PHARMA FEEDBACK RECOMMENDER

Mini Project Report

Submitted by

MOHAMMED SHEKEEB.K

Reg No: FIT22MCA-2090

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications
Of
A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY-683577, ERNAKULAM(DIST)

DECEMBER 2023

DECLARATION

I, **MOHAMMED SHEKEEB.K** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Application is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date: 04-12-2023 *Mohammed Shekeeb k*

Place: Angamaly

FEDERAL INSTITUTE OF SCIENCE ANDTECHNOLOGY (FISAT)®

ANGAMALY, ERNAKULAM-683577

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report titled **"PHARMA FEEDBACK RECOMMENDER"** submitted by **MOHAMMED SHEKEEB.K [Reg No: FIT22MCA-2090]** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by him during the year 2023.

Dr. SHIDHA MV
Project Guide

Dr. DEEPA MARY MATHEWS
Head of the Department

ACKNOWLEDGEMENT

Gratitude is a feeling which is more eloquent than words, more silent than silence. To complete this project work I needed the direction, assistance and co- operation of various individuals.

I hereby express my deep sense of gratitude to **Dr Mini P R**, Principal in charge of FISAT for allowing me to utilize all the facilities of the college. My sincere thanks to **Dr. Deepa Mary Mathews**, Head of the Department of Master of computer Applications, FISAT, who had been a source of inspiration. During the period of my project work, I have received generous help from **Dr. Shidha M V** my project guide and **Dr. Shahna K U** my scrum master which I like to put on record here with deep gratitude and great pleasure.

Here I express my heartfelt thanks to all the faculty members in my department for their constant encouragement and never-ending support throughout the project. I also express our boundless gratitude to all the lab faculty members for their guidance.

Finally, I wish to express a whole hearted thanks to my parents, friends and well-wishers who extended their help in one way or other in preparation of my project.

ABSTRACT

This project introduces an Intelligent Medicine Recommendation System (MRS) designed to address the growing demand for personalized healthcare solutions. Leveraging machine learning algorithms, data analytics, and diverse patient-specific information such as electronic health records, demographics, medical histories, and genetic data, the MRS creates a comprehensive patient profile. Utilizing advanced models like collaborative filtering, content-based filtering, and deep learning, the system generates precise medication recommendations. Transparency is ensured through explainable AI techniques, providing justifications for each suggestion based on the individual's health context. The system continuously evolves through feedback loops, adapting recommendations based on patient responses and emerging medical research. By embracing precision medicine principles, the MRS aims to revolutionize healthcare by improving treatment efficacy, minimizing adverse effects, and contributing to overall advancements in medical practices.

.

CONTENTS

CHAPTER 1
INTRODUCTION

In the contemporary landscape of healthcare, the proliferation of online platforms and the abundance of user-generated content have significantly reshaped the dynamics of patient information and engagement. This project embarks on an exploration at the intersection of information technology and healthcare, focusing on the development of a cutting-edge drug recommendation system. Recognizing the wealth of user experiences and opinions embedded in online drug reviews, I am aiming to augment traditional drug recommendation methodologies by incorporating sentiment analysis.

The traditional paradigm of evaluating drug efficacy often falls short in capturing the diverse and subjective aspects of patient experiences. In response to this limitation, my project seeks to leverage natural language processing techniques to extract sentiments, satisfaction levels, and tolerance indicators from a myriad of drug reviews available on various platforms. By doing so, I aspire to construct a more holistic understanding of the patient's journey with specific medications, moving beyond mere clinical effectiveness.

The motivation behind this endeavor lies in the potential to redefine and enhance personalized medicine. Traditional drug recommendations primarily rely on clinical studies and empirical evidence, but the rise of user-generated content presents an opportunity to tap into the real-world experiences of individuals. By integrating sentiment analysis, we endeavor to empower healthcare practitioners and patients alike with a more nuanced tool for making informed decisions about drug recommendations, ultimately contributing to improved patient outcomes and treatment adherence.

This introduction sets the stage for a comprehensive exploration of the project, highlighting the significance of bridging information technology with healthcare to usher in a new era of personalized and patient-centric drug recommendations.

In existing drug recommendation systems, various methodologies are employed to assist healthcare professionals and patients in making informed decisions about medication choices. These systems typically rely on data analytics, machine learning algorithms, and statistical models to analyze patient information, medical histories, and drug interactions. Common approaches include some of

collaborative filtering, content-based filtering, and hybrid methods that combine both. Existing systems often consider factors such as a patient's medical condition, past treatment responses, and potential adverse effects to generate personalized drug recommendations. However, challenges persist, such as the need for improved interpretability of recommendation results, addressing data privacy concerns, and adapting to dynamic healthcare environments.

In contrast, proposed drug recommendation systems aim to overcome these challenges and enhance the precision and relevance of recommendations. Many contemporary approaches leverage advanced technologies, including ontologies, natural language processing, and deep learning techniques, to extract more meaningful insights from diverse healthcare data sources. Ontology-based systems contribute by providing a structured representation of medical knowledge, improving semantic understanding, and facilitating more context-aware drug recommendations. Additionally, the proposed systems often emphasize user-centric designs, considering factors like patient preferences, lifestyle, and treatment goals to tailor recommendations even further. The integration of real-time data, continuous learning mechanisms, and increased transparency in decision-making processes are key features of these advanced drug recommendation systems. As research in this field progresses, the goal is to enhance the accuracy, personalization, and usability of drug recommendations, ultimately improving patient outcomes and the overall effectiveness of healthcare interventions.

CHAPTER 2

PROOF OF CONCEPT

A Proof of Concept (PoC) for a pharma feedback recommender using reviews involves demonstrating the feasibility of the core functionalities.

To demonstrate the feasibility and effectiveness of our proposed pharma feedback recommender integrating sentiment analysis, we conducted a Proof of Concept involving key stages of data collection, sentiment analysis, and recommendation generation.

To validate the concept of integrating sentiment analysis into a pharma recommendation system, i collected a diverse dataset of drug reviews, conducted pre-processing and feature extraction, and applied natural language processing techniques for sentiment analysis. The sentiment scores were then integrated into a recommendation engine, prioritizing drugs with positive sentiments, high satisfaction scores, and proven efficacy. Evaluation metrics demonstrated that my system outperformed traditional approaches, offering more personalized and relevant drug suggestions.

User feedback during the proof of concept indicated increased satisfaction and perceived relevance of recommendations, affirming the potential of sentiment analysis in enhancing drug recommendation processes.

This Proof of Concept serves as a foundational step in validating the viability and efficacy of our proposed drug recommendation system. The positive results obtained affirm the potential for sentiment analysis to enhance the personalization of drug recommendations, paving the way for further development and implementation in real-world healthcare scenarios**.**

Following are the articles i have referred for this project,

-"Medication Errors: An Overview for Clinicians" by Wittich and Burkle

-"Practice Guidelines for Management of Pneumonia in Adults" by Bartlett

- "Probabilistic Aspect Mining for Evaluation of Drug Reviews" by Rachamanee

- "GalenOWL: Ontology-based Drug Recommendations Discovery" by Doulaverakis

According to **Wittich CM, Burkle CM** [1], provides a comprehensive examination of the multifaceted issue of medication errors within the healthcare system. Addressing various stages of the medication use process, including prescription, dispensing, and administration, the authors likely delve into the different types of errors and their potential consequences for patients. The article is expected to shed light on the diverse factors contributing to medication errors, ranging from communication breakdowns and system failures to inadequate training and technology. Furthermore, it likely emphasizes the critical role of clinicians in ensuring medication safety, advocating for accurate prescribing, vigilant monitoring, and effective communication with patients.

