A Project report on

Drug Recommendation System based on Sentiment Analysis of Drug Reviews using Machine Learning

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

BATTA DEVARSHI (20B91A0529)

Under the Guidance of
SRI CH.VINOD VARMA
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)

[2023 – 2024]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

Chinna Amiram, Bhimavaram, West Godavari Dist., A.P.

[2023 – 2024]

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **"Drug Recommendation System based on Sentiment Analysis of Drug Reviews using Machine Learning"** is the bonafide work of **BATTA DEVARSHI** bearing **20B91A0529** who carried out the project work under my supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

SUPERVISOR
(Sri.Ch.Vinod Varma)
Assistant Professor

HEAD OF THE DEPARTMENT
(Dr. V Chandra Sekhar)
Professor

SELF DECLARATION

We hereby declare that the project work entitled **"Drug Recommendation System based on Sentiment Analysis of Drug Reviews using Machine Learning"** is a genuine work carried out by us in B.Tech., (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

BATTA DEVARSHI **(20B91A0529)**

ABSTRACT

Since the emergence of the coronavirus pandemic, there has been a significant strain on medical resources, including a shortage of healthcare professionals, essential equipment, and medicines. This strain has led to distress within the medical community and has unfortunately resulted in numerous deaths due to inadequate access to proper care. With limited access to healthcare professionals, many individuals have resorted to selfmedication without proper consultation, exacerbating their health conditions.

Recognizing the pressing need for innovative solutions, this study proposes the development of a drug recommendation system powered by machine learning. By leveraging patient reviews and sentiment analysis techniques such as TFIDF (Term FrequencyInverse Document Frequency) and Manual Feature Analysis, this system aims to assist in recommending the most suitable medication for a given disease. Various classification algorithms are employed to predict sentiment, and the performance of these models is evaluated using metrics such as precision, recall, F1 score, and accuracy.

The findings indicate that the classifier utilizing ngrams with TFIDF vectorization consistently outperforms other models, achieving an impressive accuracy rate of 93%. This research represents a significant step towards alleviating the burden on healthcare professionals and improving access to appropriate medication for patients in need.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. SVM: Support Vector Machine
2. NLP: Natural Language Processing
3. CADRE: cloud-assisted drug recommendation system

CHAPTER-1
INTRODUCTION

The COVID-19 pandemic has laid bare the critical shortage of medical professionals across nations, particularly in remote rural areas where access to specialized care is limited compared to urban centers. This scarcity is exacerbated by the lengthy training period required for doctors to attain necessary qualifications, spanning between 6 to 12 years, further deepening the crisis. Compounding this challenge is the alarming prevalence of medical errors, with prescription mistakes affecting over 200 thousand individuals in China and 100 thousand in the USA annually, underscoring the urgency for reform in medical practices to mitigate adverse outcomes.

Amidst this landscape, the high incidence of prescription errors, accounting for over 40% of medication mistakes, underscores the imperative to enhance healthcare delivery. Addressing this issue requires a nuanced understanding of patient needs and medical complexities. Patients rely on receiving the most suitable medication, necessitating doctors with comprehensive knowledge of pathogens, antibiotics, and patient profiles to ensure optimal treatment outcomes. Furthermore, the rapid pace of medical advancements compounds the challenge, making it increasingly arduous for healthcare providers to stay abreast of emerging treatments and tailored interventions tailored to individual patient histories and conditions.

In response to these challenges, recommender systems emerge as a promising avenue for improving healthcare delivery by leveraging patient feedback and sentiment analysis to suggest tailored medications for specific conditions. By harnessing patient reviews, these systems can discern patterns and preferences, guiding clinicians towards evidence-based prescribing practices. This research is delineated into five distinct sections, commencing with an introduction that provides a contextual backdrop and rationale for the study. Subsequent sections delve into a comprehensive review of related works, elucidating existing literature to inform the research approach.

The methodology section intricately details the research approach, encompassing data collection, preprocessing techniques, model selection, and evaluation metrics employed to

assess model performance. Following rigorous analysis, the results section critically evaluates the efficacy of the applied models using various metrics, shedding light on their strengths and limitations. Subsequently, the discussion section delves into nuanced insights, elucidating the constraints and implications of the framework, paving the way for future research avenues. Finally, the conclusion encapsulates the key findings and implications of the study, offering actionable insights to enhance healthcare delivery and mitigate prescription errors.

CHAPTER-2
LITERATURE SURVEY

Leilei Sun conducted research aimed at optimizing treatment prescriptions for patients by analyzing large-scale treatment records. The approach involved employing an efficient semantic clustering algorithm to estimate similarities between treatment records, facilitating the identification of the most effective treatment regimens. Additionally, a framework was developed to evaluate the efficacy of the recommended treatments, ensuring tailored prescriptions based on patients' demographic locations and medical complications. The study utilized Electronic Medical Records (EMRs) collected from multiple clinics for testing, with results demonstrating improved cure rates.

In a separate study, Xiaohong Jiang et al. compared three distinct algorithms—decision tree, support vector machine (SVM), and backpropagation neural network—on treatment data. SVM emerged as the preferred choice for the medication recommendation module due to its superior performance across various metrics such as model accuracy, efficiency, and scalability. Furthermore, the authors proposed an error-checking system to ensure diagnosis precision and service quality.

Jiugang Li et al. developed a hashtag recommender framework leveraging skipgram models and convolutional neural networks (CNN) to learn semantic sentence vectors. These vectors were then utilized to classify hashtags using LSTM RNN. The results indicated that this model outperformed conventional approaches such as SVM and Standard RNN. The study highlighted the importance of capturing semantic features, which are often lost in traditional methods, to improve prediction accuracy.

Mohammad Mehedi Hassan et al. introduced a cloud-assisted drug recommendation system (CADRE) capable of suggesting drugs based on patients' side effects and topN related prescriptions. Initially employing collaborative filtering techniques, the model clustered medications based on functional description data. However, considering limitations such as computational cost, cold start, and data sparsity, the approach shifted towards a cloud-assisted approach using tensor decomposition to enhance the quality of medication recommendations and patient experience.

Drug recommendation systems have gained significant attention in recent years due to their potential to improve healthcare outcomes and patient satisfaction. A literature survey reveals a diverse range of approaches and methodologies employed in developing these systems. Traditional methods often rely on statistical and machine learning techniques to analyze patient data, such as Electronic Health Records (EHRs) and medical histories, to recommend appropriate medications based on symptoms, diagnoses, and patient characteristics. More advanced approaches leverage natural language processing (NLP) techniques to extract insights from unstructured text data, such as patient reviews and clinical notes, to enhance the accuracy and relevance of recommendations.

Additionally, collaborative filtering techniques, inspired by recommender systems in e-commerce and social media, have been adapted to healthcare settings to provide personalized drug recommendations based on similarities between patients with similar medical profiles. Furthermore, emerging technologies like deep learning and neural networks offer promise in capturing complex patterns and relationships in medical data, leading to more accurate and context-aware recommendations. Overall, the literature reflects a growing interest in leveraging data-driven approaches to optimize drug prescriptions, with a focus on improving patient outcomes, reducing medical errors, and enhancing overall healthcare quality

CHAPTER-3
PROBLEM STATEMENT

With a growing shortage of doctors, particularly in rural areas, exacerbated by lengthy training periods, addressing patient care complexities becomes challenging amid the exponential rise in COVID19 cases.

The dynamic medical landscape introduces daily advancements, making it increasingly difficult for doctors to select appropriate treatments.

A proposed recommender framework leveraging sentiment analysis and feature engineering aims to assist doctors by recommending medications tailored to specific conditions based on patient reviews.

CHAPTER-4
METHODOLOGY

4.1 Overview of Methodology

In this section, we outline the methodology employed in this research for building the drug recommender system. We utilize the Drug Review Dataset sourced from the UCI ML repository, comprising six attributes: drug name, patient review, patient condition, useful count, review date, and patient rating. The dataset contains a total of 215,063 instances.

4.2 Data Cleaning and Visualization

We begin by cleaning the dataset to ensure data integrity and remove any inconsistencies. This involves handling null values and duplicate rows, thereby preparing the data for further analysis. We visualize the characteristics of the data using bar plots to illustrate distributions and patterns, as well as to identify any trends in null values and the most common conditions for which drugs are available.

