall  f1-score   support

           0       0.81      0.66      0.72        44
           1       0.73      0.85      0.78        47

    accuracy                           0.76        91
   macro avg       0.77      0.76      0.75        91
weighted avg       0.77      0.76      0.76        91
```

```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred4))

print("Classification Report",classification_report(y_test,y_pred4))
```

```
Confusion matrix of the model [[32 12]
 [12 35]]
Classification Report               precision    recall  f1-score   support

           0       0.73      0.73      0.73        44
           1       0.74      0.74      0.74        47

    accuracy                           0.74        91
   macro avg       0.74      0.74      0.74        91
weighted avg       0.74      0.74      0.74        91
```

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(criterion = 'gini', max_depth = 7, max_features = 'sqrt', min_samples_leaf = 2, min_samples_split = 4, n_estimators = 180)
rfc.fit(x_train, y_train)

y_pred5 = rfc.predict(x_test)

acc5=accuracy_score(y_test,y_pred5)
accuracies['RF']=acc5*100

print("Training accuracy score of the model is:",accuracy_score(y_train, rfc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred5)*100,"%")
```

```
Training accuracy score of the model is: 97.16981132075472 %
Testing accuracy score of the model is: 82.41758241758241 %
```

+ Code   + Markdown

```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred5))

print("Classification Report",classification_report(y_test,y_pred5))
```

```
Confusion matrix of the model [[32 12]
 [ 4 43]]
Classification Report               precision    recall  f1-score   support

           0       0.89      0.73      0.80        44
           1       0.78      0.91      0.84        47

    accuracy                           0.82        91
   macro avg       0.84      0.82      0.82        91
weighted avg       0.83      0.82      0.82        91
```

```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

gbc = GradientBoostingClassifier(learning_rate = 0.05, loss = 'log_loss', n_estimators = 180)
gbc.fit(x_train, y_train)

y_pred6 = gbc.predict(x_test)

acc6 = accuracy_score(y_test,y_pred6)
accuracies['GradientBoosting']=acc6*100

print("Training accuracy score of the model is:",accuracy_score(y_train, gbc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred6)*100,"%")
```

```
Training accuracy score of the model is: 100.0 %
Testing accuracy score of the model is: 79.12087912087912 %
```

```
colors = ["purple", "green", "orange", "magenta","blue","black"]

# sns.set_style("whitegrid")
plt.figure(figsize=(16,8))
plt.yticks(np.arange(0,1200,10))
plt.ylabel("Accuracy %")
plt.xlabel("Algorithms")
sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()), palette=colors )
plt.show()
```

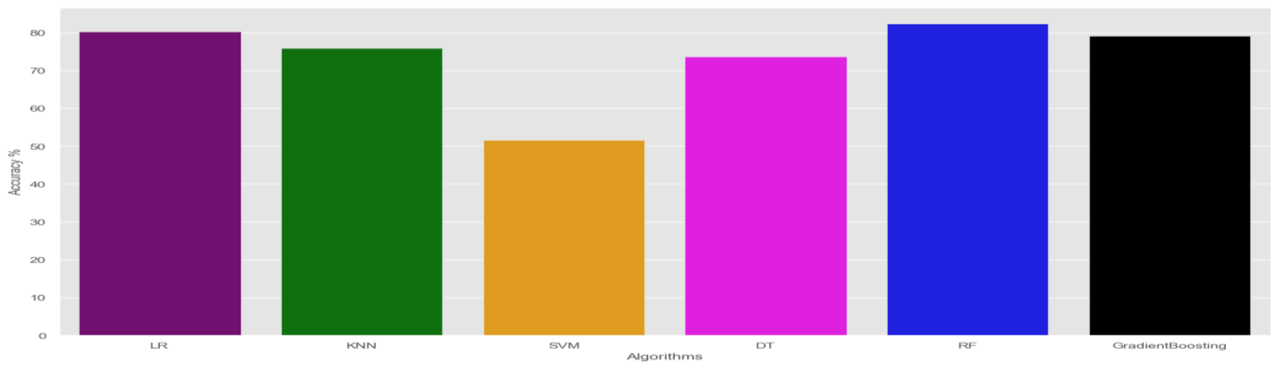
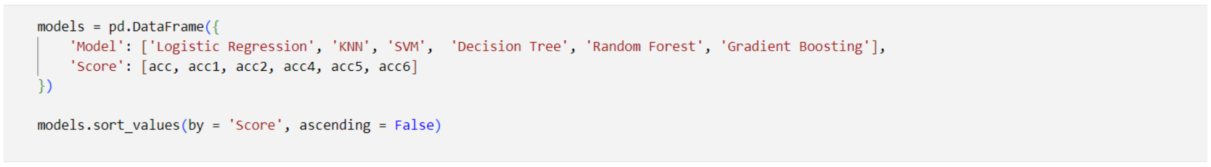
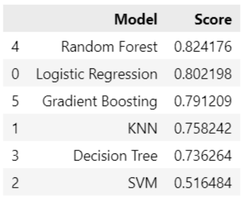


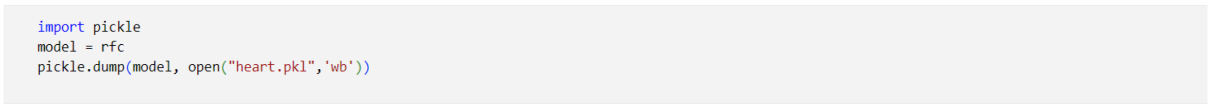
Figure 10.2.3. Bar plot using Colors



```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVM',  'Decision Tree', 'Random Forest', 'Gradient Boosting'],
    'Score': [acc, acc1, acc2, acc4, acc5, acc6]
})

models.sort_values(by = 'Score', ascending = False)
```



	Model	Score
4	Random Forest	0.824176
0	Logistic Regression	0.802198
5	Gradient Boosting	0.791209
1	KNN	0.758242
3	Decision Tree	0.736264
2	SVM	0.516484



```
import pickle
model = rfc
pickle.dump(model, open("heart.pkl",'wb'))
```

ROC Heart Disease Prediction

```
from sklearn import metrics

plt.figure(figsize=(8,5))

models = [

{

    'label': 'LR',

    'model': lr,

},

{

    'label': 'DT',

    'model': dtc2,

},

{

    'label': 'SVM',

    'model': svc,

},
{
    'label': 'KNN',
    'model': knn,
},
{
    'label': 'RF',
    'model': rfc,
},
{
    'label': 'GBDT',
    'model': gbc,
}
]
for m in models:
    model = m['model']
    model.fit(x_train, y_train)
    y_pred=model.predict(x_test)
    fpr1, tpr1, thresholds = metrics.roc_curve(y_test, model.predict_proba(x_test)[:,1])
    auc = metrics.roc_auc_score(y_test,model.predict(x_test))
    plt.plot(fpr1, tpr1, label='%s - ROC (area = %0.2f)' % (m['label'], auc))
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1 - Specificity (False Positive Rate)', fontsize=12)
plt.ylabel('Sensitivity (True Positive Rate)', fontsize=12)

plt.title('ROC - Heart Disease Prediction', fontsize=12)

plt.legend(loc="lower right", fontsize=12)

plt.savefig("roc_heart.jpeg", format='jpeg', dpi=400, bbox_inches='tight')

plt.show()
```