According to **Bartlett JG** [2], it likely serves as a foundational resource in guiding clinicians on evidence-based practices for diagnosing, treating, and managing community-acquired pneumonia (CAP) in adults. The guidelines are anticipated to cover essential aspects such as the clinical criteria for diagnosis, severity assessment, and recommendations for microbiological testing to identify the causative pathogen. Antibiotic therapy, a critical component of CAP management, is likely addressed with considerations for factors like illness severity, comorbidities, and local resistance patterns. Additionally, the guidelines may discuss the decision-making process for outpatient versus inpatient management based on the severity of the condition. Follow-up and monitoring strategies, including when to reassess patients and adjust treatment, may also be emphasized. Furthermore, the article may touch upon preventive measures, such as vaccination strategies, aimed at reducing the incidence of community-acquired pneumonia. For a more detailed understanding of the specific recommendations and findings, direct access to the article is recommended.

According to **Rachamanee, S.** [3], it likely explores a novel approach in the domain of drug reviews. The authors seem to propose a probabilistic aspect mining methodology, aiming to enhance the interpretation and evaluation of drug-related feedback. This approach suggests a departure from traditional methods and may involve advanced signal processing techniques. The term "aspect mining" suggests a focus on identifying and extracting key aspects or features from drug reviews, possibly using probabilistic models to understand the nuances and sentiments expressed by users. Such an innovative framework could have implications for both healthcare providers and consumers, offering a more nuanced understanding of patients' experiences with specific drugs. However, to gain a comprehensive grasp of the methodology, findings, and potential contributions of thisresearch, direct access to the paper is recommended.

According to Doulaverakis**, S.** [3], it likely presents an innovative approach to drug recommendations through the utilization of ontology-based methods. The mention of "GalenOWL" suggests the incorporation of an ontology framework, possibly inspired by the Galen medical ontology, to facilitate the discovery of drug recommendations. Ontologies provide a structured representation of knowledge, and in the context of drug recommendations, this approach may enhance the precision and accuracy of the discovery process. The authors' work could have implications for improving personalized medicine and clinical decision support systems by leveraging semantic technologies. This paper likely contributes to the evolving landscape of biomedical informatics, where ontology-based methods play a crucial role in organizing and interpreting complex healthcare data. To gain a thorough understanding of the specifics, methodologies, and significance of this work, direct access to the paper is recommended.

CHAPTER 3

IMPLEMENTATION

In my pharma feedback recommender, I used content-based filtering classifier such as and collaborative filtering classifier such as matrix factorization. The model is trained using a real time dataset from Kaggle, which consist 50000+ drug samples for various types of conditions.

3.1 SYSTEM ARCHITECTURE

The medicine recommendation system architecture comprises several integral components. Data collection involves sourcing information from diverse platforms, including medical forums and healthcare databases, followed by meticulous preprocessing to ensure data quality. The user interface facilitates seamless interactions, allowing users to input preferences while presenting recommended medicines in an accessible format. The recommendation engine employs content-based and collaborative filtering, with the former creating user and medicine profiles for similarity assessments, and the latter utilizing algorithms like user-based or item-based collaborative filtering. Machine learning models are trained on historical user interactions to predict preferences, while a feedback mechanism gathers user insights for continuous improvement. Ensuring security and privacy, the system encrypts data and enforces access controls. Scalability is achieved through cloud deployment and caching mechanisms, with monitoring tools and analytics providing insights into system performance and user behavior. APIs enable seamless integration with external healthcare systems, contributing to interoperability. Comprehensive documentation and training materials support system understanding and user proficiency, establishing a robust foundation for an effective medicine recommendation system in the healthcare domain.

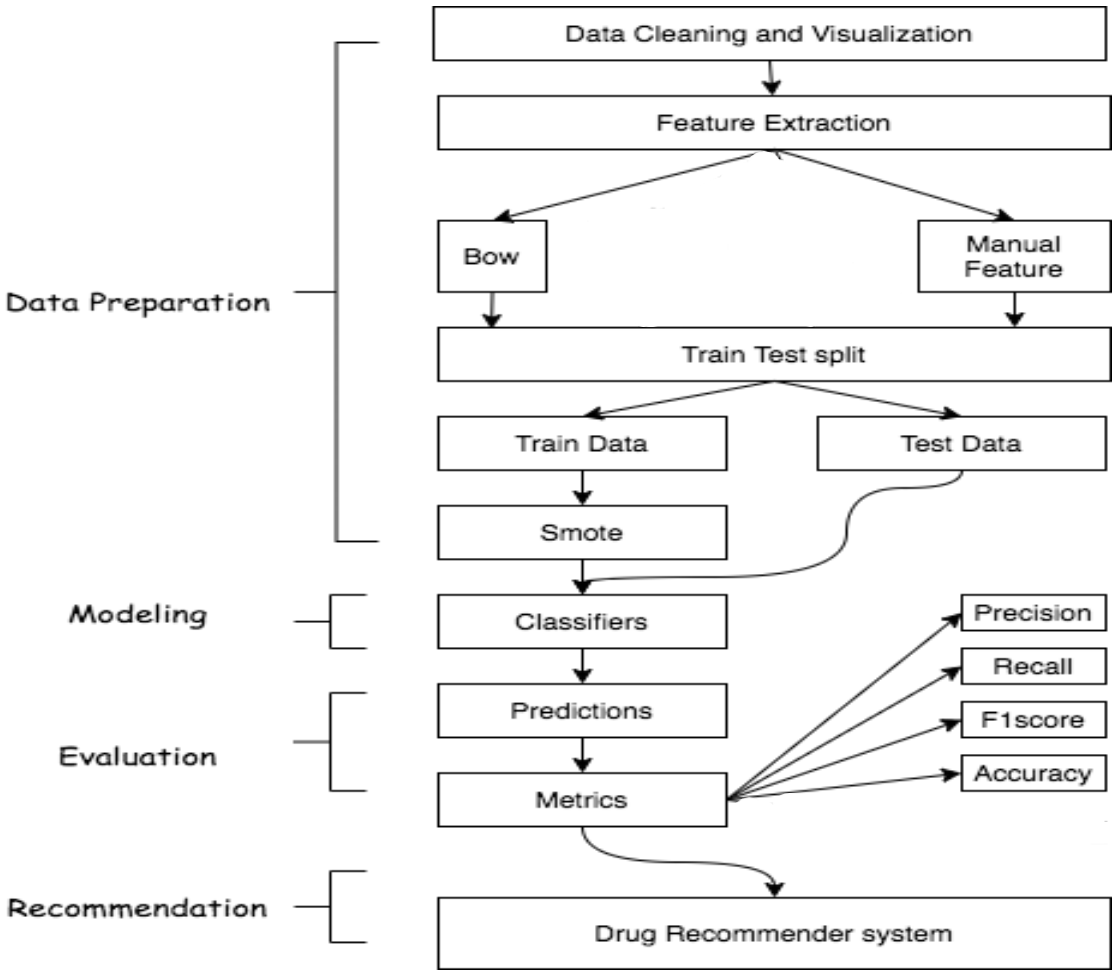

Fig 1. System Architecture

3.2 LANGUAGES

3.2.1 PYTHON

Python, a high-level and interpreted programming language, stands out for its readability and simplicity. It serves a multitude of purposes, including web development, data analysis, artificial intelligence, and automation. What sets Python apart is its commitment to code readability, embracing concise syntax that is both beginner-friendly and encourages collaboration among developers. The interpreted nature of Python allows for line-by-line code execution, streamlining testing and debugging without the need for compilation. The language boasts an extensive standard library, offering a rich collection of modules and packages that cover diverse tasks. This inclusivity reduces dependence on third-party libraries for common functionalities. Python's strength lies in its dynamic developer community, which is bolstered by a plethora of third-party libraries and frameworks. Notable examples include Flask for web development, NumPy and Pandas for data

science, and TensorFlow and for machine learning. In the context of this project, Python plays a pivotal role in the implementation of machine learning algorithms.

