Recommendation Algorithm Based on Knowledge Graph to Propagate User Preference

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By
Ravi Srivastava (191265)

Under the supervision of
Dr. Ravindara Bhatt

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" Recommendation algorithm based on knowledge graph to propagate user preference"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Dr. Ravindara Bhatt** Associate Professor Computer Science and Engineering Dept.

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Proficiency in Data Science**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Ravi Srivastava, 191265

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Ravindara Bhatt
Associate Professor
Computer Science and Engineering
Dated:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

First, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing to make it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Ravindara Bhatt, Associate Professor,** Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "Data Science" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr. Ravindara Bhatt, Department of CSE, for his kind help in finishing my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Ravi Srivastava
191265

TABLE OF CONTENT

TITLE	PAGE NO.
Candidate Declaration	I
Plagiarism Certificate	II
Acknowledgment	III
Table Of Content	IV
List of Figures	V
List of Tables	VI
Abstract	VII
Chapter-1: Introduction	1-27
Chapter 2: Literature Survey	28
Chapter-3: System Development	29-47
Chapter-4: Experiment & Result Analysis	48-54
Chapter-5: Conclusion	55-56
References	57
Appendices	58-63

List of Figures

1. Knowledge graph
2. Wikipedia's Infobox for Yellow fever
3. Data scraping and evaluation
4. Python logo
5. Jupyter Notebook logo
6. NumPy logo
7. Pandas logo
8. Sklearn logo
9. NLTK logo
10. XGBoost logo
11. Beautiful soup logo
12. System architecture
13. Symptom suggested to user
14. Symptom suggestion
15. Prediction and recommendation
16. Prediction using TF,IDF scoring
17. Using a cosine-similar scoring model and disease information predict and treatment recommendation
18. Accuracy v/s Classifiers
19. Wikipedia's box for Brucellosis
20. Dataset scraping and evaluation
21. Fetching disease
22. Fetching disease treatment
23. Random forest
24. K-Nearest Neighbor
25. SVM
26. Decision Trees
27. Taking symptom
28. Appending symptoms
29. Related symptoms
30. Scrapped data
31. Symptom matching
32. Model v/s Accuracy
33. Model v/s Cross Validation Accuracy
34. Taking symptom
35. Appending symptom
36. Adding symptom after checking similarity score
37. Top matching symptom
38. Final list of symptom
39. Top 10 highly probable disease
40. Treatment recommended
41. Fetching disease
42. Pre-processing
43. Symptom matching
44. NB classifier
45. Random Forest classifier
46. K Nearest Neighbor classifier
47. Logistic Regression
48. SVM classifier
49. Decision Tree classifier
50. MLP classifier

List of Tables

1. Comparison plot for all classifiers with accuracy
2. Comparison plot for all classifiers with cross validation accuracy.

ABSTRACT

Applications of machine learning in the biological and healthcare fields have improved early disease identification and diagnosis. This has recently improved patient treatment. Studies have revealed that consumers turn to the internet for advice on any potential health-related problems. This method has a drawback in that the search engines offer a lot of information in a disorganised way that makes it challenging to draw conclusions from. The game of disease detection based on disease is intricate. The users feed the symptoms in non-technical or natural terms because they are unfamiliar with biological terms, which makes disease prediction more difficult. The main goal is to create a novel architecture capable of accepting and handling these kinds of user queries by utilising methods like query expansion using a thesaurus, synonym matching, and symptom suggestion that will enable disease prediction with a higher degree of accuracy based on user input and recommend the best treatment possible for that disease. We collected data from the internet and created a dataset that can be applied to further study. In these situations, prediction is accomplished using matching and query search retrieval.

Chapter 1: - INTRODUCTION

1.1 Introduction

One sort of machine learning that deals with ranking or rating people or things is recommendation engines. In general, a recommender system is a system that predicts the ratings a user would give a specific item. The ranking of these projections will then be shown to the user.

Numerous well-known companies, like Instagram, Spotify, Google, Amazon, Reddit, Netflix, etc., commonly use them to increase platform and user engagement. For instance, Spotify will recommend songs that are similar to ones you've liked or listened to frequently in an effort to keep you using their platform to listen to music. Amazon provides recommendations to promote products to various customers based on the data they have gathered about a user.

Recommender systems are frequently referred to as "black boxes," because the models created by these large corporations are not very clear. Based on the results, the consumer frequently receives recommendations for products they don't immediately understand they need or want.

Creating recommender systems can be done in a variety of ways. Others use modeling-focused techniques like collaborative filtering, content-based techniques, link prediction, etc. while some use formulaic and algorithmic techniques like Page Rank. We can also get recommendations by looking at the Knowledge Graph. Each of these techniques may differ in complexity, but complexity is not always a guarantee of "excellent" performance. Simple strategies and techniques typically produce the finest results. For instance, well-known companies like Reddit, Hacker News, and Google have used simple, formulaic implementations of recommendation engines to push content on their platforms.

Google proposed the Knowledge Graph, a semantic NLP network. A graph with nodes and edges, where nodes are referred to as entities and edges as

relationships, is a simpler way to depict it. An entity in the real world could be a person, a company, or a concept, and a relation just denotes the link between two entities.

The relationships between various things are depicted in the straightforward ideal KG graph that follows.

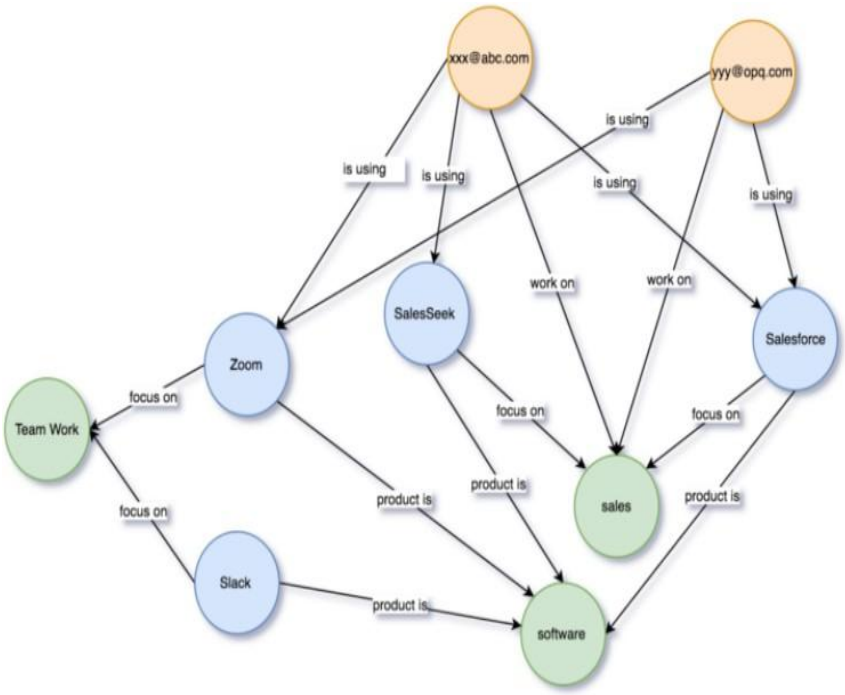


Figure1. Knowledge graph

The project is titled as “Recommendation Algorithm using Knowledge Graph to Propogate User Preference”. We are showing a knowledge graph which is showing a relation between symptoms, disease and treatment.

It is essential to offer complementary treatment plans for a disease in accordance with varied symptoms at various stages. However, the majority of classification techniques may fall short in correctly identifying a disease that exhibits numerous stages of therapy, a wide range of symptoms, and multiple aetiology. Additionally, there are little exchanges and collaborative efforts in disease diagnosis and treatment between various hospitals and departments. Therefore, emerging diseases with unusual symptoms may be challenging for novice clinicians to recognise quickly and correctly. In order to make the best use of the advanced medical technology of established institutions and the in-depth medical knowledge of experienced clinicians, a Disease Diagnosis and Treatment Recommendation System (DDTRS) is proposed in this work.

The study suggests a model in which the user can provide unstructured symptoms or choose the symptoms recommended by the system, and based on those selections, receive a list of likely diseases. Additionally, the user can choose any of the output diseases to learn more about the disease and their current health status, including additional symptoms, causes, diagnoses, potential treatments, etc. Based on the symptoms that the user has entered, the system also offers other symptoms.

The method is simple to use and can be helpful in the early detection and diagnosis of diseases. It can even be utilised by someone with limited medical understanding. Users who are hesitant to go to hospitals when they first have minor symptoms may also find it helpful. This will provide them a fundamental understanding of the disease's severity.

1.2 Problem Statement

A model is proposed in which the user can input unstructured symptoms or opt for the symptoms the system proposes, and then receive a list of possible diseases based on their choices. The user can also select any of the diseases that are output to learn more about it and their current health status, including more symptoms, causes, diagnoses, potential therapies, etc. The algorithm also

provides additional symptoms based on the user-entered symptoms. According to those symptoms treatment is recommended .

The technique is easy to apply and can aid in the early recognition and diagnosis of illnesses. Even those with little medical knowledge can make use of it. It might be useful to users who are reluctant to visit hospitals when they initially experience minor symptoms. This will provide them a fundamental comprehension of the severity of the sickness.

1.3 Objectives

Early disease identification and improved diagnosis have been made possible by machine learning applications in the healthcare and biomedical fields. Recently, this has improved patient care. According to studies, individuals consult the internet for advice on any potential health-related difficulties. The issue with this strategy is that the search engines offer a tonne of information in a dispersed fashion, making it challenging to draw conclusions.

There are numerous illness prediction systems available, including those for skin diseases, neurological disorders, and heart diseases.

However, symptom-based universal disease prediction and treatment recommendation systems are rarely used in real life. It is quite beneficial for doctors or other medical professionals to make an early diagnosis of diseases based on symptoms. Based on the highest probability and scores, likely ailments are offered to the user in response to a query.

The use of the internet and all of the user's resources allows for the appropriate diagnosis of ailments and the subsequent administration of the most effective treatment for all people. Without the use of any medical tests or other procedures, it is very beneficial for both doctors and patients to understand the disease better.

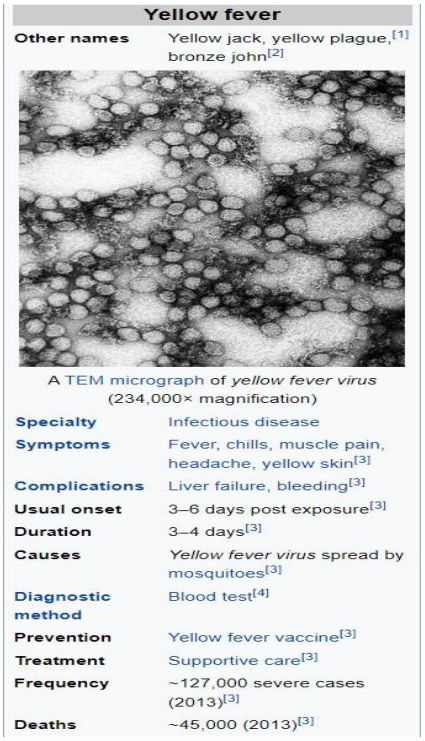
Disease-based disease detection is a tricky game. Due to the users' lack of biological terminology, the symptoms are described in non-technical or natural words, which makes disease prediction more difficult. The main goal is to create a novel architecture capable of accepting and handling these kinds of user queries by utilising methods like query expansion using a thesaurus, synonym matching, and symptom suggestion that will enable disease prediction with a higher degree of accuracy based on user input and recommend us the best solution to cure that disease. In order to create a dataset that can be used in future studies, we scraped data from the internet. Such issues employ matching and query search retrieval to achieve prediction.

1.4 Methodology

Running the Python script allowed for the web-scraping of the dataset of diseases and associated symptoms. The following sources provided the diseases and symptoms that make up the dataset:

• Diseases: The list of diseases was taken from the Centre for Health Informatics' [National Health Portal of India "(https://www.nhp.gov.in/disease-a-z)" (CHI)]. The program downloads the page's HTML code and then extracts the list of diseases by filtering the values in HTML tags.

• Symptoms: The script searches for the disease's Wikipedia page among the different search results it receives using the Google Search package. The page's HTML code is examined to retrieve the symptoms using the 'infobox' on the Wikipedia article, you may learn more about the illness. Figure 1 depicts an illustration of a Wikipedia infobox.



Yellow fever

Other names	Yellow jack, yellow plague,[1] bronze john[2]

A TEM micrograph of *yellow fever virus* (234,000× magnification)

Specialty	Infectious disease
Symptoms	Fever, chills, muscle pain, headache, yellow skin[3]
Complications	Liver failure, bleeding[3]
Usual onset	3–6 days post exposure[3]
Duration	3–4 days[3]
Causes	*Yellow fever virus* spread by mosquitoes[3]
Diagnostic method	Blood test[4]
Prevention	Yellow fever vaccine[3]
Treatment	Supportive care[3]
Frequency	~127,000 severe cases (2013)[3]
Deaths	~45,000 (2013)[3]

Figure 2: Wikipedia's Infobox for Yellow Fever.

The symptoms are all extracted, and a dictionary is made with the key being the disease and the value being the symptoms. All symptoms are regarded as distinct properties or columns, and each disease is treated as the label. The orderly progression of the stages involved in data scraping are shown in Figure 2.