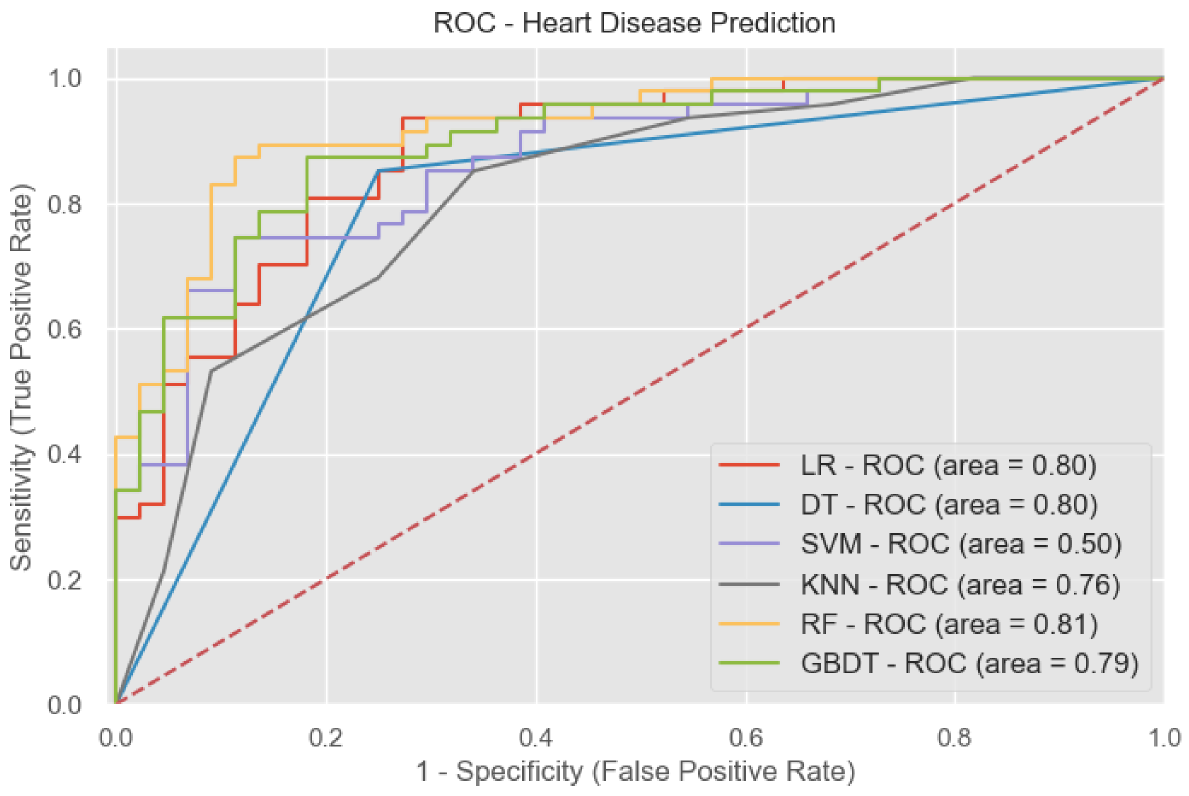


Figure 10.2.4. ROC- Heart Disease Prediction

Performance Evaluation – Heart Disease Prediction

```
from sklearn import metrics

import numpy as np

import matplotlib.pyplot as plt

# Define models

models = [
    {'label': 'LR', 'model': lr},
    {'label': 'DT', 'model': dtc2},
    {'label': 'SVM', 'model': svc},
    {'label': 'KNN', 'model': knn},
    {'label': 'RF', 'model': rfc},
    {'label': 'GBDT', 'model': gbc}
]
# Accuracy values
means_accuracy = [
    100 * round(acc, 4), 100 * round(acc4, 4), 100 * round(acc2, 4),
    100 * round(acc1, 4), 100 * round(acc5, 4), 100 * round(acc6, 4)
]
# Compute ROC AUC scores
means_roc = []
for m in models:
    model = m['model']
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    fpr1, tpr1, thresholds = metrics.roc_curve(y_test, model.predict_proba(x_test)[:, 1])
    auc = metrics.roc_auc_score(y_test, model.predict(x_test))
    means_roc.append(100 * round(auc, 4))
# Ensure both lists have the same length
n_groups = len(models)  # Fix shape mismatch
means_accuracy = tuple(means_accuracy)
```

```
means_roc = tuple(means_roc)
# Create plot
fig, ax = plt.subplots(figsize=(8, 5))
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8
rects1 = plt.bar(index, means_accuracy, bar_width,
                alpha=opacity, color='mediumpurple', label='Accuracy (%)')
rects2 = plt.bar(index + bar_width, means_roc, bar_width,
                alpha=opacity, color='rebeccapurple', label='ROC (%)')
plt.xlim([-1, n_groups])
plt.ylim([70, 105])
plt.title('Performance Evaluation - Heart Disease Prediction', fontsize=12)
plt.xticks(index + bar_width / 2, ('LR', 'DT', 'SVM', 'KNN', 'RF', 'GBDT'),
           rotation=40, ha='center', fontsize=12)
plt.legend(loc="upper right", fontsize=10)
plt.savefig("PE_heart.jpeg", format='jpeg', dpi=400, bbox_inches='tight')
plt.show()
```

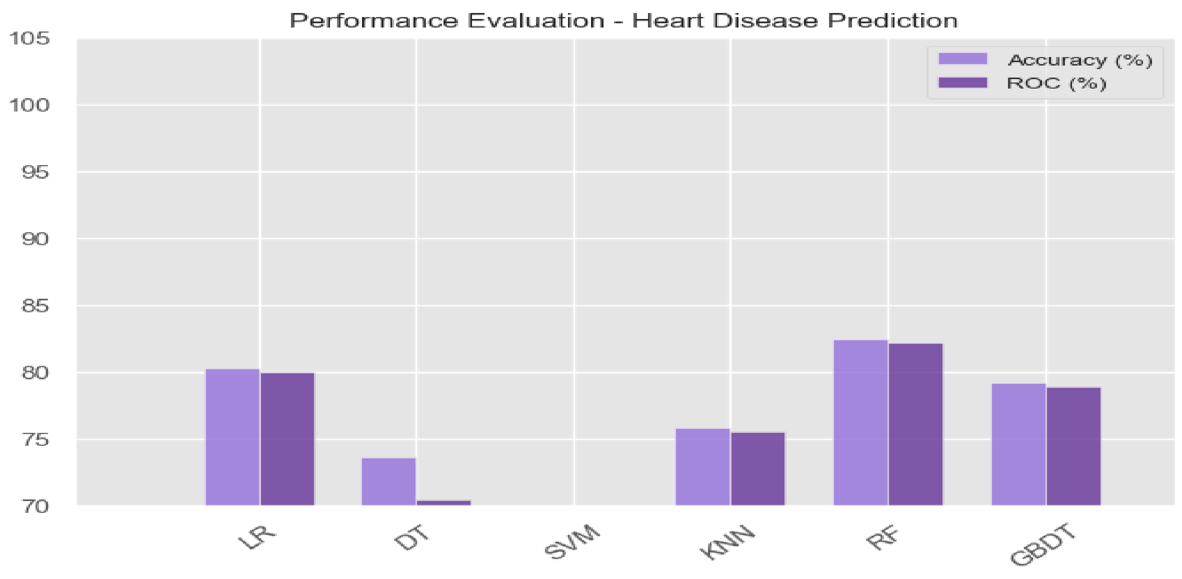


Figure 10.2.5. Performance Evaluation- Heart Disease Prediction

10.3. HARMONY CONDITIONS AND PREDICTIONS

Importing libraries

```
import pandas as pd # data preprocessing
import itertools # confusion matrix
import string
import numpy as np
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_rows', None)

import pandas as pd
df=pd.read_csv('data/drugsComTrain_raw.tsv', sep='\t')

df.to_csv('data/drugsComTrain.csv',index=False)
```

```
X = df_train.drop(['Unnamed: 0','drugName','rating','date','usefulCount'],axis=1)
```

EDA

```
# segregating dataframe for analyzing individual condition
X_birth=X[(X['condition']=='Birth Control')]
X_dep=X[(X['condition']=='Depression')]
X_bp=X[(X['condition']=='High Blood Pressure')]
X_diab=X[(X['condition']=='Diabetes, Type 2')]

for i, col in enumerate(X.columns):
    X.iloc[:, i] = X.iloc[:, i].str.replace('"', '')

X.head()
```

```
        condition                                            review
1            ADHD  My son is halfway through his fourth week of I...
2   Birth Control  I used to take another oral contraceptive, whi...
3   Birth Control  This is my first time using any form of birth ...
7  Bipolar Disorde  Abilify changed my life. There is hope. I was ...
9   Birth Control  I had been on the pill for many years. When my...
```

Stopwords

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

Lemmitization

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
porter = PorterStemmer()
lemmatizer = WordNetLemmatizer()

print(porter.stem("sportingly"))
print(porter.stem("very"))
print(porter.stem("troubled"))
```

```
sportingli
veri
troubl
```

```
import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\rajar\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
True
```

```
print(lemmatizer.lemmatize("sportingly"))
print(lemmatizer.lemmatize("very"))
print(lemmatizer.lemmatize("troubled"))
```

```
sportingly
very
troubled
```

```
from bs4 import BeautifulSoup
import re
```

```
def review_to_words(raw_review):
    # 1. Delete HTML
    review_text = BeautifulSoup(raw_review, 'html.parser').get_text()
    # 2. Make a space
    letters_only = re.sub('[^a-zA-Z]', ' ', review_text)
    # 3. lower letters
    words = letters_only.lower().split()
    # 5. Stopwords
    meaningful_words = [w for w in words if not w in stop]
    # 6. lemmitization
    lemmitize_words = [lemmatizer.lemmatize(w) for w in meaningful_words]
    # 7. space join words
    return( ' '.join(lemmitize_words))
```

```
X['review_clean'] = X['review'].apply(review_to_words)
  review_text = BeautifulSoup(raw_review, 'html.parser').get_text()
```

Creating features and Target Variable

```
X_feat=X['review_clean']
y=X['condition']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_feat, y,stratify=y,test_size=0.2,
random_state=0)

import matplotlib.pyplot as plt
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    See full source and example:
    http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Bag of Words

```
from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train)
count_test = count_vectorizer.transform(X_test)
```

```
count_train
```

```
<49910x20925 sparse matrix of type '<class 'numpy.int64'>'
        with 1535078 stored elements in Compressed Sparse Row format>
```