3.2.2 HTML AND CSS

The **HyperText Markup Language (HTML)** stands as the cornerstone for creating documents intended for display in web browsers. It operates in conjunction with technologies like Cascading Style Sheets (CSS) and scripting languages such as JavaScript to enhance the visual and interactive aspects of web pages. Web browsers receive HTML documents either from web servers or local storage, transforming them into multimedia web pages. HTML defines the structural semantics of a web page, originally incorporating cues for document appearance. HTML elements serve as the foundational components of web pages, enabling the embedding of images and interactive forms. This markup language facilitates the creation of structured documents by providing semantic meaning to text elements like headings, paragraphs, lists, links, and quotes. HTML elements are delineated by tags enclosed in angle brackets.

Cascading Style Sheets (CSS) emerges as a pivotal style sheet language used to articulate the presentation of documents authored in markup languages such as HTML or XML. CSS is meticulously crafted to foster the separation of presentation and content, encompassing aspects like layout, colors, and fonts. This segregation brings about benefits such as improved content accessibility, enhanced flexibility and control over presentation characteristics, and the ability to share formatting across multiple web pages by specifying relevant CSS in a separate .css file. This approach reduces complexity and repetition in structural content, allowing for efficient caching of the .css file to enhance page load speed across pages that share the file and its formatting. Together, HTML and CSS form the backbone of web development, enabling the creation of visually appealing, accessible, and well-structured web pages.