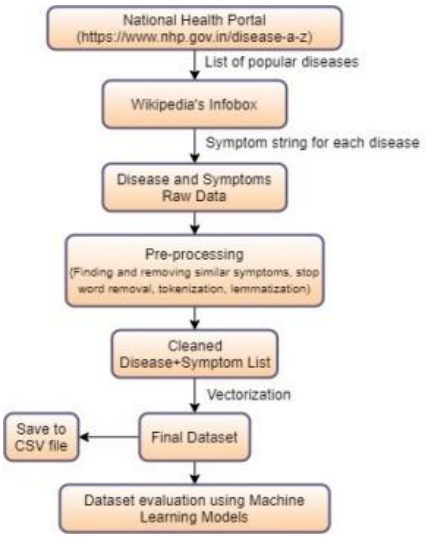


Figure3. Data Scraping and Evaluation

Let us look at the libraries and platforms that we have used for creating our project.

1.4.1 Python

Figure 4: - Python logo

The interactive, object-oriented, high-level programming, general purpose language Python is particularly well-liked. Python is a programming language that uses garbage collection and dynamic typing. It was made between 1985 and 1990 by Guido van Rossum. The GNU General Public License applies to Python, just like it does to Perl (GPL). Python is a programming language that is attractive for the development of apps since it is easy to learn but reliable and adaptable.

Python's dynamic typing, interpreted nature, and syntax make it the finest language for scripting and rapid application development. Imperative, functional, and object-oriented programming techniques are supported by Python. Python shouldn't be utilised, for instance, while developing front-end web pages. It is said to as a flexible programming language because it functions with online, enterprise, 3D CAD, and other applications. The variables are dynamically typed, therefore to give the variable an an integer value of 10, we may just write a=10.

Python provides clear and readable code. Python's simplicity allows developers to create dependable systems, but machine learning and artificial intelligence are powered by sophisticated algorithms and adaptable work routines. Instead of having to worry about the technical details of the language, developers may concentrate solely on solving an ML problem.

Due of its ease of learning, Python also appeals to many developers. Python code is comprehensible by humans, which makes it easier to develop machine learning models. When there are multiple developers participating, Python is often thought to be suitable for collaborative implementation. Python is a general-purpose language, making it possible to use it to complete a wide range of challenging machine learning tasks and to create quick prototypes for testing your product for machine learning.

Python frameworks and modules come in many varieties, which programmers use to accelerate development. Programmers can handle common programming tasks by using software libraries, which are collections of pre-written code. Thanks to its robust technology stack, Python has a wide range of AI and machine learning libraries. Here are some instances:

- Keras, Scikit-learn, Tensorflow for machine learning
- Numpy for high performance scientific computing and data analysis
- SciPy for advanced computing
- Pandas for general purpose data analysis
- Seaborn for data visualization

1.4.2 Jupyter Notebook

Jupyter Notebook serves as the platform for creating the project-specific notebook. Project Jupyter is where the Jupyter Notebook is proposed. Project Jupyter's goal is to offer interactive computing services, open standards, and open-source software for a range of programming languages. In 2014, Brian Granger and Fernando Pérez split it from IPython.

Figure 5: - Jupyter Notebook logo

Using the free and open-source Jupyter Notebook online application, you may exchange documents and write code with live code, visuals, equations, and text. Jupyter Notebook maintenance falls under the purview of the Project Jupyter team.

Jupyter Notebooks were created by the IPython project, which formerly had its own IPython Notebook project. The name Jupyter alludes to the three main programming languages it supports: Julia, Python, and R. Although Jupyter comes with the IPython kernel, which enables Python programming, there are now over 100 more kernels available.

The Jupyter Notebook App is a server-client software that enables editing and running notebook papers from a web browser. You can use the Jupyter Notebook App locally on a computer without an internet connection or remotely on a server that is reachable via the internet (as described in this paper) on a PC.

In addition to displaying, editing, and executing notebook documents, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control

panel" that exposes local files and lets you access notebook papers or kill their kernels.

Using the NumPy and Pandas libraries kicks off the data processing process.

1.4.3 NumPy

For array operations, use the NumPy Python package. Additionally, there are functions for using the Fourier transform, matrix operations, and linear algebra.

In the year 2005, Travis Oliphant developed NumPy. It is free to use because it is an open source project.

The acronym for numerical Python is NumPy.

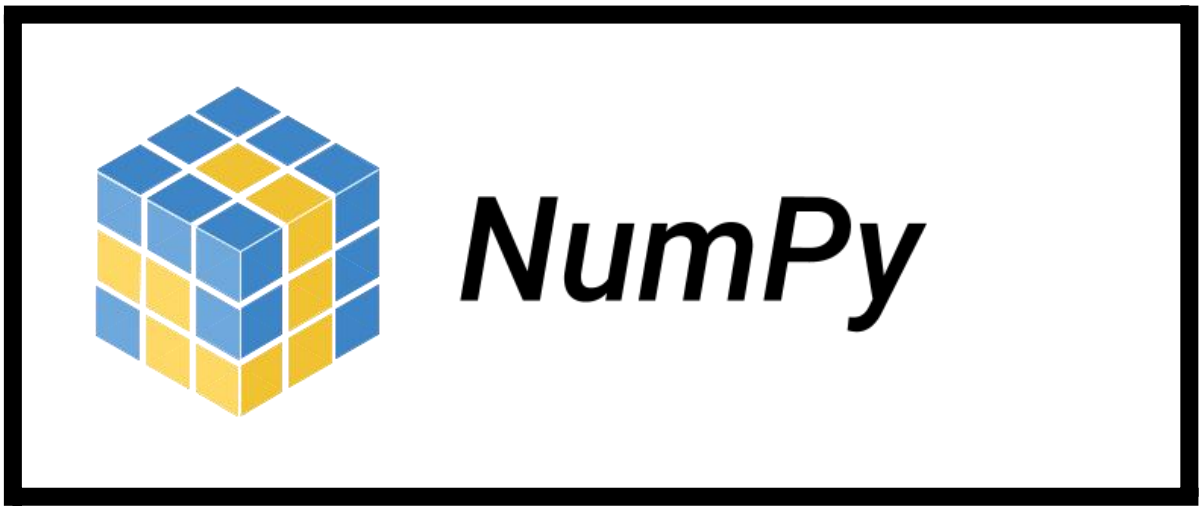


Figure 6: - NumPy Logo

Python now has access to the computing power of languages like C and FORTRAN thanks to NumPy. Using Python's NumPy module, you may work with multidimensional arrays and matrices. It is perfect for scientific or mathematical computations due to its effectiveness and quickness. Signal

processing and linear algebra are also supported by NumPy. Therefore, if you need to perform any mathematical operations on your data, NumPy is unquestionably the library for you.

There are several ways that Python lists and NumPy arrays are different from one another. First off, NumPy arrays contain more dimensions than Python lists do. Second, whereas NumPy arrays are homogeneous, Python lists are varied. This implies that each element of a NumPy array must be of the same type. Third, NumPy arrays are more efficient than Python lists. NumPy arrays can be created in a variety of ways. One method is to create an array from a Python list. Once it has been created, a NumPy array can be modified in a variety of ways. For example, you can change an array's shape or use an index to access its elements. Mathematical operations like addition, multiplication, and division can also be performed on NumPy arrays.

1.4.4 Pandas

Figure 7: - Pandas Logo

Pandas is a Python data analysis module. Wes McKinney created pandas in 2008 to satisfy the need for a consistent and flexible tool for conducting quantitative research. Pandas has since grown to be one of the most used Python packages. There is a robust contributor community.

Two crucial Python libraries—Matplotlib, which is used for data visualisation, and NumPy, which is used for mathematical operations—are the foundation upon which Pandas is constructed. Many of the matplotlib and NumPy algorithms are easier to use thanks to Pandas, which acts as a wrapper for both libraries. For instance, you can plot a chart with the least amount of code possible by using the the.plot() function from the pandas library, which combines several matplotlib routines into a single procedure.

Pandas was created to handle two-dimensional data (similar to Excel spreadsheets). Similar to how the NumPy library contains a built-in data structure called an array with particular attributes and functionalities, the pandas library offers a built-in two-dimensional data structure called a Data Frame.

This tool is essentially where your data lives. Pandas allows you to clean, modify, and analyse your data to learn more about it. In addition to being an essential component of the data science toolkit, the pandas library is used in conjunction with other libraries in that collection.

1.4.5 Sklearn

Figure 8: - Sklearn Logo

One of the most reliable and potent Python machine learning packages is Sklearn (Skit-Learn). Through a Python consistency interface, it incorporates a number of effective statistical modelling and machine learning algorithms, including classification, regression, clustering, and dimensionality reduction. This Matplotlib, SciPy, and NumPy-based library was created primarily in Python.

Almost all machine learning algorithms, such as logistic regression, linear regression, decision trees, random forests, and k-means clustering, are supported by it.

This open-source library may be used for commercial purposes under the BSD license.

David Cournapeau, a French data scientist, created Scikits.learn. An initial Google Summer of Code project was scikit-learn. The word "SciKit" (SciPy Toolkit) in its name refers to a third-party modification of SciPy that was separately created and disseminated. Different programmers later changed the original codebase. Project management was handled by Fabian Pedregosa,

Gael Varoquaux, Alexandre Gramfort, and Vincent Michel in 2010 at the Saclay, France-based French Institute for Research in Computer Science and Automation. On February 1st of that year, the project's maiden public release took place. Scikit-learn and Scikit-image were listed as scikits that were "well-maintained and popular" in November 2012. One of the most well-liked machine learning libraries on GitHub is scikit-learn.

1.4.6 NLTK

The most popular framework for creating Python programmes that use human language data is NLTK, or Natural Language Toolkit. It includes wrappers for industrial-strength NLP libraries, a discussion forum, and a collection of text processing libraries for categorization, tokenization, stemming, tagging, parsing, and semantic reasoning. It also offers simple access to more than 50 corpora and lexical resources, including WordNet.

Figure 9: - NLTK Logo

NLTK is appropriate for linguists, engineers, students, educators, academics, and business users equally because to its comprehensive API documentation and hands-on instruction covering both programming fundamentals and computational linguistics challenges. For Linux, Windows, and Mac OS X, NLTK is accessible. The best feature of NLTK is that it is a community-driven, free open-source project.

NLTK has been lauded as "an outstanding library to play with natural language" and is a fantastic tool for computational linguistics research and education using Python.

Python's Natural Language Processing offers a useful introduction to language processing programming. The principles of writing Python programming, using corpora, classifying text, studying linguistic structure, and other topics are covered step-by-step in this book. It was created by the same group as NLTK. Python 3 and NLTK 3 have been included to the book's online version.

1.4.7 XGBoost

Figure 10: - XGBoost Logo

XGBoost, a distributed gradient boosting library, was developed to be incredibly effective, adaptable, and portable. It uses the Gradient Boosting

framework to create machine learning algorithms. XGBoost, a parallel tree boosting method, can be used to quickly and accurately solve a variety of data science issues. It is sometimes referred to as GBDT or GBM. The same method, when applied to significant distributed systems (Hadoop, SGE, and MPI), may resolve problems involving more than a trillion instances.

The application of XGBoost technology results in gradient-boosted decision trees. The XGBoost models dominate a lot of Kaggle competitions.

Decision trees are created using this method sequentially. Weights are very important in XGBoost. Each independent variable is assigned a weight before entering the decision tree that forecasts results. Variables that the first decision tree mistakenly forecasted are given additional weight before entering the second decision tree. These multiple classifiers and predictors are then merged to produce an effective and trustworthy model. It can be used to address a variety of issues, including regression, classification, ranking, and custom prediction.

1.4.8 BeautifulSoup

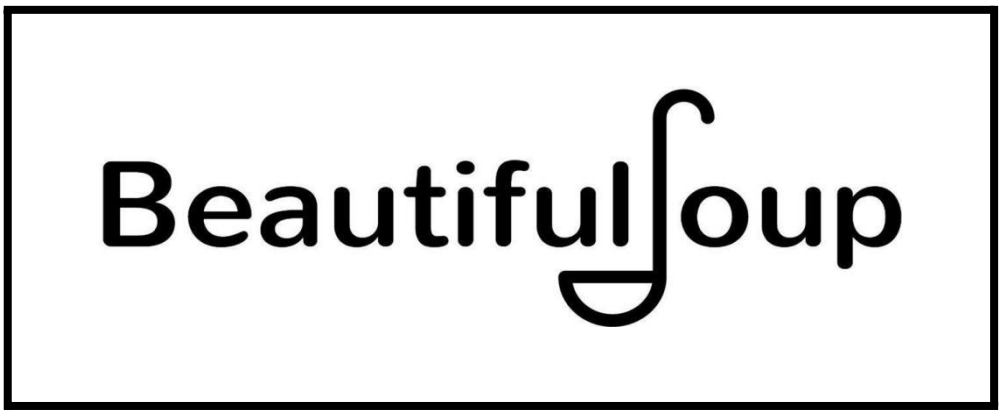


Figure 11: Beautiful soup logo

Web scraping is the process of obtaining information from the internet using a range of programmes and frameworks. By pulling data from competitors'

websites, it is occasionally used to keep an eye on changes in online prices, compare prices, and assess competitors' performance.

Web scraping has existed for as long as the internet. The World Wide Web was originally launched in 1989, and four years later, at MIT, Matthew Gray created World Wide Web Wanderer, the first web robot, with the intention of determining the size of the entire web.

Leonard Richardson made the initial suggestion, and he still contributes to the cause. Tidelift also helps with this endeavour (a paid subscription tool for open-source maintenance)

The most recent version of Beautiful Soup, 4.9.2, was officially released in May 2006 and supports Python 3 and Python 2.4.

Now, let’s look at the structure of the project.