Machine Learning Model : Passive Aggressive Classifier

```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
# Train the classifier
passive = PassiveAggressiveClassifier()
passive.fit(count_train, y_train)
# Make predictions
pred = passive.predict(count_test)
# Calculate accuracy
score = metrics.accuracy_score(y_test, pred)
print("Accuracy: %0.3f" % score)
```

```
# Compute confusion matrix
cm = metrics.confusion_matrix(y_test, pred, labels=['Birth Control',
'Depression','Pain','Anxiety','Bipolar Disorde','ADHD','Diabetes, Type 2','High Blood
Pressure'])

# Plot confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Birth Control',
'Depression','Pain','Anxiety','Bipolar Disorde','ADHD','Diabetes, Type 2','High Blood
Pressure'],
            yticklabels=['Birth Control', 'Depression','Pain','Anxiety','Bipolar
Disorde','ADHD','Diabetes, Type 2','High Blood Pressure'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.905
```

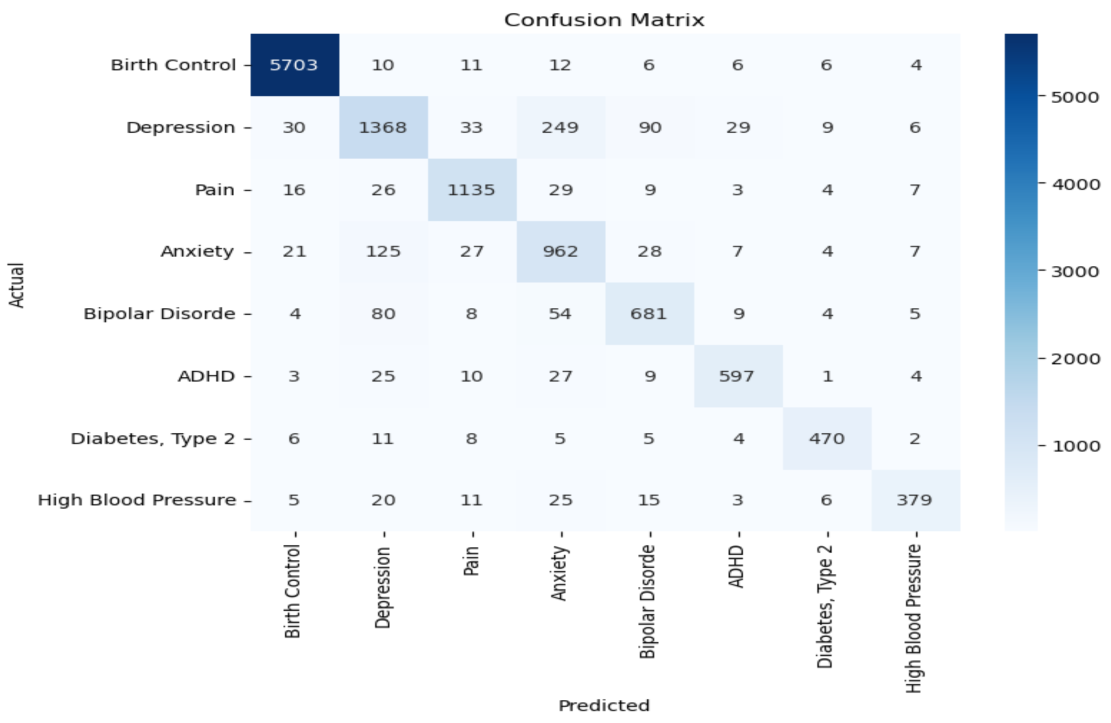


Figure 10.3.1. Confusion Matrix using Passive Aggressive Classifier

TFIDF

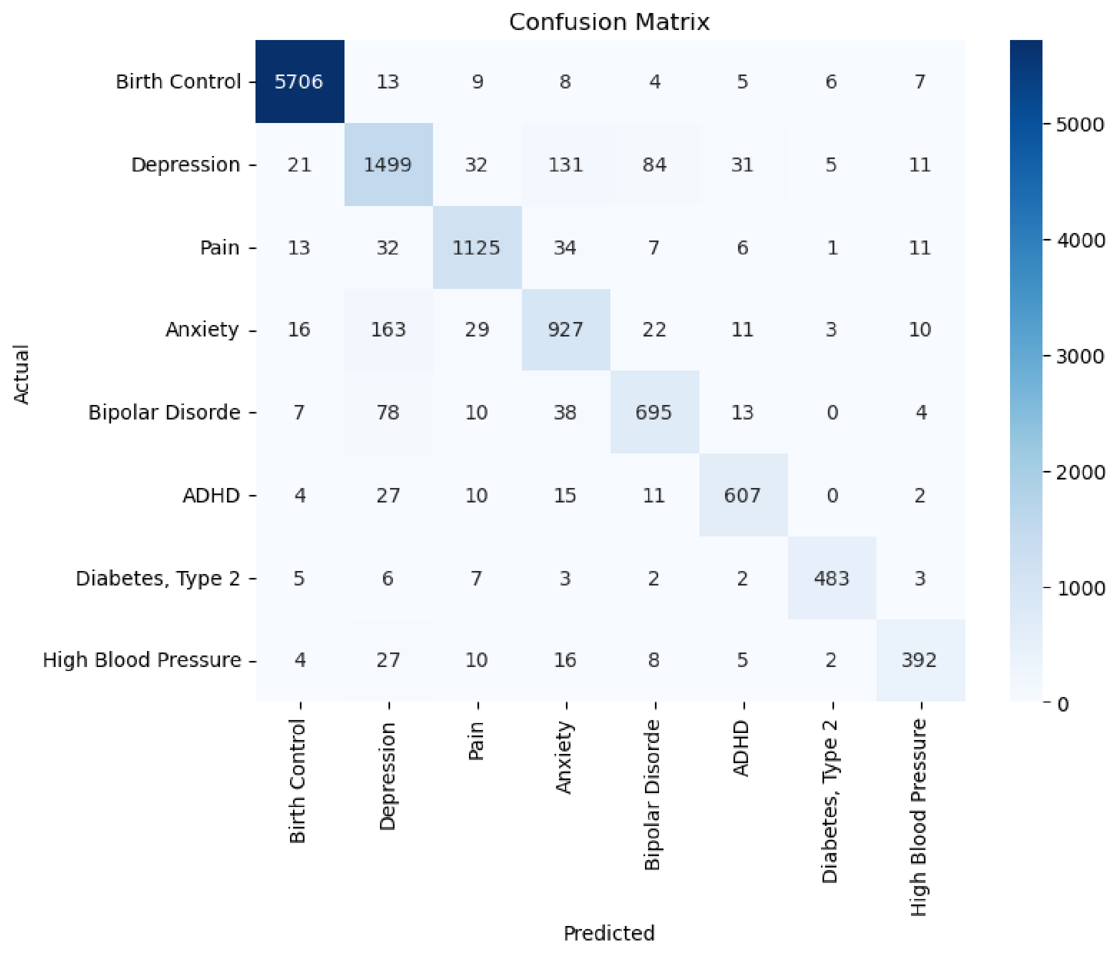
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.8)
tfidf_train_2 = tfidf_vectorizer.fit_transform(X_train)
tfidf_test_2 = tfidf_vectorizer.transform(X_test)
```

Machine Learning Model TFIDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns

# Vectorize text data
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.8)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)

# Train the Passive Aggressive Classifier
pass_tf = PassiveAggressiveClassifier()
pass_tf.fit(tfidf_train, y_train)

# Make predictions
pred = pass_tf.predict(tfidf_test)

# Calculate accuracy
score = metrics.accuracy_score(y_test, pred)
print("Accuracy: %0.3f" % score)

# Compute confusion matrix
labels = ['Birth Control', 'Depression', 'Pain', 'Anxiety', 'Bipolar Disorde', 'ADHD', 'Diabetes,
Type 2', 'High Blood Pressure']
cm = metrics.confusion_matrix(y_test, pred, labels=labels)
```

```
# Plot confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.916
```



	Birth Control	Depression	Pain	Anxiety	Bipolar Disorde	ADHD	Diabetes, Type 2	High Blood Pressure
Birth Control	5706	13	9	8	4	5	6	7
Depression	21	1499	32	131	84	31	5	11
Pain	13	32	1125	34	7	6	1	11
Anxiety	16	163	29	927	22	11	3	10
Bipolar Disorde	7	78	10	38	695	13	0	4
ADHD	4	27	10	15	11	607	0	2
Diabetes, Type 2	5	6	7	3	2	2	483	3
High Blood Pressure	4	27	10	16	8	5	2	392