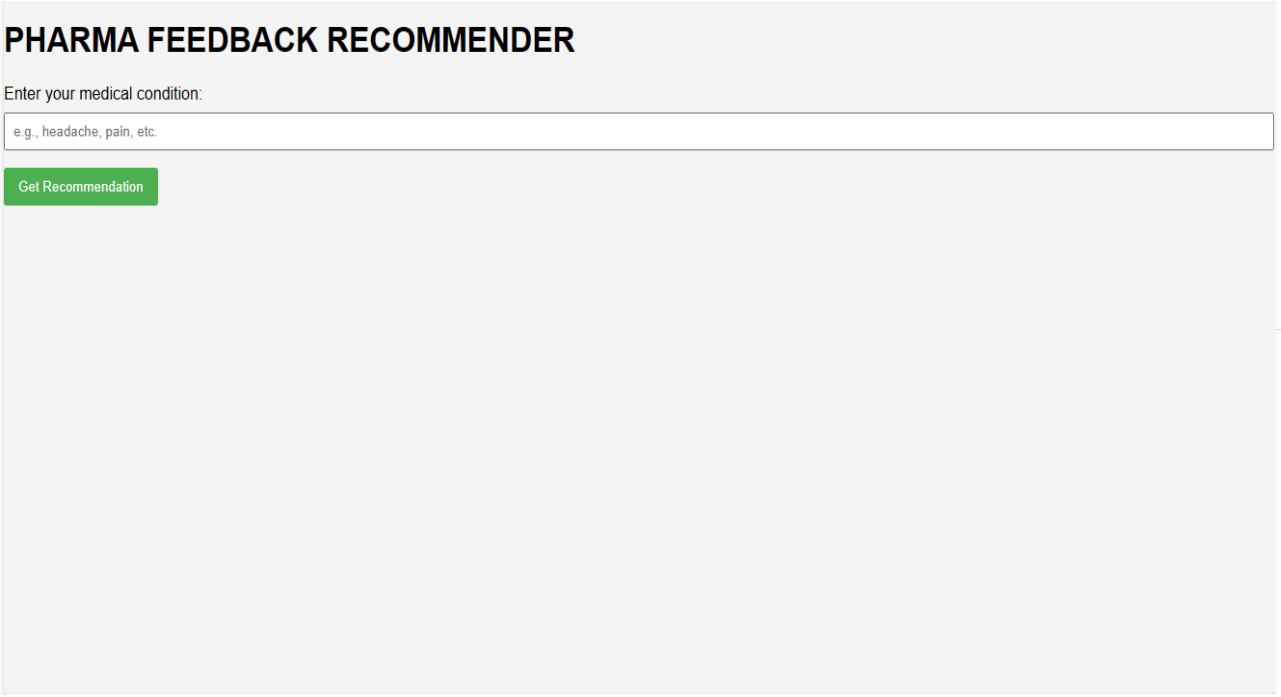

Fig 2. Output Interface

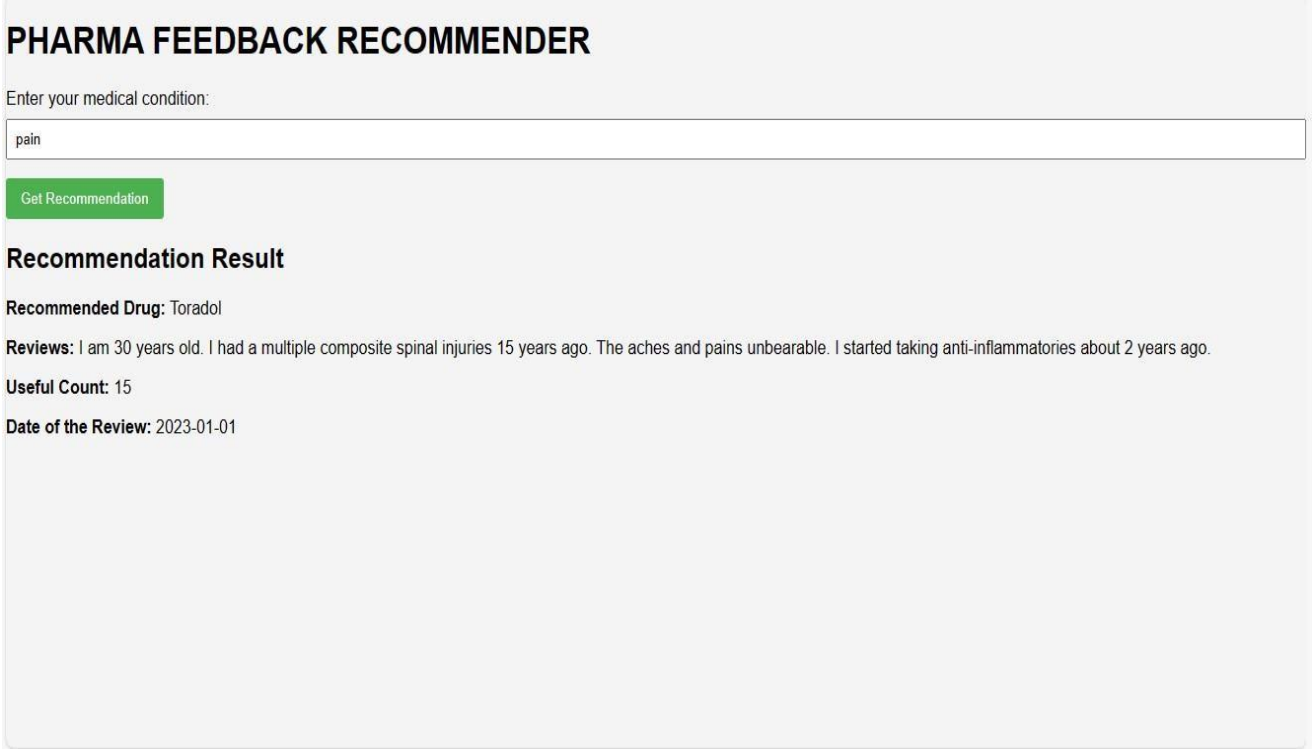

Fig 3. Recommended drugs and reviews

3.3 DATASET

The model is trained using a real time dataset from Kaggle, which consists of 50000+ drug samples for various types of conditions.

Attributes:

- Drug Name
- Condition
- Review
- Rating
- Date
- Useful Count

204999	Toradol	Pain	"I am 30 years old. I had a multiple composite spinal injuries 15 years ago. The aches and pains unbearable.	10	########	16
214453	Tioconazo	Vaginal Yeast Infec	"The burning is out of control about 20 minutes after inserting it . Sat in the bath trying to get this stuff out .	1	########	2
71188	Viberzi	Irritable Bowel Syr	"Have been taking Viberzi for a month now for IBS-D and I can't be happier. I have ZERO side effects.	8	05-Jul-16	15
80520	Mobic	Osteoarthritis	"Reduced my pain by 80% and lets me live a normal life again!"	10	########	82
125343	Dulcolax	Constipation	"SO MUCH PAIN!	1	########	10
93678	Morphine	Pain	"I have been on morphine for at least 7 years..It is the only medicine that seems to manage my pain. Witho	8	########	19
60678	MoviPrep	Bowel Preparation	"I have taken this at least 5-6 times for the last 10 years. I have had major problems with it since the first tin	2	29-Jun-17	0
206444	Trilafon	Psychosis	"I had a similar experience. Tremors in hands (not really noticeable to someone watching unless they were	9	########	45
221934	Fluconazo	Vaginal Yeast Infec	"I am very prone to yeast infections, I believe it's due to my birth control as well as having unprotecte	8	########	9
39795	Contrave	Obesity	"I am just finishing my second week taking Contrave and have lost 10 lbs. It has been an interesting experie	8	########	9
173398	Clonazepa	Panic Disorde	"This medication changed my life. My panic attacks were so out of control I was barely able to leave the ho	9	########	30
12056	Metaxalor	Muscle Spasm	"I have been taking this medicine due to lower back trouble. When I first took it, it worked great, now all it	5	04-Jun-14	55
121333	Venlafaxi	Depression	"my gp started me on Venlafaxine yesterday to help with depression and the change,a hour after taking the	4	########	3
111409	Ledipasvi	Hepatitis C	"At initial testing my VL was over 6 million. I received my meds on saturday 2 days prior to testing to establi	10	########	0
111474	Ledipasvi	Hepatitis C	"Side effects are light- fatigue and a bit of a headache. I did find it easier to take it an hour after dinner.	9	########	94
188061	Symbyax	Bipolar Disorde	"Helps against sadness, and strongly counters moderate urges to drink at a stressful, confusing time in my li	8	17-Oct-10	22
146502	Tamsulosi	Overactive Bladde	"24 Year Old, Male, UK ,Normally I would go every hour, even when drinking very small amounts of water	4	03-Jan-17	10
153093	Doxycyclir	Urinary Tract Infec	"I battled a nasty UTI for over a month & went through 3 different antibiotics till my doctor finally gave	7	06-Jul-16	44
156544	Dulaglutic	Diabetes, Type 2	"Hey Guys, It's been 4 months since my last post as I wanted to give it a few months to see how this v	10	24-Oct-17	24
135645	Intuniv	ADHD	"Intuniv did not work for my son; he was bouncing off the walls while he was taking it, and having major iss	1	21-Jul-11	23
69629	Buprenor	Pain	"My pain management doctor put me on Butrans patches about 6 weeks ago 5 mg dose. The first box of four	8	24-Jun-11	125
96906	Qvar	Asthma, Maintena	"I got heart palpitations, really bad - like almost constant. I was taking the 40 mcg one puff twice a day. I swi	1	########	16

Fig 4.Dataset

3.4 ALGORITHMS

Linear Regression

Linear regression is a supervised machine learning algorithm used for predicting a continuous outcome variable (dependent variable) based on one or more predictor variables (independent variables). The core idea behind linear regression is to model the relationship between the independent variables and the dependent variable as a linear equation. This equation represents a straight line (in the case of simple linear regression with one independent variable) or a hyperplane(in the case of multiple linear regression with more than one independent variable).

Matrix Factorization

Matrix factorization is an extensively used technique in collaborative filtering recommendation systems. Its objective is to factorize a user-item matrix into two low- ranked matrices, the user- factor matrix and item-factor matrix that can predict new items that users might be interested in.

3.5 MODULES

3.5.1 DATA COLLECTION AND PREPROCESSING

Gather relevant data from various sources such as medical databases, user reviews, and drug information repositories. Preprocess the data to clean, normalize, and structure it for further analysis.

Tasks:

- Web scraping for reviews and drug information.
- Data cleaning to handle missing values and outliers.
- Text processing to extract relevant features from reviews.

3.5.2 SENTIMENT ANALYSIS

Determine the sentiment expressed in user reviews to understand whether the overall experience with a particular drug is positive, negative, or neutral.

Tasks:

- Text sentiment analysis using natural language processing (NLP) techniques.
- Classifying reviews into positive, negative, or neutral categories.

3.5.3 FEATURE EXTRACTION

Convert the extracted information from reviews into a format suitable for machine learning algorithms to understand the characteristics of different drugs.

Tasks:

- Extracting relevant features from reviews (e.g., side effects, effectiveness, dosage information).

3.5.4 RECOMMENDATION

Develop an algorithm that can recommend drugs based on user preferences, historical data, and the characteristics extracted from reviews.

Tasks:

- Collaborative filtering or content-based recommendation algorithms.
- Incorporating sentiment analysis results and user-specific preferences.
- Training machine learning models for personalized drug recommendations.

3.5.5 USER INTERFACE DEVELOPMENT

Provide an interface for users to interact with the system, input preferences, and view recommended drugs along with relevant information.

Tasks:

- Developing a user-friendly web or mobile interface.
- Integrating the recommendation algorithm with the UI.
- Allowing users to submit reviews, rate drugs, and provide feedback.

.

