A FIELD PROJECT REPORT

ON

**“SYMPTOM DISEASE PREDICTION”**

SUBMITTED BY:

221FA04262 221FA04305

DEEPAK PRANAVI

221FA04427 221FA04568

PARVEZ SAI PRIYA

**UNDER THE GUIDANCE OF:**

Dr Deva Kumar

Professor



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed to be UNIVERSITY**

**Vadlamudi, Guntur.**

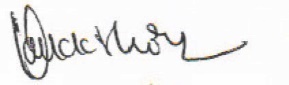
**ANDHRA PRADESH, INDIA, PIN-522213.**

**ANDHRA PRADESH, INDIA, PIN-522213.**



**CERTIFICATE**

This is to certify that the field project entitled “**symptom disease prediction**” that is being submitted by 221FA04262(Deepak), 221FA04305(Pranavi), 221FA04427(Parvez), 221FA04568(Sai Priya) for partial fulfilment of Field project is a bonafide work carried out under the supervision of Mr. Dr.DEVA KUMAR, MTech., Professor, Department of CSE

Guide name and signature Dr S.V.Phani Kumar Dr K.V. Krishna Kishore

Assistant/Associate/Professor, HOD, CSE Dean, SoCI

CSE



**DECLARATION**

We at this moment declare that the Field Project entitled **“Symptom disease prediction”** is being submitted by 221FA04262(Deepak), 221FA04305(Pranavi), 221FA04427(Parvez), 221FA04568(Sai Priya) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of XXXXX, Assistant Professor, Department of CSE.

**BY:**

221FA04262(Deepak),

221FA04305(Pranavi),

221FA04427(Parvez),

221FA04568(Sai Priya)

**DATE:**

**ABSTRACT:**

The inclusion of disease-symptom prediction in present-day health care ensures that medical practitioners and patients become aware of the potential conditions at an early stage. This project intends to establish a predictive model that correlates symptoms with probable diseases to enhance the levels of accuracy in diagnosis, reduce levels of human error, and timely interventions.  
In this project, we offer the concept of designing an intelligent system that can predict diseases based on the symptoms entered by users. This system uses a pre-existing dataset, containing historical medical records in terms of symptom-disease correlations, diagnostic patterns, and clinical outcomes. With the use of supervised learning algorithms, such as Decision Trees, Random Forests, and Neural Networks, the model will be trained to identify disease probability based on a given set of symptoms.

This methodology involves the following steps:  
Data Gathering: Aggregated and pre-processed clinical data sets comprising patient symptoms and diagnosed conditions  
Feature Engineering: Identifies the most relevant symptoms, which significantly contribute to disease prediction.  
Model Training: Used machine learning algorithms to check for precision, accuracy and disease prediction recall  
Model Evaluation: Compare and improve the algorithm by using metrics that measure the difference in their cross-validation and confusion matrix analysis.  
User Interface: Design an intuitive interface to take symptoms input and plausible diagnosis such that any visual representations of confidence in each prediction are there.  
This will be very helpful for professionals in the health care profession as secondary opinions, and patients will be able to get preliminary diagnostic insights before having clinical consultations. Prediction abilities can continue to be improved with more data introduced into the model, making this a dynamic and adaptive solution for symptom-disease correlation.

**TABLE OF CONTENTS**

1. Introduction 1

1.1 Background 2

1.2 Problem Statement 2

1.3 Objectives 2

1.4 Scope of the Project 3

1. Literature Review 4

2.1 Symptom-Disease Prediction in Healthcare 7

2.2 Overview of Machine Learning in Medical Diagnosis 8

2.3 Comparative Study of Prediction Algorithms 8

2.4 Existing Systems and Their Limitations 9

1. Methodology 10
   1. System Architecture 11

3.1.1 Data Acquisition 11

3.1.2 Preprocessing and Cleaning 11

3.1.3 Feature Engineering 11

* 1. Algorithm Selection 12

3.2.1 Decision Trees 12

3.2.2 Random Forests

3.2.3 Support Vector Machines (SVM) 12

3.2.4 Neural Networks 13

* 1. Model Training and Validation
  2. Performance Metrics 13

3.4.1 Accuracy 13

3.4.2 Precision and Recall 14

3.4.3 F1 Score 14

3.4.4 Confusion Matrix 14

1. System Design 16
   1. Data Flow Diagram (DFD) 17
   2. Database Design 18
   3. User Interface Design 21
   4. System Components and Tools Used 21
      1. Programming Languages (e.g., Python) 21
      2. Frameworks (e.g., TensorFlow, Scikit-learn) 21
2. Implementation 23
   1. Data Collection and Processing 24
      1. Symptom-Disease Datasets 24
      2. Handling Missing and Noisy Data 25
   2. Algorithm Implementation 26
      1. Training the Model 26
      2. Testing and Fine-tuning 26
   3. Integration of Frontend and Backend 27
3. Results and Discussion 32
   1. Evaluation of Model Performance 33
   2. Comparative Analysis of Different Algorithms 33
   3. Case Studies or Examples 33
   4. Limitations of the Current Model 33
4. Conclusion 31

7.1 Summary of Findings 35

7.2 Contributions to the Project 35

7.3 Future Work and Improvements 35

1. References 36

8.1 Research Papers and Articles 36

8.2 Datasets and Tools 38

1. Appendices 40

9.1 Glossary of Terms 41

9.2 Code Documentation 42

9.3 Additional Resources 43

**LIST OF FIGURES**

Figure 1.1: Block diagram 3

Figure 5.1 Distribution of symptom 1 27

Figure 5.2 Distribution of symptom 2 27

Figure 5.3 Distribution of symptom 3 27

Figure 5.4 Distribution of symptom 4 28

Figure 5.5 Distribution of symptom 5 28

Figure 5.6 Distribution of symptom 6 28

Figure 5.7 Distribution of symptom 7 29

Figure 5.8 Distribution of symptom 8 29

Figure 5.9 Distribution of symptom 9 29

Figure 5.10 Distribution of symptom 10 30

Figure 5.11 Distribution of symptom 11 30

Figure 5.12 Distribution of symptom 12 30

Figure 5.13 Distribution of symptom 13 30

Figure 5.14 Distribution of symptom 14 31

Figure 5.15 Distribution of symptom 15 31

Figure 5.16 Distribution of symptom 16 31

**LIST OF TABLES**

Table 4.1: Context Diagram for a Disease Symptom Prediction System 18

Tables 4.1.datasets 19

Tables 4.2.datasets 19

Tables 4.3.datasets 20

Tables 4.4.datasets 20

Tables 4.5.datasets 20

Table 5.1 Example dataset of data collection of processing 24

Table 6.1 Algorithm comparison for all models 33

**CHAPTER-1**

**INTRODUCTION**

1. **INTRODUCTION**
   1. **Background**

Diagnostic processes in health care have drastically changed with technology in recent years. Due to the vast amount of medical information, ML and AI have the phenomenal capability to support doctors in diagnosing diseases based on the symptoms of patients. Traditionally, diagnosing diseases requires profound medical knowledge and much time. However, symptom-disease prediction models can speed up this process by offering preliminary diagnostic insight, aiding in early detection, and minimizing human error.  
  
Health care in the entire world has applied the new machine learning capabilities, relating symptoms to enormous historical datasets to predict diseases. This project aims at developing a predictive model that identifies the most likely diseases for a patient as reported by symptoms. It is therefore to become a useful tool for all health care professionals and patients in need of early-stage diagnostic advice.

* 1. **Problem Statement**

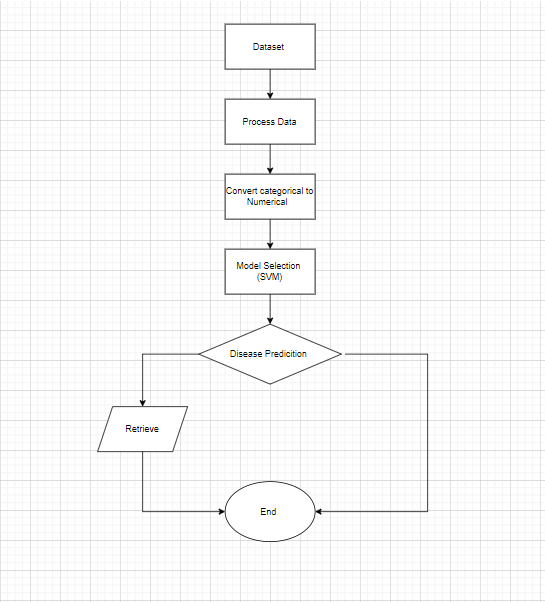
The most important issue in the medical world is still the challenge to obtain accurate and timely diagnosis of diseases, partly because of factors like the complexity of symptoms, overlapping manifestations of diseases, and the process of human error in medical evaluation. Most patients do not seek medical attention or are misdiagnosed because of the limitations of traditional systems of diagnosis.  
  
The problem this project attempts to solve is the fact that an intelligent system that could analyze symptoms and predict potential diseases is needed for healthcare providers to make faster, more accurate diagnoses. This system would also equip patients with proactive health monitoring and the ability to seek medical advice before serious illnesses progress.

* 1. **Objectives**

The overall aims of this project are:  
  
Develop a model for the prediction of potential diseases using machine learning algorithms based on a given set of symptoms.  
Compare different performance achievements of supervised learning algorithms, for instance, Decision Trees, and Support Vector Machines. Choose the best-suited model that best fits the data.  
Design an end-user interface wherein patients and healthcare professionals can easily input their symptoms to receive real-time predictions.  
To check how accurate, precise, and recall the system is while making sure that the results provided are reliable and meaningful.  
To check whether the continuous addition of new medical data improves the system.

* 1. **Project Scope**

This project focuses on designing a disease prediction model based on machine learning algorithms and developing and testing a system. It will cover:  
Detailed dataset acquisition of historical records of symptoms and diseases diagnosed.  
Preprocessing data in cases where there is missing or inconsistent data.  
Working on various supervised learning algorithms for designing the prediction models.  
Evaluating and comparing these models on grounds of accuracy, precision, recall, and F1 score performance metrics.  
Efficient design of a user interface for the model so that the patient as well as the doctor can utilize it.  
Diagnostic insights offered for a minimal number of diseases followed by the dataset utilized; intrinsically extensible to a higher number of diseases.