Figure 10.3.2. Confusion Matrix using TFIDF

TFIDF: Bigrams

```
tfidf_vectorizer2 = TfidfVectorizer(stop_words='english', max_df=0.8, ngram_range=(1,2))
tfidf_train_2 = tfidf_vectorizer2.fit_transform(X_train)
tfidf_test_2 = tfidf_vectorizer2.transform(X_test)
```

```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
# Train the Passive Aggressive Classifier
pass_tf = PassiveAggressiveClassifier()
pass_tf.fit(tfidf_train_2, y_train)
# Make predictions
pred = pass_tf.predict(tfidf_test_2)
# Calculate accuracy
score = metrics.accuracy_score(y_test, pred)
print("Accuracy: %0.3f" % score)
# Compute confusion matrix
labels = ['Birth Control', 'Depression', 'Pain', 'Anxiety', 'Bipolar Disorde', 'ADHD', 'Diabetes,
Type 2', 'High Blood Pressure']
cm = metrics.confusion_matrix(y_test, pred, labels=labels)
# Plot confusion matrix using Seaborn
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.937
```

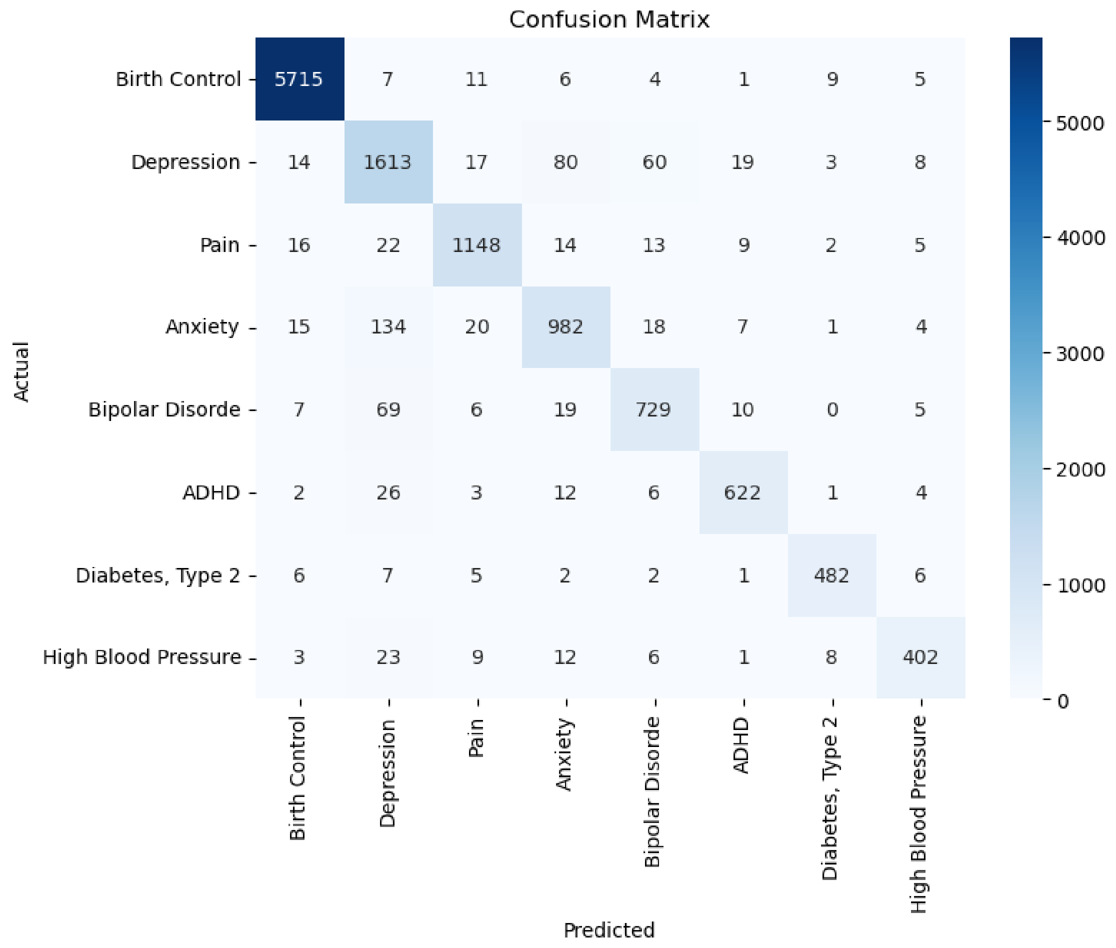


Figure 10.3.3. Confusion Matrix using TFIDF-Bigrams

TFIDF : Trigrams

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
# TF-IDF Vectorization with n-grams (1,3)
tfidf_vectorizer3 = TfidfVectorizer(stop_words='english', max_df=0.8, ngram_range=(1,3))
tfidf_train_3 = tfidf_vectorizer3.fit_transform(X_train)
tfidf_test_3 = tfidf_vectorizer3.transform(X_test)
# Train Passive Aggressive Classifier
pass_tf = PassiveAggressiveClassifier()
pass_tf.fit(tfidf_train_3, y_train)
```

```
# Make predictions
pred = pass_tf.predict(tfidf_test_3)
# Calculate accuracy
score = metrics.accuracy_score(y_test, pred)
print("Accuracy: %0.3f" % score)
# Compute confusion matrix
labels = ['Birth Control', 'Depression', 'Pain', 'Anxiety', 'Bipolar Disorde', 'ADHD', 'Diabetes,
Type 2', 'High Blood Pressure']
cm = metrics.confusion_matrix(y_test, pred, labels=labels)
# Plot confusion matrix using Seaborn
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.939
```

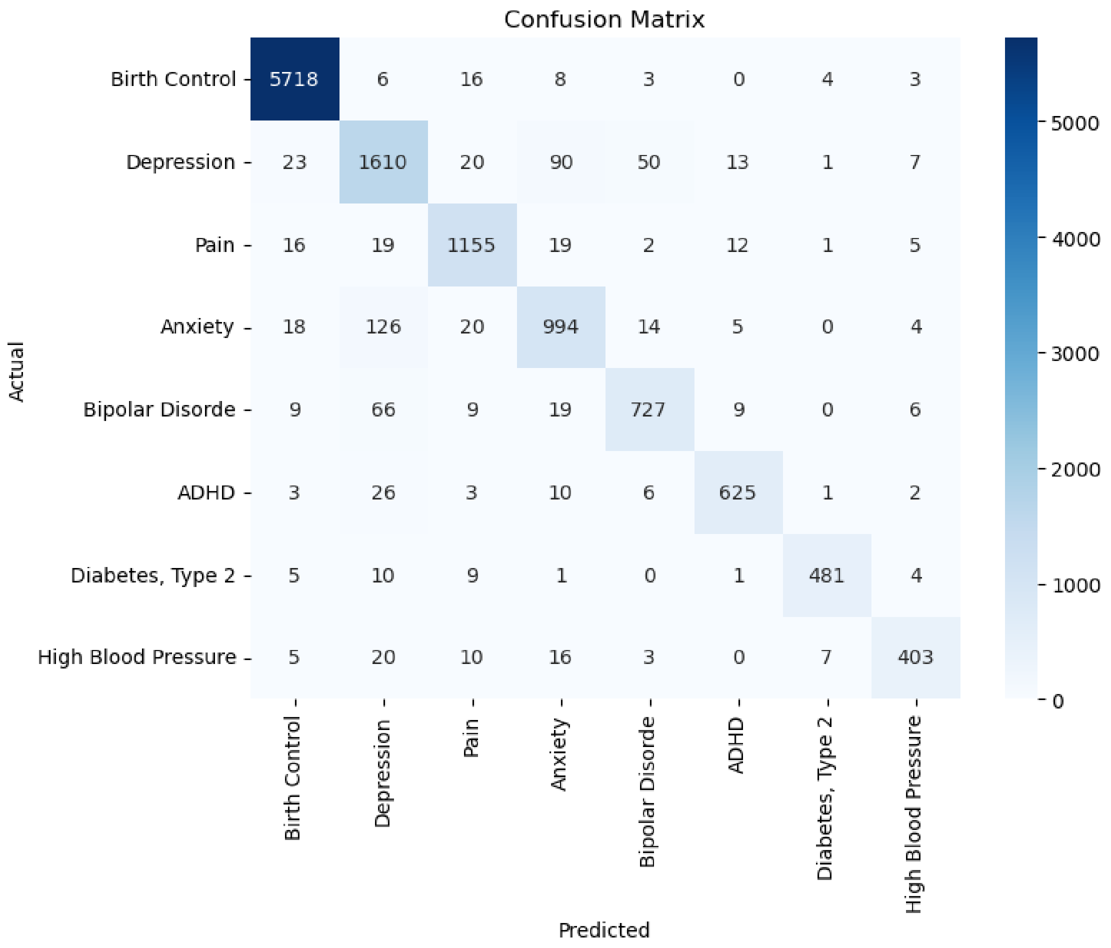


Figure 10.3.4. Confusion Matrix using TFIDF- Trigram

Most important Features