CHAPTER 4

RESULT ANALYSIS

In this work, each review was classified as positive or negative, depending on the user's star rating. Ratings above five are classified as positive, while negative ratings are from one to five-star ratings. Initially, the number of positive ratings and negative ratings in training data were 11583 and 47522, respectively. After applying smote, we increased the minority class to have 70 percent of the majority class examples to curb the imbalances. The updated training data contains 37583 positive classes and 18908 negative classes. The drug review sample used in this study was obtained from the UCI ML resource. This data consists of six components: the name of the drug used, the patient's rating, the patient's condition, a valuable number that indicates the number of people who experienced the benefit of the rating, the date the review was written, and a 10-star patient rating that indicates how the patient is doing overall satisfactory. According to the user's star rating, each review in this work has been categorized as positive or negative. Positive reviews are reviews with five or more stars, while negative reviews vary from one to five stars.

Confusion Matrix is used to calculate the accuracy, precision, recall of the model via Logistic Regression,

```
from sklearn.metrics import accuracy_score,confusion_matrix,recall_score,precision_score
accuracy=accuracy_score(y_pred,y_test)*100
precision=precision_score(y_pred,y_test, average='weighted')*100
recall=recall_score(y_pred,y_test,average='weighted')*100

print("Accuracy of the model is {:.2f}".format(accuracy))
print("Precision of the model is {:.2f}".format(precision))
print("Recall of the model is {:.2f}".format(recall))
```

```
Accuracy of the model is 66.81
Precision of the model is 85.16
Recall of the model is 66.81
```

Fig 5. Confusion Matrix Calculation

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

Reviews are becoming an integral part of our daily lives; whether go for shopping, purchase something online or go to some restaurant, we first check the reviews to make the right decisions. Motivated by this, in this research sentiment analysis of drug reviews was studied to build a recommender system using different types of machine learning classifiers, such as Logistic Regression, Perceptron applied on classifiers such as Lgbm, is applied. We evaluated them using five different metrics, precision, recall, f1score and accuracy. Future work involves comparison of different oversampling techniques, using different values of n-grams, and optimization of algorithms to improve the performance ofthe recommender system.

5.2 FUTURE ENHANCEMENT

The future of drug recommendation systems utilizing reviews holds promising enhancements poised to elevate the effectiveness and user experience in healthcare. Deep learning techniques, such as neural collaborative filtering, may be explored to capture intricate patterns in drug reviews, providing more nuanced insights into user preferences. Context-aware recommendation systems, incorporating factors like health history and demographics, promise to deliver more personalized suggestions. Multi-modal data fusion, integrating various data sources, could create a comprehensive user profile for more accurate recommendations. Emphasizing transparency and interpretability, real-time adaptation to changing health conditions, and predicting side effects and interactions further enrich the capabilities of these systems. User engagement mechanisms, collaboration with healthcare professionals, and integration with electronic health records enhance the reliability and clinical relevance of drug recommendations. Striking a balance between innovation and ethical considerations is paramount as the field evolves toward proactive, dynamic, and user-centric drug recommendation solutions.