4.2.1 Null Value Handling and Duplicate Removal

Identification and removal of null values and duplicate rows are crucial steps to ensure data cleanliness and integrity. By eliminating these discrepancies, we prepare the dataset for subsequent analysis and modeling.

4.2.2 Text Preprocessing and Feature Extraction

Text preprocessing involves cleaning the review text by removing HTML tags, punctuation, and URLs. We then tokenize the text, convert it to lowercase, and remove stopwords to normalize the data. Additionally, we apply lemmatization to further standardize the tokens. Feature extraction methods such as Bag of Words (BoW), TFIDF, and Word2Vec are employed to convert textual data into numerical format for machine learning algorithms. We also engineer manual features from reviews, including punctuation and sentiment polarity.
We can not use text directly to train our models.

We need to convert it in the form of features, only then it can be used to train any model for desired outcome and we know very well that most of the models respond to the numeric features very well. So we need to bring all these text representations in the form of numbers.

4.3 Train Test Split and SMOTE

4.3.1 Train Test Split

We divide the dataset into training and testing sets, ensuring consistent splitting across different feature extraction methods. This allows us to train our models on a subset of the data and evaluate their performance on unseen data.

4.3.2 SMOTE for Class Imbalance

To address class imbalance in the dataset, we apply the Synthetic Minority Oversampling Technique (SMOTE). This involves synthesizing new minority class data to balance class distributions in the training data, thereby improving the performance of our models.

4.4 Machine Learning Classifiers and Evaluation Metrics

4.4.1 Classifier Selection

We employ various machine learning algorithms, including Logistic Regression and Multinomial Naive Bayes, to predict sentiment based on the extracted features. By experimenting with different classifiers, we aim to identify the most effective approach for our drug recommender system.

4.4.2 Evaluation Metrics

We evaluate the performance of our classifiers using a range of metrics, including precision, recall, F1 score, accuracy, and AUC score. This allows us to assess the effectiveness of our models in sentiment prediction across different feature extraction methods.

4.5 Combined Prediction and Score Calculation

Finally, we generate combined predictions from the bestperforming classifiers and calculate drug scores by multiplying predictions with normalized useful count. This approach ensures balanced recommendations by addressing the distribution of useful count in the dataset.

Overall, this methodology provides a systematic framework for building and evaluating the drug recommender system, incorporating data cleaning, feature extraction, model training, and evaluation.

4.6 Flowchart

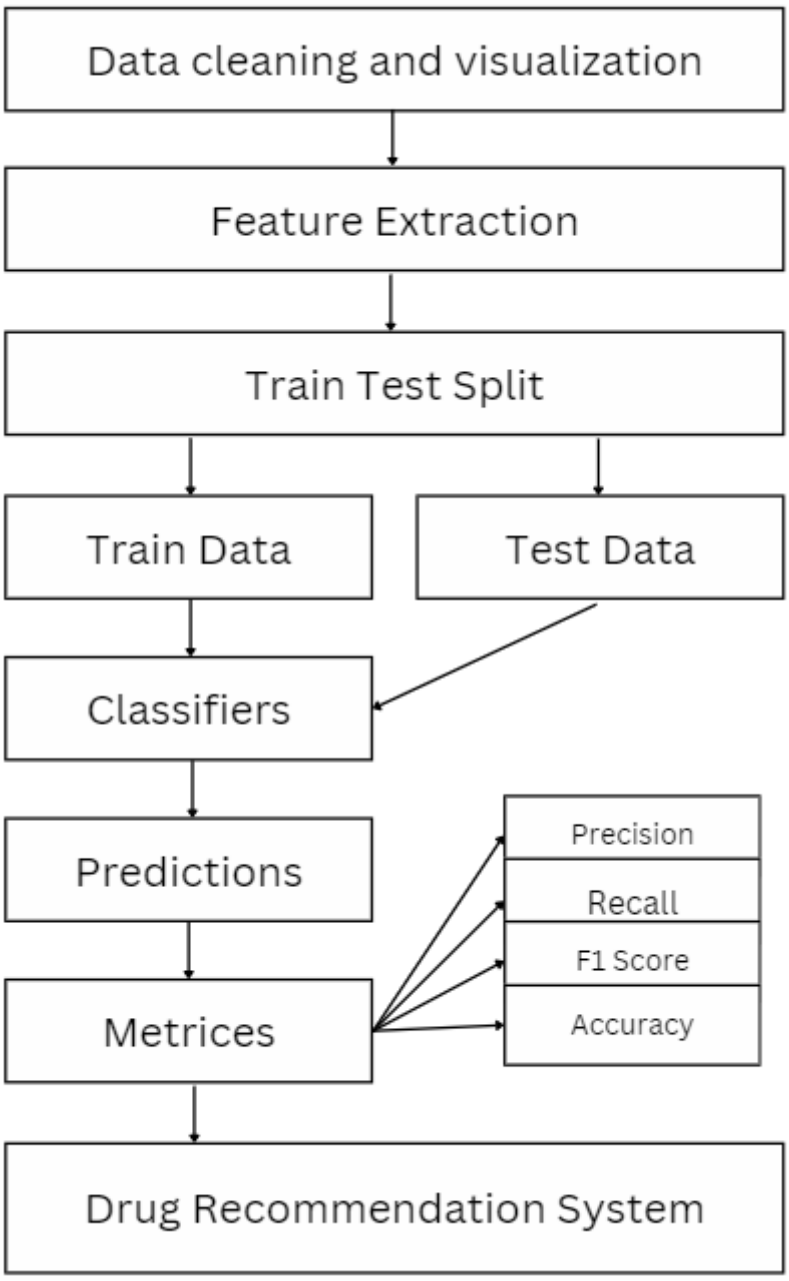

Fig 4.1 Flow chart of methodology

CHAPTER-5
IMPLEMENTATION

5.1. Data Understanding and Transformations:

Loading and Combining Data: The train and test data are loaded into separate datasets and then combined into a single dataset to streamline the analysis.

Checking Dataset Shape: The shape of the dataset, indicating the number of rows and columns, is examined to understand its size and structure.

Inspecting Columns: Each column and its respective data type are inspected to understand the variables present in the dataset.

Handling Null Values: Null values, if any, are handled by dropping rows with null values in specific columns, such as the condition column, to ensure data integrity.

Converting Date Column: The date column is converted to an appropriate data type to facilitate further analysis.

5.2. Descriptive Analysis:

Exploratory Data Analysis (EDA): EDA is conducted for each independent column to gain insights into the distribution and characteristics of the data.

Word Clouds: Word clouds are generated to visualize frequently used words in the review column for both the highest and least ratings, providing insights into the language used in reviews.

Heatmap: A heatmap is created to observe the correlation between variables, revealing any patterns or relationships in the data.

5.3. PreProcessing and Encoding Data:

Removing Special Characters: Special characters, such as unicode characters, are removed from the review column to clean the text data.

Expanding Contractions: Contractions in the review column are expanded to improve text consistency and analysis.

Text Preprocessing: Text preprocessing techniques, including lowering text, removing punctuation, lemmatization, and stopwords removal, are applied to the review column to standardize and clean the text data.

Encoding Categorical Variables: Categorical variables like condition and drugname are encoded using label encoding to convert them into numerical format for model training.

5.4. Feature Engineering:

Sentiment Analysis: Sentiment polarity is computed for the preprocessed review column using TextBlob to capture the sentiment expressed in the reviews.

Feature Extraction: Various features such as word count, unique word count, punctuation count, etc., are extracted from the original review column to provide additional information for model training.

Binary Conversion: The rating column data is converted into binary format, indicating positive or negative sentiment based on the rating (1 if rating > 5 and 0 if rating < 5), to facilitate binary classification.

5.5. Model Building:

Selection of Models: Several classification algorithms, including Logistic Regression, Linear Discriminant Analysis, K Neighbors Classifier, Decision Tree Classifier, Gaussian Naive Bayes, and Random Forest Classifier, are utilized for model building.

Training and Evaluation: The models are trained using appropriate algorithms and evaluated using various evaluation metrics such as accuracy, precision, recall, and F1score to assess their performance.

Final Results: The models are ordered based on the useful count to prioritize recommendations for drugs associated with higher utility.