```
def most_informative_feature_for_class(vectorizer, classifier, classlabel, n=10):
    labelid = list(classifier.classes_).index(classlabel)
    feature_names = vectorizer.get_feature_names_out()
    topn = sorted(zip(classifier.coef_[labelid], feature_names))[-n:]
    for coef, feat in topn:
        print (classlabel, feat, coef)
most_informative_feature_for_class(tfidf_vectorizer2, pass_tf, 'Birth Control')
```

```
Birth Control sleepiness noticed 3.3172998089336994
Birth Control kick habit 4.211012222230601
Birth Control favorite stopped 5.309167539446232
Birth Control little typically 5.55527507966304
Birth Control decision yr 5.803387698777122
Birth Control like deer 6.446130396386882
Birth Control effect anorexia 7.529165784649538
Birth Control drive stopped 7.73620072815385
Birth Control drive roof 8.01967744129424
Birth Control advise beginning 8.618121904144195
```

```
most_informative_feature_for_class(tfidf_vectorizer2, pass_tf, 'Depression')
```

```
Depression wean cold 2.981413285167493
Depression managed mood 3.135310256269496
Depression nuvaring patch 3.2675096986789782
Depression similar time 3.5955634393097764
Depression normal good 4.123108746880271
Depression heavy far 4.621315744383492
Depression good fibromyalgia 5.766497740988167
Depression bone sinus 6.080340570118933
Depression felt keel 6.337434155474702
Depression nuvaring amazing 11.766861362819565
```

```
most_informative_feature_for_class(tfidf_vectorizer2, pass_tf, 'High Blood Pressure')
```

```
High Blood Pressure lexapro depressed 2.490646751524033
High Blood Pressure crampa 3.2647885220528186
High Blood Pressure coming scalp 3.264865953647025
High Blood Pressure personal family 3.5854357225143287
High Blood Pressure eng 4.013970034794106
High Blood Pressure food fluid 4.336203932636273
High Blood Pressure difficult reason 4.3810371407695845
High Blood Pressure bit forgetful 4.443396016263653
High Blood Pressure level twice 6.07907093769977
High Blood Pressure extra needle 7.612973761627494
```

```
most_informative_feature_for_class(tfidf_vectorizer2, pass_tf, 'Diabetes, Type 2')
```

```
Diabetes, Type 2 painless light 2.1697242443304154
Diabetes, Type 2 short overall 2.428518336565371
Diabetes, Type 2 avascular necroses 2.5245077237263205
Diabetes, Type 2 ticked 2.538421658252425
Diabetes, Type 2 time encourage 2.883264753279341
Diabetes, Type 2 amazed high 2.9761233316049664
Diabetes, Type 2 enthusiasm changed 3.3276013088302343
Diabetes, Type 2 pain exertion 4.058256796958472
Diabetes, Type 2 follow hope 4.2920860835373515
Diabetes, Type 2 following normal 4.582726808101745
```

Sample Predictions

```
X.tail()

            condition  \
161283    Bipolar Disorde
161286        Depression
161287           Anxiety
161290  High Blood Pressure
161291      Birth Control

                                              review  \
161283  I was in a very bad place at the time I starte...
161286  This is the third med I&#039;ve tried for anxi...
161287  I was super against taking medication. I&#039;...
161290  I have only been on Tekturna for 9 days. The e...
161291  This would be my second month on Junel. I&#039...

                                        review_clean
161283  bad place time started taking doctor wanted we...
161286  third med tried anxiety mild depression week h...
161287  super taking medication started dealing anxiet...
161290  tekturna day effect immediate also calcium cha...
161291  would second month junel birth control year ch...
```

```
## Function for Extracting Top drugs

def top_drugs_extractor(condition):
    df_top = df[(df['rating']>=9)&(df['usefulCount']>=100)].sort_values(by = ['rating',
'usefulCount'], ascending = [False, False])
    drug_lst = df_top[df_top['condition']==condition]['drugName'].head(3).tolist()
    return drug_lst

def predict_text(lst_text):
    df_test = pd.DataFrame(lst_text, columns = ['test_sent'])
    df_test["test_sent"] = df_test["test_sent"].apply(review_to_words)
    tfidf_bigram = tfidf_vectorizer3.transform(lst_text)
    prediction = pass_tf.predict(tfidf_bigram)
    df_test['prediction']=prediction
    return df_test

sentences = [
  "I have only been on Tekturna for 9 days. The effect was immediate. I am also on a calcium
channel blocker (Tiazac) and hydrochlorothiazide. I was put on Tekturna because of
palpitations experienced with Diovan (ugly drug in my opinion, same company produces
both however). The palpitations were pretty bad on Diovan, 24 hour monitor by EKG etc.
After a few days of substituting Tekturna for Diovan, there are no more palpitations.",
    "son halfway fourth week intuniv became concerned began last week started taking highest
dose two day could hardly get bed cranky slept nearly hour drive home school vacation
unusual called doctor monday morning said stick day see school getting morning last two day
problem free much agreeable ever le emotional good thing le cranky remembering thing
overall behavior better tried many different medication far effective",
    "I just got diagnosed with type 2. My doctor prescribed Invokana and metformin from the
beginning. My sugars went down to normal by the second week. I am losing so much weight.
No side effects yet. Miracle medicine for me",


  ]

tfidf_trigram = tfidf_vectorizer3.transform(sentences)
```

```
predictions = pass_tf.predict(tfidf_trigram)

for text, label in zip(sentences, predictions):
    if label=="High Blood Pressure":
        target="High Blood Pressure"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="Depression":
        target="Depression"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="ADHD":
        target="ADHD"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="Diabetes, Type 2":
        target="Diabetes, Type 2"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="Bipolar Disorde":
        target="Bipolar Disorde"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="Anxiety":
        target="Anxiety"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    elif label=="Pain":
        target="Pain"
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
    else:
        target="Birth Control"
        print("text:", text, "\Condition:", target)
        top_drugs = top_drugs_extractor(label)
        print("text:", text, "\nCondition:", target)
        print("Top 3 Suggested Drugs:")
        print(top_drugs[0])
        print(top_drugs[1])
        print(top_drugs[2])
        print()
```

```
<>:72: SyntaxWarning: invalid escape sequence '\C'
<>:72: SyntaxWarning: invalid escape sequence '\C'
C:\Users\rajar\AppData\Local\Temp\ipykernel_13472\727699231.py:72: SyntaxWarning:
invalid escape sequence '\C'
  print("text:", text, "\Condition:", target)
```

```
text: I have only been on Tekturna for 9 days. The effect was immediate. I am also on a
calcium channel blocker (Tiazac) and hydrochlorothiazide. I was put on Tekturna because of
palpitations experienced with Diovan (ugly drug in my opinion, same company produces
both however). The palpitations were pretty bad on Diovan, 24 hour monitor by EKG etc.
After a few days of substituting Tekturna for Diovan, there are no more palpitations.
Condition: High Blood Pressure
Top 3 Suggested Drugs:
Losartan
Aldactone
Spironolactone

text: son halfway fourth week intuniv became concerned began last week started taking
highest dose two day could hardly get bed cranky slept nearly hour drive home school
vacation unusual called doctor monday morning said stick day see school getting morning
last two day problem free much agreeable ever le emotional good thing le cranky
remembering thing overall behavior better tried many different medication far effective
Condition: ADHD
Top 3 Suggested Drugs:
Adderall
```

```
Amphetamine / dextroamphetamine
Atomoxetine

text: I just got diagnosed with type 2. My doctor prescribed Invokana and metformin from the beginning. My sugars went down to normal by the second week. I am losing so much weight. No side effects yet. Miracle medicine for me
Condition: Diabetes, Type 2
Top 3 Suggested Drugs:
Victoza
Canagliflozin
Invokana
```

```
df_testsent = predict_text(sentences)
df_testsent
```

```
                                          test_sent           prediction
0  tekturna day effect immediate also calcium cha...  High Blood Pressure
1  son halfway fourth week intuniv became concern...                 ADHD
2  got diagnosed type doctor prescribed invokana ...     Diabetes, Type 2
```

```
import joblib
joblib.dump(tfidf_vectorizer3, 'tfidfvectorizer.pkl')
joblib.dump(pass_tf, 'passmodel.pkl')
```

```
['passmodel.pkl']
```

```
vectorizer = joblib.load('tfidfvectorizer.pkl')
model = joblib.load('passmodel.pkl')
```

```
test = model.predict(vectorizer.transform(["I have only been on Tekturna for 9 days. The effect was immediate. I am also on a calcium channel blocker (Tiazac) and hydrochlorothiazide. I was put on Tekturna because of palpitations experienced with Diovan (ugly drug in my opinion, same company produces both however). The palpitations were pretty bad on Diovan, 24 hour monitor by EKG etc. After a few days of substituting Tekturna
for Diovan, there are no more palpitations"]))
test[0]
```