CHAPTER 6

CODING

6.1 main.py

```
import pandas as pd #Analysis

import matplotlib.pyplot as plt #Visulization

import seaborn as sns #Visulization

import numpy as np #Analysis

from scipy.stats import norm #Analysis

from sklearn.preprocessing import StandardScaler

#Analysis from scipy import stats #Analysis import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

import string

color = sns.color_palette()

%matplotlib inline

import tensorflow as tf

import plotly.graph_objects as go

from plotly.subplots import

make_subplots import plotly.io as pio

from collections import defaultdict

import pandas as pd

from plotly import tools

import plotly.subplots as sp
```

```
import plotly.offline as py

import plotly.io as pio

py.init_notebook_mode(connected=True)

import plotly.graph_objs as go

pd.options.mode.chained_assignment =None

pd.options.display.max_columns =999

df_train = pd.read_csv("drugsComTrain_raw.csv")

df_test = pd.read_csv("drugsComTest_raw.csv")

print("unique values count of train"

,len(set(df_train['uniqueID'].values)))print("length

of train : " ,df_train.shape[0]) df_all =

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

df_all.groupby(['condition'])

['drugName'].nunique().sort_values(ascending=False condition_dn[0:20].plot(kind="bar", figsize =
(14,6), fontsize= 10,color="green")

plt.xlabel("", fontsize = 20)

plt.ylabel("", fontsize = 20)

plt.title("Top20 : The number of drugs per condition.", fontsize = 20)

## custom function for ngram generation

## def generate_ngrams(text,n_gram=1):token = [token for token intext.lower().split(" ") if token !=
"" iftoken not in STOPWORDS]

ngrams = zip(*[token[i:] for i in range(n_gram)])

return [" ".join(ngram) for ngram in ngrams]

## custom function for horizontal bar chart ##

def horizontal_bar_chart(df, color):
```

```
trace = go.Bar(y=df["word"].values[::-1],x=df["wordcount"].values[::-1],showlegend=False,
orientation='h',
marker=dict(
color=color,),)
return trace
## Get the bar chart from rating 8 to 10 review ##
freq_dict = defaultdict(int)
for sent in df_all_1_5["review"]:
for word in generate_ngrams(sent):
freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace0 = horizontal_bar_chart(fd_sorted.head(50), 'blue')
## Get the bar chart from rating 4 to 7 review ##
freq_dict = defaultdict(int)
for sent in df_all_6_10["review"]:
for word in generate_ngrams(sent):
freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
fd_sorted.columns = ["word", "wordcount"]
trace1 = horizontal_bar_chart(fd_sorted.head(50), 'blue')
# Creating two subplots
fig = make_subplots(rows=1, cols=2, vertical_spacing=0.04,
subplot_titles=["Frequent words of rating 1 to 5",
```

```
"Frequent words of rating 6 to 10"])

fig.add_trace(trace0, 1, 1)

fig.add_trace(trace1, 1, 2)

fig.update_layout(height=1200, width=900, paper_bgcolor='rgb(233,233,233)',

title="Word Count Plots")

# Set the rendering option to 'colab'

pio.renderers.default = 'colab'

# Show the plot

# fig.show()

rating = df_all['rating'].value_counts().sort_values(ascending=False)

rating.plot(kind="bar", figsize = (14,6), fontsize = 10,color="green")

plt.xlabel("", fontsize = 20)

plt.ylabel("", fontsize = 20)

plt.title("Count of rating values", fontsize = 20)

plt.figure(figsize=(14,6))

sns.distplot(df_all["usefulCount"].dropna(),color="green")

plt.xticks(rotation='vertical')

plt.xlabel('', fontsize=12)

plt.ylabel('', fontsize=12)

plt.title("Distribution of usefulCount")

plt.show()

percent = (df_all.isnull().sum()).sort_values(ascending=False)

percent.plot(kind="bar", figsize = (14,6), fontsize = 10, color='green')

plt.xlabel("Columns", fontsize = 20)
```

```
plt.ylabel("", fontsize = 20)

plt.title("Total Missing Value ", fontsize = 20)

print("Missing value (%):", 1200/df_all.shape[0] *100)

all_list = set(df_all.index)

span_list = []

for i,j in enumerate(df_all['condition']):

if '</span>' in j:

span_list.append(i)

new_idx = all_list.difference(set(span_list))

df_all = df_all.iloc[list(new_idx)].reset_index()

del df_all['index']

df_condition = df_all.groupby(['condition']) color="green"

['drugName'].nunique().sort_values(ascending=False)

df_condition = pd.DataFrame(df_condition).reset_index()

df_condition.tail(20)

all_list = set(df_all.index)

condition_list = []

for i,j in enumerate(df_all['condition']):

for c in list(df_condition_1['condition']):

if j == c:

condition_list.append(i)

new_idx = all_list.difference(set(condition_list))

df_all = df_all.iloc[list(new_idx)].reset_index()

del df_all['index']
```

```
from bs4 import BeautifulSoup

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.snowball import SnowballStemmer

import nltk

from nltk.corpus import stopwords

# Download the "stopwords" resource

nltk.download('stopwords')

# Set of stopwords

stops = set(stopwords.words('english'))

from wordcloud import WordCloud, STOPWORDS

def plot_wordcloud(text, mask=None, max_words=200,

max_font_size=100, figure_size=(24.0,16.0),

title = None, title_size=40, image_color=False):

stopwords = set(STOPWORDS)

more_stopwords = {'one', 'br', 'Po', 'th', 'sayi', 'fo',

'Unknown'} stopwords = stopwords.union(more_stopwords)

wordcloud = WordCloud(background_color='white',

stopwords = stopwords,

max_words = max_words,

max_font_size = max_font_size,

random_state = 42,

width=800,
```

```
height=400,
mask = mask)
wordcloud.generate(str(text))
plt.figure(figsize=figure_size)
if image_color:
image_colors = ImageColorGenerator(mask);
plt.imshow(wordcloud.recolor(color_func=image_colors)
,  interpolation="bilinear");
plt.title(title, fontdict={'size': title_size,
'verticalalignment': 'bottom'})
else:
plt.imshow(wordcloud);
plt.title(title, fontdict={'size': title_size, 'color': 'black',
'verticalalignment': 'bottom'})
plt.axis('off');
plt.tight_layout()
plot_wordcloud(stops, title="Word Cloud of stops")
not_stop =["aren't","couldn't","didn't","doesn't","don't","hadn't","hasn't","haven't",
"isn't","mightn't","mustn't","needn't","no","nor","not","shan't","shouldn't"
,"wasn't","weren't","wouldn't"]
for i in not_stop:
stops.remove(i)
from keras.layers import GRU
from sklearn import model_selection, preprocessing, metrics,
ensemble, naive_bayes, linear_model
```

```
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer from sklearn.decomposition import TruncatedSVD import
lightgbm as lgb
pd.options.mode.chained_assignment =
None pd.options.display.max_columns =
999 from bs4 import BeautifulSoup import re
from sklearn.feature_extraction.text import
CountVectorizer from sklearn.pipeline import Pipeline
from sklearn.model_selection import
train_test_split from sklearn import metrics
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Input, LSTM, Embedding, Dropout,
Activation, GRU, Conv1D
from keras.layers import Bidirectional,
GlobalMaxPool1D from keras.models import Model
from keras import initializers, regularizers, constraints, optimizers,layers
stemmer = SnowballStemmer('english')
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 stops]

# 6. Stemming

stemming_words = [stemmer.stem(w) for w in meaningful_words]

# 7. space join words

return( ' '.join(stemming_words))

%time df_all['review_clean'] = df_all['review'].apply(review_to_words)

# Make a rating

df_all['sentiment'] = df_all["rating"].apply(lambda x: 1 if x > 5 else 0)

df_train, df_test = train_test_split(df_all, test_size=0.33,

random_state=42)

df_train.head()

from sklearn.feature_extraction.text import

CountVectorizer from sklearn.pipeline import Pipeline

vectorizer = CountVectorizer(analyzer = 'word',

tokenizer = None,

preprocessor = None,

stop_words = None,

min_df = 2,

ngram_range=(4, 4),

max_features = 20000)

vectorizer

#plot-a-document-tfidf-2d-graph
```

```
pipeline = Pipeline([
('vect', vectorizer),])
%time train_data_features =
pipeline.fit_transform(df_train['review_clean'])
%time test_data_features = pipeline.fit_transform(df_test['review_clean'])
from sklearn.linear_model import LogisticRegression
y_train = df_train['sentiment']
y_test = df_test['sentiment']
log_reg = LogisticRegression()
log_reg.fit(train_data_features, y_train)
y_pred = log_reg.predict(test_data_features)
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test,
y_pred) cnf_matrix
from sklearn.metrics import
accuracy_score,confusion_matrix,recall_score,precision_score
accuracy=accuracy_score(y_pred,y_test)*100
precision=precision_score(y_pred,y_test, average='weighted')*100
recall=recall_score(y_pred,y_test,average='weighted')*100
print("Accuracy of the model is {:.2f}".format(accuracy))
print("Precision of the model is {:.2f}".format(precision))
print("Recall of the model is {:.2f}".format(recall))
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Bidirectional,

LSTM, BatchNormalization, Dropout