Generally speaking, the system asks the user to submit symptoms in accordance with which model forecasts diseases with the highest probability and scores .According to that disease it will predict the cause , Risk factors and recommend the best treatment possible for that disease.

The procedure for predicting diseases from user-input symptoms is shown in Figure 3. Each module is covered in detail in the following subsections.

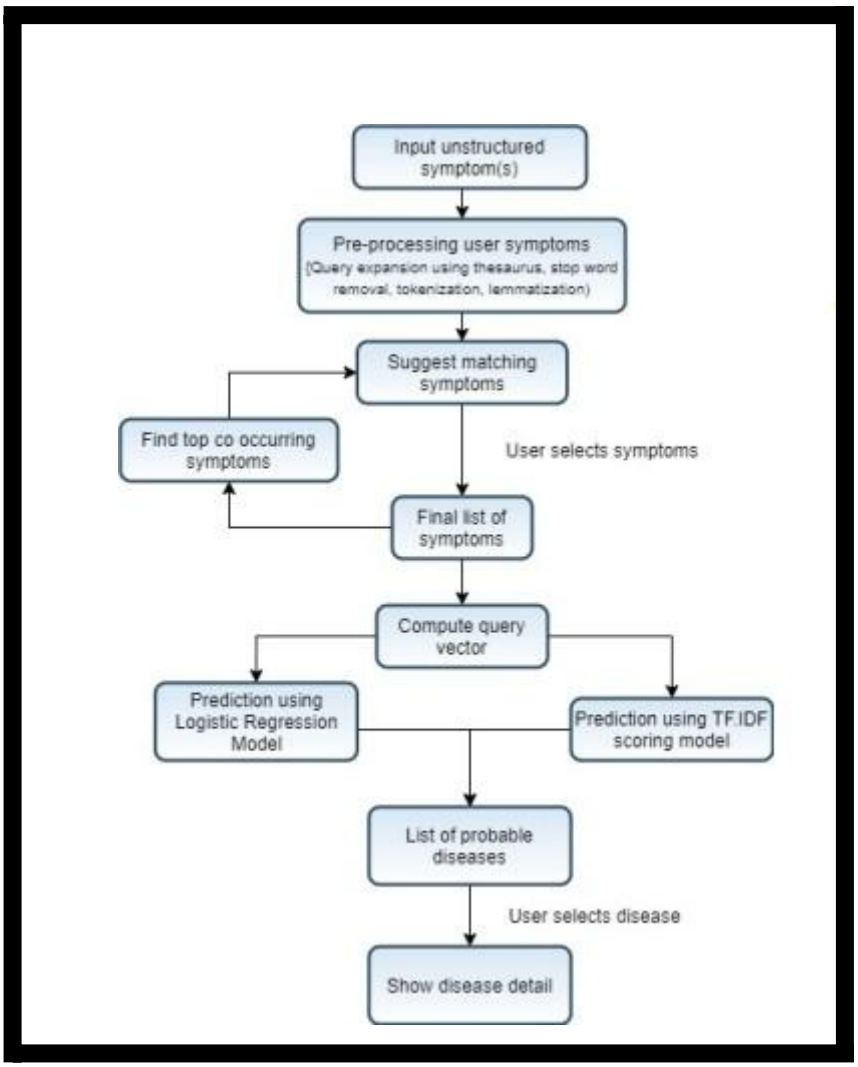


Figure 12: System Architecture

1.4.9 Symptom Preprocessing

The computer will accept one line of symptoms, with commas separating each one (,). The further preprocessing processes are as follows:

- Use commas to separate symptoms into a list.
- Lowercase the signs and symptoms.
- Eliminating stopwords
- Tokenizing symptoms to eliminate any punctuation
- Lemmatization of symptoms' tokens

The expanded symptoms are then added to the processed symptom list.

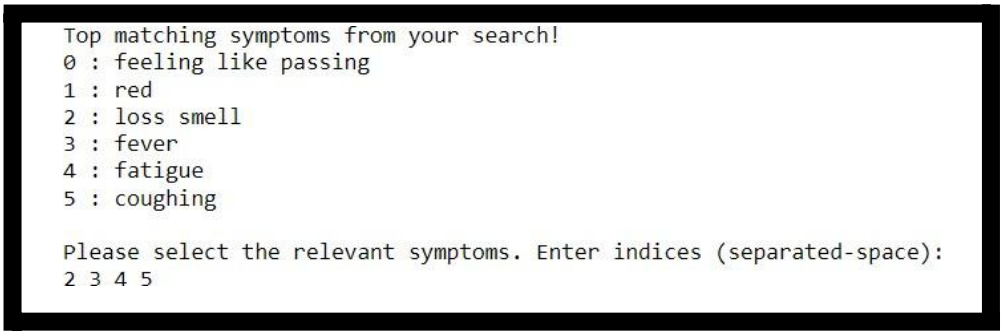


Figure 13: Symptom suggested to user

1.4.10 Symptom Preprocessing

A list of synonyms is added to each symptom to further describe it. Thesaurus.com (https://www.thesaurus.com/) is where the synonyms were found. And WordNET, a Python-based resource from Princeton University (https:// wordnet.princeton.edu/). Every symptom is separated into its subsets for locating the collection of synonyms. Figure 4 displays symptoms entered by the user as well as symptoms from the dataset that match the synonym term.

1.4.11 Symptoms Suggestion and Selection

Using the extended symptom query, the associated symptoms in the dataset are located. To identify such symptoms, each symptom in the dataset is broken down into tokens, with each token's frequency in the expanded query being assessed. This results in the generation of a similarity score; if it rises above a certain point, the symptom is deemed to be comparable to the user's symptom and recommended to the user.

From the list, the user chooses one or more symptoms. Other symptoms that are among the most common co-occurring symptoms with the ones the user initially picked are then displayed to the user for selection based on the selected symptoms. The procedure of choosing a symptom can be skipped or stopped by the user. A sample of the symptom suggestion and selection

procedure is shown in Figure 5. The final list of symptoms is then compiled in order to create the symptom vector that will be used in the prediction process.

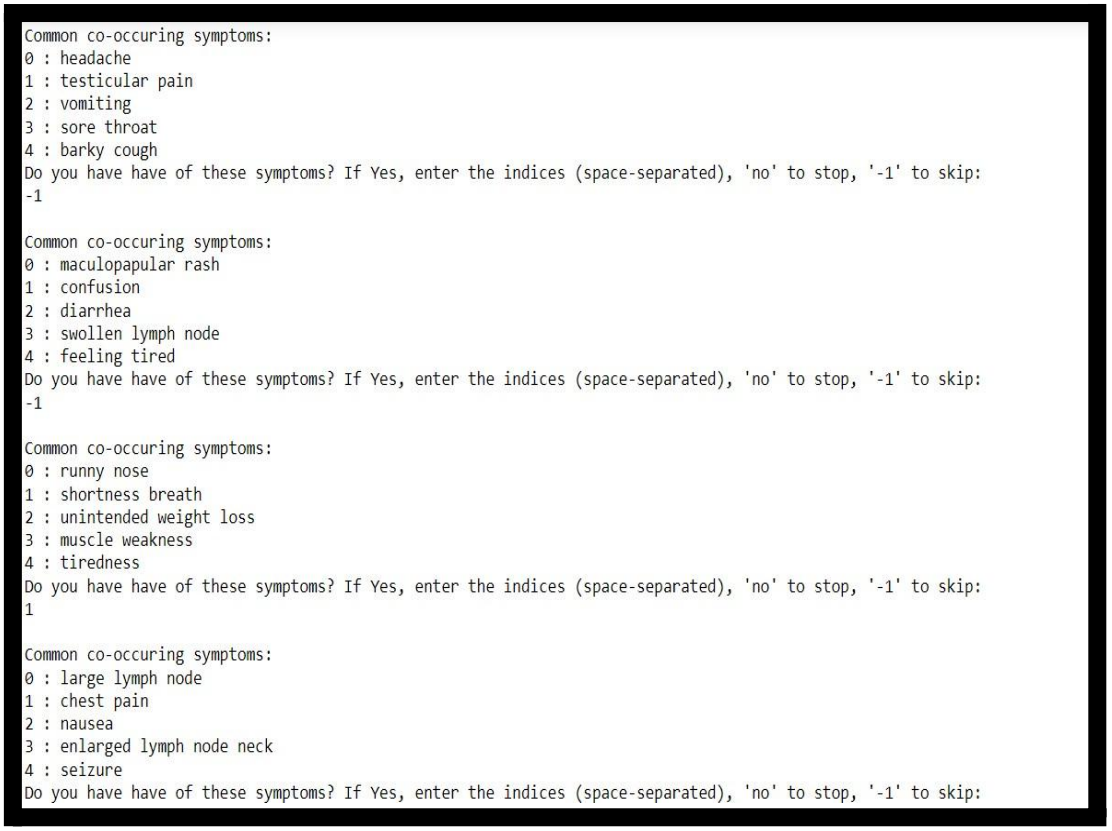


Figure 14:

Symptom suggestion and selection process

1.4.12 Disease Prediction and Treatment recommendation

The final symptom list is used to construct model-specific vectors and make disease predictions and according to that most suitable treatment is recommended.

1.4.13 Prediction using Machine Learning Model

When the symptoms in the user's selection list are present, a binary vector is generated that has the values 1 and 0, respectively. The dataset is utilised to build a machine learning model, which is then applied to this prediction. The

model takes the symptom vector as input and returns the top K diseases, ranked by decreasing individual probabilities. The likelihood of an illness is determined as follows.

```
ModelAccuracy->accuracy(model used)
DisASymp=Symptoms(DiseaseA)
match->intersect(DisASymp,userSymp)
matchScore->match/count(userSymp)
prob(DiseaseA)=matchScore*modelAccuracy
```

Figure 6 displays the list of diseases that the Logistic Regressor predicts with a probability and treatment is shown for that disease.

```
Top 10 disease based on Cosine Similarity Matching :

0. Disease : Coronavirus disease 2019 (COVID-19)        Score : 0.64
1. Disease : Brucellosis         Score : 0.52
2. Disease : Asthma      Score : 0.34
3. Disease : Influenza   Score : 0.28
4. Disease : Dehydration         Score : 0.26
5. Disease : Nasal Polyps        Score : 0.24
6. Disease : Middle East respiratory syndrome coronavirus (MERS-CoV)     Score : 0.24
7. Disease : Mouth Breathing     Score : 0.21
8. Disease : Coronary Heart Disease      Score : 0.21
9. Disease : Legionellosis       Score : 0.2

More details about the disease? Enter index of disease or '-1' to discontinue and close the system:
2

Asthma
Pronunciation -      UK:  /  ˈ  æ  s  m  ə  ,     ˈ  æ  s  θ  m  ə  /      US:  /  ˈ  æ  z  m  ə  /
Specialty -  Pulmonology
Symptoms -  Recurring episodes of wheezing, coughing, chest tightness, shortness of breath
Complications -  Gastroesophageal reflux disease (GERD), sinusitis, obstructive sleep apnea
Usual onset -  Childhood
Duration -  Long term
Causes -  Genetic and environmental factors
Risk factors -  Air pollution, allergens
Diagnostic method -  Based on symptoms, response to therapy, spirometry
Treatment -  Avoiding triggers, inhaled corticosteroids, salbutamol
Frequency -  approx 262 million (2019)
Deaths -  approx 461,000 (2019)
```

Figure 15: Prediction using logistic regression along with disease detail , its cause ,its
duration, its cause, and recommended treatment.

1.4.14 Prediction using Cosine Similarity and TF.IDF Model

By calculating the TF and IDF of the dataset's symptoms, the TF.IDF score model is trained.

• Term frequency (TF) is a measure of how frequently a disease's symptom occurs.

• Document frequency (DF) measures the prevalence of a symptom across all disorders. Equation 1 shows the computation of inverse DF.

$$IDF=\log10(\text{count(All Diseases)}/DF) \quad -(1)$$

Equation 2 illustrates how to calculate TF.IDF. The same formula is used to compute each vector element.

$$TF.IDF(sym, dis) = \log10(1 + TF)) * IDF \quad -(2)$$

In a similar manner, the user's symptom vector is likewise computed. Equation 3 illustrates how both vectors are multiplied to provide the TF.IDF score for the user's symptom query and ailment. A higher score indicates that the two vectors are quite similar to one another.

$$TF.IDF\ Score(Q, A) = dot(Q, A) \quad (3)$$

A list of the top K diseases is obtained after the scores are sorted according to decreasing score. The predicted list of diseases with TF.IDF scores is displayed in Figure 7.

```
Top 10 diseases predicted based on TF_IDF Matching :

0. Disease : Coronavirus disease 2019 (COVID-19)        Score : 13.36
1. Disease : Asthma      Score : 7.2
2. Disease : Influenza   Score : 5.75
3. Disease : Nasal Polyps        Score : 4.87
4. Disease : Brucellosis         Score : 4.47
5. Disease : Dehydration         Score : 4.47
6. Disease : Mouth Breathing     Score : 4.47
7. Disease : Anthrax     Score : 4.02
8. Disease : Legionellosis       Score : 4.02
9. Disease : Middle East respiratory syndrome coronavirus (MERS-CoV)     Score : 4.02

More details about the disease? Enter index of disease or '-1' to discontinue:
-1
```

Figure 16: Prediction using TF,IDF scoring model

1.4.15 Disease Detail

Any disease output by the model can be chosen by the user, who can then read the specifics of that disease in the console. Figures 6 and 8 display the user-selected details of the disease. As demonstrated in Figure 7, users canskip the step by inputting "-1."

