DISEASE DETECTOR

A Project Report

Submitted By:

DIVIJ SINGH (201B100)

NIKITA MISHRA (201B169)

SAMRADDHI TRIPATHI (201B236)

Under the guidance of
Dr. Rahul Pachauri

in partial fulfillment of the degree

of

Bachelor of Technology

IN
COMPUTER SCIENCE & ENGINEERING
at

JAYPEE UNIVERSITY OF ENGINEERING AND TECHNOLOGY, GUNA, MADHYA PRADESH (INDIA) – 473226

JAN 2023 – MAY 2023

DECLARATION

We hereby declare that the work reported in the 6th-semester Minor project entitled "DISEASE DETECTOR" in partial fulfillment for the award of the degree of B.Tech (CSE) submitted at Jaypee University of Engineering and Technology, Guna, as per the best of our knowledge and belief there is no infringement of intellectual property rights and copyright. In case of any violation, we will solely be responsible.

Divij Singh(201B100)

Nikita Mishra(201b169)

Samraddhi Tripathi(201b247)

Place: Jaypee University of Engineering and Technology, Guna - 473226

Date: 16-05-2023

CERTIFICATE

This is to certify that the project titled “DISEASE DETECTOR” is the bonafide work carried out by Nikita Mishra, Samraddhi Tripathi and Divij Singh students of BTech (CSE) of Jaypee University of Engineering and Technology, Guna (M.P) during the academic year 2022-23, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar tile.

Signature of the Guide

Place: Jaypee University of Engineering and Technology, Raghogarh, Guna – 473226

Date: 16-05-2023

ABSTRACT

The Disease Prediction System based on various prediction models that help to predict the disease of the user on the basis of the symptoms that user enters as an input to the system. Predictive models with the help of machine learning classification algorithms analyze the symptoms provided by the user as input and give the name and probability of the disease as an output. Disease Prediction is done by implementing the Naive Bayes Classifier, Decision tree and Random Forest Algorithm. The Naive Bayes helps to calculate the probability of the disease which is predicted. Average prediction accuracy probability 87% is obtained. The model uses a dataset with the count of 132 symptoms from which the user can select their symptoms. The user does not need to have a medical report to use this system as the prediction is based on the symptoms which will save the money. The system also has a very easy to use user interface so all the users can use it to predict genetic diseases .

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave us the opportunity to complete this project. Special thanks to our supervisor Dr. Rahul Pachauri whose help, stimulating suggestions and encouragement helped us in all the time of development process and in writing this report. We also sincerely thank you for the time spent proofreading and correcting my many mistakes. We would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited period. Last but not the least I am grateful to all the team members of DISEASE DETECTOR.

Thanking you,

Divij Singh(201b100)

Nikita Mishra(201B169)

Samraddhi Tripathi(201b247)

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem Description

Today the healthcare industry has become a big money making business. The healthcare industry uses and produces quite a large amount of data which can be used to extract information about a particular disease for a patient. This information of healthcare will further be used for effective and best possible treatment for patient's health. This area also needs some improvement by using the informative data in healthcare sciences. But a major challenge is to extract the information from the data because the data is present in a huge amount so some data mining and machine learning techniques are used.

The expected result of this project is to predict the disease beforehand so that the risk of life can be prevented at an early stage and save life of people and the cost of treatment can be reduced to a particular extent. In India also we should adopt the non-manual system of medical treatment which is the best for improving and understanding human health. The main motive is to use the concept of machine learning in healthcare to improvise the treatment of patients.

Machine learning has already made it much easier to identify and predict various diseases[7]. Predictive analysis of the disease with the help of many machine learning algorithms helps us to predict the disease and helps in treating the patients in an effective manner. Disease prediction using machine learning also uses the patient history and health data by applying vari- ous concepts like data mining and machine learning techniques and also some algorithms. Many works have also applied data mining techniques to the pathological data for prediction of some particular diseases. These approaches were intended to before- hand predict the re-occurrence of certain diseases[15]. Also, some approaches tried to do prediction while controlling the disease. The recent work of deep learning was in disparate areas of machine learning which have driven a shift to machine learning models that can learn and understand the hierarchical representations of raw data with some pre-processing. With the development of this concept called big data technology, more attention is paid to disease prediction.

1.2 Problem Overview

Disease detector apps have the potential to decrease several problems related to disease detection, including:

- Early detection: Disease detector apps can help individuals detect diseases earlier than they would have otherwise. Early detection is crucial in many diseases, as it can lead to earlier treatment and improved outcomes.

- Accessibility: Disease detector apps can increase accessibility to disease detection by allowing individuals to perform self-assessments from the comfort of their own homes, without the need to visit a healthcare provider.

- Cost-effectiveness: Disease detector apps can be cost-effective, as they may eliminate the need for expensive diagnostic tests or consultations with healthcare providers.

- Health awareness: Disease detector apps can increase awareness about specific diseases and their symptoms, helping individuals to recognize potential warning signs and seek medical attention if necessary.

1.3 Hardware Specifications

Hardware specifications for disease detection using machine learning (ML) will depend on several factors, including the complexity of the ML algorithm, the size of the dataset used for training, and the expected performance of the system.

- Processor: A high-performance processor is essential for running ML algorithms efficiently. Depending on the complexity of the algorithm, a processor with multiple cores and a clock speed of 3 GHz or higher may be required.

- Graphics processing unit (GPU): GPUs are commonly used in ML applications to accelerate processing and improve performance. A high-end GPU with dedicated memory can significantly improve the speed of training and inference.

- Memory: Sufficient memory is required to handle the large datasets used for training ML models. At least 8 GB of RAM is recommended, but more may be necessary for larger datasets.

- Storage: ML models can require significant storage space, so a fast and reliable storage system is necessary. A solid-state drive (SSD) with at least 500 GB of storage is recommended.

- Network connectivity: ML models can be computationally intensive, so it may be necessary to use cloud-based services for training and inference. A high-speed internet connection is necessary for uploading and downloading data to and from the cloud.

- Operating system: Depending on the ML framework used, the system may require a specific operating system. Linux-based systems are commonly used in ML applications due to their stability, flexibility, and performance.

- Peripheral devices: Input devices such as a keyboard and mouse, as well as output devices such as a monitor, are necessary for interacting with the system

1.4 Software Specification

Software specifications for a disease detector using machine learning (ML) will depend on the ML algorithm used, the data preprocessing steps, the platform or framework chosen for development, and the expected performance of the system. Here are some general software specifications that can be considered for a disease detector using ML:

- Programming languages: The choice of programming language will depend on the ML framework used. Python is a popular choice for ML applications due to its simplicity, readability, and support for a wide range of ML libraries.

- ML framework: There are several popular ML frameworks available, including TensorFlow, PyTorch, and scikit-learn. The choice of framework will depend on the

complexity of the ML algorithm and the size of the dataset used for training.

- Preprocessing tools: Preprocessing tools are used to clean and prepare data for use in the ML algorithm. Common preprocessing tools include NumPy, Pandas, and OpenCV.

- Data visualization tools: Data visualization tools are useful for exploring and visualizing data. Popular data visualization tools include Matplotlib, Seaborn, and Plotly.

- Cloud services: Cloud services such as Amazon Web Services (AWS) and Google Cloud Platform (GCP) can be used for scalable ML development and deployment.

- IDE: An integrated development environment (IDE) is necessary for developing and testing the code. Popular IDEs for Python include PyCharm, Spyder, and Jupyter Notebook.

- Version control: Version control systems such as Git and SVN are useful for managing changes to the code and collaborating with team members.

- Deployment platform: The choice of deployment platform will depend on the requirements of the system. Common deployment platforms include cloud-based services such as AWS and GCP, as well as on-premises server.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing System

There are several existing systems for disease detection using machine learning (ML) that have been developed for various diseases. Here are some examples:

- Diabetic Retinopathy Detection: This system uses ML algorithms to detect and classify diabetic retinopathy, a complication of diabetes that affects the eyes. The system analyzes retinal images to identify signs of the disease, such as hemorrhages and exudates.

- Malaria Detection: This system uses ML algorithms to detect malaria parasites in blood smear images. The system can help healthcare providers diagnose and treat malaria more efficiently, improving patient outcomes.

- Parkinson's Disease Detection: This system uses ML algorithms to analyze speech recordings and detect early signs of Parkinson's disease. The system can provide early intervention and treatment for patients with the disease, improving their quality of life.

- COVID-19 Detection: This system uses ML algorithms to analyze chest X-rays and detect signs of COVID-19 pneumonia. The system can provide a rapid and accurate diagnosis, allowing for prompt treatment and isolation to prevent the spread of the diseases.