5.6. Predictions:

Top 5 Drug Recommendations: The trained models are used to predict the top 5 drugnames for a given ailment, leveraging the insights obtained from the data preprocessing and feature engineering steps.

In summary, this comprehensive approach encompasses data preparation, exploratory analysis, preprocessing, feature engineering, model selection and evaluation, and ultimately, generating recommendations for drug prescriptions based on the trained models.

5.7 Requirements and Algorithms:

5.7.1 Requirements:

To successfully run the program, you will need Python version 3.6 or higher installed on your system. Additionally, the following Python libraries are required: pandas, numpy, matplotlib, seaborn, sklearn, Textblob, nltk, xgboost, lightgbm, and spacy.

Pandas is a powerful data manipulation library used for data analysis tasks, while numpy provides support for numerical computing and array operations. Matplotlib and seaborn are essential for data visualization, allowing you to create various types of plots and charts to explore and communicate your data effectively.

Scikit-learn (sklearn) is a machine learning library that provides a wide range of algorithms and tools for classification, regression, clustering, and dimensionality reduction tasks. Textblob and NLTK are natural language processing (NLP) libraries used for text processing and analysis, including tasks such as sentiment analysis and text tokenization.

Xgboost and LightGBM are gradient boosting libraries widely used for building powerful machine learning models, especially in structured/tabular data scenarios. They offer high performance and efficiency in handling large datasets.

Spacy is a library for advanced natural language processing tasks, such as named entity recognition, part-of-speech tagging, and dependency parsing. It provides pre-trained

models and pipelines for various NLP tasks, making it suitable for building sophisticated NLP applications.

Overall, these libraries form the foundation for data analysis, machine learning, and natural language processing tasks, enabling you to perform advanced analytics and develop predictive models with ease and efficiency. To successfully run the program, you will need Python version 3.6 or higher installed on your system. Additionally, the following Python libraries are required: pandas, numpy, matplotlib, seaborn, sklearn,

As a whole, data pipelines serve as a fundamental blocks of data cohesion for implementing activities of data analysis, machine learning, and natural language processing, thus, permitting you to perform advanced analytics and design predictive models effortlessly. Running the program will be successful if you had Python 3.6 and above version installed on your system. Additionally, the following Python libraries are required: ,contains pandas, numpy, matplotlib, seaborn, sklearn, Textblob, nltk, xgboost, lightgbm, and spacy.

Textblob, nltk, xgboost, lightgbm, and spacy.To implement this project we have done Data preparation, Feature extraction, Test Train split after these preprocessing steps, we have implemented various classification models to choose the most accurate one.

5.7.2 Algorithms:

The implemented models include:

1. Logistic Regression
2. Linear Discriminant Analysis
3. K neighbors Classifier
4. Decision Tree Classifier
5. Gaussian NB
6. Random Forest Classifier

5.7.2.1. Logistic Regression:

Type: Supervised learning algorithm for binary classification.

Explanation: It models the probability that a given input belongs to a particular category. Despite its name, logistic regression is used for classification rather than regression. It calculates the probability of the binary outcome utilizing a logistic function.

5.7.2.2. Linear Discriminant Analysis (LDA):

Type: Supervised learning algorithm.

Explanation: LDA is a dimensionality reduction technique and a classification algorithm. It finds the linear combinations of features that best separate different classes. It makes assumptions about the distribution of the data and computes the posterior probabilities of the classes given the input features using Bayes' theorem.

5.7.2.3. K Neighbors Classifier:

Type: Non parametric lazy learning algorithm.

Explanation: KNN is a simple and intuitive classification algorithm. It classifies a data point based on the majority class among its 'k' nearest neighbors in the feature space. It doesn't make any assumptions about the underlying data distribution.

5.7.2.4. Decision Tree Classifier:

Type: Supervised learning algorithm.

Explanation: Decision trees recursively split the data into subsets based on the features that lead to the most significant reduction in impurity (e.g., Gini impurity or entropy). Each split creates branches until a stopping criterion is met, resulting in a tree like model for decision making.

5.7.2.5. Gaussian Naive Bayes (Gaussian NB):

Type: Supervised learning algorithm.

Explanation: Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features. Gaussian NB specifically assumes

that features follow a Gaussian distribution. It calculates the likelihood of a class given a feature vector and

chooses the class with the highest probability.

5.7.2.6. Random Forest Classifier:

Type: Ensemble learning method.

Explanation: Random forests build multiple decision trees during training and output the mode of the classes (classification) or the mean prediction (regression) of the individual trees. It introduces randomness in two ways: by selecting a random subset of features for each tree and by bootstrapping the training data. This helps to reduce overfitting and improve generalization.

5.7.2.7. ngrams Model:

An ngram model is a probabilistic language model used in natural language processing (NLP) and machine learning tasks, particularly in text analysis. It models the probability of a sequence of n words occurring together in a given text corpus. Ngrams are contiguous sequences of n items (words, characters, or tokens) extracted from the text.

1. Definition: An ngram is a sequence of n tokens (words, characters, or other linguistic units) extracted from a text corpus. For example, in the sentence "The quick brown fox jumps," the 2grams (bigrams) are "The quick," "quick brown," "brown fox," and "fox jumps."

2. Modeling: The ngram model calculates the probability of a word sequence using the chain rule of probability. It assumes that the probability of a word depends only on the previous (n1) words in the sequence. For example, the probability of the next word in a sequence given the previous words can be expressed as P(word | previous_words) = P(word | word1, word2, ..., word(n1)).

3. Application: Ngram models are widely used in various NLP tasks such as language modeling, speech recognition, machine translation, and text generation. They are particularly useful for tasks where the context of the words matters, such as predicting the next word in a sentence or detecting spam emails.

4. Advantages:

Ngram models are simple and easy to implement.They capture local word dependencies and context effectively.They can handle large text corpora efficiently.

5. Challenges:

The main challenge with ngram models is sparsity, especially for higherorder ngrams. As the value of n increases, the number of possible ngrams grows exponentially, leading to sparse data and estimation problems.Ngram models may not capture longrange dependencies in the text effectively.

6. Parameter Tuning: In practice, the choice of the value of n (the order of the ngram) is crucial and depends on the specific task and the characteristics of the text data. Higher values of n capture more context but may suffer from sparsity issues, while lower values of n may not capture enough context for accurate predictions.

CHAPTER-6

RESULT ANALYSIS

6.1 Evaluation Metrics: The system evaluates the performance of classifiers using metri cs such as precision, recall, F1 score, accuracy.

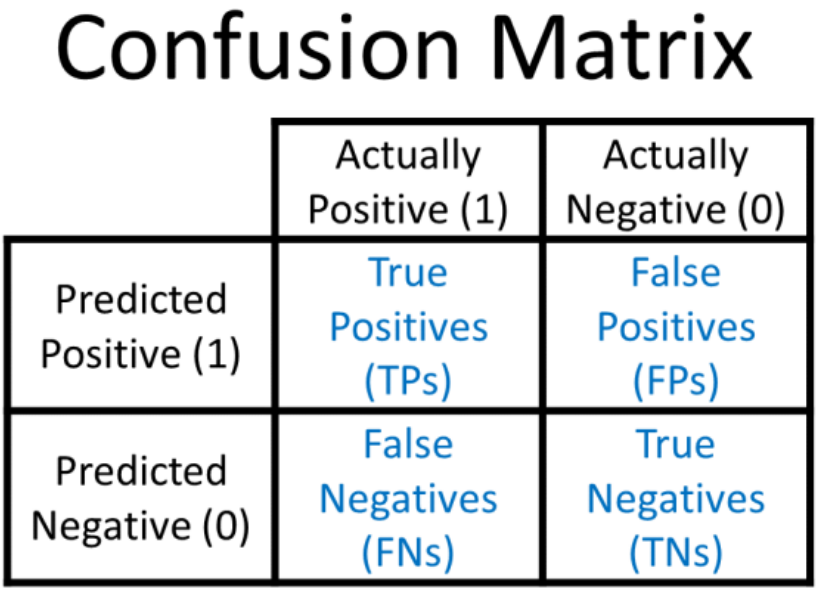

Fig 6.1 Confusion Matrix

6.2 What is a Confusion Matrix?

The confusion matrix is an indicator that is afterwards also used for the evaluation of the efficiency of machine learning model and it provides more specific data about the prediction. This reduces it down to seeing how fit this model is at drawing right boundaries between the different classes. On equating the number of accurate and false predictions, the model's precision level is assessed. In classification tasks, where the goal is to assign labels to input data, the confusion matrix breaks down the model's predictions into four categories: positive true cases, negative true cases, positive false cases and negative false cases are the four true to equate with positive predicted value and negative predicted value. The performance accuracy of the model will be determined by scrutinizing the account and getting reported information of it. We will look into the strength points, its weaknesses and its overall output at the end