```
'High Blood Pressure'
```

10.4. OUTPUT

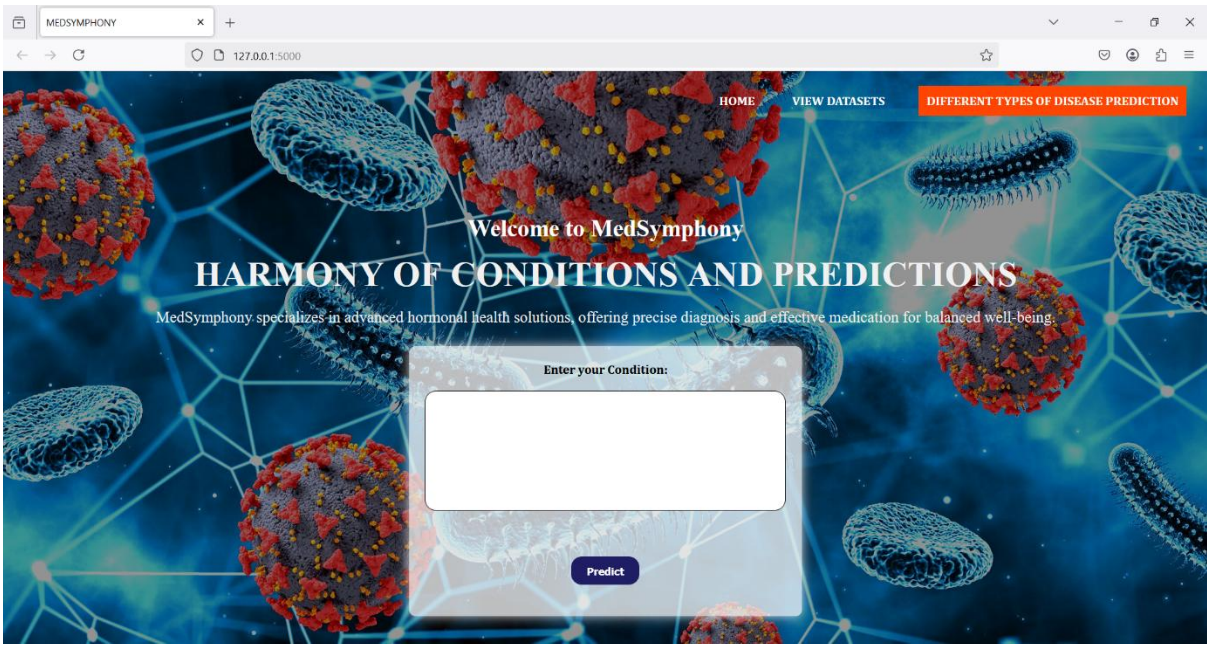
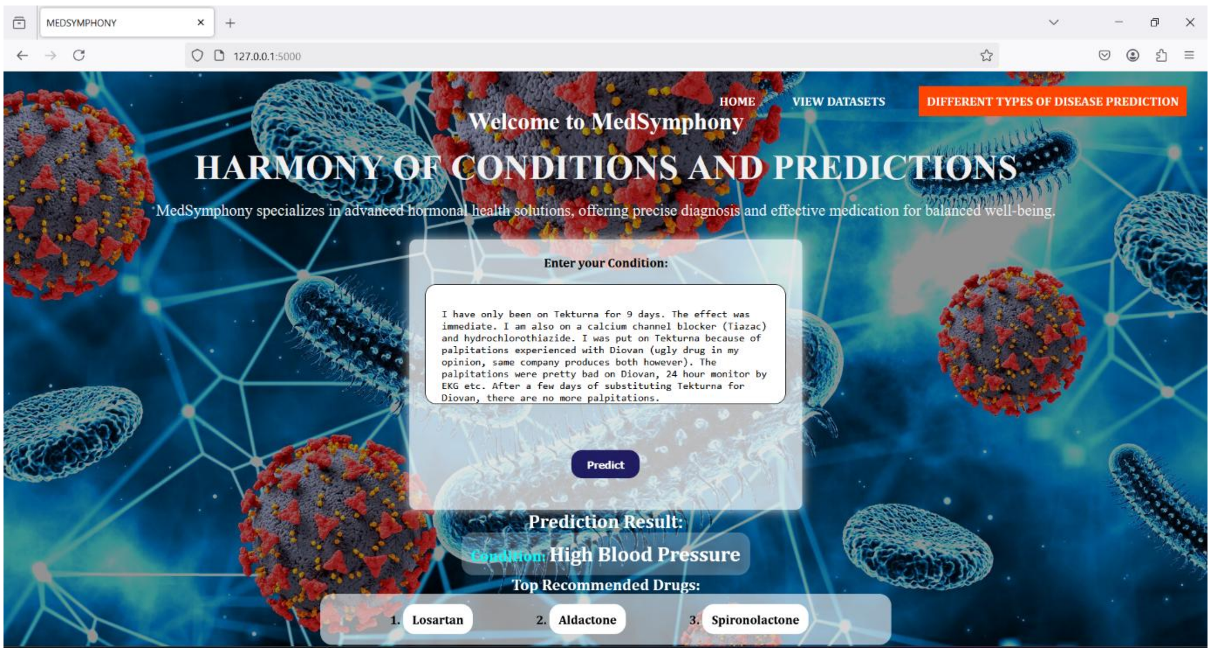

MEDSYMPHONY
127.0.0.1:5000
HOME
VIEW DATASETS
DIFFERENT TYPES OF DISEASE PREDICTION
Welcome to MedSymphony
HARMONY OF CONDITIONS AND PREDICTIONS
MedSymphony specializes in advanced hormonal health solutions, offering precise diagnosis and effective medication for balanced well-being.
Enter your Condition:
Predict

Figure 10.4.1. Welcome Page


MEDSYMPHONY
127.0.0.1:5000
HOME
VIEW DATASETS
DIFFERENT TYPES OF DISEASE PREDICTION
Welcome to MedSymphony
HARMONY OF CONDITIONS AND PREDICTIONS
MedSymphony specializes in advanced hormonal health solutions, offering precise diagnosis and effective medication for balanced well-being.
Enter your Condition:
Predict

```
I have only been on Tekturna for 9 days. The effect was
immediate. I am also on a calcium channel blocker (Tiazac)
and hydrochlorothiazide. I was put on Tekturna because of
palpitations experienced with Diovan (ugly drug in my
opinion, same company produces both however). The
palpitations were pretty bad on Diovan, 24 hour monitor by
EKG etc. After a few days of substituting Tekturna for
Diovan, there are no more palpitations.
```

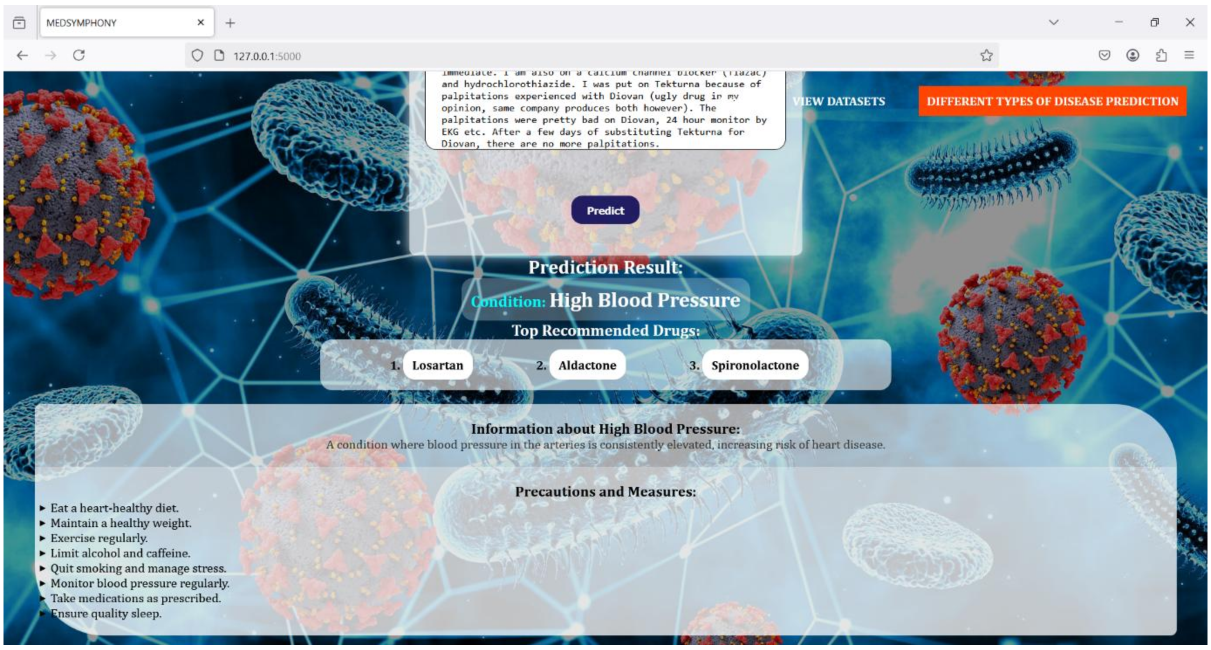
Prediction Result:

Condition: High Blood Pressure

Top Recommended Drugs:

1. Losartan
2. Aldactone
3. Spironolactone

Figure 10.4.2. Enter Your Condition