2.2 Proposed System

There are following steps involved in our proposed methodology:

1. First we will collect the datasets of symptoms and their functional problems in the body.

2. Then the cleaning, training and testing of the given data will be performed using Confusion Matrix and Label Encoding(for converting the data into a numerical type.)
3. Then we will collect the information that will associate the symptoms to possible diseases thus related disease information will be collected. (Kaggle Dataset)
4. Then we will get the symptoms as input from the patient and process it by SVM, Naive Bayes and Random Forest Classifier.
5. After that these machine learning algorithms will predict the diseases that may be possible for those acquired symptoms based on the symptom entered by the user.
6. Then the system will show the diagnosis in the form of the most likely possible disease through the web application we built using Anvil.

2.3 Feasibility Study

2.3.1 Introduction

A feasibility study is an analysis of the potential of a project to succeed based on its technical, economic, and operational aspects. Here is a brief feasibility study for a disease detector using machine learning:

2.3.2 Technical Feasibility

Disease detection using machine learning is technically feasible as it has been successfully implemented for various diseases such as diabetic retinopathy, skin cancer, and Parkinson's disease. ML algorithms are able to analyze large amounts of data and identify patterns that may not be visible to human experts. The technical requirements for such a system would include a suitable hardware and software infrastructure for data collection, preprocessing, model training, and deployment.

2.3.3 Operational Feasibility

The operational feasibility of a disease detector using ML would depend on factors such as the availability of data, expertise, and infrastructure required to develop and deploy the system. Additionally, the system would need to be integrated into existing healthcare workflows and comply with regulatory requirements to ensure its effectiveness and safety.

2.3.4 Economic Feasibility

The economic feasibility of a disease detector using ML would depend on various factors

such as the cost of data collection, hardware and software infrastructure, and expertise required to develop and maintain the system. Additionally, the system could potentially provide cost savings by reducing the need for manual diagnosis and enabling early detection of diseases, which can improve patient outcomes and reduce healthcare costs.

2.4 Identification of Research Gap and Problems

- The model given by [1] uses KNN and CNN algorithms which is more time consuming as it involves both the structured and the unstructured data so the time taken to process the data is more as compared to the dataset which contains only the structured data as in the proposed project which contains only the structured data and the classification algorithms used in the proposed project are decision tree, Naive Bayes and Random forest.The accuracy of the model given by [1] is above 90% which is not good for a ML model as it is said to be in an over fitting situation whereas the proposed model has accuracy of about 86% which is good enough for a model of disease prediction.

- The model given by [2] has a very limited scope as it is only meant for the prediction of diabetes and hypertension whereas the proposed model is used for the prediction of the basic general disease. The model given by [2] needs the blood report of the patient or the user for the prediction of the diabetes or the hypertension and the algorithms used in this model are ensemble learning techniques whereas the predicted model does not need any blood report or physical presence of the user or the patient. The system contains a list of symptoms from which the user can select the symptoms which the user is facing and can predict the disease very easily and the algorithms used are different from the given model. The input required in the given model are based on the medical report of the user like cholesterol, blood glucose etc whereas the proposed system does not require any type of blood report for the prediction of the disease.

- The model given by [4] is best for the prediction of disease related to breast cancer, diabetes and heart related problems and has different dataset for all the three different kind of disease which is different from the proposed model as the proposed model

helps in the prediction of the general diseases with the help of the symptoms and has a single dataset for all the diseases.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

3.1 Algorithms Used

In this project different ML algorithms are used and several techniques of data mining are also used to check the dataset that whether it is a balanced dataset or not and check for the data is structured or not for the disease prediction.
The various ML algorithms used in this project are:

3.1.1 Support Vector Classifier

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

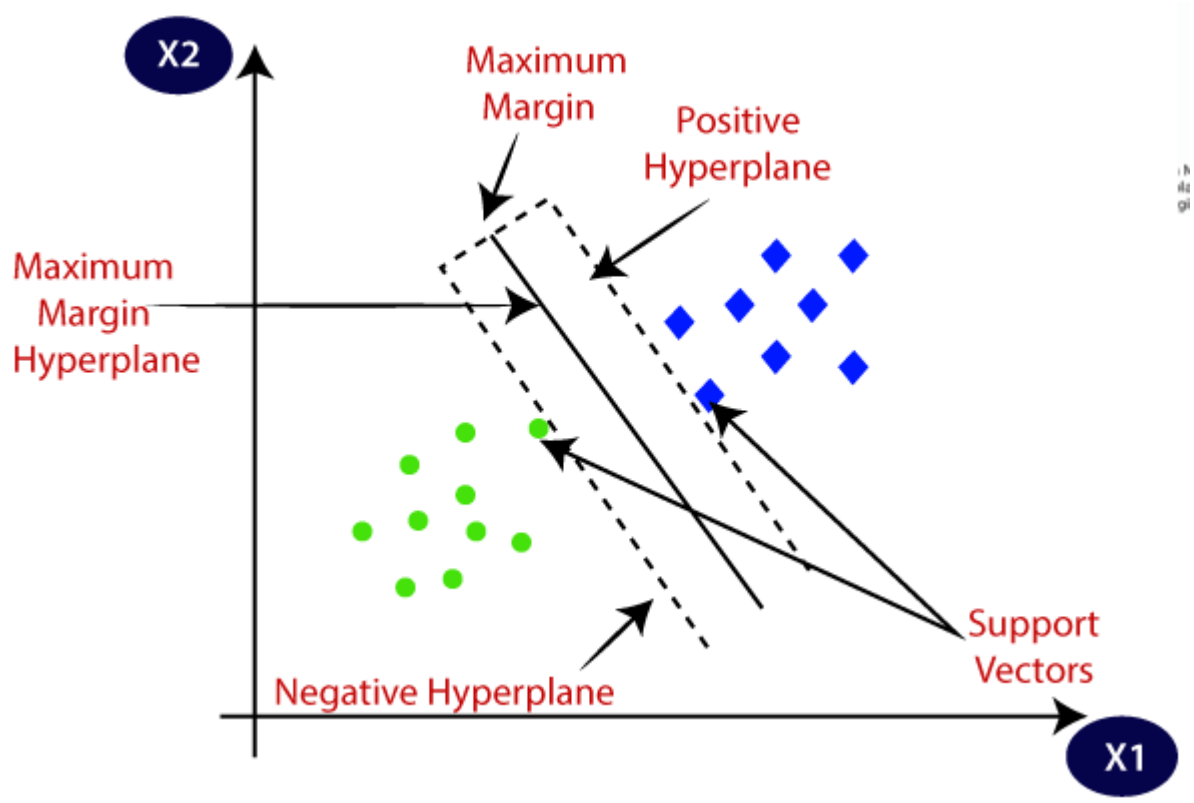


Figure 3.1 : SVM Overview

SVM algorithm can be used for Face detection, image classification, text categorization, etc. SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

3.1.2 Naive Bayes Classifier

The Naive Bayes is used as a classification technique in the ML which is used to classify the things and give the answer on the basis of the classification.The main working of the Naive Bayes Classifier is based on the bayes theorem of the statistics and the bayes theorem states that :

$$P(X|Y) = P(Y|X)P(X)P(Y)$$

By making the use of bayes theorem we can find the probability of X to occur when the probability of Y occurrence is given to us. For example:

S.No.	Temperature	Weather	Rain	Humid	Play Tennis
1	Hot	Sunny	No	High	Yes
2	Mild	Overcast	Yes	Low	Yes
3	Mild	Rainy	Yes	High	NO
4	Cold	Rainy	Yes	High	NO
5	Hot	Sunny	No	Low	Yes

Table 3.1: Prediction Using Naive Bayes

This is the kind of input which is given to the Naive Bayes Classifier for the prediction. On This kind of table input the bayes theorem is used the ML algorithm for the prediction, with this table the prediction can be done as probably of playing tennis when the temperature is mild, weather is rainy, rain is yes, humid is high like this the prediction is done in theML model also.

There are three different types of naive bayes classifier:–

1. Multinomial Naive Bayes: This is mainly used for the classification of big files and documents by dividing them into different categories as weather a file or document is of sports category, politics, technology or something else. This method of classification is very widely used in Machine Learning algorithms.
2. Bernoulli Naive Bayes: This method of classification is very similar to the multinomial naive bayes but the result of this type of naive bayes is only in either yes or no.
3. Gaussian Naive Bayes: This type of naive bayes is used for the prediction of the

continuous values while the other two types were used for the discrete value prediction.In this way the Naive Bayes Classifier is used in different ways in the ML models for the process of prediction and analysis

3.1.3 Decision Tree Algorithm

The decision tree algorithm is a type of a supervised learning algorithm which can be used in the case of classification as well as regression; it has capability to solve both kinds of problems. In the decision tree for predicting the value we start from the root of the tree and then form a sub-tree from that root and finally come to a conclusion which is nothing but the predicted value.