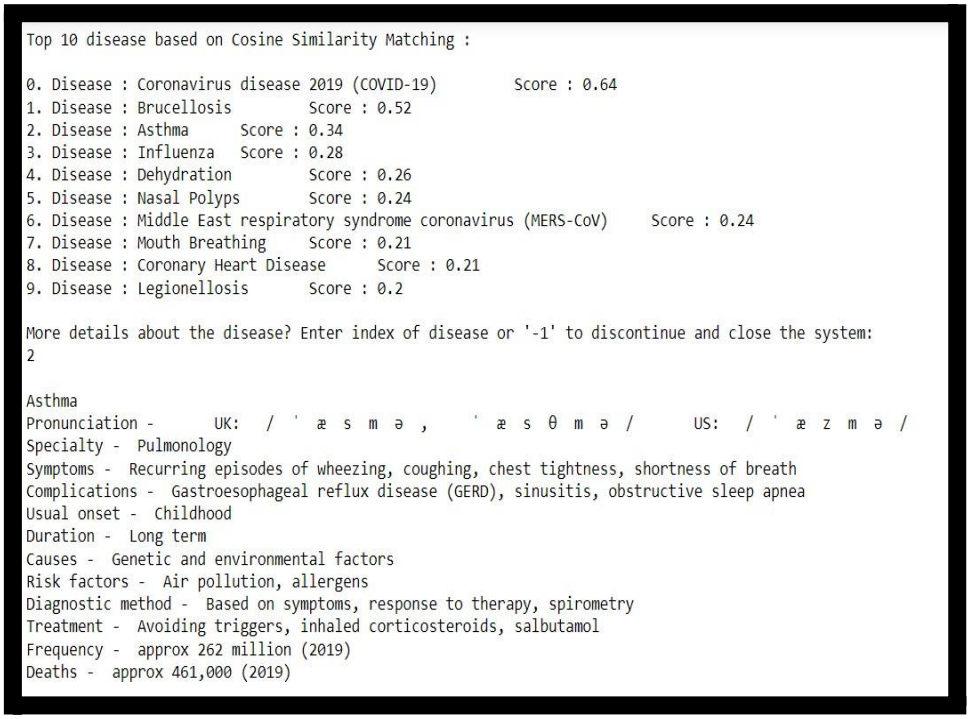


Figure 17: Using a cosine-similar scoring model and disease information predict and treatment recommendation.

1.5 Organization

The organization of the report is as follows:

Chapter 1: - It gives information about the project and the idea that inspired it. The primary focus of chapter 1 is the issue that we are addressing, the project's main goal, its various components, and how we plan to put everything together. Main focus in chapter 1 is the introduction of our project , what we want to show basically what is the objective of ours .

Chapter 2: - In order to gain a clear understanding of the many project intuitions and the potential applications of various models, several journals,

articles, and research papers on the internet have been investigated. The impact of various models on the project's evaluation indicators is also taken into account. Also how we can add more features in our project alongside the important ones is taken into consideration.

Chapter 3: - This chapter let us to go deeper into the analytical, mathematical and experimental interpretation of our data set and how we can manipulate or modify our data set to suit our conditions. It also takes into account the use of various libraries in order to give the best results.

Chapter 4: - It basically covers the classifiers that will define the accuracy of our models and each and every model is discussed in detail. A total of 5 classifiers, namely the Multinomial NB classifier ,RandomForest classifier, KNeighbours classifier, Logistic Regression, SVM classifier, Decision Tree classifier and MLP classifier are taken into account for determining the accuracy of our model. And along with that why and how each model is used in a particular context is also mentioned here.

The associated symptoms in the dataset are located using the expanded symptom query. Each symptom in the dataset is divided into tokens, and the frequency of each token in the enlarged query is calculated in order to find such symptoms. A similarity score is calculated using this information, and if it exceeds a particular threshold, the symptom is considered to be equivalent to the user's symptom and is recommended to the user. The user selects one or more symptoms from the list. The user is then presented with additional symptoms for selection depending on the initial symptoms they chose that are among the most frequent co-occurring symptoms. The user has the option of skipping or stopping the symptom selection process.

The final symptom list is used to construct model-specific vectors and make disease predictions and according to that most suitable treatment is recommended.

Chapter 5: - Under this chapter, the conclusion of our project and the future scope associated with it have been discussed in regards to further techniques

and optimizations that can be implemented in our dataset for better accuracy in a relatively less amount of time.

By using several machine learning algorithms and evaluating the accuracy they provide, the dataset is evaluated. Also a comparison between different model accuracies is shown. The highest accuracy is reported by Random Forest (89.59%) and K Nearest Neighbour (89.48%) while the lowest is of Multinomial Naive Bayes (84.62%).

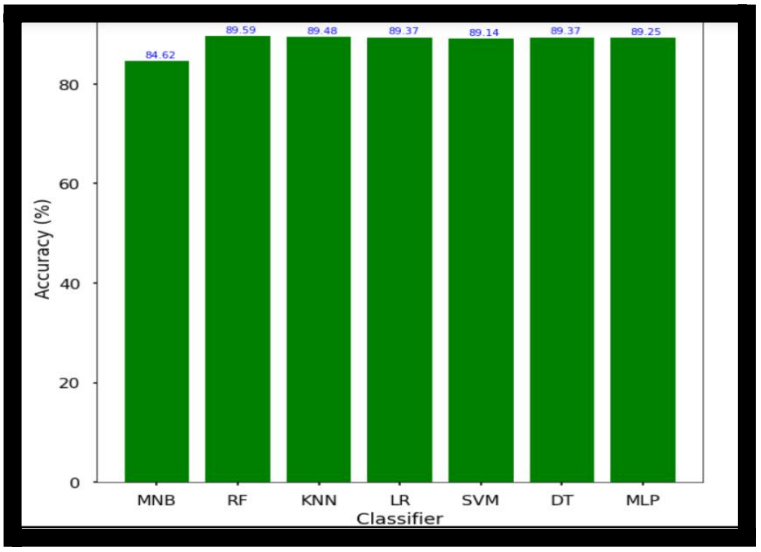


Figure 18: Accuracy v/s classifiers

Chapter 2

LITERATURE SURVEY

Much research has been done to identify diseases based on probable patients' symptoms and other medical health information. In a study titled "Semantic text categorization of disease reporting" (Zhang and Liu, 2007), Y. Zhang and B. Liu describe the training and successful prediction of an infectious disease model using sentence-level semantics. The research suggested in (Wang X, 2008) focuses on disease prediction from clinical data supplied by New York's Presbyterian Hospital. Automated disease prediction is significantly different and simpler using these clinical data than it is with user text input. Since users avoid using clinical terminology that suggest greater complexity when matching symptom names with the user's input, the authors made this observation.

Sen et al. (Kumar Sen, 2013) put into practise a system that analyses symptoms and other patients' specific text to predict coronary heart diseases. In order to take feedback, rate, and analyse comments in order to improve the model, Petrov et al. (Slav Petrov, 2013) propose a natural language processing model.

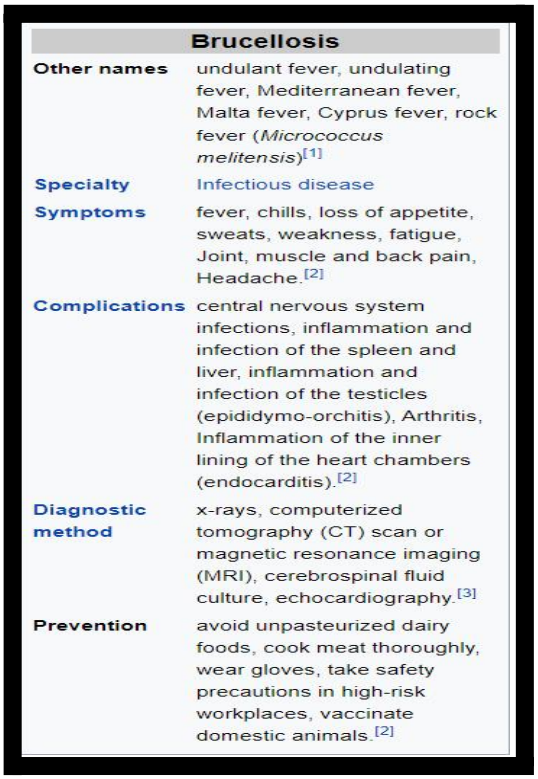
Chapter 3: - SYSTEM DEVELOPMENT

3.1 Analytical / Experimental

The analytical treatment of the model is based on the fact that we have to analyze our dataset and pick up the best and the most reasonable features that we require and drop off the features that are redundant. By running the Python script, a dataset of diseases and their symptoms was scraped from the internet. The following sources were used to gather the dataset, which includes diseases and their symptoms:

• Diseases: The Center for Health Informatics' National Health Portal of India (https://www.nhp.gov.in/disease-a-z) was used to compile the list of diseases (CHI). After downloading the HTML code of the page, the application filters the values in the HTML tags to extract the list of diseases.

• Symptoms: Among the several search results it receives from the Google Search package, the script looks for the Wikipedia page for the ailment. By using the "infobox" on the Wikipedia article, you can learn more about the illness by looking at the HTML code of the page to retrieve the symptoms. A Wikipedia infobox is illustrated in Figure 1.



Brucellosis	
Other names	undulant fever, undulating fever, Mediterranean fever, Malta fever, Cyprus fever, rock fever (*Micrococcus melitensis*)[1]
Specialty	Infectious disease
Symptoms	fever, chills, loss of appetite, sweats, weakness, fatigue, Joint, muscle and back pain, Headache.[2]
Complications	central nervous system infections, inflammation and infection of the spleen and liver, inflammation and infection of the testicles (epididymo-orchitis), Arthritis, Inflammation of the inner lining of the heart chambers (endocarditis).[2]
Diagnostic method	x-rays, computerized tomography (CT) scan or magnetic resonance imaging (MRI), cerebrospinal fluid culture, echocardiography.[3]
Prevention	avoid unpasteurized dairy foods, cook meat thoroughly, wear gloves, take safety precautions in high-risk workplaces, vaccinate domestic animals.[2]

Figure 19: Wikipedia's Infobox for Brucellosis

The symptoms are all extracted, and a dictionary is made with the key being the disease and the value being the symptoms. All symptoms are treated as distinct attributes or columns, and each disease is treated as the label. The orderly progression of the steps involved in data scraping are shown in Figure 2.

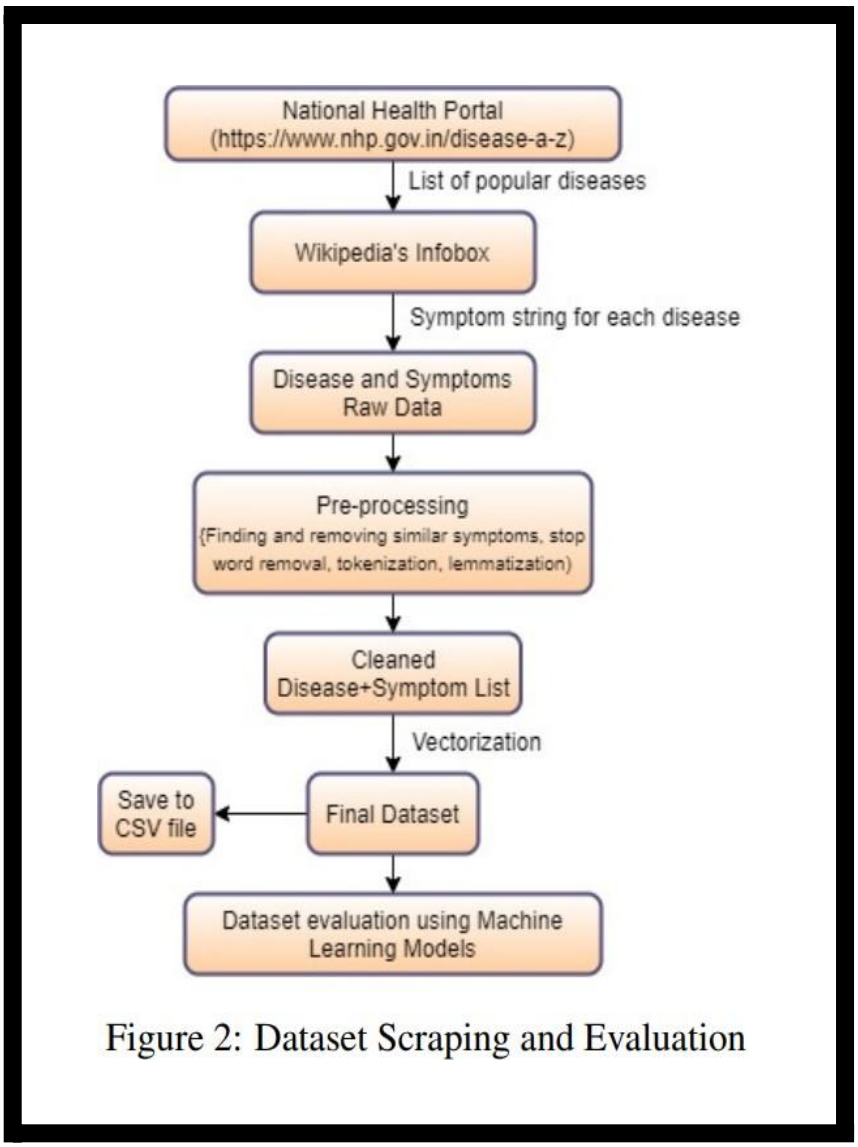


Figure 2: Dataset Scraping and Evaluation

The scraping software retrieves 500+ symptoms and over 261 different diseases that make up the label. The symptoms are then pre-processed to get rid of symptoms that are the same but have different names (For example, headache and pain in the forehead). This is accomplished by computing the Jaccard Coefficient for pairs of symptoms and discovering the synonyms for each symptom. Both symptoms are quite similar and can be eliminated if the score is higher than the cutoff.