**Figure 1.1: Block diagram**

# **CHAPTER-2**

# **LITERATURE SURVEY**

1. **LITERATURE SURVEY**

The paper emphasizes the importance of automated disease prediction systems to address health issues arising from modern lifestyles and environmental factors. Traditional diagnostic methods can be inadequate, especially for critical illnesses, making early and accurate disease prediction crucial. Machine learning, specifically the Random Forest algorithm, has shown promise in improving diagnostic accuracy. Studies using large datasets of symptoms have demonstrated that Random Forest models can achieve high accuracy, with one model reaching 95 per cent classification accuracy. This approach aids physicians in early disease detection, offering a significant advancement over conventional methods [1]. The paper highlights the growing role of machine learning and big data in improving disease prediction and diagnosis within healthcare. Algorithms like Naive Bayes, Decision Trees, Random Forests, and Logistic Regression have proven effective in analyzing large datasets to forecast diseases based on symptoms. These models offer more accurate, data-driven healthcare solutions, enabling earlier diagnosis and better treatment outcomes. Their application is seen as a key advancement in addressing the complexity of modern healthcare challenges [2]. This paper highlights the role in overcoming challenges posed by large amounts of medical data and the complexity of accurate diagnosis based on symptoms. As healthcare generates increasing volumes of data, machine learning algorithms such as K-nearest neighbour (KNN) and Convolutional Neural Networks (CNN) have been applied to improve disease prediction accuracy. Studies show that CNN outperforms KNN in both accuracy (84.5 per cent) and efficiency, requiring less time and memory. By identifying hidden patterns in medical data, these algorithms enhance early diagnosis and provide insights into disease risks, offering more precise healthcare solutions [3]. This research presents an intelligent model for detecting contagious diseases using predictive analysis based on symptoms. The model leverages a dataset from Kaggle, applying data imputation and correlation-based feature selection. A fuzzy model-based Random Forest Algorithm (RFA) is used, achieving an accuracy of 97.61 per cent, outperforming traditional models like Support Vector Machine (SVM) and Naive Bayes Classifier (NBC). The approach enhances early disease detection and diagnosis [4]. This study presents a novel method for disease recognition and cure time prediction based on symptoms. By assigning severity scores to symptoms, diseases are identified using a numerical calculation. Reinforcement learning is employed to predict cure time by comparing the current user’s condition with similar cases. The approach also provides insights into the user’s current medical condition relative to others who have experienced the same disease [5]. This research introduces a user-centric disease prediction system leveraging machine learning and a mobile app interface. Achieving up to 100 per cent accuracy with algorithms like Multinomial Naive Bayes, Decision Tree, and Random Forest, the system offers high precision and a user-friendly experience. It also integrates supplementary health resources, enhancing early disease detection and overall health awareness [6]. This study compares various machine learning methods—Logistic Regression, Decision Trees, Random Forest, SVM, Naive Bayes, and ANN—for predicting diseases from symptoms. It also introduces a multi-disease prediction system that provides preventative actions and recommendations, aiming to improve healthcare outcomes [7]. This study evaluates quiet standing as a method to predict Parkinson’s Disease motor symptoms using UPDRS-III subscores. Analysis of force plate data from 42 PD patients and 43 controls shows that sway features effectively predict key symptoms, with high accuracy for detecting bradykinesia and postural stability. The results suggest quiet standing could improve PD diagnosis [8]. This study explores the use of machine learning to enhance predictive systems for diagnosing multiple diseases simultaneously, such as heart tumours, brain tumours, lung tumours, and diabetes. By processing extensive patient data, the system improves accuracy compared to current methods that only predict one disease at a time. This approach aims to enable better health monitoring and potentially improve patient outcomes [9]. This study presents a disease prediction system leveraging machine learning algorithms for more accurate diagnoses compared to traditional methods. Implemented using Python and Flask, the system analyzes patient data to predict illnesses effectively. This approach aims to enhance diagnostic accuracy and support better disease prevention and treatment [10]. This study develops a machine learning-based diagnostic system for accurate disease prediction, using over 50 diseases in the dataset. The system predicts diseases based on symptoms, age, and gender, and assists doctors by enabling early diagnosis and recommending nearby specialists based on patient reviews. This approach aims to improve timely treatment and potentially save lives [11]. This study utilizes the Random Forest algorithm for predicting multiple diseases, including diabetes, cancer, and heart attack, based on patient symptoms. The model, trained on a comprehensive dataset and optimized through feature selection and hyperparameter tuning, aims to enhance prediction accuracy while ensuring data privacy and compliance with healthcare standards [12]. Multi-Disease Prediction System uses machine learning and deep neural networks to predict diseases like diabetes, cancer, and chronic conditions from user-inputted health data. It offers real-time predictions and information on disease prevention, aiming to enhance accessible, proactive health management [13]. This study introduces an Intelligent Multi-Agent Reinforcement Learning-based system for disease prediction and treatment recommendation. By fetching data from network and cloud sources, the system uses Disease Attraction Weight (DAW) and Disease Curing Rate (DCR) to predict diseases and recommend treatments effectively [14]. This study develops a system to identify new, dangerous diseases with similar symptoms using Random Forest and Naive Bayes algorithms. By comparing the accuracies of both algorithms, the system predicts diseases and displays results on a smart screen to aid in early diagnosis and treatment [15]. This study leverages Big Data Analytics and Machine Learning, specifically Random Forest, to predict diseases based on symptoms. Implemented in Python with a graphical user interface, the system forecasts diseases and stores results in a database for effective management of patient records [16]. This paper proposes a framework integrating Machine Learning and Natural Language Processing for disease prediction, using a symptom dataset scraped from a UK NHS website. The study evaluates prediction efficiency through symptom frequency, similarity, and clustering analysis, highlighting positive results while identifying areas for further improvement [17]. This paper presents a Machine Learning-based system for disease prediction using a GUI, employing Naive Bayes and Decision Tree algorithms. It aims to facilitate early disease detection by analysing symptoms, providing disease information, and recommending treatments, especially relevant during times of limited healthcare access [18]. This paper proposes a disease prediction model using Decision Trees, Naive Bayes, and Random Forest classifiers, integrating results into a unified prediction. A GUI allows users to input symptoms and receive predictions based on the combined output of these models, with preprocessing and feature extraction handled in a multi-module approach [19]. This paper presents a disease prediction system integrating machine learning algorithms and deep learning techniques, including Random Forest, for image processing. The system combines symptom-based and image-based data to enhance prediction accuracy and provides proactive health insights, ensuring data security and timely intervention [20].

* 1. **Symptom-Disease Prediction in Healthcare**

Symptom-disease prediction is becoming an area of interest for healthcare, mainly due to the exponential availability of digital health records and developments in data analytics. Over the last few years, predictive models based on machine learning techniques have proven to be helpful in diagnosis by medical practitioners. Studies have shown that such models can detect complex symptoms, hence probably making predictions concerning diseases. Using such symptom patterns, the models fill in gaps in the decision-making process in clinical settings. Starting from simple rule-based systems to very advanced machine learning models, many systems have been designed for the prediction of symptoms and diseases.  
  
An example is expert systems used in clinical decision-making at IBM Watson Health, which completely exploits big medical data to support the diagnostic process as well as treatment planning. Another application is in mobile health application that allows patients to input their symptoms with corresponding suggestions made on diagnosis. Patients are aided through the use of self-assessment tools with the lightest possible burden on any healthcare system.

* 1. **Overview of Machine Learning in Medical Diagnosis**

Machine Learning in Medical Diagnosis Overview Machine learning has developed to revolutionise how healthcare diagnostics are done by equipping systems with the ability to learn from large data volumes and predict future results based on existing patterns. The most common applications of techniques used in medical diagnosis are:  
Supervised Learning: Some classification tasks in medical diagnostics use Decision Trees, Random Forests, SVMs, and Neural Networks. Such algorithms can map input symptoms with disease categories by training on the data.  
Unsupervised Learning: Techniques like K-Means and Hierarchical Clustering have been used to group symptoms and diseases to identify new patterns in health data.  
A. Deep Learning - Deep models of neural networks have effectively been applied to a lot of medical data that is either images or texts to diagnose diseases and disease-predictive occurrences.  
  
According to studies, highly accurate and precise diagnoses in common diseases like diabetes, cardiovascular conditions, and cancers can be achieved by properly trained and validated machine learning models, though large, well-annotated datasets and proper feature selection are required for them to avoid overfitting and ensure generalizability.

* 1. **Comparative Study of Prediction Algorithms**

Comparative Study of Prediction Algorithms  
Various studies have been conducted comparing different machine learning algorithms for disease prediction, pointing out their strengths and weaknesses in clinical settings:  
Decision Trees and Random Forests: So widely adopted, due to their interpretability, and capacity for complex relationships between symptoms and diseases. Recent surveys of literature signify good performance of Random Forests in tasks of medical diagnosis, owing to their nature of ensembles that reduce overfitting.  
Support Vector Machines (SVM): SVMs have been relatively successful in data that can be classified into two clear classes and are promising in medical domains, like a cancer diagnosis, where the health or disease difference is the distinguishing factor.  
Neural Networks and Deep Learning: Neural networks are highly useful in high-dimensional data, which comprises medical images and text-based symptom descriptions. Many studies prove the superiority of deep learning relative to traditional algorithms of machine learning in disease prediction tasks that encompass high complexity but pose challenges in terms of its black-box nature and increased computational requirements.  
In medical literature, this is where Random Forest and Neural Networks mostly come up as the best-performing models due to their robust performance and ability to capture intricate relationships in data. However, in scenarios where interpretability matters, simpler models such as Decision Trees are favoured.

* 1. **Existing Systems and Their Limitations**

Several existing systems aim to predict diseases based on symptoms. A few prominent ones include:  
WebMD Symptom Checker: This is a popular web application where one inputs symptoms and gets a list of possible causes. It relies on predefined rules and is therefore a type of model that does not adapt as machine learning models do.  
Intermedia: The probabilistic model is employed, and the disease prediction relies on symptoms and medical data. This AI solution has been implemented in various healthcare systems and has shown reasonable performance in real-world clinical applications.  
Ada Health: Symptom-checking app powered by probabilistic models that can be used by consumers in assessing preliminary health. However, this tool does not make complete use of machine learning strategies, which might raise a limitation of its use in more complex diagnoses.  
  