```
immediate. I am also on a calcium channel blocker (Tiazac)
and hydrochlorothiazide. I was put on Tekturna because of
palpitations experienced with Diovan (ugly drug in my
opinion, same company produces both however). The
palpitations were pretty bad on Diovan, 24 hour monitor by
EKG etc. After a few days of substituting Tekturna for
Diovan, there are no more palpitations.
```

Prediction Result:

Condition: High Blood Pressure

Top Recommended Drugs:

1. Losartan
2. Aldactone
3. Spironolactone

Information about High Blood Pressure:

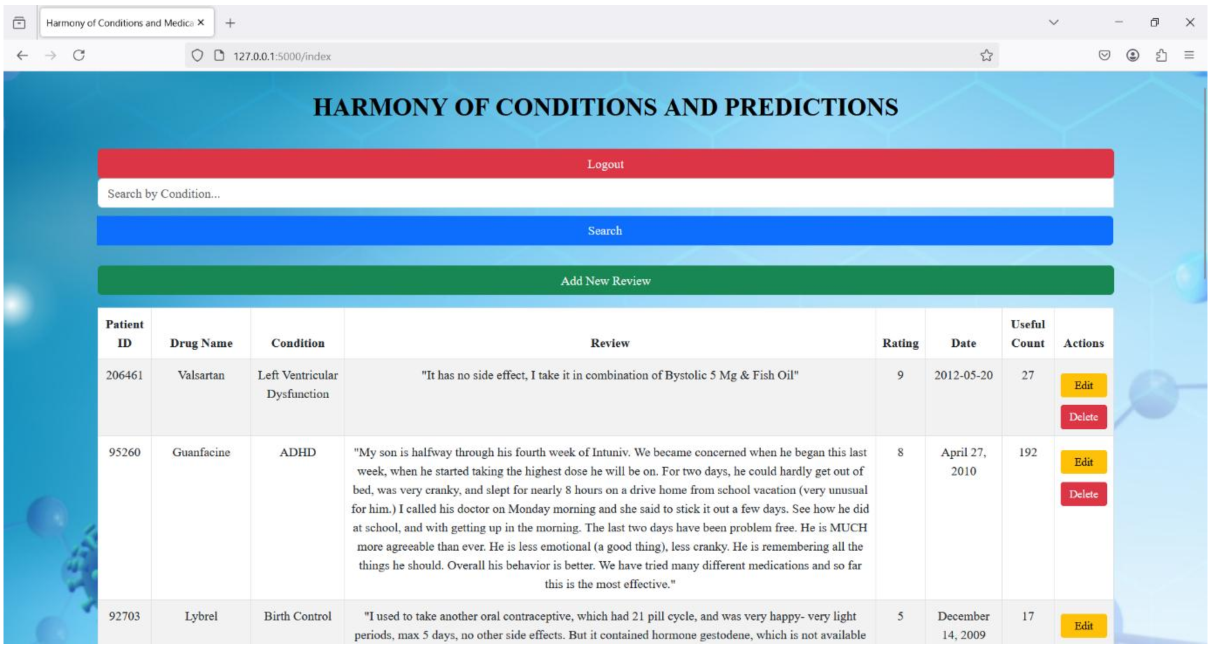
A condition where blood pressure in the arteries is consistently elevated, increasing risk of heart disease.

Precautions and Measures:

- Eat a heart-healthy diet.
- Maintain a healthy weight.
- Exercise regularly.
- Limit alcohol and caffeine.
- Quit smoking and manage stress.
- Monitor blood pressure regularly.
- Take medications as prescribed.
- Ensure quality sleep.

Figure 10.4.3. Prediction and Medications

DATABASE:

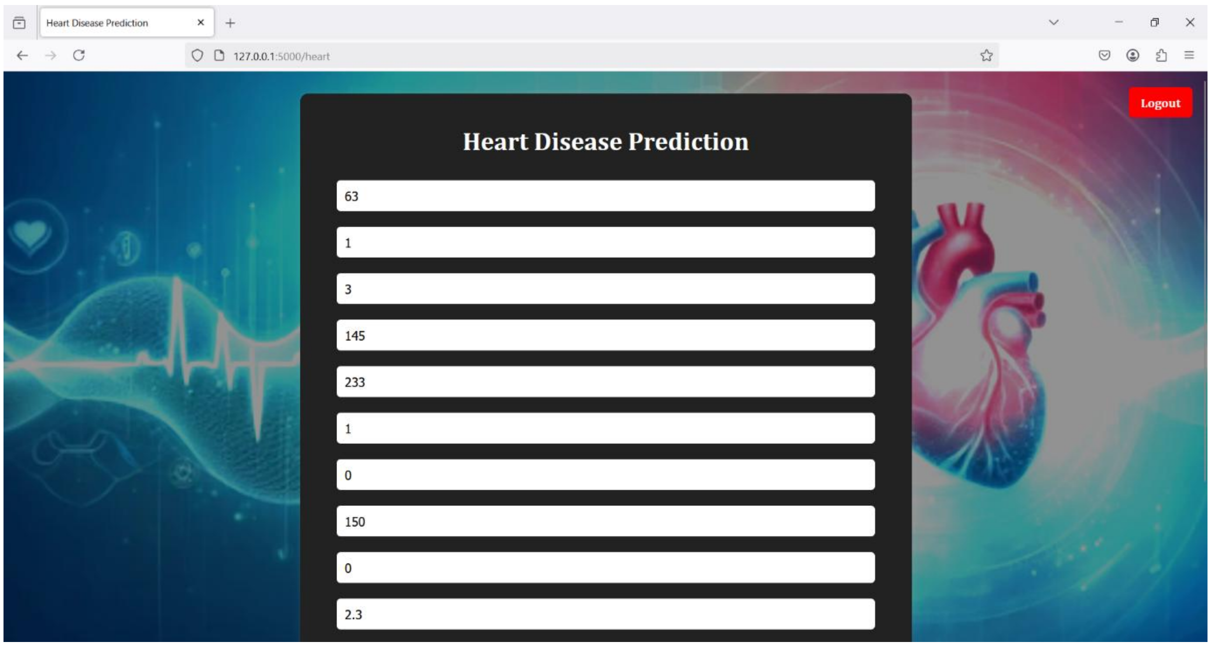


HARMONY OF CONDITIONS AND PREDICTIONS

Patient ID	Drug Name	Condition	Review	Rating	Date	Useful Count	Actions
206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combination of Bystolic 5 Mg & Fish Oil"	9	2012-05-20	27	Edit Delete
95260	Guanfacine	ADHD	"My son is halfway through his fourth week of Intuniv. We became concerned when he began this last week, when he started taking the highest dose he will be on. For two days, he could hardly get out of bed, was very cranky, and slept for nearly 8 hours on a drive home from school vacation (very unusual for him.) I called his doctor on Monday morning and she said to stick it out a few days. See how he did at school, and with getting up in the morning. The last two days have been problem free. He is MUCH more agreeable than ever. He is less emotional (a good thing), less cranky. He is remembering all the things he should. Overall his behavior is better. We have tried many different medications and so far this is the most effective."	8	April 27, 2010	192	Edit Delete
92703	Lybrel	Birth Control	"I used to take another oral contraceptive, which had 21 pill cycle, and was very happy- very light periods, max 5 days, no other side effects. But it contained hormone gestodene, which is not available	5	December 14, 2009	17	Edit

Figure 10.4.4. Database with CRUD

HEART DISEASE PREDICTION:


Heart Disease Prediction

127.0.0.1:5000/heart

Logout

Heart Disease Prediction

63

1

3

145

233

1

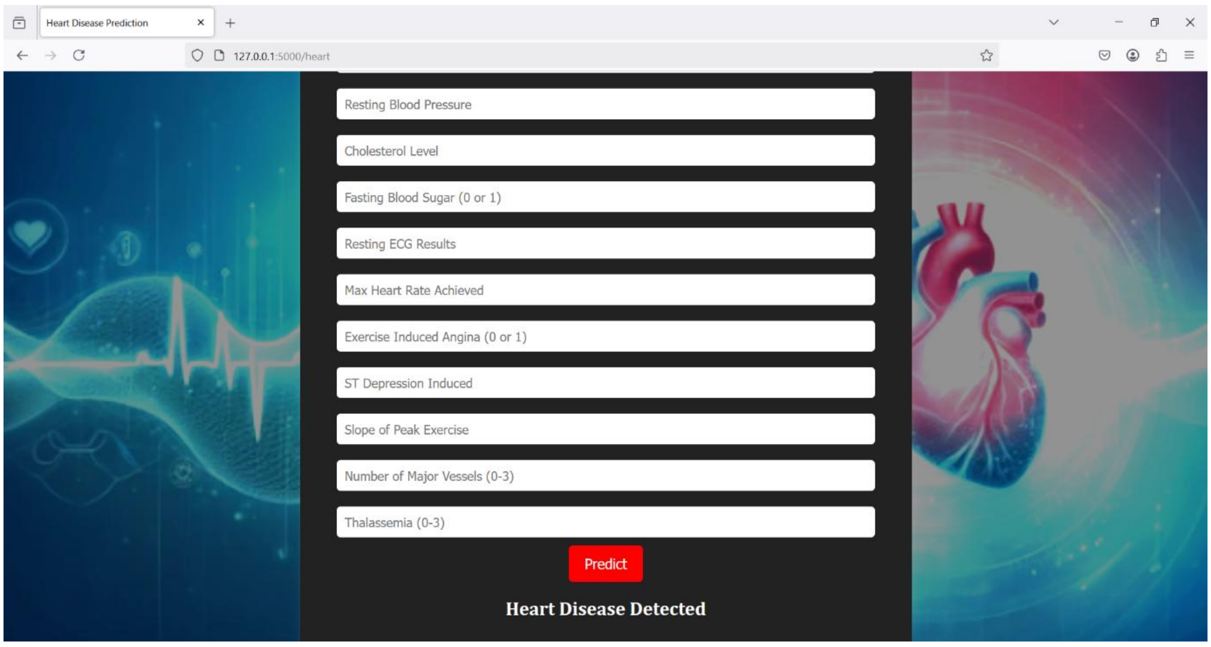
0

150

0

2.3

Figure 10.4.5. Heart Disease Predication


Heart Disease Prediction

127.0.0.1:5000/heart

Resting Blood Pressure

Cholesterol Level

Fasting Blood Sugar (0 or 1)

Resting ECG Results

Max Heart Rate Achieved

Exercise Induced Angina (0 or 1)

ST Depression Induced

Slope of Peak Exercise

Number of Major Vessels (0-3)

Thalassemia (0-3)

Predict

Heart Disease Detected

Figure 10.4.6. Heart Disease Predication- Result

11. CONCLUSION

The development of the MedSymphony platform demonstrates the effective application of machine learning and data analytics in building an intelligent system that can predict diseases and recommend medications based on user input. By combining a symptom-based disease prediction model with a drug review management system, this project showcases how modern AI tools can improve decision-making and user engagement through a web-based interface.

The system leverages a machine learning pipeline that includes data preprocessing, text vectorization (TF-IDF), and a trained classification model to predict possible health conditions. It further enriches the experience by analyzing real-world drug reviews, helping users understand the effectiveness of medications based on ratings, usefulness, and sentiment. The integration of review-based recommendations adds value by providing practical guidance after prediction, connecting the user's symptom entry with real, data-driven medication feedback.

Furthermore, the project combines user input analysis, NLP techniques, and database management within a unified Flask-based application. The synergy between predictive modeling and review analytics allows for a seamless transition from prediction to action, ultimately offering a meaningful and informed user experience.

11.1. KEY TAKEAWAYS

- **Practical Integration of Machine Learning:** The project successfully demonstrates the practical implementation of machine learning models for predicting user-defined conditions from textual input. The use of preprocessing, TF-IDF vectorization, and classification helps in converting unstructured health descriptions into accurate predictions.
- **Impact of Drug Review Analytics:** By extracting insights from large drug review datasets, the system helps users better understand which medications are effective for particular conditions. This adds a user-centric layer of value to the platform beyond prediction alone.

- **Importance of Preprocessing and Feature Engineering:** Key to the success of the system was the quality of text preprocessing steps such as stopword removal, lemmatization, and feature extraction. These techniques improved model accuracy and helped reduce noise in both prediction and review components.
- **User Interface and Experience:** The web interface built using Flask provides an intuitive experience, making it easy for users to submit symptoms, get predictions, and view helpful drug recommendations and precautions in one place.