The symptoms of each disease are selected, generated into combinations, and added to the dataset as additional rows in order to multiply it. For instance, the dataset now contains (25-1) records for disease A with 5 symptoms. The

dataset has about 8835 rows with 489 distinct symptoms after preprocessing and multiplication.

The data was scraped using several libraries, including beautifulsoup. Web scraping is the process of obtaining information from the internet using a range of programmes and frameworks. By pulling data from competitors' websites, it is occasionally used to keep an eye on changes in online prices, compare prices, and assess competitors' performance. Web scraping has existed for as long as the internet. The World Wide Web was originally launched in 1989, and four years later, at MIT, Matthew Gray created World Wide Web Wanderer, the first web robot, with the intention of determining the size of the entire web. The idea was first put out by Leonard Richardson, who still contributes to this project, and Tidelift (a paid subscription tool for open-source maintenance) also supports this initiative. The most recent version of Beautiful Soup, 4.9.2, was officially released in May 2006 and supports Python 3 and Python 2.4..

Data Scraping is done as by firstly fetching diseases :

```
# Fetch disease list from 'www.nhp.gov.in'
small_alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', '
diseases=[]
for c in small_alpha:
    URL = 'https://www.nhp.gov.in/disease-a-z/'+c
    time.sleep(1)
    page = requests.get(URL,verify=False)

    soup = BeautifulSoup(page.content, 'html.parser')

    all_diseases = soup.find('div', class_='all-disease')

    for element in all_diseases.find_all('li'):
        diseases.append(element.get_text().strip())

with open('list_diseaseNames.pkl', 'rb') as handle:
    diseases2 = pickle.load(handle)
```

Figure 21: Fetching disease

And then scraping treatment of that diseases via Wikipedia as:

```
# Search diseases on google, open wikipedia page and fetch trt from infobox

dis_trt={}
# dis1=['anthrax']
for dis in c:
  query = dis+' wikipedia'
   # search "disease wilipedia" on google
  for sr in search(query,tld="co.in",stop=10,pause=0.5):
       # open wikipedia link
    match=re.search(r'wikipedia',sr)
    filled = 0
    if match:
      wiki = requests.get(sr,verify=False)
      soup = BeautifulSoup(wiki.content, 'html.parser')
       # Fetch HTML code for 'infobox'
      info_table = soup.find("table", {"class":"infobox"})
      if info_table is not None:
          # Preprocess contents of infobox
        for row in info_table.find_all("tr"):
          data=row.find("th",{"scope":"row"})
          if data is not None:
            data=data.get_text()
            if data=="Treatment":
              trt=str(row.find("td"))
              trt= trt.replace('.','')
              trt= trt.replace(';',',')
              trt= re.sub(r'<b.*?/b>:',',',trt) # Remove bold text
              trt= re.sub(r'<a.*?>','',trt) # Remove hyperlink
              trt= re.sub(r'</a>','',trt) # Remove hyperlink
              trt= re.sub(r'<[^<]+?>',', ',trt) # All the tags
              trt= re.sub(r'\[.*\]','',trt) # Remove citation text
              trt=' '.join([x for x in trt.split() if x != ','])
              dis_trt[dis]=trt
              # print(dis_trt[dis])
              filled = 1
              break
```

Figure 22: Fetching disease treatment

The Natural Language Toolkit, generally known as NLTK, is a set of tools and programmes for symbolic and statistical natural language processing for English. It was developed in Python and is used for pre-processing. The term "lemmatizer" refers to the act of analysing a word's various inflected forms by merging them into a single unit through the process of lemmatization. Lemmatization gives context to the words, much like stemming does. This links words with similar meanings together.

Lemmatization and stemming are both involved in text preparation. Users frequently become confused by these two terms. These two are sometimes

compared. Because lemmatization does a morphological examination on the words, it is really preferred to stemming.

And after that we iterated over all disease and preprocessed the symptoms string and break it into the individual symptom. We also removed the ‘none’ from symptom . After that we initialized two dataframes, one for normal dataset and one for combination dataset , then we read each disease and symptom list and convert that into dictionary and add that to dataframe.

With the aid of the xgboost package, symptom matching is carried out. In order to be extremely effective, adaptable, and portable, the distributed gradient boosting library XGBoost was developed. The Gradient Boosting framework is used to construct machine learning algorithms. A number of data science issues can be successfully resolved by using the parallel tree boosting method known as XGBoost, sometimes referred to as GBDT or GBM.

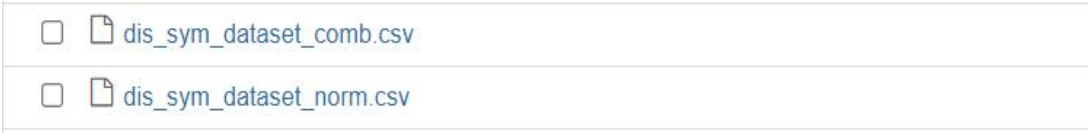
We iterated over all diseases and preprocess symptoms string and break it into individual symptom.

Then we stored each symptom’s synonyms as a list of words. Then for symptom pair in dataset , we found out Jaccard score of their synonym list. If Jaccard > threshold it means that those synonyms are similar and one of them can be used in place of other.

After removing similar symptoms , final list of symptoms is stored in new_symptoms.

We initialized two dataframes , one for normal dataset and one for combination dataset. We read each disease and symptom list and converted into dictionary and add to dataframe. And if the symptom is similar , replace it with the value in dictionary.

Then we exported the datset into csv files .



Then we used model classifiers to predict the accuracy of disease prediction through user input symptoms . We used 7 model classifiers to predict the accuracy. We first loaded our dataset scraped from NHP and Wikipedia , then we splitted the data for training the classifiers and testing . Classifiers used are:

- Multinomial Naïve byes classifier - For the examination of numerical data, there are thousands of programmes and tools at our disposal, but there are very few for the study of texts. Multinomial Naive Bayes is one of the most popular supervised learning classifications for the analysis of categorical text data. The volume of information that needs to be processed from emails, papers, webpages, and other online sources has led to an increase in the prevalence of text data classification. Understanding the context of a certain type of text might help one forecast how consumers will react to a piece of software or a product. The Multinomial Naive Bayes algorithm, a probabilistic learning method, is used in Natural Language Processing (NLP). A text, such as an email or news story, can have its tag predicted using a Bayes theorem-based approach. It determines the odds of each tag for a specific sample and outputs the tag with the highest probabilities. A fundamental principle shared by all Naive Bayes classifier algorithms is that each feature being classified is independent of all other properties. The existence or absence of one trait has no bearing on the existence or absence of the other feature.

  The accuracy and Cross Validation accuracy we got by NB Classifier is-
  Accuracy – 84.62%
  Cross Validation Accuracy (MNB) – 84.50%

- Random Forest Classifier - The well-known machine learning algorithm Random Forest is used in the supervised learning process. AI and regression It can be applied to resolve problems with classification. The principle of ensemble learning, which is the act of combining different classifiers to solve a difficult problem and improve the performance of the model, serves as its theoretical underpinning. The Random Forest classifier, as its name suggests, uses decision trees to boost the predicting accuracy of the supplied dataset. In order to improve the dataset's predicted accuracy, Random Forest "contains a number of decision trees on various subsets of the provided dataset and takes the average." In order to predict the outcome based on which predictions earned the most votes, the random forest includes predictions from all of the decision trees rather than just one. The greater number of trees in the forest prevents overfitting and improves accuracy. The graphic below shows how the Random Forest algorithm works:

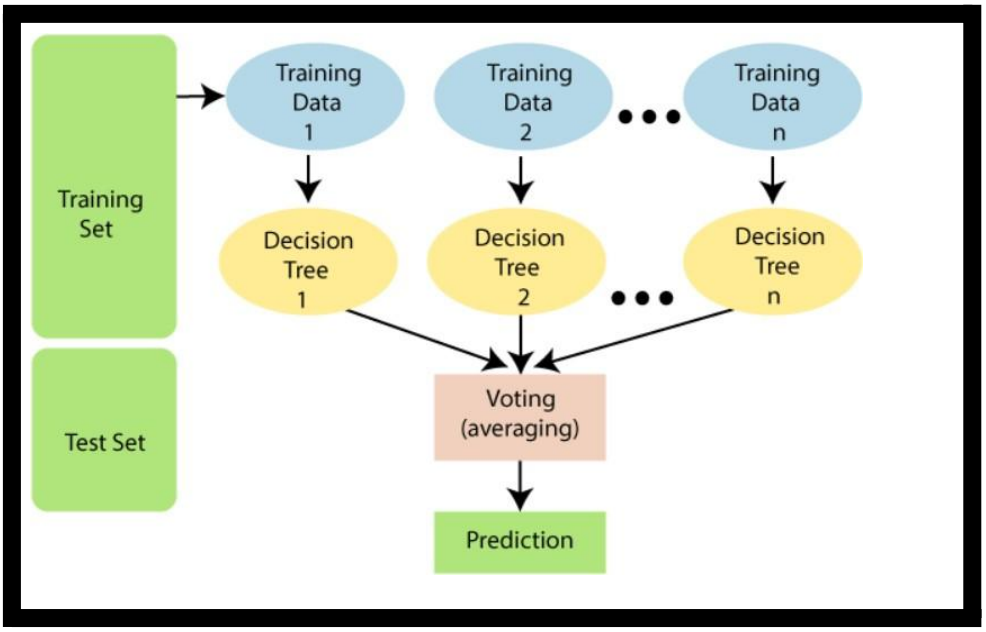


Figure 23: Random Forest

To create the random forest, N decision trees are combined, and then, in the second phase, predictions are made for each of the trees from the first phase.

The following steps and diagram provide an explanation of the working process:

Choose K data points at random from the training set in step 1.

Create the decision trees linked to the chosen data points in step two (Subsets).

Step 3: Select the decision tree N that you want to construct.

Step 4: Re-do Steps 1 and 2.

Step 5: For any new data points, locate each decision tree's predictions and group the new data points into the category with the most support.

The accuracy and Cross Validation accuracy we got by Random Forest Classifier is-

Accuracy – 89.59%

Cross Validation Accuracy (MNB) – 86.84%

- K Nearest Neighbors Classifier - The k-nearest neighbours classifier (kNN) is a supervised machine learning approach that is non-parametric. It classifies items based on those of their close neighbours and is based on distance. Although kNN can also be usedto address regression concerns, classification problems represent its primary use. A non-parametric model does not require parameter adjusting during the training phase. In spite of the fact that k is a hyperparameter, it resembles an algorithm parameter in some ways. Both during training and inference, it is manually chosen and fixed. closest neighbours in K The algorithm is also non-linear. It works well with data when the connection between the independent variable (x) and the dependent variable (y) is not a straight line, in contrast to simpler models like linear regression.

The number of labelled points (neighbours) taken into consideration for classification is represented by the parameter k in kNN.

The value of k represents the number of these points required to compute the result. Our tasks are to calculate the distance and to determine which categories are closest to our unknown entity. The k-nearest neighbour algorithm's fundamental concept is as follows. We can make an effort to identify which points in our feature space are near a point whose class we are unsure of. These points are the neighbours within k. Given that related entities frequently occupy nearby spots in feature space, it is very likely that the point belongs to the same class as its neighbours. That allows a new point to be categorised as belonging to one class or another.

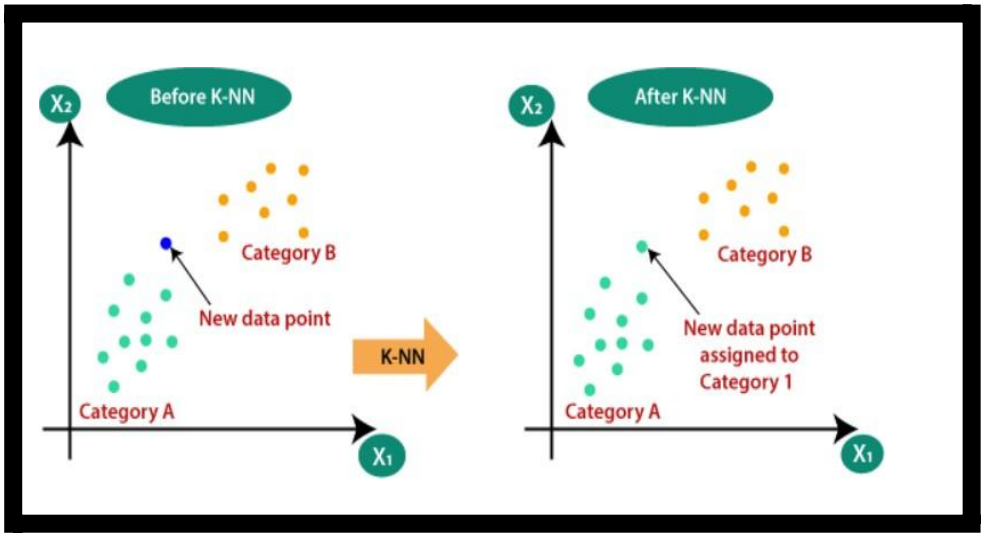


Figure 24: K Nearest Neighbor

The accuracy and Cross Validation accuracy we got by K Nearest Neighbor Classifier is-

Accuracy – 89.48%

Cross Validation Accuracy (MNB) – 85.25%

- Logistic Regression - Machine learning has adapted the logistic regression classification technique from statistics. The statistical

method of logistic regression can be used to locate one or more independent variables that influence a result in a dataset. The objective of logistic regression is to find the model that most accurately depicts the connection between the dependent and independent variables.
Logistic regression is a technique for classification used in machine learning. A logistic function is used to model the dependent variable. Due to the dependent variable's dichotomous character, only two classes are possible (eg.: either the cancer is malignant or not). Consequently, this technique is used while working with binary data.