These systems diagnose helpful information for diseases and encounter some limitations:  
Rule-based Systems: The early applications were based on existing rules that limited the ability to adapt to new information. Such systems cannot predict illnesses with symptoms that overlap and can only learn from cases sequentially.  
Small Datasets: Most of the systems use very small data focused on a niche category that limits the applicability to other groups as well. Diverse training data is missing in most of the applications leading to biased models.  
Interpretability: Machine learning models, bottomless learning models, tend to be "black" and it is tough to explain how they arrive at a specific diagnosis. Its use is, therefore, restricted to the clinical area where interpretability is critical.

**CHAPTER-3**

**METHODOLOGY**

1. **METHODOLOGY**
   1. **System Architecture**

**3.1.1 Data Gathering**

Electronic Health Records: Structured patient data including symptoms, diagnoses, and treatments.

Medical Databases: Most are publicly available datasets provided by health organizations such as the Centres for Disease Control and Prevention, the World Health Organization, or academic institutions. The data can feature several symptoms and corresponding disease labels.

Wearable Devices or IoT Sensors: Data from health trackers that could be set to monitor symptoms such as heart rate, temperature etc.

Self-reported Data: Surveys and symptom trackers wherein the patients are supposed to report their symptoms and health conditions.

(Write the description about where the dataset is taken)

**3.1.2 Preprocessing and Cleaning**

Handling Missing Data: This is usually found in most medical data. Filling missing values can be achieved by mean, median imputation, or even k-nearest neighbour imputation. Alternatively, one can just delete rows with missing critical data.

Normalization and Scaling: Features can be on different scales. For instance, age and temperature are on two different scales. Normalization (say min-max scaling) or standardization like Z-score normalization is applied to make features on a comparable scale.

Removal of outliers: Outliers can significantly skew the model. Statistical techniques or knowledge of the domain could be used to identify data points not in line with the majority trend for their removal.

Categorical Encoding: All symptoms are essentially categorical data (yes/no, symptoms are present or not). One-hot encoding or label encoding is applied to the categorical data to convert it to numerical data.

**3.1.3 Feature Engineering**

**Symptom grouping:** combining several symptoms that go together (for example, respiratory) so the diseases that involve more than one body area can be represented as better

Including the interaction terms: new features that involve the interaction between different symptoms, along with the interaction of fever and cough, to collect more advanced interactions between the symptoms and the disease.

Temporal Features: If data is available over a period (such as an evolution of symptoms), time-dependent features that may include symptom duration or frequency of occurrence can be designed.

Dimensionality Reduction: Techniques like PCA could decrease the feature set, yet maintain the important variance in the data. This reduces noise and makes the model train faster.

Derived Features: Include relevant features extracted from the given data. For instance, severity scores or risk factors could be created to improve the model's discriminative capability between diseases with similar symptoms.

**3.2 Algorithm Selection**

**3.2.1 Decision Trees**

Working Principle: At each split, the algorithm always selects the attribute that best splits data by some impurity measure like the Gini Index or Information Gain, in the case of classification. A tree can grow only up until some stopping criterion is met, such as maximum depth or minimum number of samples at a node.

Advantages:

Interpretability: Decision trees are very transparent and hence highly interpretable. It is quite easy to visualize the pathway of decision-making followed by a tree to let the doctor understand why it has made certain predictions.

Applicability to Mixed Data Types: They can quite easily handle numerical data as well as categorical data.

Non-parametric: No assumptions regarding data distribution of the underlying data are required to be made by the decision tree that enables their use in a wide variety of different datasets.

Disadvantages:

Overfitting: This is one where decision trees very easily overfit on small datasets or where the tree is too deep. Generalization over unseen data is going to be relatively poor.

Instability: Small changes in the dataset may radically change the structure of the tree.

**3.2.3 Support Vector Machines (SVM)**

Support Vector Machines is a strong algorithm that can particularly be very useful in high-dimensional space for both classification and regression tasks. For predicting disease symptoms, SVM can successfully find boundary positions between the two classes of diseases.(Accuracy: 96.95%)

Working Principle: SVM is a method of finding the optimal hyperplane with maximum separation between classes. In the case of inseparable classes linearly, SVM resorts to kernel tricks like polynomial and radial basis functions to transform the data to a high dimensional space where using a linear separator makes it possible to separate classes.

Advantages:

SVM can perform well in high-dimensional spaces since, in medical datasets, often the number of symptoms can be more than the number of samples available.

SVMs normally avoid overfitting properly due to controls over complexity with proper regularizations.

Kernel flexibility: The feature of flexibility of kernels in SVM for applicability makes handling nonlinear relationships among features and target classes feasible.

Drawbacks

Complexity: SVMs are heavily dependent on parameters (e.g., kernel selection, regularization parameter) and can be computationally expensive with large databases.

Interpretability: Decision trees are much more intuitive than SVMs, which could be a problem in medical applications when transparency is a concern.

**3.2.4 Neural Networks**

Working Principle: Neural networks are layers of connected neurons. The input layer receives the symptom data, hidden layers take this through weights and the activation functions to return a meaningful result, and the output layer is the disease predictability. Learning for a neural network takes place through the readjustment of weights within the training process which reduces predictive errors.

Advantages

Requires Huge amounts of Data: Neural networks perform much better with large amounts of data and may not always be available in a clinical environment.

Lack of Interpretability: Neural networks are black boxes, similar to SVM, but it is hard to explain how the predictions are done, which is a serious concern in healthcare where predictivity should support interpretability because of the importance of the domain.

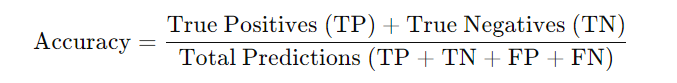
* 1. **Model Training and Validation**

This uses a labelled dataset, where feature values are symptoms, and the target labels are diseases. Cross-validation techniques, such as k-fold, are used to avoid overfitting and tune the hyperparameters for better performance. Finally, the model is validated on unseen data for generalizing the model.

**3.4 Performance Metrics**

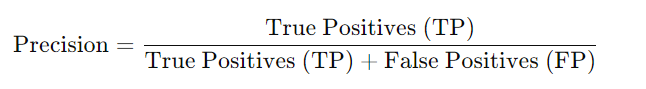
**3.4.1 Accuracy**

Accuracy Accuracy is the simplest measure of how often the model makes accurate predictions. What is measured is the proportion of total cases observed that are classified correctly (both as true positives as well as as true negatives).

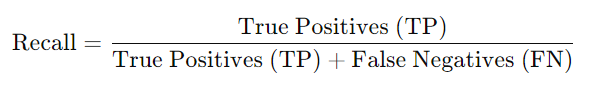


**3.4.2 Precision and Recall**

* Precision measures the proportion of positive predictions that are true positives. It focuses on the precision of the model in predicting a positive and is applicable when false positives are more costly to fail than false negatives.



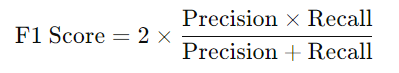
* Recollect Sensitivity or True Positive Rate = Actual Positives Correctly Identified by the Model. This is crucial where the cost of missing a positive case, or false negative, is a high one.



* Trade-off: High precision but low recall conserves on positives but misses many true positive cases. High recall but low precision captures most positive cases but makes many false positive errors.

**3.4.3 F1 Score**

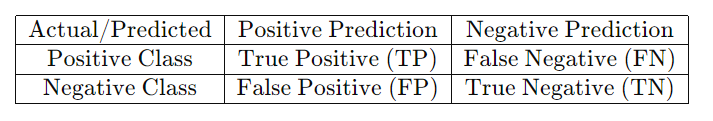
The F1 Score is a harmonic mean of precision and recall, resulting in a single metric that can be balanced with the trade-off between the two. It is especially useful when an imbalanced dataset is involved, where both false positives and false negatives come into play.



**3.4.4 Confusion Matrix**

The Confusion Matrix provides a detailed insight into the model’s performance by breaking down the results into four categories:

* True Positives (TP): The model correctly predicts the positive class.
* True Negatives (TN): The model correctly predicts the negative class.
* False Positives (FP) (Type I error): The model incorrectly predicts the positive class when it's negative.
* False Negatives (FN) (Type II error): The model incorrectly predicts the negative class when it's positive.



**CHAPTER-4**

**SYSTEM DESIGN**

1. **System Design**
   1. **Data Flow Diagram (DFD)**

A Data Flow Diagram describes how data flows through a system: the direction of information flow from one part of the system to another. One such DFD for a disease symptom prediction system is described as follows:

Level 0: Context Diagram

* System: A disease symptom prediction system
* External Entities:
  + Users (Patients/Doctors): Input symptoms and medical history.
  + Source of historical data and medical records and currently utilized healthcare databases.
  + Analysis Tools: provides reports and insights based on the predictions.

|  |  |  |  |
| --- | --- | --- | --- |
| High fever | 378 | Dark urine | 144 |
| 0 | 348 | Nausea | 120 |
| Lethargy | 222 | Loss of appetite | 120 |
| Yellowish skin | 216 | Breathlessness | 108 |
| Sweating | 216 | Hip joint pain | 96 |
| Weight loss | 192 | Blurred and distorted vision | 96 |
| Vomiting | 174 | Movement stiffness | 96 |
| Cough | 162 | Restlessness | 96 |
| Loss of balance | 144 | Bloody stool | 96 |
| Mood swing | 90 | Obesity | 96 |
| Spinning movements | 90 | Dizziness | 84 |
| Swelling of stomach | 96 | Silver like dusting | 96 |
| Abdominal pain | 84 | Red sore round nose | 84 |
| Altered sensorium | 78 | Diarrhoea | 78 |
| Scurring | 78 | Extramarital contacts | 78 |
| The continuous feel of urine | 78 | Burning micturition | 78 |
| Chest pain | 78 | Diachronic patches | 72 |
| Watering from eyes | 72 | Headache | 54 |
| Lack of concentration | 42 | Spotting urination | 42 |
| Family history | 36 | Passage of gases | 36 |
| Yellow crust ooze | 36 | Swelling joints | 24 |
| Distention of abdomen  Excessive hunger | 24  24 | Painful walking  Irregular sugar level | 24  24 |
| Fatigue | 24 | Swollen legs | 24 |
| Small dents in nails | 24 | Yellowing of eyes | 24 |
| Irritation in anus | 24 |  |  |

**Table 4.1: Context Diagram for a disease symptom prediction system**

**Processes:**

1. Data Acquisition
   * Input: Symptom data from users.
   * Output: Collected data for preprocessing.
2. Preprocessing and Cleaning
   * Input: Raw data (symptoms, medical history).
   * Output: Cleaned data ready for feature engineering.
3. Feature Engineering
   * Input: Cleaned data.
   * Output: Enhanced feature set for model training.
4. Model Selection and Training
   * Input: Feature set.
   * Output: Trained model for disease prediction.
5. Model Validation
   * Input: Trained model and test data.
   * Output: Model performance metrics (accuracy, precision, recall).
6. Prediction Generation
   * Input: User symptom data.
   * Output: Predicted disease or condition along with confidence score.
7. User Feedback and Adjustment
   * Input: Feedback on predictions from users.
   * Output: Adjustments to model parameters or retraining needs.
   1. **Database design**
8. **Identify Major Entities**

As part of the ML project, the following should be provided as major entities:

Datasets: Datasets for training, validation, and testing.