A pictorial representation is illustrated in the following figure :-

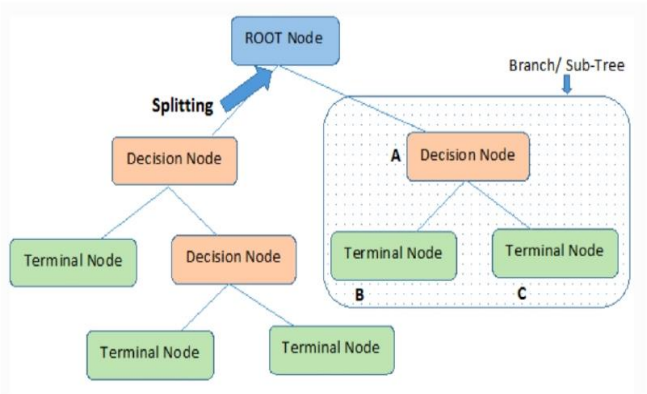


Figure 3.2: Decision Tree Overview

Decision tree helps in the method of classification as it sorts the values while traversing down the tree and predicts the right value.The decision tree starts from the root node on each step of moving downward in the tree the Information Gain and entropy are calcu-lated, after that the branch with lowest entropy or the highest information gain is selected and the information gain and the entropy are calculated again.

Entropy =cXi=1−pilog2pi

Information Gain =entropy(parent)−[average entropy(children)]

When the entropy is high then it is concluded that the randomness of the dataset is high and then it is hard to predict the answer whereas the information gain tells us about how well

structured the dataset is and if the information gain is high then it is easy to predict the answer. All these steps are repeated until the decision tree reaches the leafnode i.e. the final predicted value by the decision tree.

3.1.4 Random Forest Algorithm

The random forest algorithm is a type of supervised learn-ing algorithm which is used for both classification as well as regression as the basic idea behind the random forest algorithm is the decision tree algorithm, this algorithm creates multiple decision trees and then predicts the values.

multiple decision trees in a random forest algorithm.

A pictorial representation is illustrated in the following figure :-

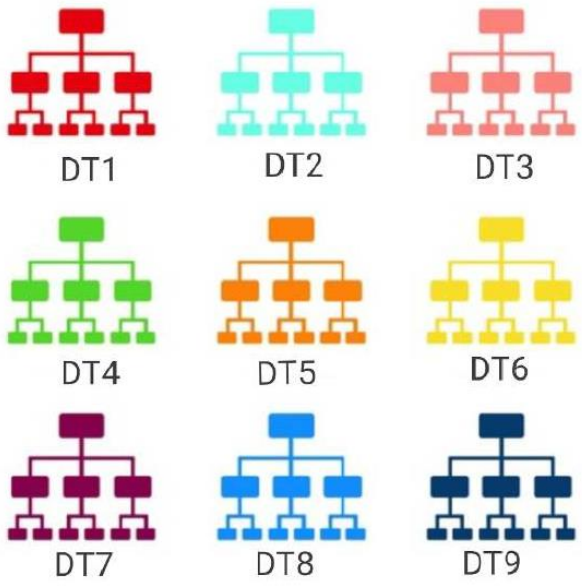


Figure 3.3: Random Forest Overview

Many of these decision trees will not be performing good enough and the best performing decision trees are selected for prediction.

3.2 Database

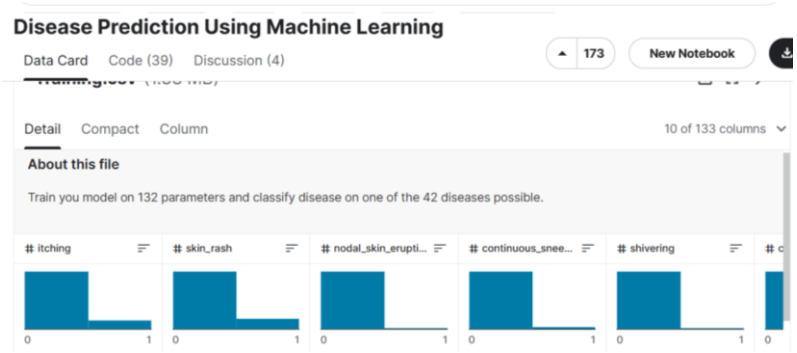
We have used a dataset from “Kaggle” for this project. This dataset has 132 parameters on which 42 different types of diseases can be predicted. This dataset is a clean dataset with no null values and all the features consist of 0’s and 1s. This dataset is a balanced dataset i.e. there are exactly 120 samples for each disease, and no further balancing is required.

We have two csv files in this dataset namely:

- Training
- Testing

We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models.

We can notice that our target column i.e. prognosis column is of object datatype, this format is not suitable to train a machine learning model. So, we will be using a label encoder to convert the prognosis column to the numerical datatype. A Label Encoder converts the labels into numerical form by assigning a unique index to the labels. If the total number of labels is n, then the numbers assigned to each label will be between 0 to n-1. A visualization of this dataset is shown in the following figure :-



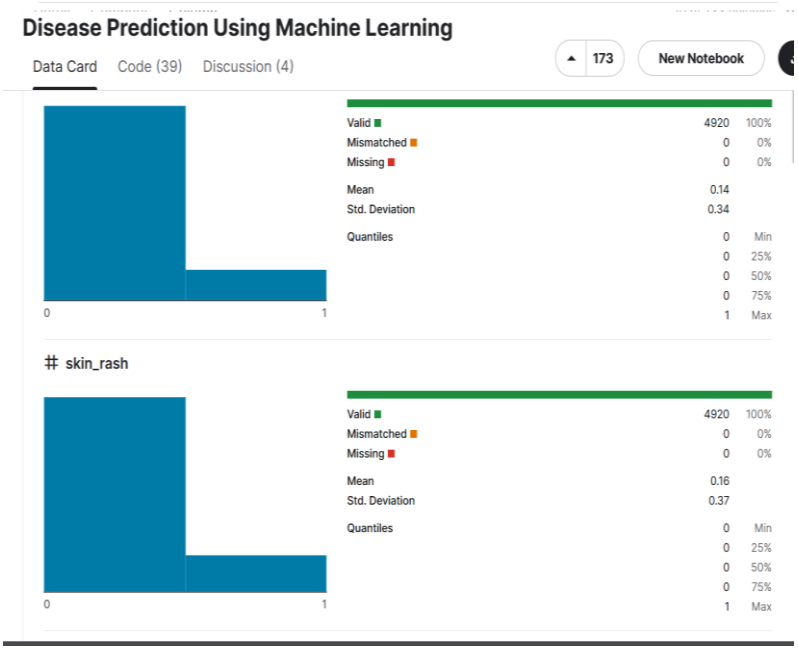


Figure 3.4: Representation of dataset required

3.3 Machine Learning Model

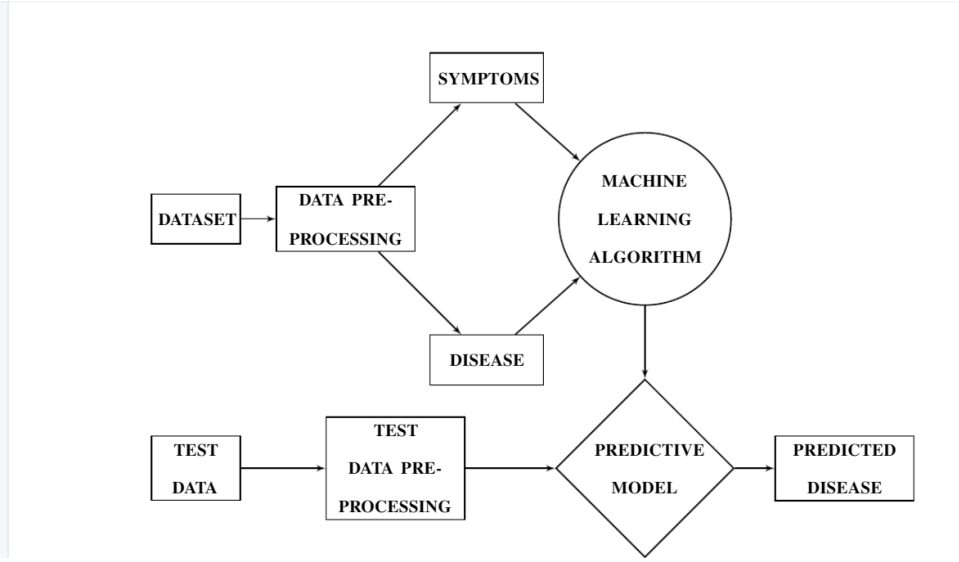


Figure 3.5 : Detailed Design of the Model

1. Confusion Matrix: In machine Learning, Classification is the process of categorizing a given set of data into different categories. In Machine Learning, To measure the

performance of the classification model we use the confusion matrix. A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

2. K-Fold Cross-Validation: K-Fold cross-validation is one of the cross-validation techniques in which the whole dataset is split into k number of subsets, also known as folds, then training of the model is performed on the k-1 subsets and the remaining one subset is used to evaluate the model performance.