The sigmoid function is used by the logistic regression technique to convert predicted values into probabilities. Any real value can be converted into a number between 0 and 1 using this technique. Every point in this function has exactly one inflection point and a non-negative derivative.
With the aid of a cost function, the discrepancy between expected and predicted values can be mathematically stated. In other words, a cost function measures how well the model predicts the link between x and y. The terms cost, loss, or simple error might be used to characterise the value that the cost function returns.

$$\text{Cost}(h_\Theta(x), Y(\text{actual})) = -\log(h_\Theta(x)) \text{ if } y=1$$
$$-\log(1- h_\Theta(x)) \text{ if } y=0$$

Because the loss function minimization during training must maximise probability, a negative function is produced. When samples are selected from identically independent distributions, the maximum probability will rise as the cost goes down.

The accuracy and Cross Validation accuracy we got by Logistic Regression Classifier is-

Accuracy – 89.37%

Cross Validation Accuracy (MNB) – 89.19%

- Support Vector Machine (SVM) Classifier - A hyperplane is the line that SVM uses to divide the classes. The data points closest to the hyperplane on either side of the hyperplane are called support vectors, and they are used to plot the boundary line. Data can be used in linear or non-linear ways for SVM classification. An SVM classifier can be built up using many kernels. We can label the kernel as "linear" for a dataset with just linear data.

  On the other hand, two kernels for a non-linear dataset go by the names "rbf" and "polynomial". Drawing the hyperplane is made easier by the data's mapping to a higher dimension. The lower dimension is reached after that.

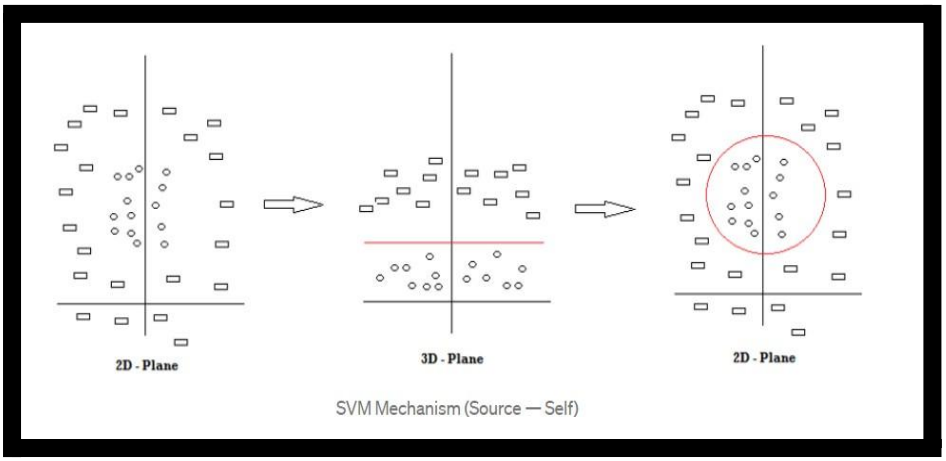


Figure 25: SVM

The diagram above illustrates two classes of shapes: circles and rectangles. Because it is difficult to create an SVM line in the 2D Plane, we move the data points to a higher dimension (3D Plane). After

that, the SVM Classifier is marked in red and the plane is returned to its original position.

The accuracy and Cross Validation accuracy we got by SVM Classifier is-

Accuracy – 89.14%

Cross Validation Accuracy (MNB) – 88.62%

- Decision Tree Classifier - Decision Tree is a supervised machine learning system that uses a set of rules to simulate human decision-making. You may consider an algorithm for machine learning categorization as being built with decision-making in mind.

  However, in practise, the model cannot anticipate the class of a brand-new, previously unobserved input before the algorithm decides which class to assign.

  There are rule-based algorithms as well as probabilistic ones for classification, such as Naive Bayes. Every day, humans make decisions based on rules. You follow a set of rules when organising your upcoming vacation. Depending on how long your trip will last, your financial situation, and whether or not your extended family will be joining you, you may choose a different location.

  The resolution is based on the responses to these inquiries. And if you continually eliminate potential vacation spots based on your responses to each query, you can picture this decision-making procedure as a (decision) tree. As a result of their versatility in performing classification and regression tasks, decision trees are sometimes referred to as CART algorithms: Classification and Regression Trees. All tree-based algorithms, not just decision trees, fall under this general term. With Decision Trees, the goal is to isolate all of the data points that fall into each class by continually splitting the dataset into yes/no questions using the attributes of the dataset.

Through this procedure, a tree structure for the data is being created.Each time you ask a question, you add a node to the tree, growing it. The very first node is known as the root node.

The response to a question creates new nodes by partitioning the dataset according to the value of a feature. If you decide to stop the process after a split, the last nodes that are produced are referred to as leaf nodes.

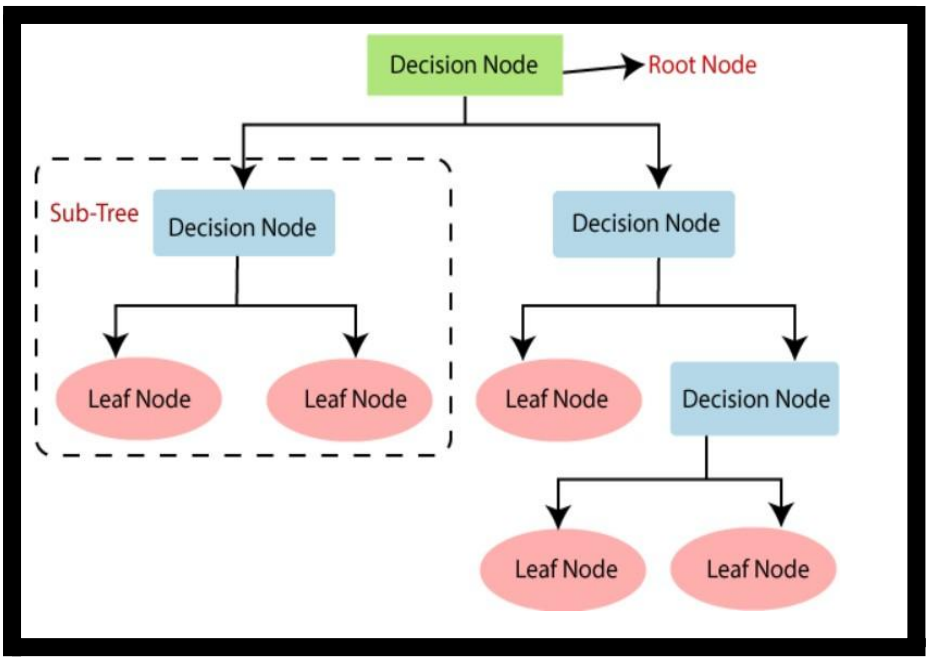


Figure 26: Decision Tree

The accuracy and Cross Validation accuracy we got by Decision Tree Classifier is-

Accuracy – 89.37%

Cross Validation Accuracy (MNB) – 83.66%

- MLP Classifier - A feedforward fully connected artificial neural network (ANN) is referred to as a multilayer perceptron (MLP). Uncertain usage of the term "MLP" results in instances where it is

broadly used to refer to any feedforward ANN and instances where it is specifically used to refer to networks composed of many layers of perceptrons. Multilayer perceptrons, especially those with a single hidden layer, are sometimes referred to as "vanilla" neural networks.

An MLP has at least three layers of nodes, including an input layer, a hidden layer, and an output layer.

Except for the input nodes, every node is a neuron that uses a nonlinear activation function. MLP employs backpropagation, a supervised learning technique. MLP uses non-linear activation and has more layers than a linear perceptron. Data that cannot be linearly separated can nevertheless be distinguished.

The accuracy and Cross Validation accuracy we got by MLP Classifier is-
Accuracy – 89.37%
Cross Validation Accuracy (MNB) – 83.66%

Then we plotted the graph between model and accuracy to compare the accuracies of various classifiers and we have seen that Random Forest gave us the highest accuracy compared to the other classifiers.

Next what we did is we loaded our dataset which is scrapped from NHP and Wikipedia and then we took input of symptom from the user and preprocessed the input symptoms.

```
Please enter symptoms separated by comma(,):
coughing,pyrexia,tire,loss of smell
```

Figure 27: Taking symptom

After this we took the user symptoms and found all its synonyms and appended it to the pre-processed symptom string.

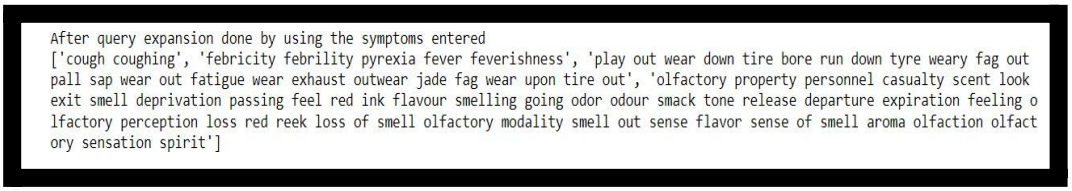


Figure 28: Appending of symptoms

Then we looped over all the symptoms in dataset and checked its similarity score to the synonym string of the user input symptoms. If similarity > 0.5 , add the symptom to the final list.

Then we print out all the symptoms and made the user to select related symptoms from the dataset.

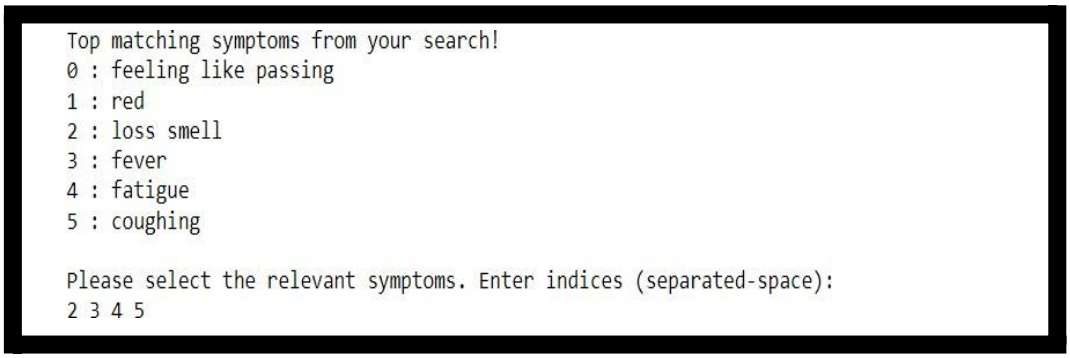
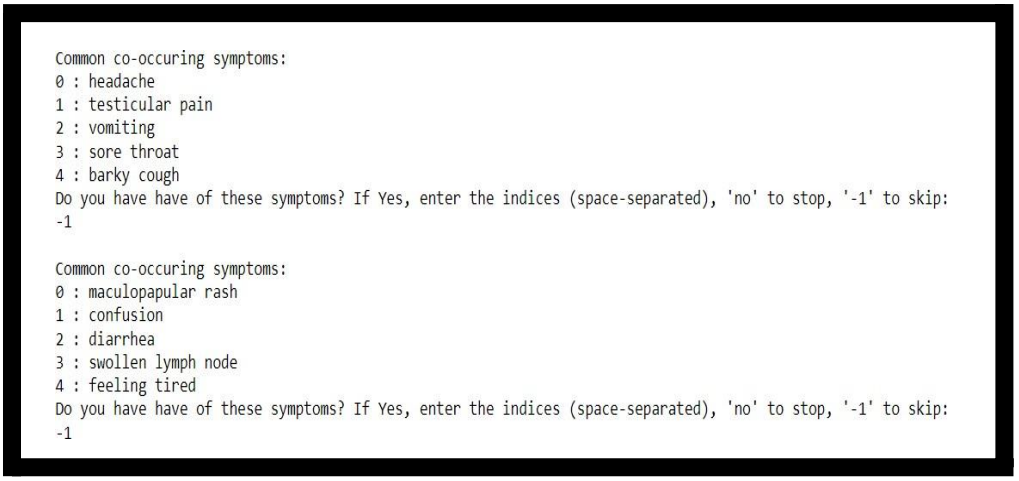


Figure 29: Related symptom

Iteratively, suggest top co-occuring symptoms to the user and asked to select the ones applicable.



For getting information about the recommended treatment user can enter the corresponding index to know more details.