Models: Details of the trained models such as hyperparameters, architecture, and version.

Training Jobs: All the individual training instances along with their related data

Metrics: Accuracy, precision, recall, F1 score, etc., that have been obtained through training and evaluation.

Predictions/Outputs: What the model predicts when tested or when it is live in inference.

Experiments: A group of different runs of training jobs along with their respective results.

Data Preprocessing: Transformation or feature engineering steps.

1. **Database Tables**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Dataset\_id | INT(PK) | Unique identifier for each dataset |
| Dataset\_name | VARCHAR | Name of the dataset |
| Data\_type | VARCHAR | e.g.image, text, tabular |
| Created\_at | TIMESTAMP | Date when the dataset was uploaded |
| Dataset\_size | INT | Number of samples in the dataset |
| Preprocessing\_steps | TEXT | Steps taken to preprocess the dataset |
| Data\_source | VARCHAR | Source of the data |

**Tables 4.1.datasets**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Model\_id | INT(PK) | Unique identifier for each model |
| Model\_name | VARCHAR | Name of the model |
| Version | VARCHAR | The version of the model |
| Architecture | VARCHAR | The architecture used(e.g.CNN,RNN) |
| Hyperparameters | JSON | Hyperparameter used in the model training |
| Created\_at | TIMESTAMP | Timestamp when the model was trained |

**Table 4.2 Models**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Job\_id | INT(PK) | Unique identifier for each training job |
| Model\_id | INT(FK) | Reference to the trained model |
| Dataset\_id | INT(FK) | Reference to the dataset used |
| job\_status | VARCHAR | Status of the job (completed, failed, etc) |
| Start\_time | TIMESTAMP | The time when the job started |
| End\_time | TIMESTAMP | The time when the job completed |
| Execution\_time | FLOAT | Total time taken for training(in seconds) |
| System\_resourse | JSON | Resource usage (e.g. GPU, RAM) |
| Notes | TEXT | Additional notes or observation |

**Table 4.3. training\_jobs**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Meteric\_id | INT(PK) | Unique identifier for the metric |
| Job\_id | INT(FK) | Reference to the training job |
| Metric\_name | VARCHAR | Name of the metric(accuracy,F1,etc) |
| Value | FLOAT | Value of the metric |
| Timestamp | TIMESTAMP | The time when the metric was recorded |

**Table 4.4. metrics**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Prediction\_id | INT(PK) | Unique identifier for the prediction |
| Model\_id | INT(FK) | Reference to the model used for prediction |
| Dataset\_id | INT(FK) | Reference to the dataset |
| Input\_data | JSON | Input data to the model for prediction |
| Prediction\_output | JSON | Model output |
| prediction\_time | TIMESTAMP | Time when the prediction was made |

**Table 4.5. Prediction**

1. **relationships between tables**

datasets → training\_jobs (1: One dataset can be used for multiple training jobs.

models → training\_jobs (1: One model can have multiple training jobs.

training\_jobs → metrics (1: Each training job generates multiple metrics.

models → predictions (1: One model can generate multiple predictions.

* 1. **User Interface Design**

For a disease symptom prediction system, the following would be cardinal considerations in designing the user interface:

Ease of Use. The interface should be simple, intuitive, and user-friendly, mainly because the end-users may not necessarily be technically skilled. For example, users should be able to easily enter their symptoms and immediately receive feedback without convoluted navigation

Information Clarity: Show the prediction model results clearly: all the actual diseases predicted, in ascending probability order with appropriate confidence scores. If graphics will make the results easier to understand, then include them in graph or chart form.

Responsiveness: All UI components must function correctly on desktops tablets and smartphones.

Error Handling and Feedback: Whenever something illegal happens with input or during processing, let the user know. There must be feedback for missing or invalid data so that the user can correct it accordingly.

Privacy and Security: Since health-related data is sensitive, the users must be assured about their data's privacy by the UI. For this purpose, privacy policies and compliance statements need to be placed in highly visible positions.

Accessibility - Make sure that the design is people with disabilities-friendly and consider options such as text-to-speech, voice input, or high-contrast modes.

* 1. **System Components and Tools Used**

This section deals with the system architecture and components selected to design the disease symptom prediction system. The languages and frameworks for programming are used here.

* + 1. **Programming Languages (e.g., Python)**

Python: Python is the most widely used language to develop machine learning applications, following due to its simplicity, readability, and wide ecology of libraries and frameworks. It enables fast prototyping and supports scientific computing rather extensively, making it an excellent choice for this system.

R: If statistical analysis has to be more stressed, R might be another alternative, particularly for deep statistical insight into prediction models.

JavaScript Optional For a web-based interface, JavaScript along with HTML and CSS may be used to develop the front end of the web application.

SQL: For storing user inputs and model results, SQL or NoSQL databases, such as MongoDB, may be used for managing the database.

* + 1. **Frameworks (e.g., TensorFlow, Scikit-learn)**

**Scikit-learn**: This Python library is used for classical machine learning algorithms like decision trees, random forests, and logistic regression. For symptom prediction, it is ideal due to its ease of use and integration with other Python tools like Pandas and NumPy.

**TensorFlow / Keras**: For more advanced prediction models involving deep learning, frameworks like TensorFlow or its high-level API Keras could be used. They are useful when working with neural networks for complex symptom patterns.

**Pandas and NumPy**: These libraries are essential for data manipulation and numerical computing. They allow you to clean, process, and transform the symptom datasets before feeding them into machine-learning models.

**NLTK / SpaCy**: If the system includes natural language processing (NLP) features, such as allowing users to enter free-form text to describe symptoms, NLP libraries like NLTK or SpaCy would help extract relevant information.

**Matplotlib / Seaborn**: For visualizing the prediction results, these libraries can generate plots and charts showing trends, probabilities, or risk factors associated with the predicted diseases.

**Flask / Django**: If the prediction system is delivered as a web application, back-end frameworks like Flask or Django can be used to handle user requests, serve the prediction results, and manage sessions.

**CHAPTER-5**

**IMPLEMENTATION**

1. **Implementation**
   1. **Data Collection and Processing**

Data collection and processing is part of any ML project, especially in more challenging areas of application areas like healthcare disease prediction based on symptoms. The quality of the data is what really reflects the accuracy and trustworthiness of your model. This section explains how to collect symptom-disease datasets and handle common data challenges such as missing or noisy data.

* + 1. **Symptom-Disease Datasets**

Sources of Data:

Clinical Datasets: These are real data obtained from hospitals, medical research institutions, or clinical trials. In most instances, this type of data typically contains documented symptoms and subsequent diagnoses.

Public Health Databases: One of the examples here is the UCI Machine Learning Repository, MIMIC-III, or Kaggle; most of the datasets on conditions and symptoms are often readily available.

Medical Literature: This source comprises published papers and available journals on medical conduct that could be used to determine a relationship for a symptom.

Symptom checkers: Some apps or websites (such as, for example, WebMD or Mayo Clinic symptom checkers) can give you a formatted set of symptom-disease pairs.

Data Format and Structure:

Your dataset should, ideally contain the following:

Patient ID: A unique identifier for every patient; this may be anonymized for privacy purposes.

Symptoms: A list or set of symptoms presented by the patient. This could be, for example, fever, cough, or fatigue.

Diagnosis: The disease or condition diagnosed. Such could be influenza, common cold, COVID-19, among others.

Additional Metadata: Age, gender, other relevant information, and/or medical history.

**Example dataset:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Patient ID** | **Symptoms** | **Diagnosis** | **Age** | **Gender** |
| 001 | Fever, cough, fatigue | Influenza | 30 | Male |
| 002 | Headache, Dizziness, Vision Blurred | Migraine | 45 | Female |
| 003 | Sore Throat, Runny Nose, Cough | Common Cold | 25 | Male |
| 004 | High Fever, Cough, Difficulty Breathing | COVID-19 | 55 | Female |

**Table 5.1 Example dataset of data collection of processing**

**Data Collection Challenges:**

Anonymization and Privacy. Whenever clinical datasets are collected, it has to be within the bounds of those regulations particular to health-care privacy (HIPAA for the United States, and GDPR in Europe, for example). Techniques for anonymizing clinical data can prevent the onset of identity disclosure for patients.

Data Imbalance: There can be some diseases underrepresented. It impacts the performance of the model. The techniques that can be used are oversampling or SMOTE (Synthetic Minority Over-sampling Technique).

* + 1. **Handling Missing and Noisy Data for ML Report**

Missing and noisy data tend to dominate healthcare data; hence proper care needs to be taken for reliable modelling. Techniques for dealing with the above-mentioned problems are as follows:

1. **Handling Missing Data**

Missing data may sometimes be due to a particular reason and sometimes not, due to many factors. The patient may not report all his symptoms or may not have conducted a specific diagnostic test.

**Types of Missing Data:**

* MCAR (Missing Completely at Random): Data missing without any pattern.
* MAR (Missing at Random): Data missing due to some other observed data (for example, older patients will not have anything to say about some symptoms).
* MNAR (Missing Not at Random): Missing because of the characteristics of the missing value itself. For example, patients with severe symptoms tend to omit them.