3. SciPy: SciPy is a scientific computation library that uses NumPy underneath.SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely. SciPy was created by NumPy's creator Travis Olliphant.

4. Sklearn: This stands for Scikit learn and is built on the Scipy package. It is the primary package being used in this project. It is used for providing interfaces for supervised and unsupervised learning algorithms. Following groups of models are provided by sklearnClustering, Cross Validation, Datasets, Dimensionality Reduction, Ensemble methods,Feature extraction, Feature selection, Parameter Tuning, Manifold Learning, SupervisedModels.

5. Numpy : It is a library for the Python programming language, adding support for multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It provides functions for Array Objects, Routines,Constants, Universal Functions, Packaging etc. In this project it is used for performing multi-dimensional array operations.

6. Pandas : This library is used to provide high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It provides functionalities liketable manipulations, creating plots, calculating summary statistics, reshape tables, combine data from tables, handle times series data, manipulate textual data etc. In this project it is used for reading csv files, comparing null and alternate hypothesis etc.

7. Matplotlib : It is a library for creating static, animated, and interactive visualizations inPython programming language. In this project it is used for creating simple plots, sub-plots and its object is used alongside with the seaborn object to employ certain functions such as show, grid etc. A %matplotlib inline function is also used for providing more concise plots right below the cells that create that plot.

8. Seaborn : It provides an interface for making graphs that are more attractive and interactive in nature. It is based on the matplotlib module. These graphs can be dynamic and are much more informative and easier to interpret. It provides different presentation formats for data such as Relational, Categorical, Distribution, Regression, Multiples and style and color of all these types. In this project they are used for creating complex plots that use various attributes.

9. Warning : It is used for handling any warnings that may arise when the program is running. It is a subclass of Exception

All these libraries are used to create a model with the help of the dataset. The model created by applying all these data mining techniques is a binary file so that the model is secure from any kind of modifications and other security threats related to the system.

3.4 Requirement Specification

3.4.1 HTML

HTML (Hypertext Markup Language) is a markup language used for creating web pages and applications. It is typically used in the front-end development of web-based applications, including disease detectors using machine learning.
In a disease detector using ML, HTML can be used to create the user interface for the application. This may include designing the layout of the web page, creating forms for inputting data, and displaying the results of the ML algorithm.

In addition to forms, HTML can also be used to create tables, graphs, and other visualizations to display the results of the ML algorithm. Overall, HTML is an important component of the front-end development of disease detectors using machine learning, as it enables the creation of user-friendly and interactive web-based applications.

3.4.2 CSS

CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in HTML or XML. It is used to control the visual layout of web pages and web applications.
In a disease detector using ML, CSS can be used to style the HTML code and create an appealing and user-friendly interface. CSS can be used to define the font styles, colors, background, borders, and positioning of HTML elements.
CSS is an important component of the front-end development of disease detectors using machine learning, as it enables the creation of a visually appealing and user-friendly interface that can enhance the user experience.

3.4.3 Python

Python can be used in disease detection in a variety of ways, depending on the specific application and type of disease. Here are a few examples:

Data Analysis: Python has powerful data analysis libraries such as Pandas and NumPy that can be used to process large datasets of medical records, lab results, and other health-related data. This data can be analyzed to identify patterns or trends that may indicate the presence of a disease.

Machine Learning: Python has several machine learning libraries such as scikit-learn and TensorFlow that can be used to develop disease detection models. These models can be trained on large datasets of medical images, genetic data, or other health-related information to identify patterns or features that are indicative of a particular disease.

Image Processing: Python has several image processing libraries such as OpenCV and Pillow that can be used to analyze medical images such as X-rays, CT scans, and MRI images. These libraries can be used to extract features from images that can be used to diagnose diseases such as cancer or tuberculosis.

Natural Language Processing: Python has several natural language processing libraries such as NLTK and spaCy that can be used to analyze medical text such as doctor's notes or patient records. These libraries can be used to extract information about symptoms, medical history, and other factors that may indicate the presence of a disease.

Overall, Python's versatility and powerful libraries make it a popular choice for developing disease detection systems.

3.4.4 Google Colab

Google Colab is a cloud-based platform provided by Google that allows users to write, run, and share code in a collaborative environment. It offers a Jupyter Notebook interface and provides access to powerful hardware resources, including GPUs, which make it ideal for machine learning and data analysis tasks. Users can execute Python code, create visualizations, and manipulate data using popular libraries. Colab integrates seamlessly with Google Drive, allowing for easy access to files and collaboration with others. With its free access and pre-installed libraries, Colab has become a popular choice for students, researchers, and developers to experiment, learn, and collaborate on coding projects.

3.4.5 Anvil

Anvil is a web development platform that allows developers to build web applications using Python. Anvil provides a drag-and-drop interface for building user interfaces, and it includes a built-in Python backend for handling server-side logic and database operations.

Anvil can be used in disease detection in a few different ways. For example:

- Data Collection: Anvil can be used to build web-based data collection forms that allow doctors or researchers to collect data from patients or study participants. This data can be used to identify patterns or risk factors that may be associated with a particular disease.

- Machine Learning: Anvil can be used to build web-based machine learning applications that allow users to interact with disease detection models in real-time. For example, a doctor could input patient data into an Anvil app, which could then use a machine learning model to provide a diagnosis or recommend treatment options.

- Telemedicine: Anvil can be used to build web-based telemedicine applications that allow doctors to remotely diagnose and treat patients. For example, an Anvil app could allow a doctor to conduct a virtual exam of a patient using a video call, and the app could use machine learning models to help diagnose the patient's condition.

Overall, Anvil's web development capabilities and built-in Python backend make it a useful tool for building web-based disease detection applications.

3.5 FLOW CHARTS

3.5.1 Flow of events in the proposed model

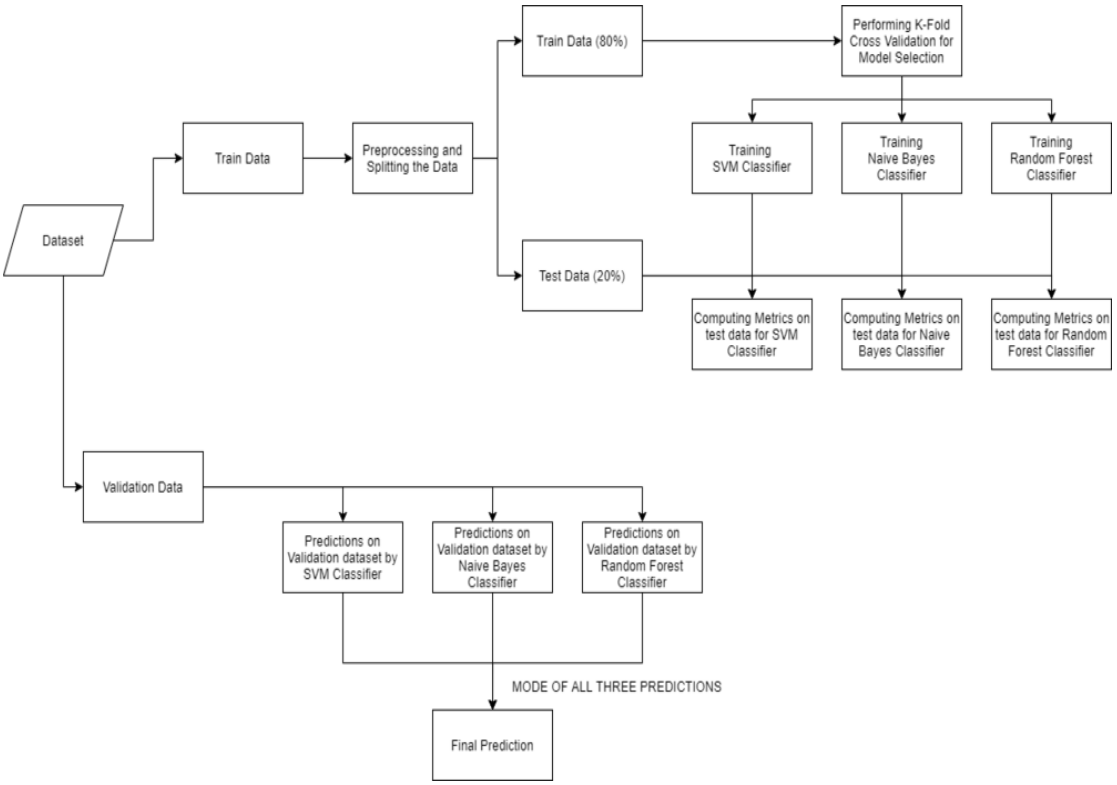


Fig 3.6 Flowchart of the proposed model

3.5.2 Class Diagram

A class diagram for a disease detector can help to identify the main classes involved in the system, their attributes and methods, and the relationships between them. It can be a useful tool for designing and understanding the structure of the disease detector system.