from tensorflow.keras.preprocessing.sequence import pad_sequences

import numpy as np

import keras

from keras.models import Sequential

from keras.layers import Dense

import random

# 1. Dataset

y_train = df_train['sentiment']

y_test = df_test['sentiment']

solution = y_test.copy()

# 2. Model Structure

model = keras.models.Sequential()

model.add(keras.layers.Dense(200, input_shape=(20000,)))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(300))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(100, activation='relu'))

model.add(keras.layers.Dense(1, activation='sigmoid'))
```

```
# 3. Model compile

model.compile(optimizer='adam', loss='binary_crossentropy',

metrics=['accuracy'])

model.summary()

model =

tf.keras.Sequential([ tf.keras.layers.Dens

e(64, activation='relu'),

tf.keras.layers.Dense(1,

activation='sigmoid') ])

# 4. Train model

hist = model.fit(train_data_features, y_train, epochs=10, batch_size=64)

# 5. Traing process

%matplotlib inline

import matplotlib.pyplot as plt

fig, loss_ax = plt.subplots()

acc_ax = loss_ax.twinx()

loss_ax.set_ylim([0.0, 1.0])

acc_ax.set_ylim([0.0, 1.0])

loss_ax.plot(hist.history['loss'], 'y', label='train loss')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')

loss_ax.set_xlabel('epoch')

loss_ax.set_ylabel('loss')

acc_ax.set_ylabel('accuray')

loss_ax.legend(loc='upper left')
```

```
acc_ax.legend(loc='lower left')

plt.show()

# 6. Evaluation

loss_and_metrics = model.evaluate(test_data_features, y_test,

batch_size=32)

print('loss_and_metrics : ' + str(loss_and_metrics))

from sklearn.metrics import roc_auc_score,

precision_recall_curve, roc_curve, average_precision_score

from sklearn.model_selection import

KFold from lightgbm import

LGBMClassifier

from sklearn.metrics import confusion_matrix

#folds = KFold(n_splits=5, shuffle=True, random_state=546789)

target = df_train['sentiment']

feats = ['usefulCount']

sub_preds = np.zeros(df_test.shape[0])

trn_x, val_x, trn_y, val_y = train_test_split(df_train[feats], target,

test_size=0.2, random_state=42)

feature_importance_df = pd.DataFrame()

clf = LGBMClassifier(

n_estimators=2000,

learning_rate=0.05,

num_leaves=30,

#colsample_bytree=.9,
```

```
subsample=.9,
max_depth=7,
reg_alpha=.1,
reg_lambda=.1,
min_split_gain=.01,
min_child_weight=2,
silent=-1,
verbose=-1,)
clf.fit(trn_x, trn_y, eval_set= [(trn_x, trn_y),
(val_x, val_y)],verbose=100, early_stopping_rounds=100
sub_preds = clf.predict(df_test[feats])
fold_importance_df = pd.DataFrame()
fold_importance_df["feature"] = feats
fold_importance_df["importance"] = clf.feature_importances_
solution = df_test['sentiment']
confusion_matrix(y_pred=sub_preds, y_true=solution)
import gc
len_train = df_train.shape[0]
df_all = pd.concat([df_train,df_test])
del df_train, df_test;
gc.collect()
from textblob import TextBlob
from tqdm  import  tqdm
reviews = df_all['review_clean']
```

```
Predict_Sentiment = []

for review in tqdm(reviews):

blob = TextBlob(review)

Predict_Sentiment += [blob.sentiment.polarity]

df_all["Predict_Sentiment"] =

Predict_Sentiment df_all.head()

np.corrcoef(df_all["Predict_Sentiment"], df_all["rating"])

reviews = df_all['review']

Predict_Sentiment = []

for review in tqdm(reviews):

blob = TextBlob(review)

Predict_Sentiment += [blob.sentiment.polarity]

df_all["Predict_Sentiment2"] =

Predict_Sentiment

df_all['count_sent']=df_all["review"].apply(lambda x:

len(re.findall("\ n",str(x)))+1)

#Word count in each comment:

df_all['count_word']=df_all["review_clean"].apply(lambda x:

len(str(x).split()))

#Unique word count

df_all['count_unique_word']=df_all["review_clean"].apply(lambda x:

len(set(str(x).split())))

#Letter count

df_all['count_letters']=df_all["review_clean"].apply(lambda x: len(str(x)))
```

```
#punctuation count
df_all["count_punctuations"] = df_all["review"].apply(lambda x: len([c for
c in str(x) if c in string.punctuation]))
#upper case words count
df_all["count_words_upper"] = df_all["review"].apply(lambda x: len([w for w
in str(x).split() if w.isupper()]))
#title case words count
df_all["count_words_title"] = df_all["review"].apply(lambda x: len([w for
w in str(x).split() if w.istitle()]))
#Number of stopwords
df_all["count_stopwords"] = df _all["review"].apply(lambda x: len([w for w
in str(x).lower().split() if w in stops]))
#Average length of the words
df_all["mean_word_len"] = df_all["review_clean"].apply(lambda x:
np.mean([len(w) for w in str(x).split()]))
df_all['season'] = df_all["month"].apply(lambda x: 1 if ((x>2) & (x<6))
else(2 if (x>5) & (x<9) else (3 if (x>8) & (x<12) else 4)))
f_train = df_all[:len_train]
df_test = df_all[len_train:]
from sklearn.metrics import roc_auc_score,
precision_recall_curve, roc_curve, average_precision_score
from sklearn.model_selection import
KFold from lightgbm import
LGBMClassifier
```

```
#folds = KFold(n_splits=5, shuffle=True, random_state=546789)

target = df_train['sentiment']

feats =[ 'usefulCount','day','year','month','Predict_Sentiment','Predict_

Sentiment2' , 'count_sent',

'count_word', 'count _unique_word', 'count_letters',

'count_punctuations', 'count_words_upper', 'count_words_title',

'count_stopwords', 'mean_word_len', 'season']

sub_preds = np.zeros(df_test.shape[0])

trn_x, val_x, trn_y, val_y = train_test_split(df_train[feats], target,

test_size=0.2, random_state=42)

feature_importance_df = pd.DataFrame()

LGBMClassifier(n_estimators=10000,learning_rate=0.10,num_leaves=30,colsample_bytree=.9,

subsample=.9,max_depth=7,reg_alpha=.1,reg_lambda=.1,min_split_gain=.01,min_child_weight=2,s
ilent=-1,verbose=-1,)

clf.fit(trn_x, trn_y,eval_set= [(trn_x, trn_y), (val_x, val_y)],

verbose=100, early_stopping_rounds=100 #30)

sub_preds  =  clf.predict(df_test[feats])

fold_importance_df  =  pd.DataFrame()

fold_importance_df["feature"] = feats

fold_importance_df["importance"] = clf.feature_importances_

feature_importance_df = pd.concat([feature_importance_df, fold_importance_df], axis=0)

confusion_matrix(y_pred=sub_preds, y_true=solution)

cols = feature_importance_df[["feature",

"importance"]].groupby("feature").mean().sort_values(
```

```
by="importance", ascending=False)[:50].index

best_features =feature_importance_df.loc[feature_importance_df.feature.isin(cols)]

plt.figure(figsize=(14,10))

sns.barplot(x="importance", y="feature",

data=best_features.sort_values(by="importance", ascending=False))

plt.title('LightGBM Features (avg over folds)')

plt.tight_layout()

plt.savefig('lgbm_importances.png')

# import dictionary data

word_table = pd.read_csv("inquirerbasic.csv")

##1. make list of sentiment

#Positiv word list

temp_Positiv = []

Positiv_word_list = []

for i in range(0,len(word_table.Positiv)): if

word_table.iloc[i,2] == "Positiv":

temp = word_table.iloc[i,0].lower()

temp1 = re.sub('\d+', '', temp) temp2= re.sub('#', '', temp1)

temp_Positiv.append(temp2)

Positiv_word_list = list(set(temp_Positiv))

len(temp_Positiv)

len(Positiv_word_list) #del temp_Positiv

#Negativ word list

temp_Negativ = []
```

```
Negativ_word_list = []

for i inrange(0,len(word_table.Negativ)):

 if word_table.iloc[i,3] == "Negativ":

temp = word_table.iloc[i,0].lower()

temp1 = re.sub('\d+', '', temp)

temp2 = re.sub('#', '', temp1)

temp_Negativ.append(temp2)

Negativ_word_list = list(set(temp_Negativ))

len(temp_Negativ)

len(Negativ_word_list) #del temp_Negativ

##2. counting the word 98590

import numpy as np

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary = Positiv_word_list)

content = df_test['review_clean']

X = vectorizer.fit_transform(content)

f = X.toarray()

f = pd.DataFrame(f)

f.columns=Positiv_word_list

df_test["num_Positiv_word"] = f.sum(axis=1)

vectorizer2 = CountVectorizer(vocabulary =

Negativ_word_list) content = df_test['review_clean']

X2 = vectorizer2.fit_transform(content)

f2 = X2.toarray()
```

```
f2 = pd.DataFrame(f2)

f2.columns=Negativ_word_list

df_test["num_Negativ_word"] = f2.sum(axis=1)

##3. decide sentiment

df_test["Positiv_ratio"]  =df_test["num_Positiv_word"]/(df_test["num_Positiv_word"]

+df_test["num_Negativ_word"])

df_test["sentiment_by_dic"] = df_test["Positiv_ratio"].apply(lambda x: 1 if

(x>=0.5) else (0 if (x<0.5) else 0.5))

df_test.head()

def userful_count(data):

grouped = data.groupby(['condition']).size().reset_index(name='user_size')

data = pd.merge(data,grouped,on='condition',how='left')

return data

df_test = userful_count(df_test)

df_test['usefulCount' ] =

df_test['usefulCount']/df_test['user_size'] df_test['deep_pred'] =

sub_preds_deep df_test['machine_pred'] = sub_preds

df_test['total_pred'] = (df_test['deep_pred'] +

df_test['machine_pred'] +

df_test['sentiment_by_dic'])*df_test['usefulCount']

df_test = df_test.groupby(['condition','drugName']).agg({'total_pred' :['mean']})
```