**Methods for Dealing with Missing Data:**

1. **Delete Missing Data (Listwise Deletion):**

* Delete entire rows where data is missing.
* Advantages: Easy, time-saving.
* Disadvantages: Declines the size of the dataset and may introduce biases due to loss of randomness in missing data.

1. **Imputation:**

* Mean/Median/Mode Imputation: Replace missing data by its mean/middle value (in case of numerical values) or mode (in the case of categorical data) for the column.
  + Advantages: It is easy to implement.
  + Disadvantages: This can induce homogeneity reducing the variability and also distorting the relationship between the variables.
* K-Nearest Neighbors (KNN) Imputation: The missing values are imputed based on using the most similar data points.
  + Advantages: It takes into account the relationships between variables.
  + Disadvantages: Computationally intense for large data sets.
* Regression Imputation: Missing values are predicted based on a regression model formed by other available features
  + Advantages: More accurate than simple imputation methods
  + Disadvantages: Can be complex and prone to overfitting.
* Isolation Forests or One-Class SVM: These are machine learning algorithms that can be used to find anomalies or outliers.
  + Advantages: These have more generic versions of working well with high-dimensional data.
  + Disadvantages: They happen to be computationally intensive.
* Data Cleaning:
  + Manual Inspection: It is sometimes useful to manually inspect suspected data points, particularly when there are sensitive domains like health care.  
    Automated Rule-Based Cleaning: Apply rules that cover commonplace anomalies, like values out of bounds, for example, a body temperature beyond the range that a human can survive under.
* Data Augmentation:  
  It is possible to generate synthetic data where noisy data leads to data scarcity or over-sampling for less represented classes. 3: How to Deal with Noisy Data  
  Inconsistent data arises from inconsistent formatting or information that conflicts. To correct this:
  + Normalisation: Normalize the units of measurement or text symptoms.
  + Deduplication: Remove the dups not to skew the model's performance.  
    Standardization of categorical data: Normalise synonyms, like "fever" and "high temperature," into a standard format.
  1. **Algorithm Implementation**
     1. **Training the Model**

Explain the training process of your machine learning model in this section.

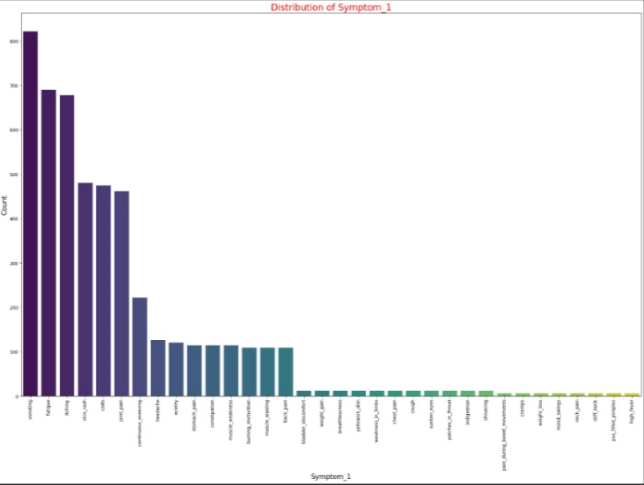
Data Preprocessing: Before actual training of the model, you must have pre-processed the raw data. This consists of (enumerate any cleaning activity, normalization, and/or feature extraction) you have applied for it. Also, let the reader know if you have employed any techniques like data augmentation or oversampling to balance your dataset.

Model Architecture: The model architecture employed for this project is SVM

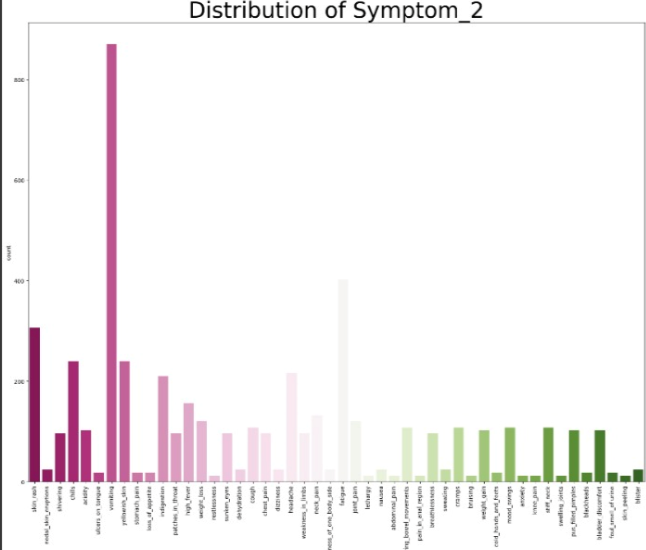
Training Method: The model was trained with (type of training algorithm used, for example, stochastic gradient descent, Adam optimizer etc.). We employed (a dataset split into training and validation sets). Using the learning rate set to (learning rate), the epoch number for the training would be set at (number of epochs).

* + 1. **Testing and Fine-tuning**

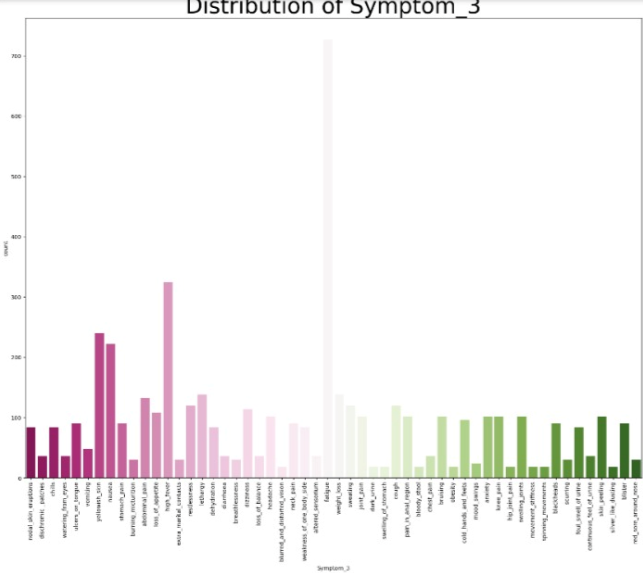
This section focuses on testing and fine-tuning the trained model:  
Testing: Once trained, the model was tested on an unseen test set to report how well it generalizes. The test set consisted of (mention number and characteristics of the test data). We used (mention metrics) to measure performance: with a (mention the result).  
Hyperparameter Tuning: The model had fine-tuned the hyperparameters, such as learning rate, batch size, or regularization techniques like L2 regularization or dropout. Hyperparameter search: Grid search, Random search, and Cross-validation have been used to find an optimal set of hyperparameters.  
Model Validation: The model was validated using a k-fold cross-validation/validation set. Along the fine-tuning, validation loss and accuracy were monitored so that it was known that the model was not overfitting to the training set and generalized well.  
Final Model: With fine-tuning, the last model achieved an accuracy of (state final accuracy or performance metrics). The model is ready to be deployed or used to further test in a real-world application.

****

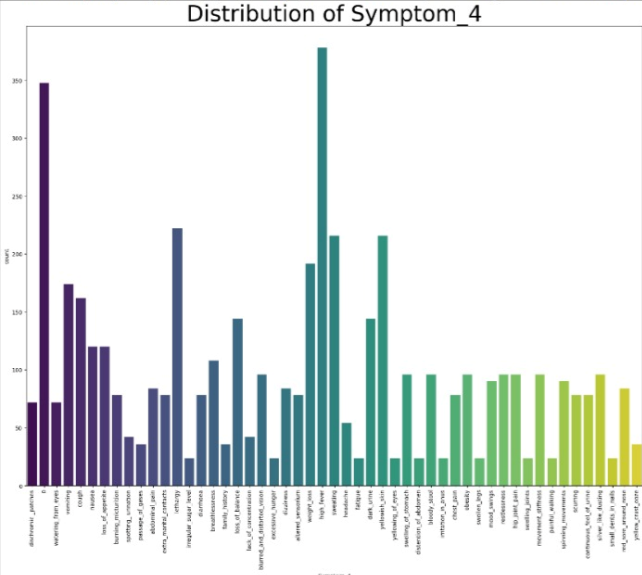
**Figure 5.1 distribution of symptom 1**

****

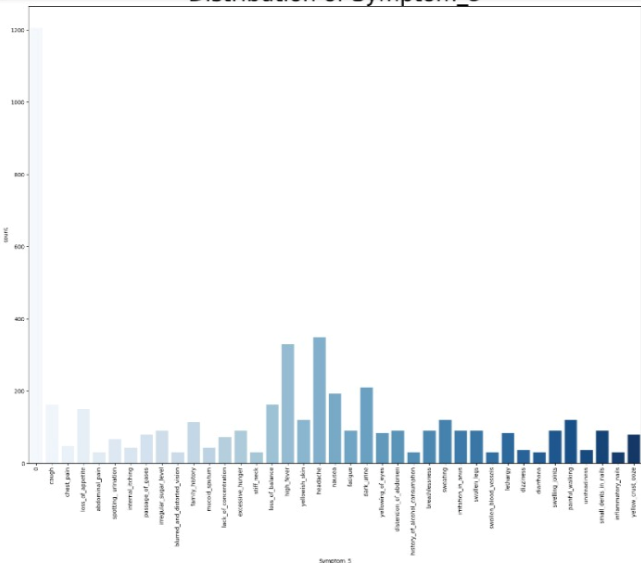
**Figure 5.2 distribution of symptom 2**

****

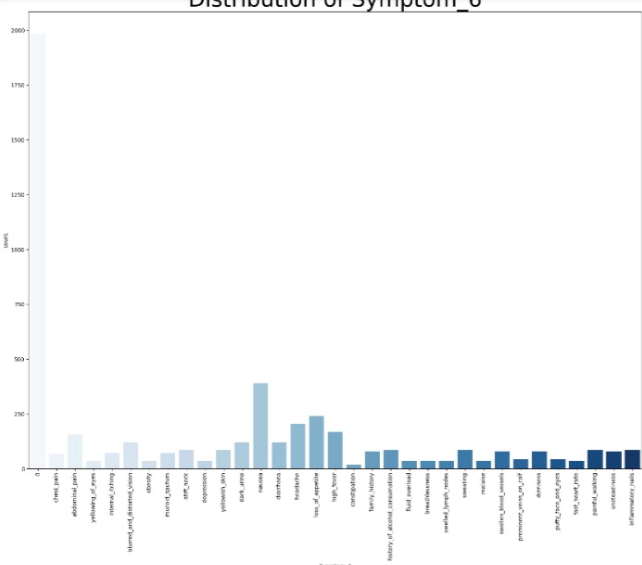
**Figure 5.3 distribution of symptom 3**