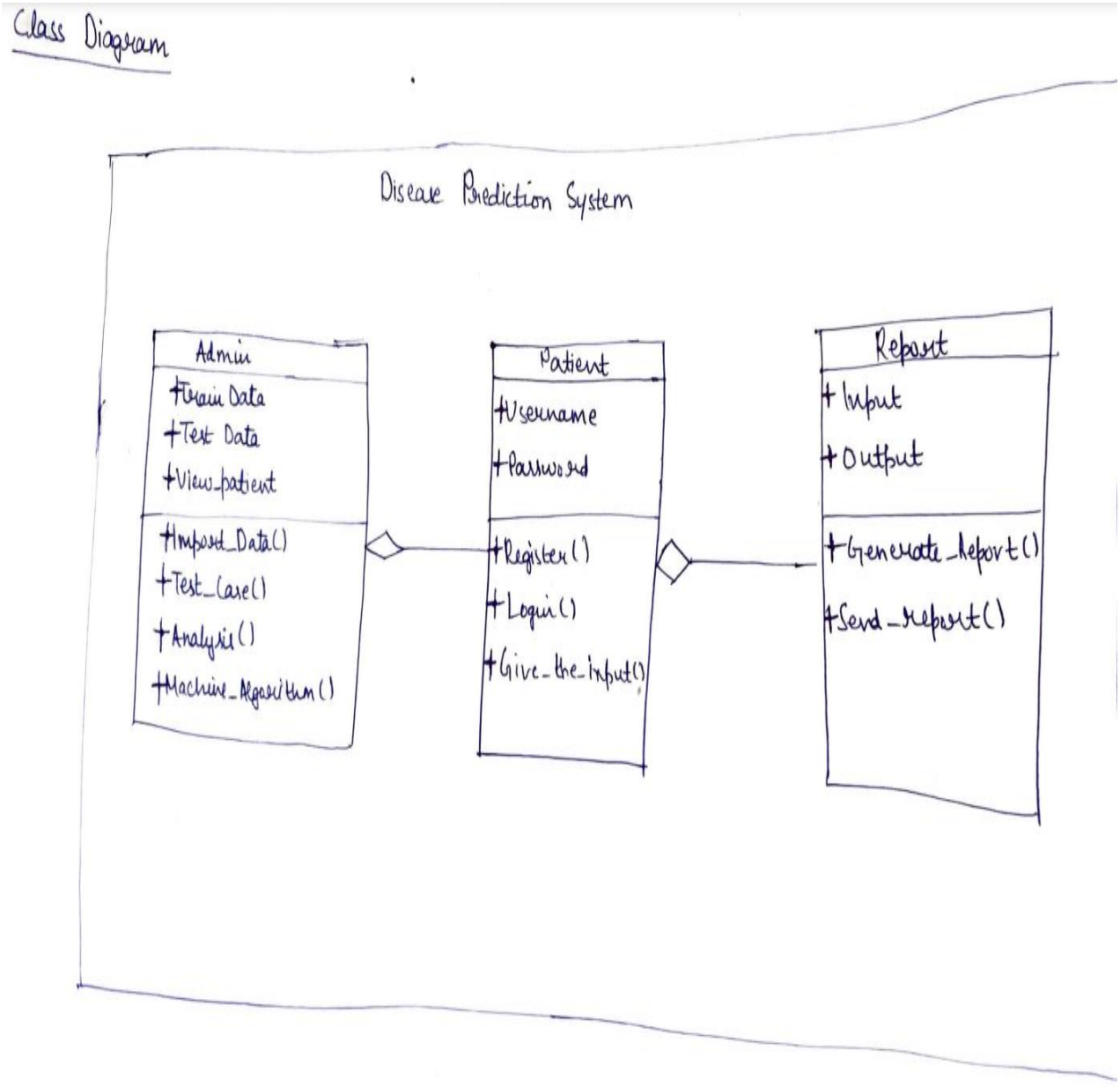


Fig 3.7 Class Diagram

3.5.3 State Diagram

A state diagram for disease detection using machine learning can help to identify the states that the system can be in, the events that cause it to change state, and the actions that occur as a result. It can be a useful tool for designing and understanding the behavior of the disease detector system.

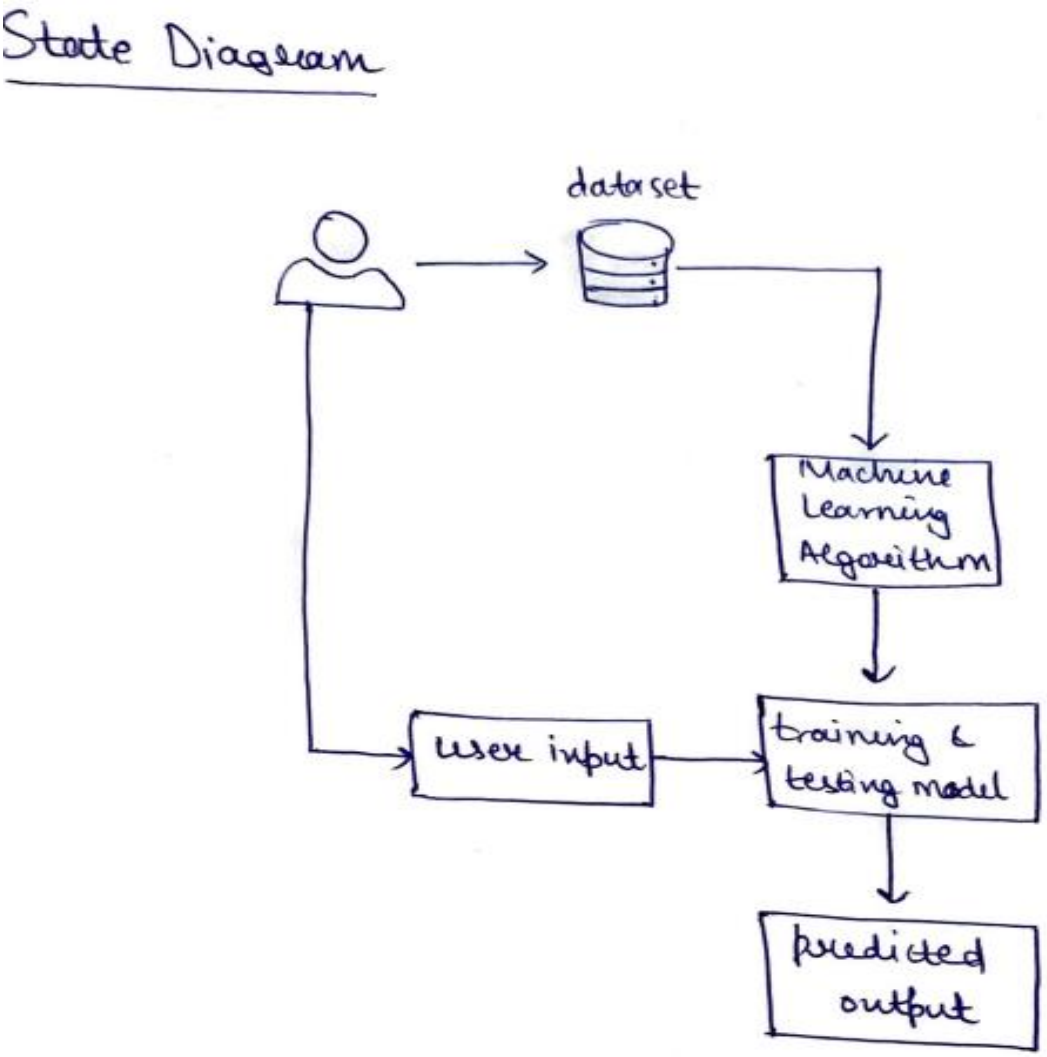


Fig 3.8 State Diagram

3.5.4 Sequence Diagram

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction.