Chapter 4 – PERFORMANCE ANALYSIS

500+ symptoms and over 261 different diseases are retrieved using the scraping script. The symptoms are then pre-processed to exclude ailments with similar names but differing appearances (For example, headache and pain in the forehead). To do this, the synonyms for each symptom are determined, and the Jaccard Coefficient is calculated for pairings of symptoms. If the score is higher than the cutoff, one of the symptoms can be eliminated because they are both very similar. The scraped dataset is stores as dictionary and the converted to csv format :

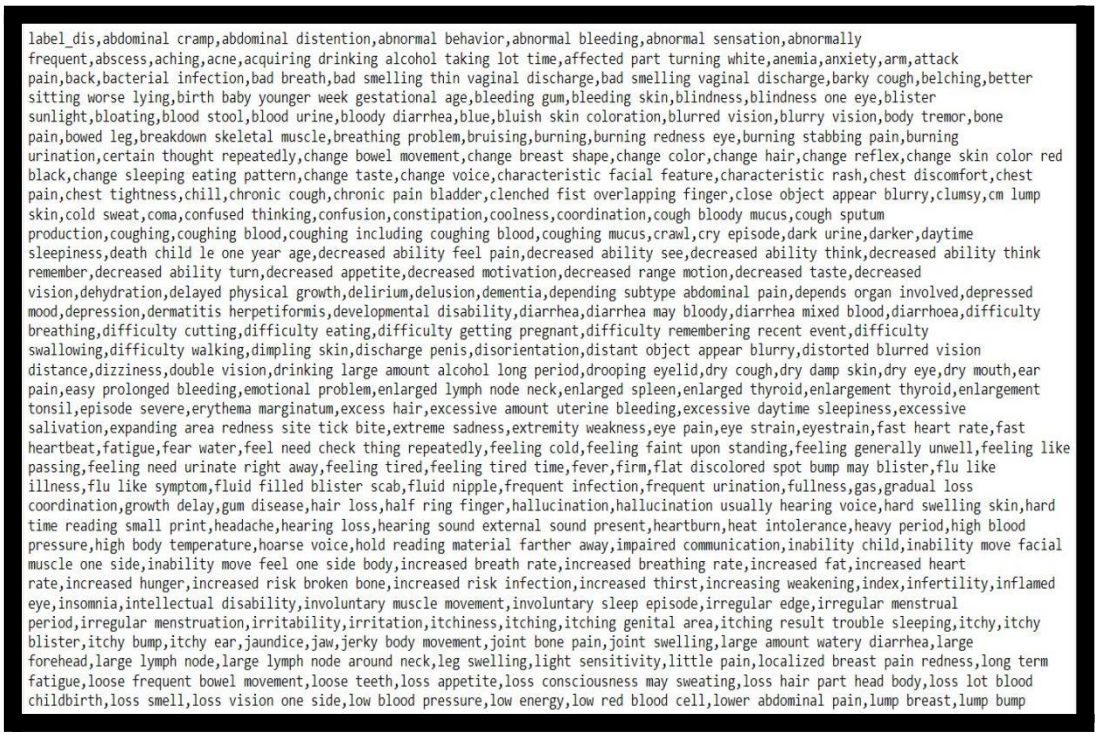


Figure 30: Scrapped data

Pre-processing is done with the help of Lemmatizer , Lemmatization is the process of combining a word's various inflected forms so they can be examined as a single entity. Lemmatization is similar to stemming, but it gives the words context. It so connects words with related meanings into a single term. Lemmatization and stemming are both parts of text preparation. These

two terms are frequently misunderstood by the public. Some individuals equate these two. Since lemmatization performs morphological analysis on the words, it is actually preferred to stemming.

After pre-processing , symptom matching is done. For each symptom pair in dataset, find Jaccard score of their synonym list. If Jaccard>threshold, it means that those synonyms are similar and one of them can be used in place of other.

```
percutaneous coronary intervention pci -> percutaneous coronary intervention
antiviral medication tenofovir -> antiviral medication
continuous birth control pill -> hormonal birth control
antibiotic supportive care -> immediate supportive care
antibiotic supportive care -> treatment supportive care
antibiotic supportive care -> supportive care
antibiotic supportive care -> prenatal care
antibiotic supportive care -> dental care
folic acid supplementation -> tranexamic acid
folic acid supplementation -> salicylic acid
folic acid supplementation -> acetic acid
folic acid supplementation -> folic acid
symptomatic dietary change -> dietary change
antibiotic drop ofloxacin -> eye drop
drinking sufficient fluid -> drinking enough fluid
drinking sufficient fluid -> drinking fluid
hyperbaric oxygen therapy -> oxygen therapy
immediate supportive care -> supportive care
incision drainage abscess -> incision drainage
treatment supportive care -> supportive care
```

Figure 31: Symptom matching

After symptom matching model classifiers is done to check the accuracy of predicted disease according to the selected symptoms.

We used 7 model classifiers to predict the accuracy. We first loaded our dataset scraped from NHP and Wikipedia , then we splitted the data for training the classifiers and testing . Classifiers used are:

- Multinomial Naïve Byes classifier
- Random Forest classifier
- K Nearest Neighbor classifier
- Logistic Regression classifier
- SVM classifier

- Decision Tree classifier
- MLP classifier

Comparison plot for all classifiers with their accuracy is:

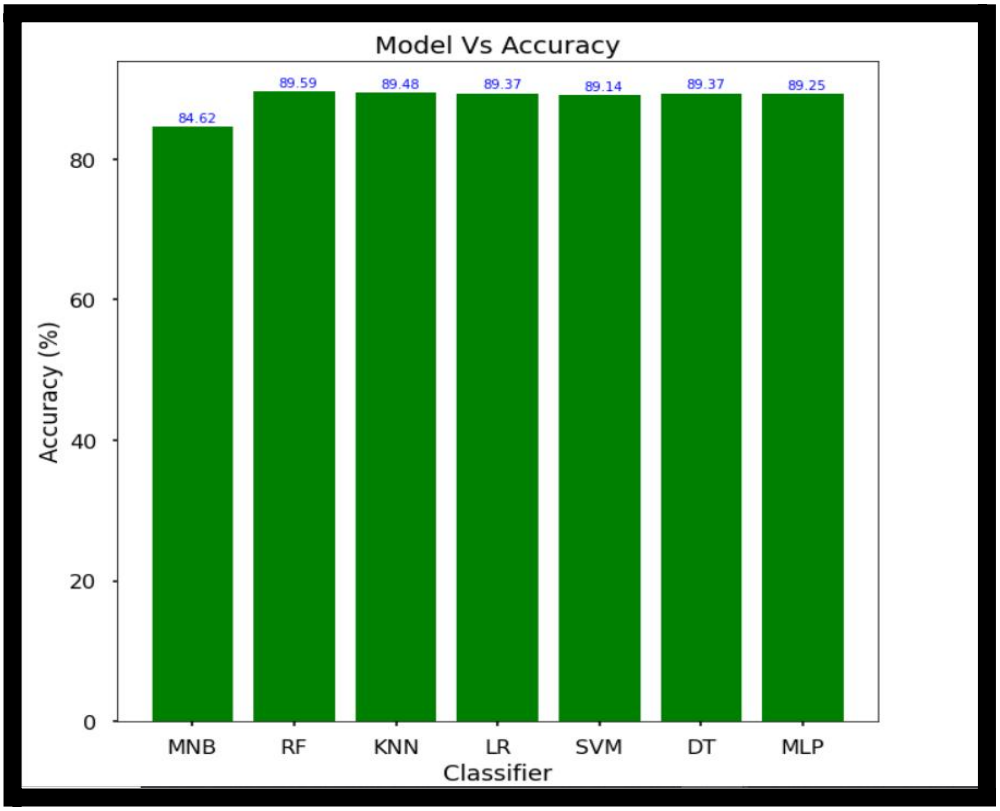


Figure 32: Model v/s Accuracy

Comparison plot for all classifiers with their cross validation accuracy is:

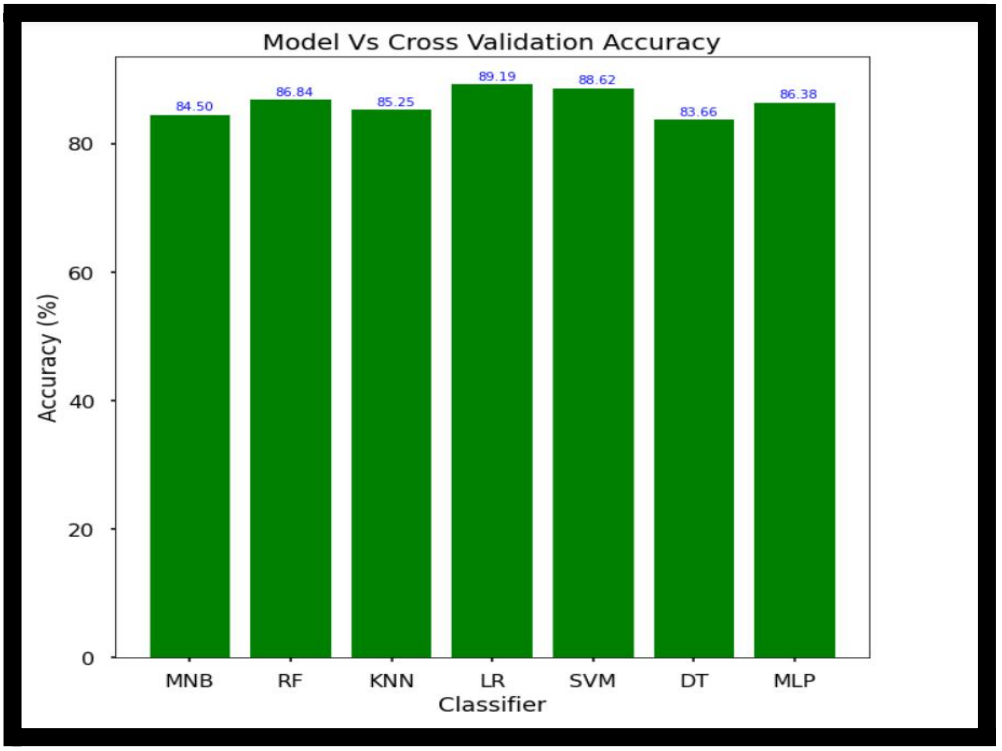


Figure 33: Model v/s Cross Validation Accuracy

Here, we have seen that the accuracy of Random Forest classifier is highest as compared to other classifiers and cross validation accuracy of Logistic Regression is more than other classifiers.

Then we took the input from the user for symptoms he/she is having:

```
Please enter symptoms separated by comma(,):
coughing,pyrexia,tire,loss of smell
```

Figure 34: Taking symptom

Taking each user symptom and finding all its synonyms and appending it to the pre-processed symptom string.

```
After query expansion done by using the symptoms entered
['cough coughing', 'febricity febrility pyrexia fever feverishness', 'play out wear down tire bore run down tyre weary fag out
pall sap wear out fatigue wear exhaust outwear jade fag wear upon tire out', 'olfactory property personnel casualty scent look
exit smell deprivation passing feel red ink flavour smelling going odor odour smack tone release departure expiration feeling o
lfactory perception loss red reek loss of smell olfactory modality smell out sense flavor sense of smell aroma olfaction olfact
ory sensation spirit']
```

Figure 35: Appending symptom

Loop over all the symptoms in dataset and check its similarity score to the synonym string of the user-input symptoms. If similarity>0.5, add the symptom to the final list .

```
['feeling like passing', 'red', 'loss smell', 'fever', 'fatigue', 'coughing']
```

Figure 36: Adding symptom after checking similarity score

We printed all found symptoms and showed the related symptoms found in the dataset and asked user to select among them. After that we find other relevant symptoms from the dataset based on user symptoms based on the highest co-occurrence with the ones that is input by the user.

```
Top matching symptoms from your search!
0 : feeling like passing
1 : red
2 : loss smell
3 : fever
4 : fatigue
5 : coughing

Please select the relevant symptoms. Enter indices (separated-space):
2 3 4 5
```

Figure 37: Top matching symptom

Iteratively, top co-occurring symptoms are suggested to the user and asked to select the ones applicable. After that calculation of TF-IDF and Cosine Similarity is done using matched symptoms.

Final list of symptoms used for prediction comes out to be –

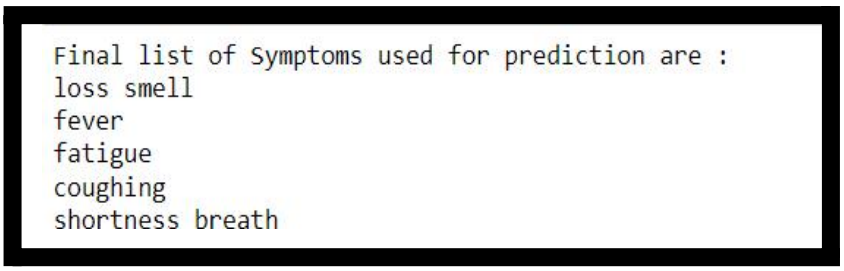


Figure 38: Final list of symptom

For getting information about the suggested treatments, user can enter the corresponding index to know more details .

Also, top 10 highly probable disease are shown to the user.