****

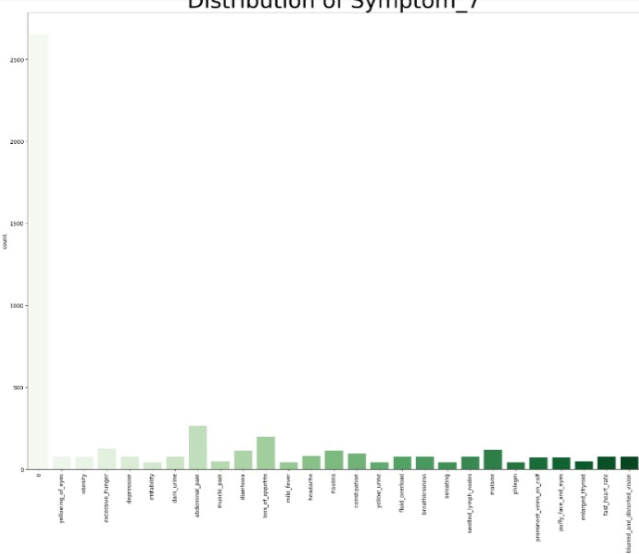
**Figure 5.4 distribution of symptom 4**

****

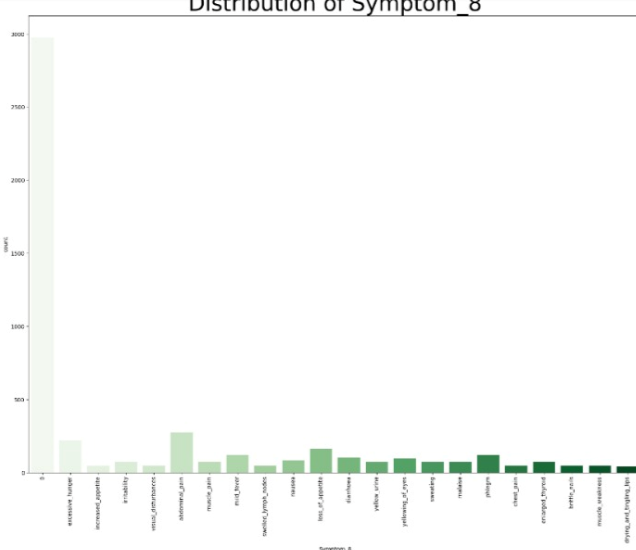
**Figure 5.5 distribution of symptom 5**

****

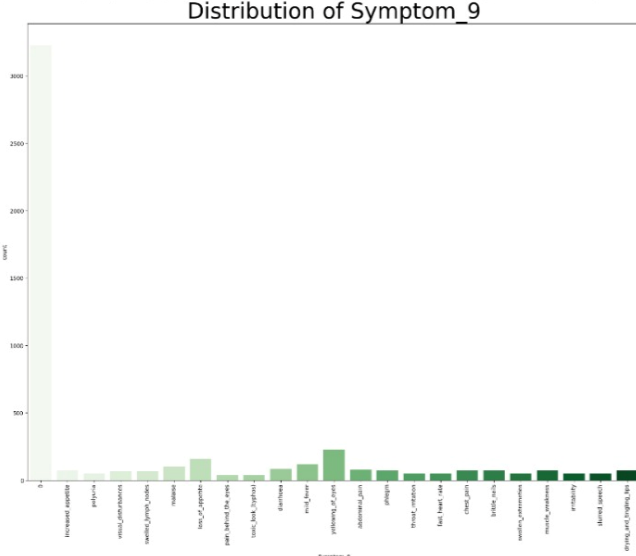
**Figure 5.6 distribution of symptom 6**

****

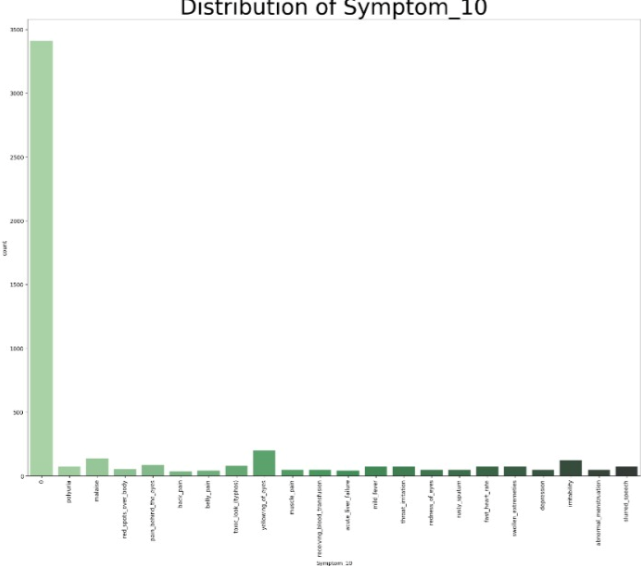
**Figure 5.7 distribution of symptom 7**

****

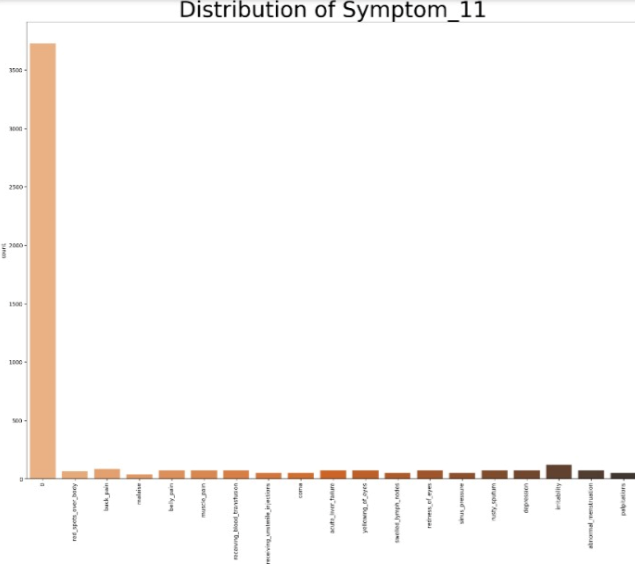
**Figure 5.8 distribution of symptom 8**

****

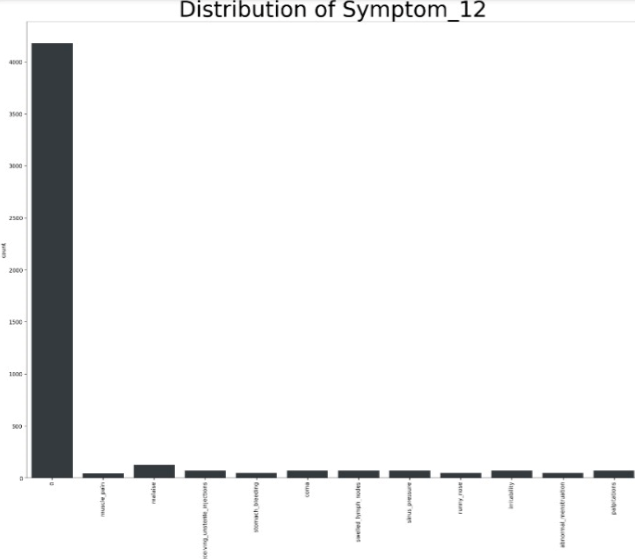
**Figure 5.9 distribution of symptom 9**

****

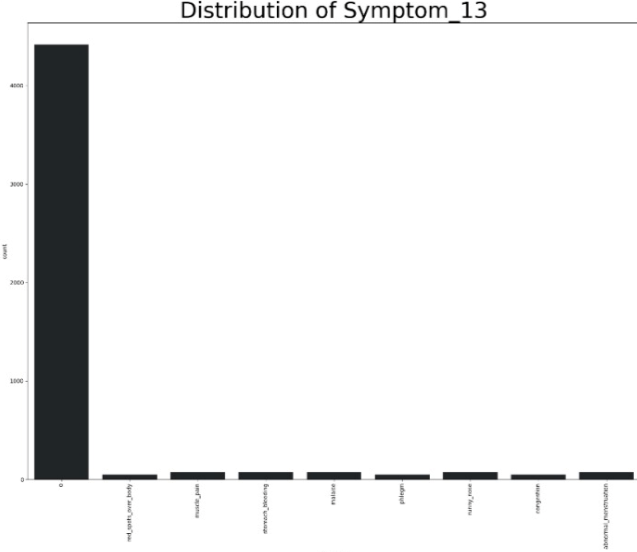
**Figure 5.10 distribution of symptom 10**

****

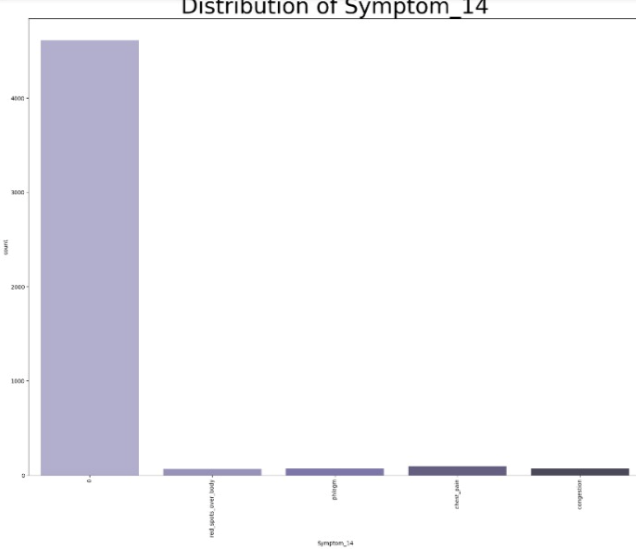
**Figure 5.11 distribution of symptom 11**