6.3 Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

- **Classification Accuracy:** It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

- **Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

$$\text{Error rate}=\frac{FP+FN}{TP+FP+FN+TN}$$

- **Precision:** It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision}=\frac{TP}{TP+FP}$$

- **Recall:** It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall}=\frac{TP}{TP+FN}$$

- **Fmeasure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use Fscore. This score helps us to evaluate the recall and precision at the same time. The Fscore is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure}=\frac{2*Recall*Precision}{Recall+Precision}$$

Logistic Regression

LR: 0.704049 (0.001033)

	precision	recall	f1score	support
0	0.56	0.01	0.02	16119
1	0.70	1.00	0.82	37352
accuracy			0.70	53471
macro avg	0.63	0.50	0.42	53471
weighted avg	0.66	0.70	0.58	53471

Linear Discrimant Analysis

LDA: 0.746986 (0.001593)

	precision	recall	f1score	support
0	0.67	0.29	0.41	16119
1	0.75	0.94	0.84	37352
accuracy			0.74	53471
macro avg	0.71	0.61	0.62	53471
weighted avg	0.73	0.74	0.71	53471

K Nearest Neighbours

KNN: 0.710040 (0.002445)

	precision	recall	f1score	support
0	0.55	0.45	0.49	16119
1	0.78	0.84	0.81	37352

accuracy			0.72	53471
macro avg	0.67	0.64	0.65	53471
weighted avg	0.71	0.72	0.71	53471

Decision Tree

DT: 0.825503 (0.001557)

	precision	recall	f1score	support
0	0.77	0.78	0.78	16119
1	0.91	0.90	0.90	37352
accuracy			0.86	53471
macro avg	0.84	0.84	0.84	53471
weighted avg	0.87	0.86	0.87	53471

Neive Baise

NB: 0.732615 (0.001610)

	precision	recall	f1score	support
0	0.56	0.54	0.55	16119
1	0.80	0.82	0.81	37352
accuracy			0.73	53471
macro avg	0.68	0.68	0.68	53471
weighted avg	0.73	0.73	0.73	53471

Random Forest

RF: 0.873084 (0.000923)

	precision	recall	f1score	support
0	0.88	0.78	0.83	16119
1	0.91	0.96	0.93	37352
accuracy			0.90	53471
macro avg	0.90	0.87	0.88	53471
weighted avg	0.90	0.90	0.90	53471

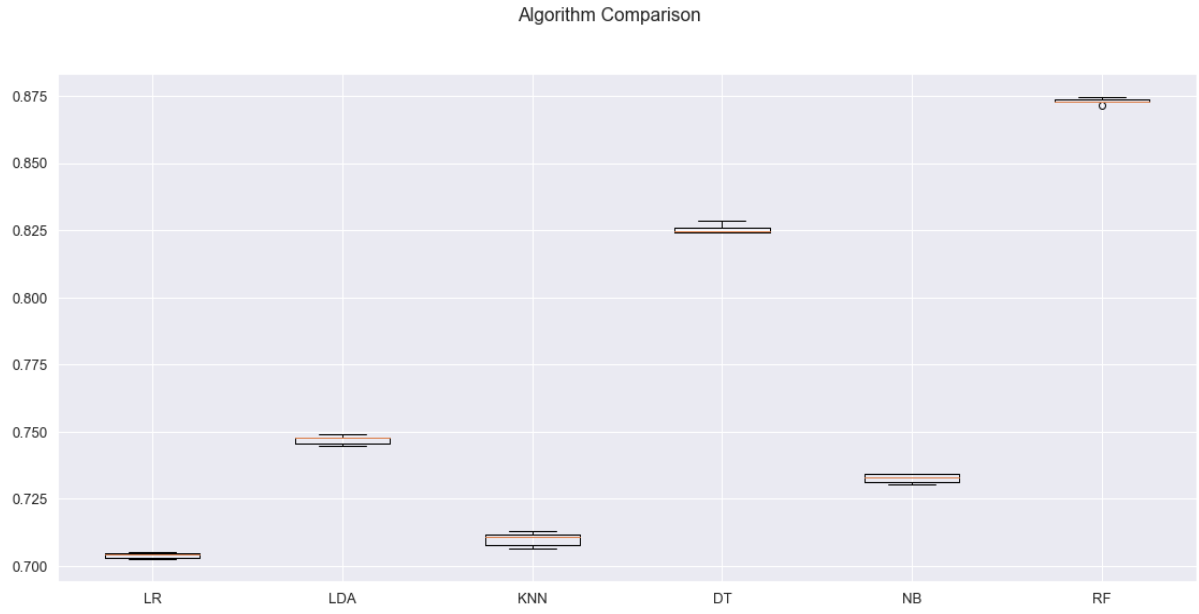

Fig 6.2 Algorithm Comparison

Confusion Matrix of ngrams model:

The confusion Matrix is

	precision	recall	f1score	support
0	0.88	0.78	0.83	16119
1	0.91	0.96	0.93	37352
accuracy			0.93	53471
macro avg	0.90	0.87	0.88	53471
weighted avg	0.90	0.90	0.90	53471

Drug Name Prediction:

```
enter a condition Anxiety
```

	drugname	usefulcount
172347	Buspirone	585
208378	BuSpar	322
170568	Gabapentin	253
213123	Lexapro	249
205565	Cymbalta	236

Fig 6.3 Sample Result-1

```
enter a condition Chronic Pain
```

	drugname	usefulcount
197455	OxyContin	184
163217	Acetaminophen / oxycodone	163
208376	Duloxetine	155
172431	Cymbalta	155
179398	Percocet	140

Fig 6.4 Sample result-2

```
enter a condition Major Depressive Disorde
```

	drugname	usefulcount
171551	Bupropion	159
211200	Fetzima	155
212361	Wellbutrin XL	151
174712	Wellbutrin	147
194525	Mirtazapine	141

Fig 6.5 Sample Result-3

```
enter a condition Migraine
```

	drugname	usefulcount
161268	Gabapentin	80
173607	Cyproheptadine	78
209877	Cyclobenzaprine	57
195552	Diclofenac	52
192410	Flexeril	51

Fig 6.6 Sample Result-4

```
enter a condition Bacterial Infection
```

	drugname	usefulcount
194010	Doxycycline	171
205126	Cephalexin	98
186129	Metronidazole	88
165010	Cleocin	87
162995	Keflex	70

Fig 6.7 Sample Result-5

```
enter a condition Diabetes, Type 2
```

	drugname	usefulcount
188537	Liraglutide	198
173017	Metformin	185
168081	Januvia	140
201890	Glimepiride	130
173830	Canagliflozin	123

Fig 6.8 Sample Result-6

```
enter a condition High Blood Pressure
```

	drugname	usefulcount
165233	Losartan	291
176396	Amlodipine	280
179056	Norvasc	280
173250	Cozaar	186
210004	Metoprolol	167

Fig 6.9 Sample Result-7

```
enter a condition Acne
```

	drugname	usefulcount
202908	Adapalene	159
161314	Differin	159
187251	Tretinoin	132
205120	Accutane	131
163047	Spironolactone	115

Fig 6.10 Sample Result-8

CHAPTER-7
CONCLUSION & FUTURE SCOPE

7.1 CONCLUSION

Research Objective: The study aimed to develop a recommender system for drug selection based on sentiment analysis of drug reviews using various machine learning classifiers and feature extraction techniques.

Classifier Performance: Multiple classifiers were employed, including Logistic Regression, LDA, Gaussian NB, KNN, Decision Tree, Random Forest, ngrams.

Evaluation was done using precision, recall, f1 score, accuracy, and AUC score metrics.

Best Performing Model: ngrams on TF IDF achieved the highest accuracy of 93%, outperforming all other models.

Worst Performing Model: Logistic Regression classifier on TF IDF showed the lowest accuracy at 70%.