- **Scalability and Modularity:** The modular nature of the system allows for scalability. New conditions, improved models, or additional review data can be integrated easily. The current architecture supports updates and feature enhancements without major redesign.
- **Real-World Application Potential:** This project serves as a foundation for a broader application where users can get preliminary predictive feedback and community-driven medication information. Its usefulness spans personal health tracking, educational tools, and initial guidance systems.

12. FUTURE WORK

While the MedSymphony platform successfully integrates disease prediction and drug review analysis into a single system, there are several areas where the platform can be enhanced further to improve performance, usability, and applicability. As the project evolves, expanding its features, accuracy, and adaptability will make it even more valuable for users seeking insights and recommendations based on symptoms and drug effectiveness.

12.1. INTEGRATION OF REAL-TIME SYMPTOM SUGGESTION

To enhance user experience, future versions can include real-time auto-suggestions as users type their symptoms. This can be done using symptom ontologies or auto-complete systems trained on existing medical datasets. It would improve input accuracy and reduce user confusion while entering symptom descriptions.

12.2. SENTIMENT ANALYSIS OF DRUG REVIEWS

While reviews are currently analyzed based on rating and usefulness, adding sentiment analysis using NLP models (e.g., VADER or BERT) could further refine how effective a drug is perceived. This could identify subtle indicators in language beyond numeric ratings—for instance, highlighting frustration, improvement, or side-effect mentions.

12.3. MULTI-CLASS AND MULTI-LABEL PREDICTION SUPPORT

Currently, the model predicts a single condition per input. Extending this to support multi-class and multi-label predictions would allow the system to recognize and return multiple potential conditions when symptoms are ambiguous or overlap between diseases.

12.4. EXPANDING THE DRUG DATABASE

The current database includes a limited number of drug reviews. In future versions, this could be expanded by integrating public APIs like the FDA drug review database or other community-contributed data sources to keep the system updated with newer medications and real-world user reviews.

12.5. PERSONALIZED DRUG RECOMMENDATIONS

Incorporating user-specific data such as age, gender, or medical history (optionally and securely) could lead to more personalized drug suggestions. This would increase the system's relevance, as drug effectiveness and reactions can vary significantly across demographics.

12.6. EXPLAINABLE PREDICTIONS (XAI)

To enhance transparency and trust, integrating explainable AI tools like SHAP or LIME could help show users why a certain condition was predicted based on their input. This would also help developers fine-tune the model and detect any biases.

12.7. CHATBOT OR VIRTUAL ASSISTANT INTEGRATION

A chatbot interface could be added for users who prefer interactive communication. The bot could guide them through the process of symptom input, clarify ambiguous statements, and provide instant condition summaries and medication feedback.

12.8. MOBILE APPLICATION DEVELOPMENT

To improve accessibility, the platform can be transformed into a mobile application using technologies like Flutter or React Native. This would allow users to use the prediction system on-the-go and receive medication suggestions instantly.

12.9. INTEGRATION WITH HEALTH MONITORING SYSTEMS

The platform could be enhanced to ingest data from wearables or health monitoring apps, allowing predictions based on biometric inputs like heart rate, activity level, or blood sugar readings. This would create a more comprehensive view of the user's health status.

12.10. LANGUAGE AND ACCESSIBILITY FEATURES

Future updates could introduce multi-language support and text-to-speech capabilities, making the platform more inclusive for non-English speakers and visually impaired users.

13. REFERENCES

- "UCI Machine Learning Repository - Drug Review Dataset." https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29
- "Python Programming Language – Official Site." https://www.python.org
- "Flask – Web Development with Python." https://flask.palletsprojects.com
- "SQLite – Lightweight Relational Database." https://www.sqlite.org
- "Jupyter Notebook – Open Source Interactive Computing." https://jupyter.org
- "Git – Distributed Version Control System." https://git-scm.com
- "NumPy – The Fundamental Package for Scientific Computing with Python." https://numpy.org
- "Pandas – Python Data Analysis Library." https://pandas.pydata.org
- "Matplotlib – Python Plotting Library." https://matplotlib.org
- "Seaborn – Statistical Data Visualization Library." https://seaborn.pydata.org
- "Scikit-learn – Machine Learning in Python." https://scikit-learn.org
- "Keras – Deep Learning for Humans." https://keras.io
- "NLTK – Natural Language Toolkit." https://www.nltk.org
- "BeautifulSoup – Library for Pulling Data Out of HTML and XML Files." https://www.crummy.com/software/BeautifulSoup
- "TF-IDF Vectorizer – Scikit-learn Feature Extraction." https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html