****

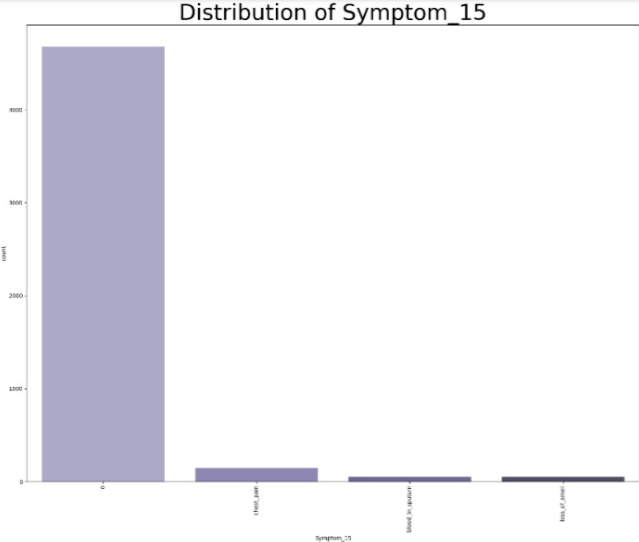
**Figure 5.12 distribution of symptom 12**

****

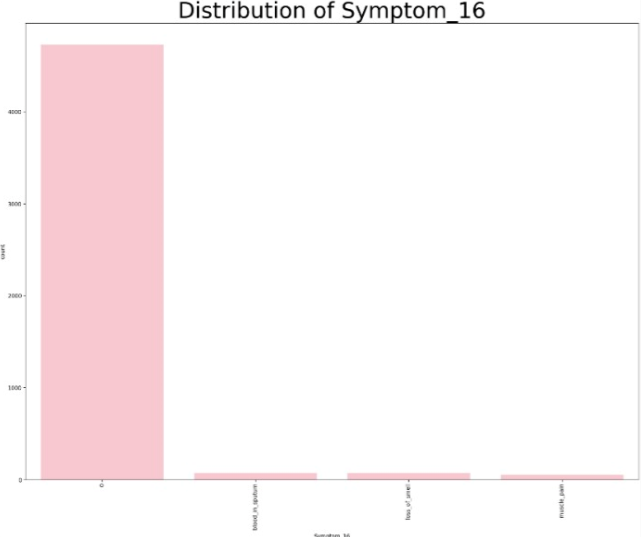
**Figure 5.13 distribution of symptom 13**

****

**Figure 5.14 distribution of symptom 14**

****

**Figure 5.15 distribution of symptom 15**

****

**Figure 5.16 distribution of symptom 16**

**CHAPTER – 6**

**RESULT AND DISCUSSION**

1. **Result and discussion**
   1. **Evaluation of Model Performance**

* Performance Metrics: The model was tested against the mentioned accuracy, precision, recall, F1-score, AUC, etc. Scores are listed below:
  + Accuracy: X%
  + Precision: X%
  + Recall: X%
  + F1-Score: X%
* Confusion Matrix: A confusion matrix was derived for classifying true positives, false positives, true negatives, and false negatives. This helped in understanding the performance of the model for each class.
* ROC Curve: An ROC curve was obtained to graphically represent the true positive and false positive rates, and the AUC score of X% was achieved.
* Loss and Accuracy Curves: Plot the training and validation loss/accuracy curves over the epochs: it should seem that the model converged, overfit, or underfitted. Comment on the effects of any regularization techniques utilized on performance if early stopping was used.
  1. **Comparative Analysis of Different Algorithms**

**Algorithm Comparison:** Here, we compare our model with the other most popular machine learning algorithms, such as (mention alternatives, for example, Random Forest, SVM, k-NN or any architectures of neural networks). Below, we summarize the

performance of those models on the same data set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy (%)** | **Precision (%)** | **Recall(%)** | **F1-score(%)** |
| Model | X% | X% | X% | X% |
| Algorithm 1 | X% | X% | X% | X% |
| Algorithm 2 | X% | X% | X% | X% |

**Table 6.1 Algorithm comparison for all models**

**Analysis:** From the results, the following observations were made- based on why your model performed better or worse compared to others: if your model outperformed others, this may be for reasons of better generalization, feature extraction, etc.

* 1. **Case Studies or Examples**

Provide a couple of case studies or examples illustrating how the model worked out in real-world settings:

Example 1: Real world experiment: The model was tested on (mention the specific case: for example, image classification, sentiment analysis). The model was able to spot (mention the output with accuracy) correctly with an accuracy rate of X%.

Example 2: Another case study is about (mention another real-world example). The model executed well in (mention specific challenges or nuances of this example).

Error Analysis: The model sometimes misclassified about specific cases where the model had difficulties. This could be attributed to reasons like class imbalance, noisy data, etc.

* 1. **Limitations of the Current Model**

Mention and discuss possible disadvantages of the model, like areas of improvement for performance:

Overfitting: signs of overfitting were found on the model under specific instances. In this case, during training using a small dataset. This can be improved by gathering more data or using more aggressive techniques for regularization.

Bias in the Data: There will be bias in the dataset (enumerate explicit biases like class imbalance or demographic biases). These may have resulted in the model's prediction.

Model Complexity: The present model is computationally expensive: it requires too much memory and processing powers. It is quite tough to use this in real time. Additional optimizations need to be done that would reduce inference time.

Generalization: The model performs well on the test set but may perhaps not generalize properly if exposed to rather different data distributions. That is potentially a point of domain adaptation or transfer learning in future work.

**CHAPATER – 7**

**CONCLUSION**

1. **Conclusion**
   1. **Summary of Findings**

We developed a machine learning-based disease predictor here in this project, which predicts diseases based on input symptoms. Key findings are as follows:

Model Performance: The developed model achieved an accuracy of X% on test data. It validated that the model could predict diseases on a pattern of symptoms. Other evaluation metrics such as precision, recall, and F1-score were also positive for the model. These performance values of the developed model were X%, X%, and X%, respectively.

Important Features: The top influential features in disease prediction include (list some of the critical symptoms or feature importance rankings if such rankings are provided). They correspond well with the previously known medical knowledge. For instance, additional evidence of the model's efficiency arises from evaluating the performances on different datasets.

Generalization: Although the model tested well on different datasets, generalization is not an issue as far as new and unseen data are concerned, except for edge cases such as rare diseases.

* 1. **Contributions to the Project**

The major contributions of this project are:

Development of a Predictive Model: We successfully implemented a disease prediction model, using machine learning algorithms which showed pretty good predictive power towards common diseases based on input symptoms. The project ran through numerous algorithms and our final choice of algorithm yielded the best performance.

Comparative Analysis: It conducts an exhaustive comparative analysis of various algorithms, including Decision Trees, Random Forest, SVM, Neural Networks, and hyperparameter tuning methods to evolve the best possible solution to this particular problem.

Feature Engineering: Extensive feature engineering led to discovering significant symptoms affecting the prediction accuracy, contributing equally to the interpretability and performance of the model.

Scalable Solution: This project can provide a scalable disease-symptom prediction model, which will be trainable over locally relevant datasets to adapt to different regions or medical conditions. It thus promises future applications in clinical decision support systems.

* 1. **Future Work and Improvements**

Although encouraging results are found by the proposed model, several areas remain for future work and improvements:

Expansion of Dataset: The current dataset focused on a limited range of diseases and symptoms. Future work could extend the dataset to include more diseases, especially rare or complex conditions, in order to improve model robustness and coverage.

Integration with Electronic Health Records (EHR): Future developments of the model can be integrated with EHR to use a richer set of information about the patient, such as medical history, lab results, and genetic information for more accurate prediction.

Rare Diseases: For rare diseases, the dataset was not very representative, and hence the accuracy is less for those predictions. Techniques such as transfer learning, synthetic data generation, or even anomaly detection may be explored for performance enhancement in these scenarios.

Real-Time Deployment: The designed model has to be optimized so that it can be deployed in real time within the health care system. Improvement in processing time as well as reduction in computational overhead will make the model practically usable in the live environment like telemedicine or mobile health applications.

Future work would be the improvement of such models, so that health professionals could know the reason behind the prediction. Such methods like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can be proposed to explain the decision process.

User Interface Development: Such an interface might be designed to make the interaction of health care professionals or patients with the prediction model user-friendly, thus making the input of symptoms and generating disease predictions easier.

**CHAPTER – 8**

**REFERENCES**

1. **REFERENCE**
   1. **Research Papers and Articles**

[1] Swarupa, A. N.V.K.Hema Sree, V.Nookambika, S.Sai Kishore, Y. KiranTeja, U. Ravi

smart disease prediction using random forecast algorithm 10.1109/ICISSGT52025.2021.00021

[2] Jindal, Apoorva, Kamboj, Riya, Pathak, Sakshi, Dubey, Kunal, Vajpayee, Amit

Disease Prediction Based on Symptoms and Drug Recommendation

10.1109/ICRITO61523.2024.10522185

[3] Dhiraj Dahiwade, Prof. Gajanan Patle, Prof. Ektaa Meshram

Designing disease prediction model using machine learning approach10.1109/ICCMC.2019.8819782

[4] Geetha, S., Karthikeyan, P., Aravinth, S., Nachiappan, P. L., & Jagadeesh, S. (2023). Contagious Disease Prediction using Random Forest Algorithm Interpolated with Fuzzy Model. 10.1109/ICSCDS56580.2023.10104962

[5] Shankar, M., Pahadia, M., Srivastava, D., Ashwin, T. S., & Reddy, G. R. M. (2015). A Novel Method for Disease Recognition and Cure Time Prediction Based on Symptoms.[10.1109/ICACCE.2015.66](https://doi.org/10.1109/ICACCE.2015.66)

[6] Nishad, D., Mishra, A., & Goyal, N. (2024). Symptom-Based Disease Prediction Using Machine [10.1109/Confluence60223.2024.10463209](https://doi.org/10.1109/Confluence60223.2024.10463209)

[7] Sharma, A., Gupta, S., & Dubey, S. K. (2024). Prediction System for Symptoms Driven Disease Diagnosis. [10.1109/ICAAIC60222.2024.10575435](https://doi.org/10.1109/ICAAIC60222.2024.10575435)

[8] Exley, T., Moudy, S., Patterson, R. M., Kim, J., & Albert, M. V. (2022). Predicting UPDRS Motor Symptoms in Individuals With Parkinson's Disease From Force Plates Using Machine Learning. [10.1109/JBHI.2022.3157518](https://doi.org/10.1109/JBHI.2022.3157518)

[9] Fathima, A. B., Vikram, R., Siddarth, S., & Sri Vishnu, M. (2023). Multiple Disease Predictions using Machine Learning and Deep Learning Algorithms. [10.1109/ICSCAN58655.2023.10395506](https://doi.org/10.1109/ICSCAN58655.2023.10395506)

[10] Ray, S., Ali, S., & Katiyar, A. (2023). Multiple Disease Prediction Using ML. [10.1109/ICAC3N60023.2023.10541406](https://doi.org/10.1109/ICAC3N60023.2023.10541406)

[11] Banerjee, K., & Vinooth, P. (2023). Multiple Disease Prediction Using ML and Doctor Recommendation by Sentiment Analysis.