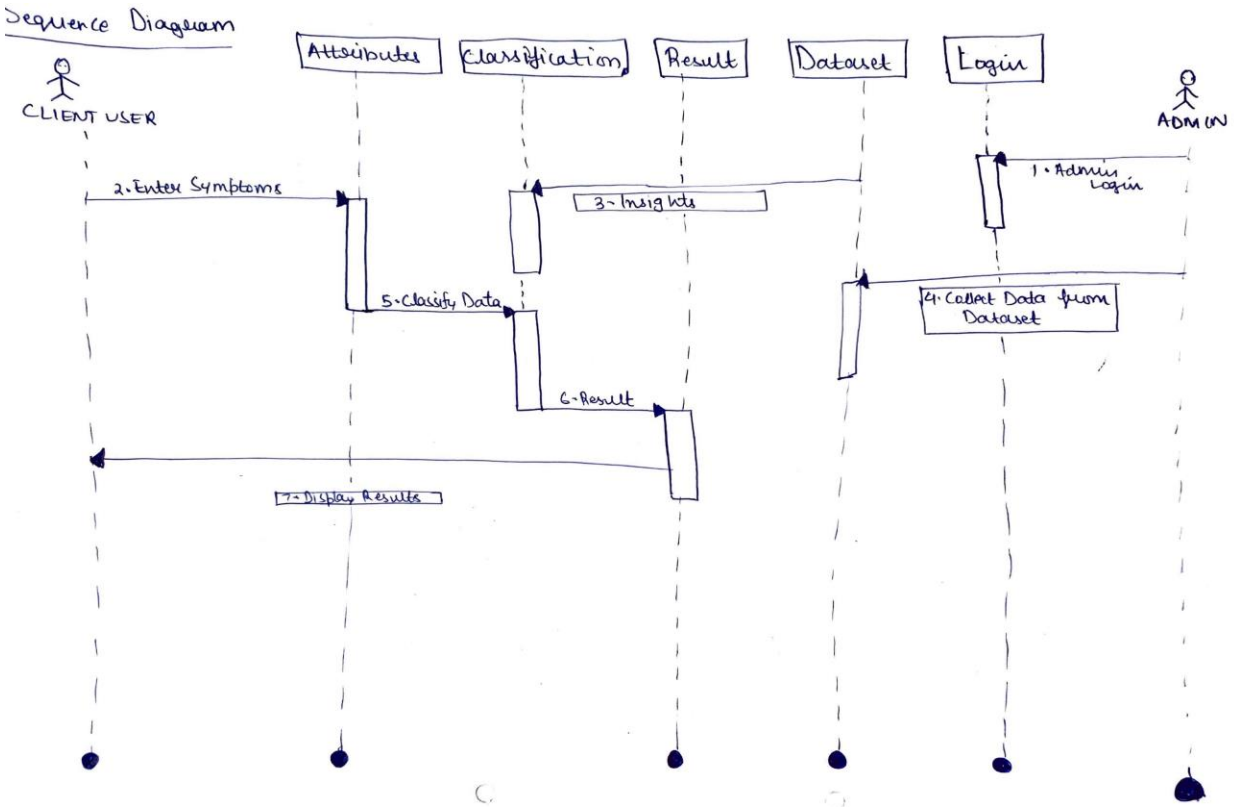


Fig 3.9 Sequence Diagram

3.5.5 Data Flow Diagram

A data flow diagram is a graphical representation of the flow of data and processes involved in detecting diseases using various tools and technologies.

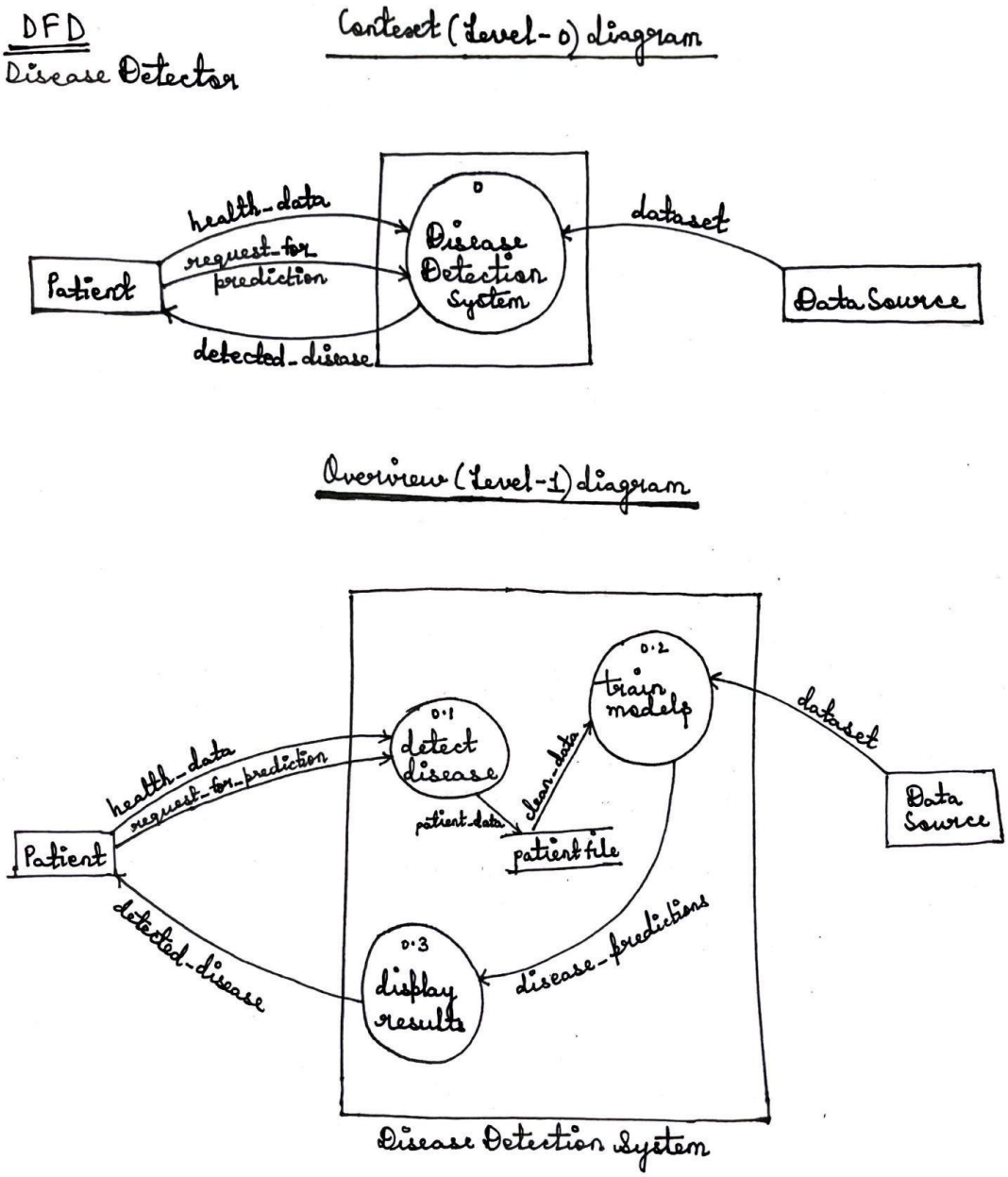


Fig 3.10 Data Flow Diagram

CHAPTER 4

RESULTS/OUTPUTS

By the analysis of results we can compare how much better this proposed system is performing. In result analysis we will see accuracy of different diseases that are predicted using our proposed system. We have taken datasets of 100 cases for result analysis.

Disease based accuracy analysis for 100 cases:

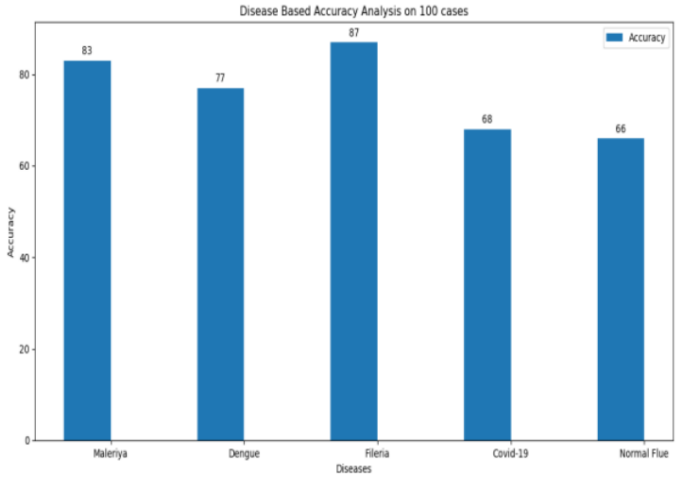


Fig 4.1 Accuracy Analysis

Above diagram shows the accuracy of 5 diseases that are malaria, dengue, filaria, covid-19 and normal flu.