Overall Recommendations: The best predicted emotion values from each method were used to calculate an overall score of the drug by condition. ngrams on TF IDF (93%) and Random Forest on manual features (88%) contributed to the overall score.

7.2 FUTURE SCOPE

The future scope of this project is outlined as follows: Deep Learning models may be used to reduce the Type II errors. User interface can be created and continuous feedback from users can be used to improve the model predictions

CHAPTER-8
REFERENCES

Garg, Satvik. "Drug recommendation system based on sentiment analysis of drug reviews using machine learning." 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2021.

Hossain, Md Deloar, et al. "Drugs rating generation and recommendation from sentiment analysis of drug reviews using machine learning." 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE). IEEE, 2020.

Koganti, Gunakar, et al. "Drug recommendation system based on analysis of drug reviews using machine learning." AIP Conference Proceedings. Vol. 2869. No. 1. AIP Publishing, 2023.

APPENDIX-I

DESIGN DOCUMENTS

UML DESIGN

UML aims to establish a consistent method for illustrating system designs, akin to blueprints used in other engineering disciplines. It serves as a universal modeling language within software engineering, offering a standardized approach to visualizing system designs.

A.I.1 USE CASE DIAGRAM

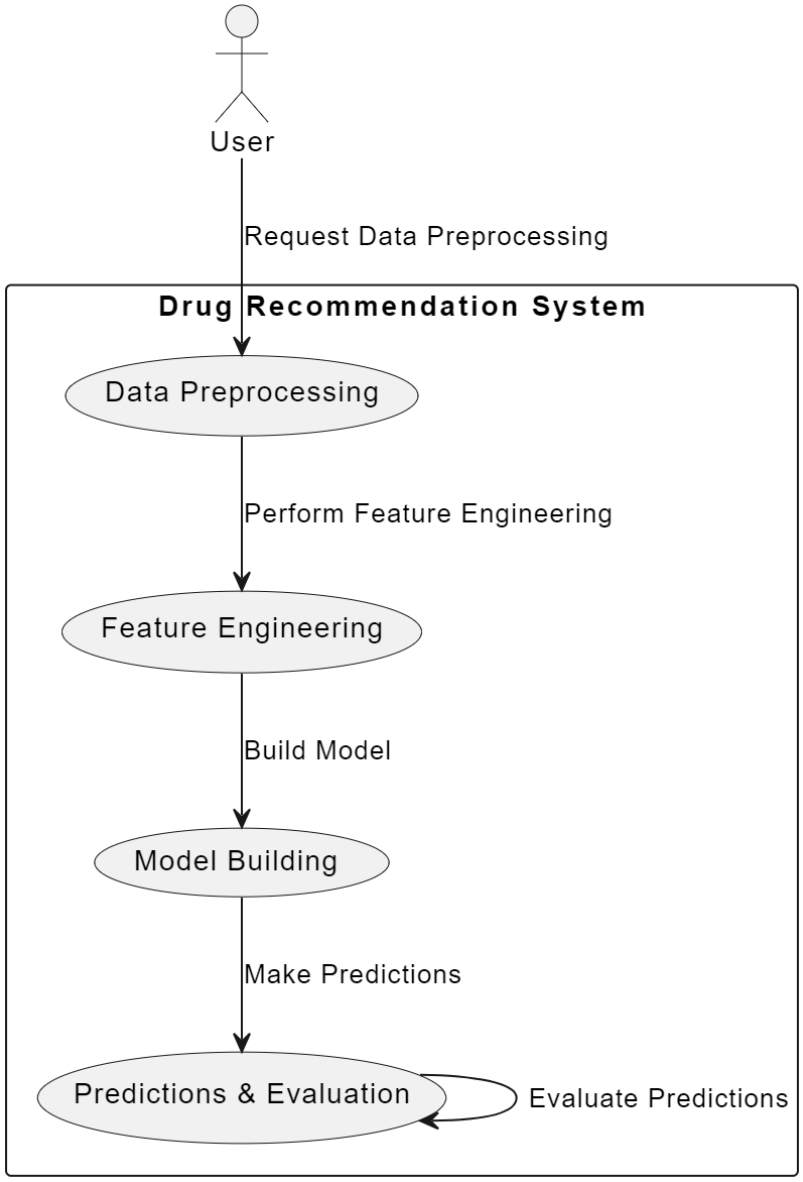

Fig. A.I.1: Use Case Diagram

A.I.2 CLASS DIAGRAM

A class diagram is a visual representation of the structure and relationships among classes in a system. It depicts the static structure of the system, including classes, attributes, operations, and their associations. Class diagrams are crucial in software design for modeling the entities and their relationships in a system, aiding in communication among stakeholders, guiding the implementation process, and facilitating the understanding of system architecture. Key properties include showing the hierarchy of classes, associations between classes, attributes, and methods, and providing a blueprint for system design and development.

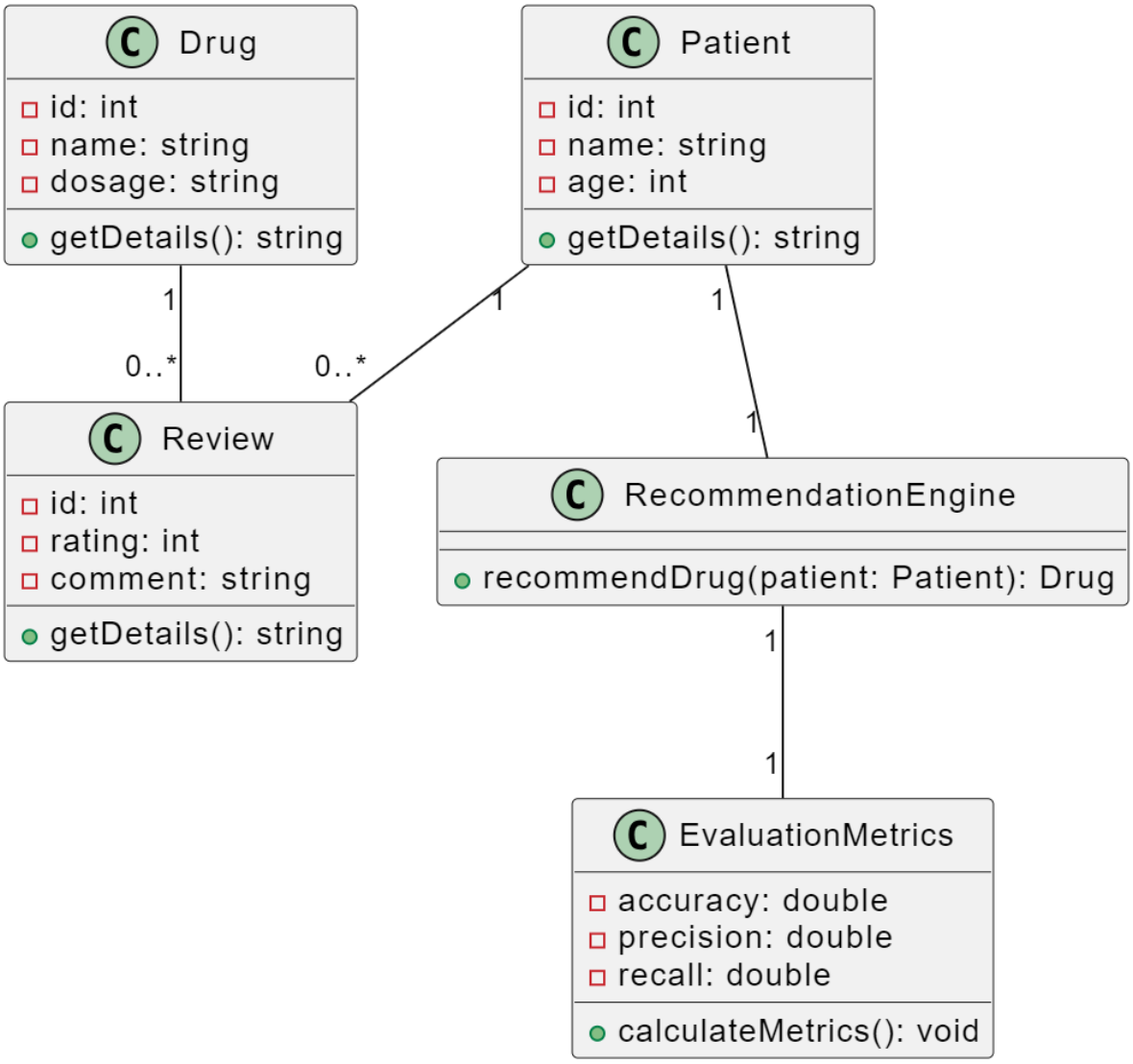

Fig. A.I.2: Class Diagram

A.I.3 SEQUENCE DIAGRAM

A sequence diagram is a visual representation of interactions between objects or components in a system, showing the sequence of messages exchanged over time. It helps in understanding the flow of control and data during execution. Sequence diagrams are essential in software design for illustrating the dynamic behavior of systems, aiding in communication among stakeholders, identifying potential issues in system interactions, and facilitating the design of efficient and reliable systems. Key properties include depicting the chronological order of messages, showing concurrent activities, and highlighting the dependencies between components.