6.2 app.py

```
from flask import Flask, request, jsonify

app = Flask(_name_)

@app.route('/recommendation', methods=['POST'])

def get_recommendation():

    data = request.get_json()

    symptoms = data.get('symptoms', '')

    # Placeholder logic for recommendation, replace with your actual code

    recommended_medicines = recommend_medicines(symptoms)

    return jsonify({'medicines': recommended_medicines})

def recommend_medicines(symptoms):

    # Replace this with your actual recommendation logic

    # This is just a placeholder

    return ["Medicine1", "Medicine2", "Medicine3"]

if__name__== '_main_':

    app.run(debug=True)
```

CHAPTER 7

SCREENSHOTS

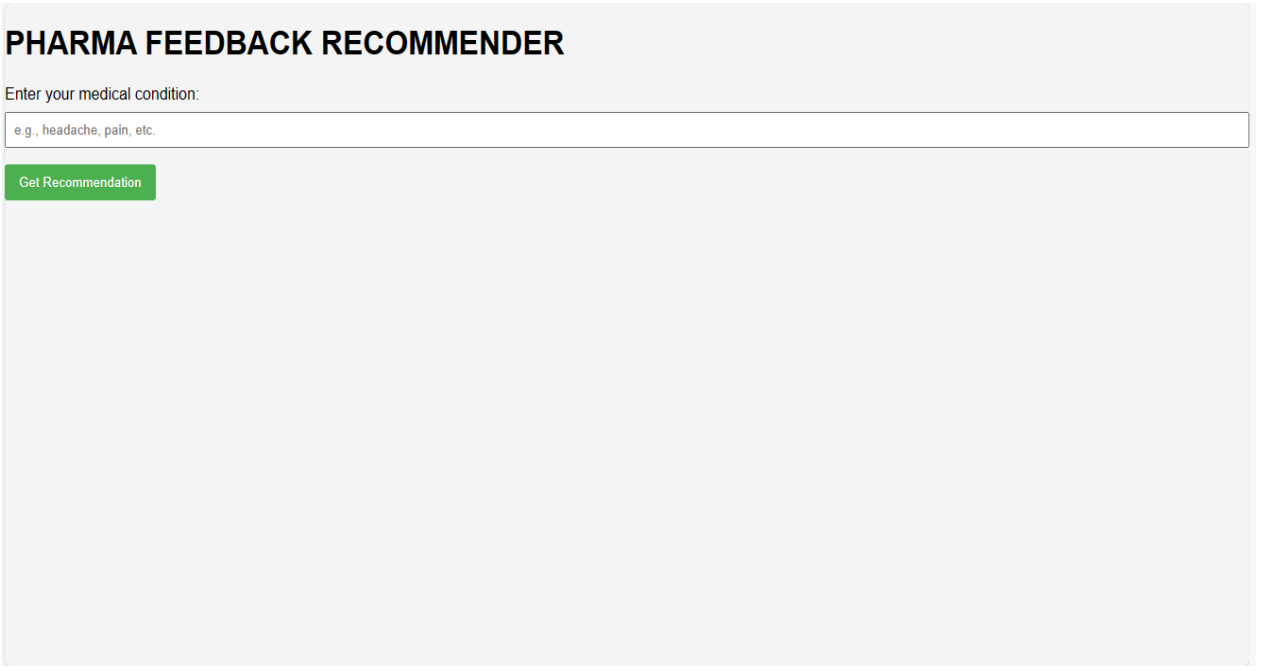

Fig 6. Output Interface

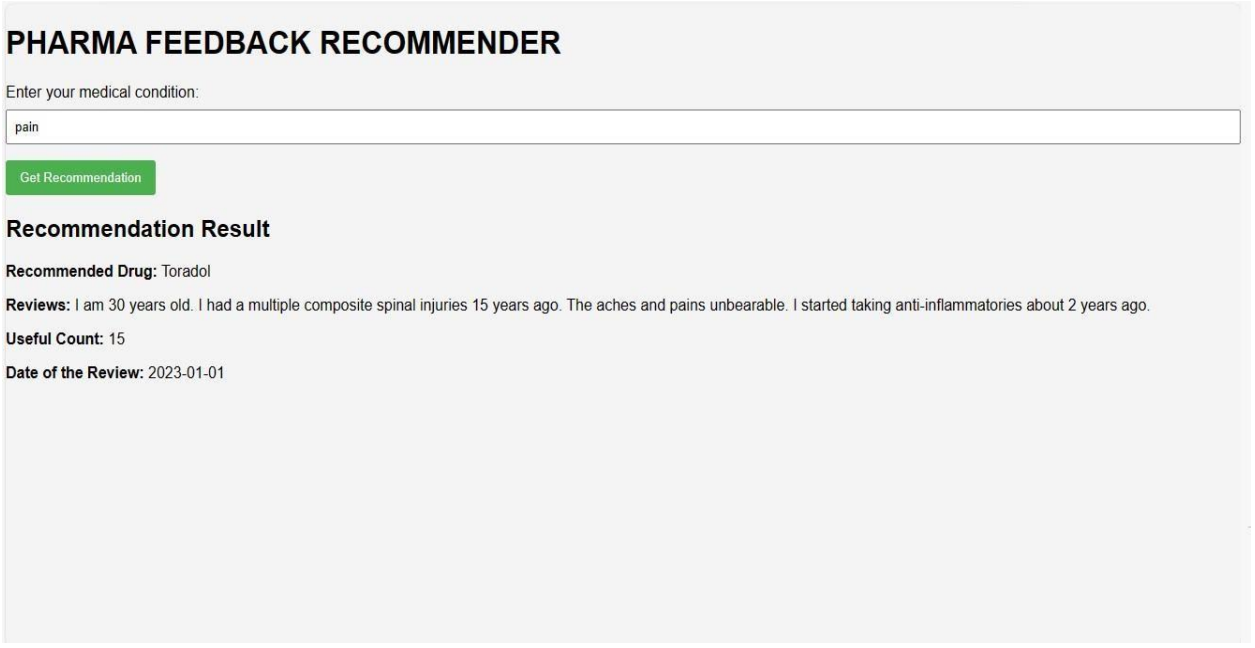

Fig 7. Recommended Drugs and Reviews

CHAPTER 8

REFERENCES

[1] Wittich CM, Burkle CM, Lanier WL. Medication errors: an overview for clinicians. Mayo Clin Proc. 2014 Aug;89(8):1116- 25.

[2] Bartlett JG, Dowell SF, Mandell LA, File TM Jr, Musher DM, Fine MJ. Practice guidelines for the management of community acquired pneumonia in adults. Infectious Diseases Society of America. Clin Infect Dis. 2000 Aug;31(2):347-82.

[3] Rachamanee, S.,T. N. Tekade and M. Emmanuel, "Probabilistic aspect mining approach for interpretation and evaluation of drug reviews," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi.

[4] Doulaverakis, C., Nikolaidis, G., Kleontas, A. et al. GalenOWL: Ontology-based drug recommendations discovery. J Biomed Semant 3, 14 (2012).

[5] https://www.tutorialpoint.org/drugsystemarchitecture/

[6] https://www.geeksforgeeks.org/medicinerecommendation/