4.1 Screenshots

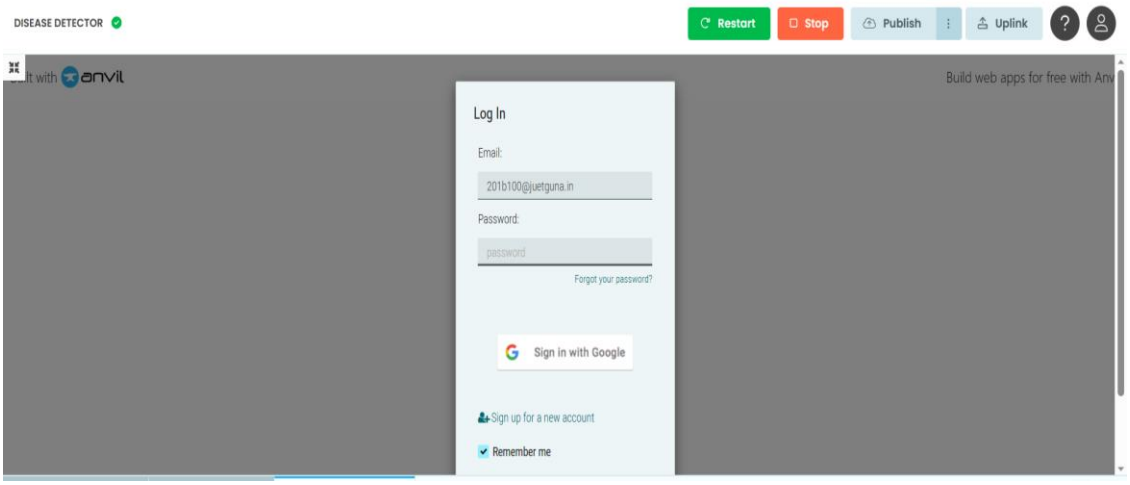


Fig 4.2 Login pop-up display

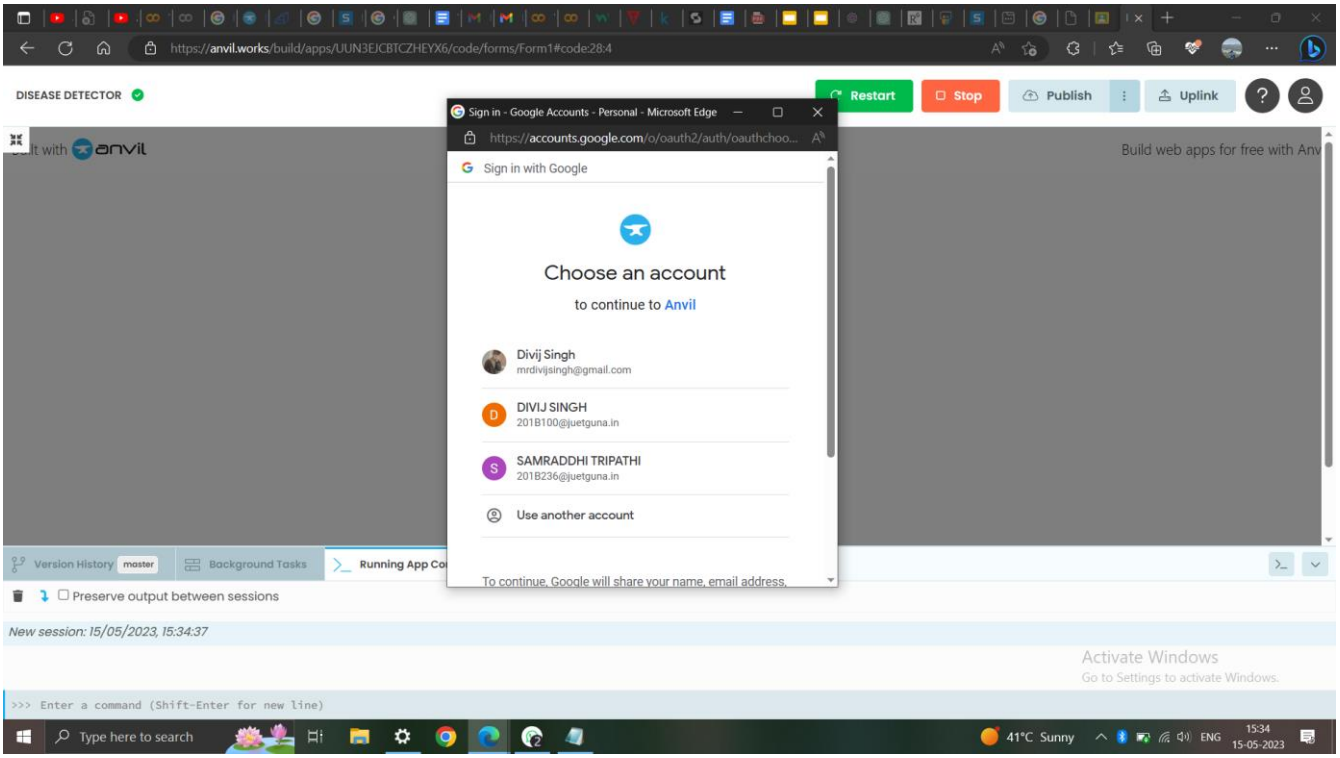


Fig 4.3 Sign-in with Google

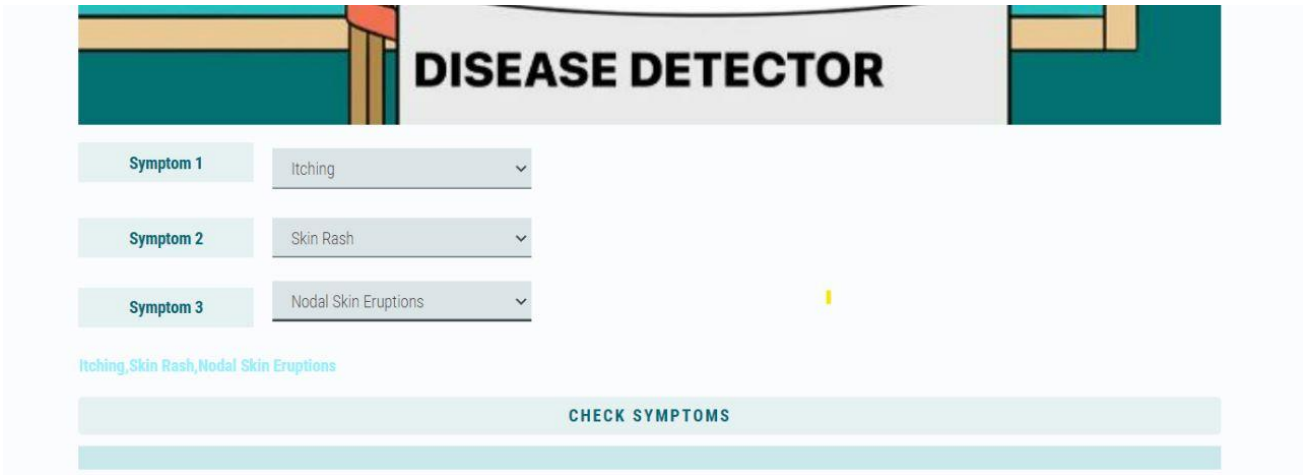


Fig 4.4 Interactive drop down lists

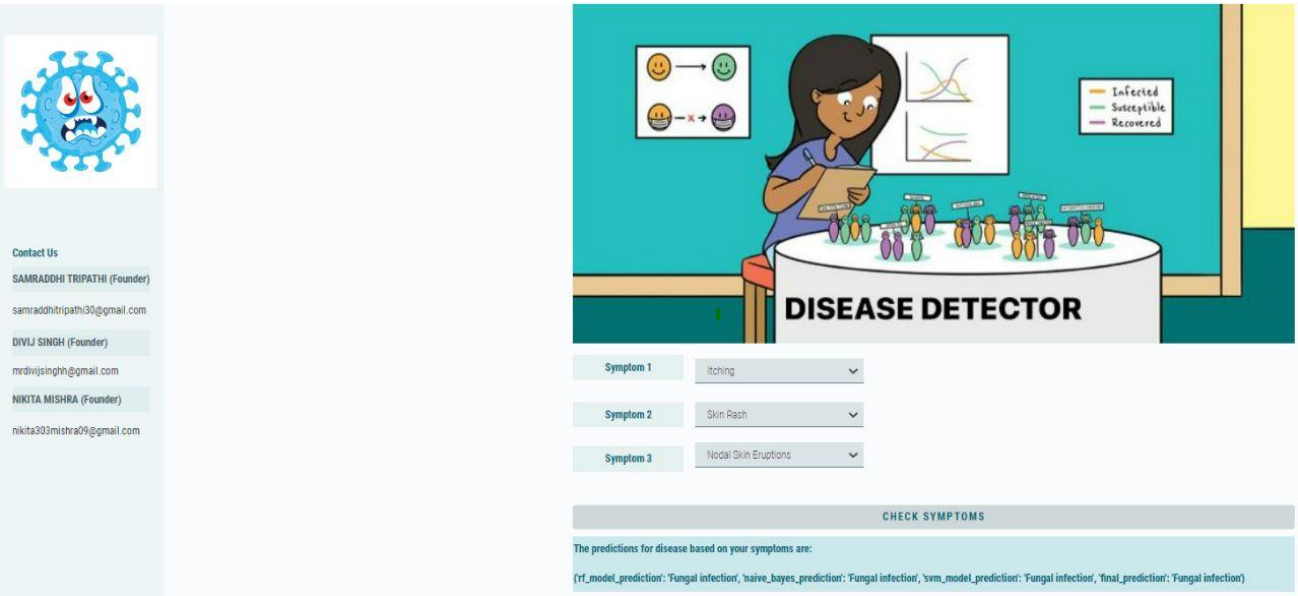


Fig 4.5 Landing Page

Fig 4.6 Side panel for team's contact details

CHAPTER 5
CONCLUSIONS/RECOMMENDATIONS

- The outcome which is expected from the project is given with the help of machine learning models that are used in the project to predict the disease on the basis of input provided by the user as a content of symptoms which are selected from a given list of symptoms provided to the user
- The expected outcome observed will also have an UI , so that it will be more easier, understandable and interactive to help a user to operate and predict the disease on the basis of input provided and becomes an easier task to perform
- The project is absolutely user friendly and essential machine learning models like Naive Bayes, Random Forest and SVM are applied which will help in predicting the disease in a better and easier way.
- The dataset used in the project contains 132 symptoms which are related to almost every kind of general disease and contains data in the form of 0 and 1 where 1 means that the symptom is present and 0 means that the symptom is not present.
- The system also shows the probability of the disease and how many chances there are that the user is suffering from the disease which is predicted by the system which helps in better identification of the disease and a better diagnosis.
- As the result obtained is specific and depends upon the input provided by the users as symptoms should be accurate with prior knowledge, so there is no misinterpretation of disease.
- The accuracy of the disease prediction is about 86% so we can say that the dataset is not in over-fitting situation and predicts the correct

CHAPTER 6
REFERENCES

[1] Dahiwade, D., Patle, G., and Meshram, E. (2019). “Designing disease prediction models using machine learning approaches.” 2019 3rd International Conference on Computing Method-
ologies and Communication (ICCMC), IEEE. 1211–1215.

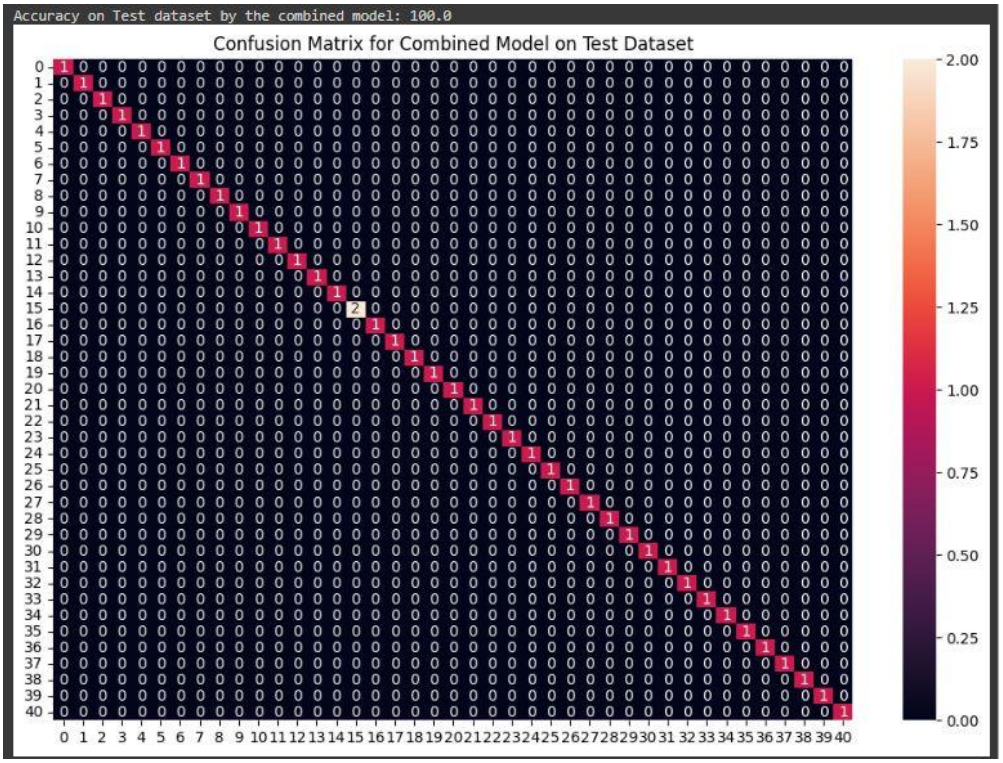
[2] Fitriyani, N. L., Syafrudin, M., Alfian, G., and Rhee, J. (2019). “Development of disease prediction model based on ensemble learning approach for diabetes and hypertension.” IEEE
Access, 7, 144777–144789.

[4] Kohli, P. S. and Arora, S. (2018). “Application of machine learning in disease prediction.” 2018 4th International Conference on Computing Communication and Automation
(ICCCA), IEEE. 1–4

APPENDIX 1

1. **Data Collection**
    - **Kaggle Dataset:** This dataset consists of two CSV files one for training and one for testing. There are a total of 133 columns in the dataset out of which 132 columns represent the symptoms and the last column is the prognosis.
2. **Data Cleaning and Preprocessing**
    - In our dataset all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a label encoder.
    - Technique used for feature engineering: Feature Scaling
3. **Model Building**