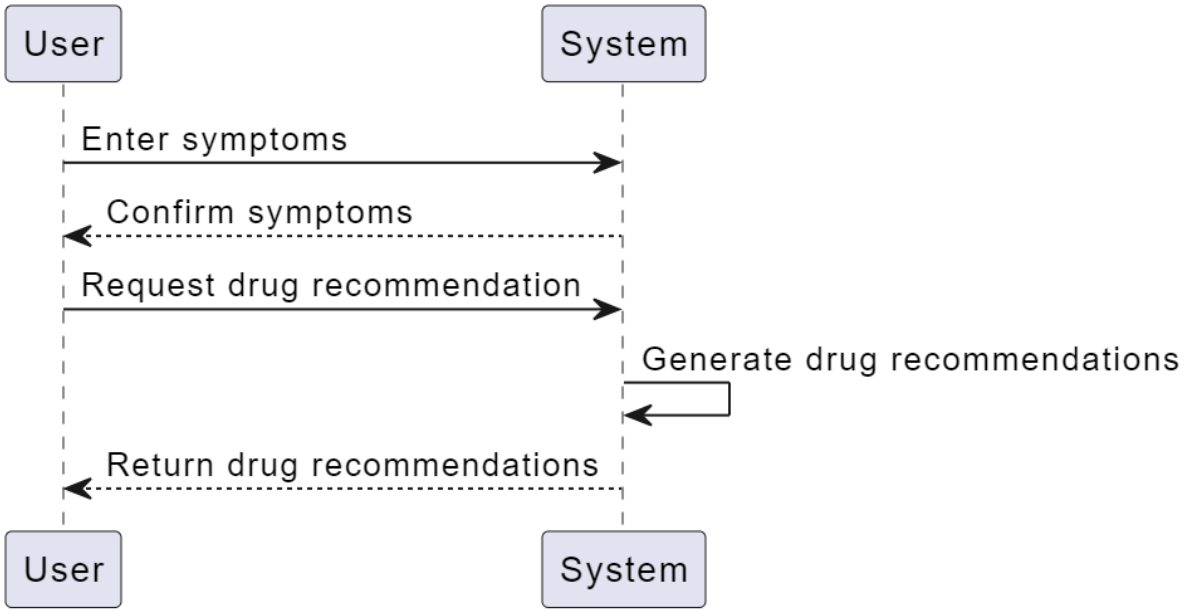

Fig. A.I.3 Sequence Diagram

A.I.4 ACTIVITY DIAGRAM

An activity diagram is a graphical representation of workflows or processes in a system, illustrating the sequence of activities and their transitions. It helps in modeling the dynamic aspects of a system, including decision points, parallel activities, and synchronization. Activity diagrams are essential in software design for modeling business processes, algorithmic workflows, and system behavior, aiding in communication among stakeholders, identifying bottlenecks, and ensuring the clarity of system logic. Key properties include depicting the flow of activities, decision nodes, concurrency, and loops, providing a visual guide for understanding and optimizing system behavior.

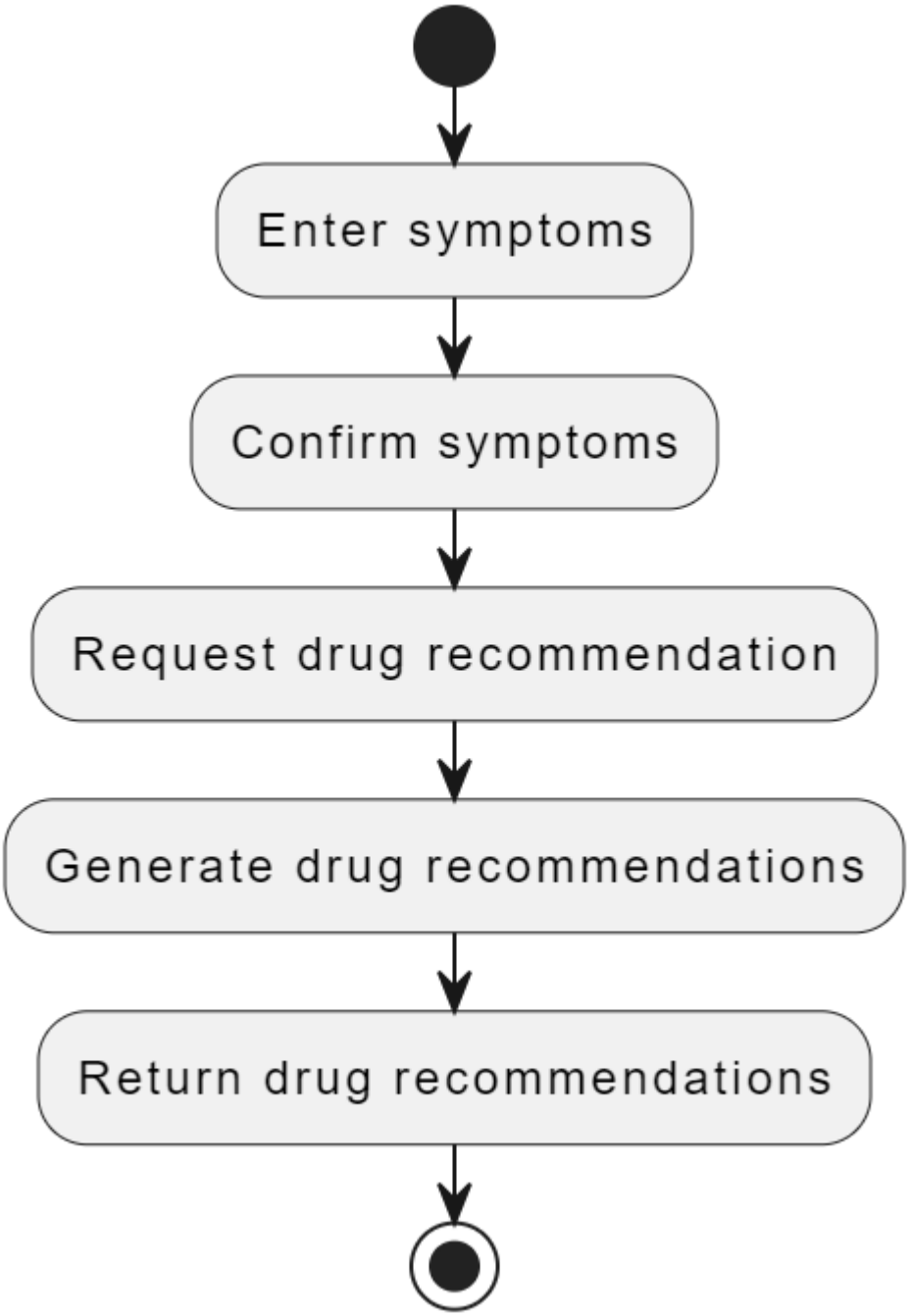

Fig. A.I.4: Activity Diagram

APPENDIX-II

SAMPLE CODE

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from wordcloud import WordCloud

from textblob import TextBlob

from nltk.corpus import stopwords

from collections import Counter

import warnings; warnings.filterwarnings('ignore')

import nltk

import re

import string

from string import punctuation

from nltk import ngrams

from nltk.tokenize import word_tokenize

from nltk.stem import SnowballStemmer


from sklearn import model_selection

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier, plot_importance
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import train_test_split
import spacy



from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk import pos_tag
nltk.download('punkt')
nltk.download('stopwords')

nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
exclude_list = ['but','no','because','against','not','than']
stop_words = set(stopwords.words('english'))
stop_words.difference_update(exclude_list)
df_train = pd.read_csv(r'drugsComTrain_raw.csv')
df_test = pd.read_csv(r'drugsComTest_raw.csv')
```