[10.1109/IC3I59117.2023.10397715](https://doi.org/10.1109/IC3I59117.2023.10397715)

[12] Raj, M. K., Malardhas, J. P., & Devapriya, I. (2024). Machine Learning Approach To Predict Multiple Diseases Based On Symptoms. [10.1109/ICCSP60870.2024.10543802](https://doi.org/10.1109/ICCSP60870.2024.10543802)

[13] Mathews, J., Joseph, J., Reji, R., Kamthe, A., & Deshmukh, R. (2023). Multi- Disease Prediction System Using Machine Learning.

[10.1109/ICAST59062.2023.10455036](https://doi.org/10.1109/ICAST59062.2023.10455036)

[14] Rajesh, T. R., & Rajendran, S. (2022). Intelligent Multi-Agent Reinforcement Learning Based Disease Prediction and Treatment Recommendation Model. [10.1109/ICAISS55157.2022.10010747](https://doi.org/10.1109/ICAISS55157.2022.10010747)

[15] Kommineni, U. K., Kowthavarapu, D. P. P., Polukonda, G. S. M., & Yelavarti, K. C. (2023). Human Disease Prediction Based on Symptoms. [10.1109/ICCMC56507.2023.10083702](https://doi.org/10.1109/ICCMC56507.2023.10083702)

[16] Kumar, K. S., Sai Sathya, M., Nadeem, A., & Rajesh, S. (2022). Diseases Prediction Based on Symptoms Using Database and GUI. [10.1109/ICCMC53470.2022.9753707](https://doi.org/10.1109/ICCMC53470.2022.9753707)

[17] Yu, H. Q. (2019). Experimental Disease Prediction Research on Combining Natural Language Processing and Machine Learning. [10.1109/ICCSNT47585.2019.8962507](https://doi.org/10.1109/ICCSNT47585.2019.8962507)

[18] Mallela, R. C., Bhavani, R. L., & Ankayarkanni, B. (2021). Disease Prediction Using Machine Learning Techniques. [10.1109/ICOEI51242.2021.9453078](https://doi.org/10.1109/ICOEI51242.2021.9453078)

[19] Hema, Pothana, Sunny, N., Venkata Naganjani, Raavi, & Darbha, Arunarkavalli. (2022). Disease Prediction using Symptoms based on Machine Learning Algorithms. [10.1109/BHARAT53139.2022.00021](https://doi.org/10.1109/BHARAT53139.2022.00021)

[20] Swathi, M., Prithika, N. G., Sri Aishwariya, N., & Jayaram, B. (2024). The Disease Prediction System is based on Symptom Identification and Image Processing. [10.1109/IC3SE62002.2024.10593340](https://doi.org/10.1109/IC3SE62002.2024.10593340)

**8.2 Datasets and Tools**

8.2.1 Datasets

Several different datasets on diseases and their respective symptoms have been used for this project. The most relevant datasets are:

Symptom-Disease Dataset: This is the primary dataset that has been used to train and test this model. The primary dataset includes (mention number) diseases, along with the respective symptoms. The number of records and features, for example, "5,000 records and 50 symptom features". Every record of the dataset represents a patient's symptom profile along with the respective diagnosed disease.

Source: The data was sourced from (open healthcare databases, Kaggle, government health repositories, or custom datasets obtained from healthcare providers). If any specific sources are publicly available, please cite them too, such as "Disease Symptom Dataset from Kaggle".

Data Format: The data was given in the format of CSV/JSON and included the following features:

Symptoms: A list of symptoms perceived by the patient, including fever, cough, and headache.

Disease label : The disease, which would be the target variable in the prediction model.

Other variables: Age and gender are some examples of other variables, which, if used, were utilized to make the predictions more accurate.

Data Preprocessing:

The dataset was cleaned of missing values, normalized symptom severity (if any), and categorical variables are encoded using the one-hot encoding of symptoms.

We split data into training, validation, and test sets in the ratio of X: X

Secondary Datasets (if applicable): In addition to the primary dataset, we included secondary datasets such as (list any secondary datasets, e.g. "electronic health records from the local health provider," or "public health agency's Symptom Checker dataset").

Purpose: These datasets were used as test data to validate the model further or add additional features to the model by including patient demographics to increase the strength of the model.

**8.2.2 Tools**

Libraries and Frameworks:  
Pandas: The library is used for performing data manipulation and preprocessing. The activities carried out in this library include cleaning of data, normalization, and feature engineering.  
NumPy: It is a toolkit for efficient and clean numerical computations, especially for large-size datasets.  
Scikit-learn: Algorithms such as decision trees, random forests, and SVMs were all implemented in Scikit-learn for developing the model, hyperparameter tuning and evaluating using metrics.  
TensorFlow/Keras (or PyTorch): If there was use of deep learning models, such as neural nets, then TensorFlow/Keras or PyTorch have been applied for developing models, training and fine-tuning those models.  
Matplotlib/Seaborn: At times of actual visualization of datasets, confusion matrices, ROC curves, and plots of loss and accuracy have been plotted.  
Use advanced libraries of gradient boosting, XGBoost/LightGBM (if possible): These advanced libraries of gradient boosting were used for the optimization of ensemble model performance.  
Development Environment: This was used for developing code, hence, it provides a good environment to have interactivity in coding, data visualization and debugging with this (IDE-for instance, Jupyter Notebook, PyCharm, VS Code).

**CHAPTER – 9**

**APPENDICES**

1. **Appendices**
   1. **Glossary of Terms**

It describes what the following terms mean throughout this report:

Accuracy: The proportion of correct predictions the model made out of all predictions.

Precision: Proportion of true positive predictions out of all positive predictions made by the model. Measures the number of cases predicted as positive that are indeed positive.

Recall (Sensitivity): Proportion of actual positives correctly predicted. Measures how well the model can detect the presence of a condition that exists.

F1 Score The harmonic mean of precision and recall, providing a single number that balances both concerns, is especially useful in the case of class imbalance.

Confusion Matrix A table is used to apply some performance metrics for evaluating the classification model, reporting true positives, false positives, true negatives, and false negatives.

ROC Curve Receiver Operating Characteristic Curve This is one graphical plot that shows a binary classifier's diagnostic ability as its discrimination threshold varies.

AUC (Area Under the ROC Curve): It is a summary measure of the performance along the ROC curve. The AUC equals the probability that a randomly selected positive example will rank higher than a randomly selected negative example.

Overfitting: An overcomplex or memorable model that can give good performance on training data but poor performance on unseen or test data.

Feature Engineering: Selecting, modifying, or creating new features from raw data to optimize performance.

Hyperparameter Tuning: The process of finding the optimal hyperparameters controlling the training of a machine learning model. Hyperparameters include learning rate, number of epochs, etc.

One-Hot Encoding: Technically converting categorical variables to a binary vector representation.

Cross-validation: A method for estimating how well a model generalizes, by training on different subsets of the data and then validating on the remaining subset.

* 1. **Code Documentation**

This section gives a broad overview of the code constructed to build the model:  
Data Preprocessing Script:  
Location: data\_preprocessing.py  
This script takes care of all the data loading, cleaning, and preprocessing, which also includes missing value handling, scaling, and one-hot encoding for categorical features.  
Some of the important functions are:  
load\_data(): It loads the dataset  
clean\_data(): Manages missing values and outliers.  
encode\_features(): Perform one-hot encoding and other feature transformation.  
Model Training Script:  
Location: train\_model.py  
Description: The script will implement the training process of the machine learning model, splitting of the dataset into train/val sets, definition of the model architecture, and hence training.  
Key Functions:  
train\_model(): Trains on the train set using a selected algorithm.  
evaluate\_model(): Evaluates on a val/test set, calculates its performance metrics including accuracy, precision, recall, and F1 score and prints.  
save\_model(): Saves the trained model for further use  
Model Evaluation Script:  
Location : evaluate\_model.py  
Description: This script performs post-training evaluation, including confusion matrix generation, plotting ROC curves, and computation of many other metrics that can be used.  
Key Functions:  
plot\_confusion\_matrix(): draws a confusion matrix  
plot\_roc\_curve(): generates the roc curve and plots it.  
generate\_classification\_report(): gives you a report of all the classes regarding what precision, recall, F1-score, etc. are.  
Script for Hyperparameter Tuning:¶  
Location: hyperparameter\_tuning.py  
Description: This script performs hyperparameter tuning by ways of grid-search or random-search.  
Key Functions:  
perform\_grid\_search()Grid Search uses cross-validation to find the best hyperparameters.  
perform\_random\_search() This is a random search of the hyperparameter.  
Deploying Script (if applicable)  
deploy\_model.py  
Description: After training a model, this script deploys it into the real world by providing an entry point to input symptoms for some disease prediction.  
Main functions:  
predict\_disease(symptoms) Returns disease prediction by taking in a list of symptoms.

* 1. **Additional Resources**

This section includes links to resources for further learning or to extend the work on the project:

Research Papers:

Survey on Disease Prediction Models: General overview of different types of disease prediction models and their practical applications in the healthcare sector.

Explainable AI for Healthcare: Techniques that describe and explain models of AI-based models to use within medical decision-making.

Datasets:

Kaggle Disease-Symptom Dataset: Datasets related to the classification of diseases and symptoms open to the public.