    - After splitting the data, we will be now working on the modeling part. We will be using K-Fold cross-validation to evaluate the machine-learning models.
    - Support Vector Classifier: Support Vector Classifier is a discriminative classifier i.e. when given a labeled training data, the algorithm tries to find an optimal hyperplane that accurately separates the samples into different categories in hyperspace.
    - Gaussian Naive Bayes Classifier: It is a probabilistic machine learning algorithm that internally uses Bayes Theorem to classify the data points.
    - Random Forest Classifier: Random Forest is an ensemble learning-based supervised machine learning classification algorithm that internally uses multiple decision trees to make the classification. In a random forest classifier, all the internal decision trees are weak learners, and the outputs of these weak decision trees are combined i.e. mode of all the predictions is as the final prediction.
    - We used a confusion matrix to determine the quality of the models.
4. **Model Evaluation**
    - Fitting the model on whole data and validating on the Test dataset and it can clearly see that our combined model has classified all the data points accurately.



5. Inference

- After training the three models we will be predicting the disease for the input symptoms by combining the predictions of all three models. This made the overall prediction more robust and accurate.

6. Disease Prediction Function

```
# Input: string containing symptoms separated by commas
# Output: Generated predictions by models
def predictDisease(symptoms):
    symptoms = symptoms.split(",")

    # creating input data for the models
    input_data = [0] * len(data_dict["symptom_index"])
    for symptom in symptoms:
        index = data_dict["symptom_index"][symptom]
        input_data[index] = 1

    # reshaping the input data and converting it
    # into suitable format for model predictions
    input_data = np.array(input_data).reshape(1,-1)

    # generating individual outputs
    rf_prediction = data_dict["predictions_classes"][final_rf_model.predict(
    nb_prediction = data_dict["predictions_classes"][final_nb_model.predict(
    svm_prediction = data_dict["predictions_classes"][final_svm_model.predic

    # making final prediction by taking mode of all predictions
    final_prediction = mode([rf_prediction, nb_prediction, svm_prediction])[
    predictions = {
        "rf_model_prediction": rf_prediction,
        "naive_bayes_prediction": nb_prediction,
        "svm_model_prediction": svm_prediction,
        "final_prediction":final_prediction
    }
    return predictions

# Testing the function
print(predictDisease("Itching,Skin Rash,Nodal Skin Eruptions"))
```

7. Web App Development with Anvil

- **Anvil:** Creating web apps with Anvil is simple. No need to wrestle with HTML, CSS, JavaScript or PHP. We can do everything in Python.
- Steps to put a web front-end on a Google Colab notebook :-
    1) Create your Anvil app
    2) Design your page
    3) Add a button click event
    4) Enable the uplink
    5) Install the Uplink Library in our Colab Environment
    6) Connect Script
    7) Creating a callable function
    8) Publish and deploy the model