```
print ("The shape of the train set given is : ", df_train.shape)

print ("The shape of the test set given is : ", df_test.shape)



# Merging the test and train data

df_data = pd.concat([df_train, df_test])


print (df_data.shape)


df_data.head()


df_data.columns = df_data.columns.str.lower()


df_data.columns


df_data.info()


cat = []

num = []

for i in df_data.columns:

    if df_data[i].dtypes=='object':

        cat.append(i)

    else:
```

```
        num.append(i)
print('number of numeric variables are ',len(num))
print('number of categorical variables are ',len(cat))


df_data.describe()


df_data.isnull().sum()


percent_missing = df_data.isnull().sum() * 100 / len(df_data)
missing_value_df = pd.DataFrame({'column_name': df_data.columns, 'percent_missing':
percent_missing})
missing_value_df


print('Shape of df before dropping null values ',df_data.shape)
df_data.dropna(inplace=True)
df_data.reset_index(drop=True)
print('Shape of df after dropping null values ',df_data.shape)


#date is not in datetime format, need to convert it
df_data['date'] = pd.to_datetime(df_data['date'])


df_data.nunique()


# Some of the conditions that people are suffering with
df_data['condition'].value_counts().head(10)
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')

# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [15, 8]

# Create a dictionary with top 10 conditions and their counts
top_conditions = df_data['condition'].value_counts().head(10).to_dict()

# Create bar plot
fig = sns.barplot(x=list(top_conditions.keys()), y=list(top_conditions.values()))

# Set title and axis labels
fig.set_title("Top 10 Conditions")
fig.set_xlabel("Conditions")
fig.set_ylabel("Count")

# Rotate xaxis labels for better readability if necessary
plt.xticks(rotation=45)
# Show plot
```

```
plt.show()

# Top 10 drugs which got good rating(10)

df_data.loc[df_data['rating'] == 10]['drugname'].value_counts().head(10)


import seaborn as sns

import matplotlib.pyplot as plt


# Set seaborn font scale and style

sns.set(font_scale=1.2, style='darkgrid')


# Set matplotlib figure size

plt.rcParams['figure.figsize'] = [20, 8]


# Filter the DataFrame to get drugs with rating 10, count their occurrences, and select the top 10

top_drugs = df_data[df_data['rating'] == 10]['drugname'].value_counts().head(10)

# Create bar plot

fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)


# Set title and axis labels

fig.set_title("Top 10 Drugs for Rating 10")

fig.set_xlabel("Drugs")

fig.set_ylabel("Rating Count")


# Rotate xaxis labels for better readability
```

```
plt.xticks(rotation=30)

# Show plot
plt.show()



# 10 drugs which users are not satisfied with and given 1 ating
df_data.loc[df_data['rating'] == 1]['drugname'].value_counts().head(10)

import seaborn as sns
import matplotlib.pyplot as plt

# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')

# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [20, 8]

# Filter the DataFrame to get drugs with rating 1, count their occurrences, and select the top 10
top_drugs = df_data[df_data['rating'] == 1]['drugname'].value_counts().head(10)
# Create bar plot
fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)

# Set title and axis labels
```

```
fig.set_title("Top 10 Drugs for Rating 1")

fig.set_xlabel("Drugs")

fig.set_ylabel("Rating Count")


# Rotate xaxis labels for better readability

plt.xticks(rotation=30)


# Show plot

plt.show()



sns.set(font_scale = 1.2, style = 'darkgrid')

plt.rcParams['figure.figsize'] = [15, 8]


fig = sns.histplot(data=df_data, x="rating", kde=True)

fig.set_title("Ratings count")

fig.set_xlabel("Rating")

fig.set_ylabel("Count for each rating");


df_data.groupby(['condition'])['drugname'].nunique().sort_values(ascending =
False).head(40).plot.bar(figsize = (19, 7), color = 'green')

plt.title('Most drugs available per Conditions in the Patients', fontsize = 30)

plt.xlabel('Conditions', fontsize = 20)

plt.ylabel('count')

plt.show()
```

```
# now extracting year from date
df_data['year'] = df_data['date'].dt.year

# extracting the month from the date
df_data['month'] = df_data['date'].dt.month

# extracting the days from the date
df_data['day'] = df_data['date'].dt.day

# Check if 'year' column exists in df_data
if 'year' in df_data.columns:
    # Create the count plot using the 'year' column data
    sns.countplot(x='year', data=df_data, palette='dark')

    # Set title and axis labels
    plt.title('The Number of Reviews in Each Year', fontsize=30)
    plt.xlabel('Year', fontsize=15)
    plt.ylabel('Count of Reviews', fontsize=15)

    # Show plot
    plt.show()
else:
    print("The 'year' column does not exist in the DataFrame.")
import seaborn as sns
```

```
# Show plot
plt.show()
# Top 10 drugs which got good rating(10)
df_data.loc[df_data['rating'] == 10]['drugname'].value_counts().head(10)


import seaborn as sns
import matplotlib.pyplot as plt


# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')


# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [20, 8]


# Filter the DataFrame to get drugs with rating 10, count their occurrences, and select the top 10
top_drugs = df_data[df_data['rating'] == 10]['drugname'].value_counts().head(10)
# Create bar plot
fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)


# Set title and axis labels
fig.set_title("Top 10 Drugs for Rating 10")
fig.set_xlabel("Drugs")
fig.set_ylabel("Rating Count")
```

```
# Rotate xaxis labels for better readability
plt.xticks(rotation=30)

# Show plot
plt.show()



# 10 drugs which users are not satisfied with and given 1 ating
df_data.loc[df_data['rating'] == 1]['drugname'].value_counts().head(10)

import seaborn as sns
import matplotlib.pyplot as plt

# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')

# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [20, 8]

# Filter the DataFrame to get drugs with rating 1, count their occurrences, and select the top 10
top_drugs = df_data[df_data['rating'] == 1]['drugname'].value_counts().head(10)
# Create bar plot
fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)
```

```
# Set title and axis labels
fig.set_title("Top 10 Drugs for Rating 1")
fig.set_xlabel("Drugs")
fig.set_ylabel("Rating Count")

# Rotate xaxis labels for better readability
plt.xticks(rotation=30)

# Show plot
plt.show()



sns.set(font_scale = 1.2, style = 'darkgrid')
plt.rcParams['figure.figsize'] = [15, 8]

fig = sns.histplot(data=df_data, x="rating", kde=True)
fig.set_title("Ratings count")
fig.set_xlabel("Rating")
fig.set_ylabel("Count for each rating");

df_data.groupby(['condition'])['drugname'].nunique().sort_values(ascending =
False).head(40).plot.bar(figsize = (19, 7), color = 'green')
plt.title('Most drugs available per Conditions in the Patients', fontsize = 30)
plt.xlabel('Conditions', fontsize = 20)
plt.ylabel('count')
```

```
import matplotlib.pyplot as plt

# Set the figure size
plt.rcParams['figure.figsize'] = (19, 8)

# Check if 'day' column exists in df_data
if 'day' in df_data.columns:
    # Create the count plot using the 'day' column data
    sns.countplot(x='day', data=df_data, palette='colorblind')

    # Set title and axis labels
    plt.title('The Number of Reviews in Each Day', fontsize=30)
    plt.xlabel('Days', fontsize=15)
    plt.ylabel('Count of Reviews', fontsize=15)

# Check if 'day' column exists in df_data
if 'day' in df_data.columns:
    # Create the count plot using the 'day' column data
    sns.countplot(x='day', data=df_data, palette='colorblind')

    # Set title and axis labels
    plt.title('The Number of Reviews in Each Day', fontsize=30)
    plt.xlabel('Days', fontsize=15)
    plt.ylabel('Count of Reviews', fontsize=15)
```

```
df_test = pd.read_csv(r'drugsComTest_raw.csv')


print ("The shape of the train set given is : ", df_train.shape)

print ("The shape of the test set given is : ", df_test.shape)