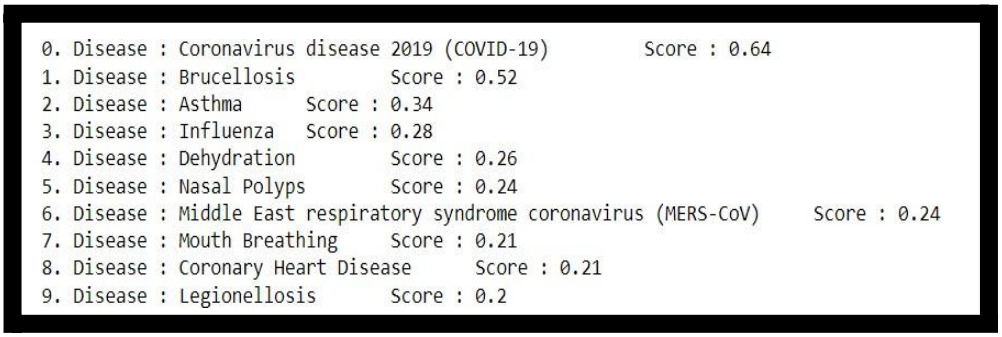


Figure 39: Top 10 highly probable disease

Also, other than disease , Treatment is recommended –

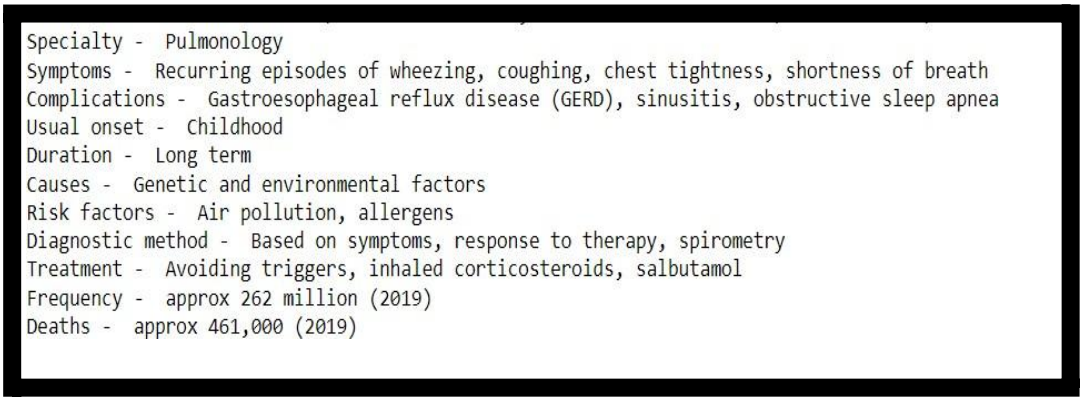


Figure 40: Treatment recommended

Chapter 5- CONCLUSION

The difficulty a common user faces in seeking medical assistance through some technology aid as an alternative to medical assistance has been partially alleviated by the suggested approach. The proposed system's excellent accuracy of more than 89% makes it trustworthy and reliable in and of itself. One of the most significant industries where machine learning can be applied to a variety of jobs is the health care industry .Making better forecasts also benefitted by scraping datasets and expanding the amount of data. Furthermore, highly developed and tuned models can further improve accuracy and produce a much more accurate model. To increase the number of samples in the dataset, disease and symptom data can also be combined from various sources, such as CDC databases, UCI databases, etc. which makes the system highly accurate.

Using the aforementioned criteria, the Random Forest model had the maximum accuracy of 89.59% in predicting illnesses. Nearly every ML model produced good accuracy values. Some models couldn't predict the disease and had low accuracy rates because they were dependent on the parameters. We could easily manage the medicine resources needed for the treatment once the disease was predicted. This model would aid in reducing the expense involved in treating the disease and would also speed up the healing process.

Finally, I want to emphasise how important this project—disease prediction and treatment recommendation using machine learning—is to everyone's daily activities, but particularly to those working in the healthcare industry, which routinely uses these systems to forecast patients' diseases based on their demographic data and symptoms. Given the large role the health sector today plays in treating patients' ailments, it frequently serves the industry's interests well to inform the user. The user can learn about the condition they are suffering by simply entering the symptoms and any other relevant information,

and the sector can profit from this system. This is advantageous for the user even if they don't want to go to a hospital or other clinics.

If the healthcare sector adopts this idea, doctors' workloads will be reduced and they will be better able to predict a patient's illness. A method known as disease prediction allows doctors to anticipate the development of a variety of common diseases that, if untreated or ignored, can cause death and a host of other issues for the patient and their family.

REFERENCES

Zhisheng Yang , Jinyong Cheng. *Recommendation Algorithm Based on Knowledge Graph to Propagate User Preference,* 2021.

Dr. D. P. Shukla Kumar Sen, Shamsher Bahadur Patel. 2013. A data mining technique for prediction of coronary heart disease using neuro-fuzzy. *International Journal Of Engineering And Computer Science*, 2:2663–2671.

Md Tahmid Rahman Laskar, Md Hossain, Abu Kamal, and Nafiul Rashid. 2016. Automated disease prediction system (adps): A user input-based reliable architecture for disease prediction. *International Journal of Computer Applications*, 133:24–29.

Ryan McDonald Slav Petrov, Dipanjan Das. 2013. A universal part of-speech tagset. *Proceedings of the Eight International Conference on Language Resources and Evaluation* (LREC 2012).

APPENDICES

Fetching diseases from site – www.nhp.gov.in

```
small_alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w'
diseases=[]
for c in small_alpha:
    URL = 'https://www.nhp.gov.in/disease-a-z/'+c
    time.sleep(1)
    page = requests.get(URL,verify=False)

    soup = BeautifulSoup(page.content, 'html.parser')

    all_diseases = soup.find('div', class_='all-disease')

    for element in all_diseases.find_all('li'):
        diseases.append(element.get_text().strip())

with open('list_diseaseNames.pkl', 'rb') as handle:
    diseases2 = pickle.load(handle)

#print(len(diseases2))
#print(len(diseases))
#print(len(set(diseases).intersection(set(diseases2))))

a=set(diseases)
b=set(diseases2)
c=list(a.union(b))
c.sort()

#print(c)

# Search diseases on google, open wikipedia page and fetch trt from infobox
```

Figure 41: Fetching disease

Iterate over all disease and preprocess symptoms string and break it into individual symptom-

```
for key in sorted(dis_trt.keys()):
    value = dis_trt[key]
    list_trt = re.sub(r"\[\S+\]", "", value).lower().split(',')
    temp_trt = list_trt
    list_trt = []
    for trt in temp_trt:
        if len(trt.strip())>0:
            list_trt.append(trt.strip())
    # Remove 'none' from symptom
    if "none" in list_trt:
        list_trt.remove("none");
    if len(list_trt)==0:
        continue
    temp = list()
    for trt in list_trt:
        trt=trt.replace('-',' ')
        trt=trt.replace("'",'')
        trt=trt.replace('(','')
        trt=trt.replace(')','')
        trt = ' '.join([lemmatizer.lemmatize(word) for word in splitter.tokenize(trt) if word not in stop_words and not word[0].
        total_trt.add(trt)
        temp.append(trt)
    diseases_trt_cleaned[key] = temp

total_trt = list(total_trt)
total_trt.sort()
total_trt=['label_dis']+total_trt

print(len(diseases_trt_cleaned))
```

Figure 42: Pre-processing

For each symptom pair in dataset, find Jaccard score of their synonym list. If Jaccard>threshold, it means that those synonyms are similar and one of them can be used in place of other.

Symptom matching is done as-

```
total_symptoms = sorted(total_symptoms, key=len, reverse=True)
symptom_match=dict()
new_symptoms = set()
for i,symi in enumerate(total_symptoms):
    for j in range(i+1,len(total_symptoms)):
        symj=total_symptoms[j]
        syn_symi = set(sym_syn[symi])
        syn_symj = set(sym_syn[symj])
        jaccard = len(syn_symi.intersection(syn_symj))/len(syn_symi.union(syn_symj))
        if jaccard>0.75:
            print(symi,"->",symj)
            # Store similar symptoms in dictionary, replace symj with symi, so
            # symptom_match[symj] = symi
            if symi in symptom_match.keys():
                symptom_match[symj] =  symptom_match[symi]
            else:
                symptom_match[symj] = symi
new_symptoms = set(total_symptoms).difference(set(symptom_match.keys()))
print(len(new_symptoms))
```

```
percutaneous coronary intervention pci -> percutaneous coronary intervention
antiviral medication tenofovir -> antiviral medication
continuous birth control pill -> hormonal birth control
antibiotic supportive care -> immediate supportive care
antibiotic supportive care -> treatment supportive care
antibiotic supportive care -> supportive care
antibiotic supportive care -> prenatal care
antibiotic supportive care -> dental care
folic acid supplementation -> tranexamic acid
```

Figure 43: Symptom matching

Various models used for disease prediction –

```
# Multinomial NB Classifier
mnb = MultinomialNB()
mnb = mnb.fit(X, Y)
# prediction of labels for the test data
mnb_pred = mnb.predict(x_test)
# calculation of accuracy score based on predictions performed
# converting to Decimal as rounding with float is inaccurate
acc_mnb = round(Decimal(accuracy_score(y_test, mnb_pred) * 100), 2)
accuracy_list.append(acc_mnb)
model_list.append("MNB")
print(f"Accuracy (MNB) : {acc_mnb}%")

# Cross Validation Accuracy MNB
# performing cross validation with 5 different splits
scores_mnb = cross_val_score(mnb, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_mnb.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (MNB): {score}%")
```

```
Accuracy (MNB) : 84.62%
Cross Validation Accuracy (MNB): 84.50%
```

Figure 44:NB Classifier

```
# RF Classifier
rf = RandomForestClassifier(n_estimators=10, criterion='entropy')
rf = rf.fit(X, Y)
# prediction of labels for the test data
rf_pred = rf.predict(x_test)
acc_rf = round(Decimal(accuracy_score(y_test, rf_pred) * 100), 2)
accuracy_list.append(acc_rf)
model_list.append("RF")
print(f"Accuracy (RF) : {acc_rf}%")

# Cross Validation Accuracy RF
# performing cross validation with 5 different splits
scores_rf = cross_val_score(rf, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_rf.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (RF): {score}%")
```

```
Accuracy (RF) : 89.59%
Cross Validation Accuracy (RF): 86.84%
```

Figure 45: Random Forest classifier

```
# KNN Classifier
knn = KNeighborsClassifier(n_neighbors=7, weights='distance', n_jobs=4)
knn = knn.fit(X, Y)
# prediction of labels for the test data
knn_pred = knn.predict(x_test)
acc_knn = round(Decimal(accuracy_score(y_test, knn_pred) * 100), 2)
accuracy_list.append(acc_knn)
model_list.append("KNN")
print(f"Accuracy (KNN) : {acc_knn}%")

# Cross Validation Accuracy KNN
# performing cross validation with 5 different splits
scores_knn = cross_val_score(knn, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_knn.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (KNN): {score}%")
```

```
Accuracy (KNN) : 89.48%
Cross Validation Accuracy (KNN): 85.25%
```

Figure 46: K Nearest Neighbor classifier

```
# LR Classifier
lr = LogisticRegression()
lr = lr.fit(X, Y)
# prediction of labels for the test data
lr_pred = lr.predict(x_test)
acc_lr = round(Decimal(accuracy_score(y_test, lr_pred) * 100), 2)
accuracy_list.append(acc_lr)
model_list.append("LR")
print(f"Accuracy (LR) : {acc_lr}%")

# Cross Validation Accuracy LR
# performing cross validation with 5 different splits
scores_lr = cross_val_score(lr, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_lr.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (LR): {score}%")
```

```
Accuracy (LR) : 89.37%
Cross Validation Accuracy (LR): 89.19%
```

Figure 47: Logistic Regression

```
# SVM Classifier
svm = SVC()
svm = svm.fit(X, Y)
# prediction of labels for the test data
svm_pred = svm.predict(x_test)
acc_svm = round(Decimal(accuracy_score(y_test, svm_pred) * 100), 2)
accuracy_list.append(acc_svm)
model_list.append("SVM")
print(f"Accuracy (SVM) : {acc_svm}%")

# Cross Validation Accuracy SVM
# performing cross validation with 5 different splits
scores_svm = cross_val_score(svm, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_svm.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (SVM): {score}%")
```

```
Accuracy (SVM) : 89.14%
Cross Validation Accuracy (SVM): 88.62%
```

Figure 48: SVM classifier

```
# DT Classifier
dt = DecisionTreeClassifier()
dt = dt.fit(X, Y)
# prediction of labels for the test data
dt_pred = dt.predict(x_test)
acc_dt = round(Decimal(accuracy_score(y_test, dt_pred) * 100), 2)
accuracy_list.append(acc_dt)
model_list.append("DT")
print(f"Accuracy (DT) : {acc_dt}%")

# Cross Validation Accuracy DT
# performing cross validation with 5 different splits
scores_dt = cross_val_score(dt, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_dt.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (DT): {score}%")
```

```
Accuracy (DT) : 89.37%
Cross Validation Accuracy (DT): 83.66%
```

Figure 49: Decision Tree classifier

```
# MLP Classifier
mlp = MLPClassifier(hidden_layer_sizes=(32, 32, 32), activation='relu', solver='adam', max_iter=50)
mlp = mlp.fit(X, Y)
# prediction of labels for the test data
mlp_pred = mlp.predict(x_test)
acc_mlp = round(Decimal(accuracy_score(y_test, mlp_pred) * 100), 2)
accuracy_list.append(acc_mlp)
model_list.append("MLP")
print(f"Accuracy (MLP) : {acc_mlp}%")

# Cross Validation Accuracy MLP
# performing cross validation with 5 different splits
scores_mlp = cross_val_score(mlp, X, Y, cv=5)
# mean of cross val score (accuracy)
score = round(Decimal(scores_mlp.mean() * 100), 2)
cross_accuracy_list.append(score)
print(f"Cross Validation Accuracy (MLP): {score}%")
```

```
Accuracy (MLP) : 89.25%
Cross Validation Accuracy (MLP): 86.38%
```

Figure 50: MLP classifier

major_projectnew.pdf 26 04 2023

ORIGINALITY REPORT

17%	6%	6%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	1%
2	www.ijraset.com Internet Source	1%
3	Submitted to University of Sunderland Student Paper	1%
4	Submitted to Centurion University of Technology & Management Student Paper	1%
5	"Pattern Recognition and Data Analysis with Applications", Springer Science and Business Media LLC, 2022 Publication	1%
6	Submitted to Rochester Institute of Technology Student Paper	1%
7	Submitted to University of Salford Student Paper	1%
	Submitted to University of Hertfordshire	