# Merging the test and train data

df_data = pd.concat([df_train, df_test])


print (df_data.shape)


df_data.head()


df_data.columns = df_data.columns.str.lower()


df_data.columns


df_data.info()


cat = []

num = []

for i in df_data.columns:

    if df_data[i].dtypes=='object':

        cat.append(i)
```

```
    else:
        num.append(i)
print('number of numeric variables are ',len(num))
print('number of categorical variables are ',len(cat))

df_data.describe()

df_data.isnull().sum()

percent_missing = df_data.isnull().sum() * 100 / len(df_data)
missing_value_df = pd.DataFrame({'column_name': df_data.columns, 'percent_missing': percent_missing})
missing_value_df

print('Shape of df before dropping null values ',df_data.shape)
df_data.dropna(inplace=True)
df_data.reset_index(drop=True)
print('Shape of df after dropping null values ',df_data.shape)

#date is not in datetime format, need to convert it
df_data['date'] = pd.to_datetime(df_data['date'])

df_data.nunique()

# Some of the conditions that people are suffering with
```

```
df_data['condition'].value_counts().head(10)


import seaborn as sns

import matplotlib.pyplot as plt


# Set seaborn font scale and style

sns.set(font_scale=1.2, style='darkgrid')


# Set matplotlib figure size

plt.rcParams['figure.figsize'] = [15, 8]


# Create a dictionary with top 10 conditions and their counts

top_conditions = df_data['condition'].value_counts().head(10).to_dict()


# Create bar plot

fig = sns.barplot(x=list(top_conditions.keys()), y=list(top_conditions.values()))


# Set title and axis labels

fig.set_title("Top 10 Conditions")

fig.set_xlabel("Conditions")

fig.set_ylabel("Count")


# Rotate xaxis labels for better readability if necessary

plt.xticks(rotation=45)
```

```
# Show plot
plt.show()
# Top 10 drugs which got good rating(10)
df_data.loc[df_data['rating'] == 10]['drugname'].value_counts().head(10)


import seaborn as sns
import matplotlib.pyplot as plt


# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')


# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [20, 8]


# Filter the DataFrame to get drugs with rating 10, count their occurrences, and select the top 10
top_drugs = df_data[df_data['rating'] == 10]['drugname'].value_counts().head(10)
# Create bar plot
fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)


# Set title and axis labels
fig.set_title("Top 10 Drugs for Rating 10")
fig.set_xlabel("Drugs")
fig.set_ylabel("Rating Count")
```

```
# Rotate xaxis labels for better readability
plt.xticks(rotation=30)

# Show plot
plt.show()



# 10 drugs which users are not satisfied with and given 1 ating
df_data.loc[df_data['rating'] == 1]['drugname'].value_counts().head(10)

import seaborn as sns
import matplotlib.pyplot as plt

# Set seaborn font scale and style
sns.set(font_scale=1.2, style='darkgrid')

# Set matplotlib figure size
plt.rcParams['figure.figsize'] = [20, 8]

# Filter the DataFrame to get drugs with rating 1, count their occurrences, and select the top 10
top_drugs = df_data[df_data['rating'] == 1]['drugname'].value_counts().head(10)
# Create bar plot
fig = sns.barplot(x=top_drugs.index, y=top_drugs.values)
```

```
# Set title and axis labels
fig.set_title("Top 10 Drugs for Rating 1")
fig.set_xlabel("Drugs")
fig.set_ylabel("Rating Count")

# Rotate xaxis labels for better readability
plt.xticks(rotation=30)

# Show plot
plt.show()



sns.set(font_scale = 1.2, style = 'darkgrid')
plt.rcParams['figure.figsize'] = [15, 8]

fig = sns.histplot(data=df_data, x="rating", kde=True)
fig.set_title("Ratings count")
fig.set_xlabel("Rating")
fig.set_ylabel("Count for each rating");

df_data.groupby(['condition'])['drugname'].nunique().sort_values(ascending =
False).head(40).plot.bar(figsize = (19, 7), color = 'green')
plt.title('Most drugs available per Conditions in the Patients', fontsize = 30)
plt.xlabel('Conditions', fontsize = 20)
plt.ylabel('count')
```

```
plt.show()
# now extracting year from date
df_data['year'] = df_data['date'].dt.year

# extracting the month from the date
df_data['month'] = df_data['date'].dt.month

# extracting the days from the date
df_data['day'] = df_data['date'].dt.day

# Check if 'year' column exists in df_data
if 'year' in df_data.columns:
    # Create the count plot using the 'year' column data
    sns.countplot(x='year', data=df_data, palette='dark')

    # Set title and axis labels
    plt.title('The Number of Reviews in Each Year', fontsize=30)
    plt.xlabel('Year', fontsize=15)
    plt.ylabel('Count of Reviews', fontsize=15)

    # Show plot
    plt.show()
else:
    print("The 'year' column does not exist in the DataFrame.")
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the figure size
plt.rcParams['figure.figsize'] = (19, 8)

# Check if 'day' column exists in df_data
if 'day' in df_data.columns:
    # Create the count plot using the 'day' column data
    sns.countplot(x='day', data=df_data, palette='colorblind')

    # Set title and axis labels
    plt.title('The Number of Reviews in Each Day', fontsize=30)
    plt.xlabel('Days', fontsize=15)
    plt.ylabel('Count of Reviews', fontsize=15)
```

APPENDIX-III

PLAGIARISM REPORT

A-07 CVV.pdf

ORIGINALITY REPORT

11%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.javatpoint.com Internet	66 words — 1%
2	dspace.daffodilvarsity.edu.bd:8080 Internet	38 words — 1%
3	ia804701.us.archive.org Internet	35 words — 1%
4	fastercapital.com Internet	27 words — 1%
5	www.geeksforgeeks.org Internet	25 words — < 1%
6	Clift, Jennifer E.. "A Predictive Model to Increase Technology Transfer and Transition", The George Washington University, 2023 ProQuest	23 words — < 1%
7	id.scribd.com Internet	19 words — < 1%
8	robots.net Internet	18 words — < 1%
9	azkurs.org Internet	17 words — < 1%

10	dr.library.brocku.ca Internet	17 words — < 1%
11	www.ijraset.com Internet	17 words — < 1%
12	www.iprism.eu Internet	17 words — < 1%
13	www.michael-grogan.com Internet	16 words — < 1%
14	Ton Duc Thang University Publications	15 words — < 1%
15	deepai.org Internet	15 words — < 1%
16	www.coursehero.com Internet	15 words — < 1%
17	www.mdpi.com Internet	15 words — < 1%
18	assets.researchsquare.com Internet	14 words — < 1%
19	teav.ankara.edu.tr Internet	14 words — < 1%
20	Moshiur Bhuiyan, M.M.Zahidul Islam, Aneesh Krishna, Aditya Ghose. "Integration of Agent-Oriented Conceptual Models and UML Activity Diagrams Using Effect Annotations", 31st Annual International Computer Software and Applications Conference - Vol. 1- (COMPSAC 2007), 2007 Crossref	13 words — < 1%

21	www.arxiv-vanity.com Internet	12 words — < 1%
22	Ramos, Kevin Almeida. "Attribute-Value Inference Using Deep Neural Networks", ISCTE - Instituto Universitario de Lisboa (Portugal), 2023 ProQuest	11 words — < 1%
23	www.emanuelesorbello.com Internet	11 words — < 1%
24	Bharathi Mohan G, Prasanna Kumar R, Elakkiya R, Prasanna Kumar B G. "Medical Recommendations: Leveraging CRNN with Self-Attention Mechanism for Enhanced Systems", 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT), 2023 Crossref	10 words — < 1%
25	Mohammad Shahin, F. Frank Chen, Ali Hosseinzadeh. "Harnessing customized AI to create voice of customer via GPT3.5", Advanced Engineering Informatics, 2024 Crossref	10 words — < 1%
26	dspace.chitkara.edu.in Internet	10 words — < 1%
27	repository.ittelkom-pwt.ac.id Internet	10 words — < 1%
28	dailydrop.hrbrmstr.dev Internet	9 words — < 1%
29	"Security Analysis", Security Engineering for Service-Oriented Architectures, 2009 Crossref	8 words — < 1%

30	machinelearningmodels.org Internet	8 words — < 1%
31	www.iieta.org Internet	8 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF