Ramakrishna Mission

Vivekananda Centenary College, Rahara

Department of Computer Science (CMSA)

CMSA DSE 5

Final Year Project Report

Disease Classification System Using Symptoms as Features

Submitted by

Anik Ghoshal and Soumya Dey

Supervised by

Dr. Biswajit Biswas
Department of Computer Science
Ramakrishna Mission Vivekananda Centenary College

June 24, 2021

Certificate

I hereby certify that the project report titled "Disease Classification System Using Symptoms as Features" which is submitted by Soumya Dey (Registration No.: A01-1112-117-004-2018, Examination Roll No.: 2021161263) and Anik Ghoshal (Registration No.: A01-1112-117-013-2018, Examination Roll No.: 2021161270) to the faculty of the Computer Science Department of Ramakrishna Mission Vivekananda Centenary College in partial fulfillment of the requirements for the Degree of Bachelor of Science(Honours) in Computer Science, is a record of the project work carried out by the students under my supervision in the academic session of semester-VI of 2020-2021. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this institution or elsewhere.

Biswajit Biswas

Date: 18/07/2021

Dr. Biswajit Biswas
(Supervisor)

Declaration

I, Soumya Dey (Registration No.: A01-1112-117-004-2018, Examination Roll No.: 2021161263), student of Bachelor of Science(Honours) in Computer Science, hereby declare that this project report titled “Disease Classification System Using Symptoms as Features” which is submitted by me to the Department of Computer Science, Ramakrishna Mission Vivekananda Centenary College,Rahara in partial fulfilment of the requirement for the award of degree of Bachelor of Science(Honours) in Computer Science, is a presentation of my own original work and not copied from any other source without any proper citation and has not been previously included in a project report submitted to this or any other institution for a degree, diploma or any other qualification. The project work was done in the academic session of semester-VI of 2020-2021 under the supervision of Dr. Biswajit Biswas.

Soumya Dey

Date: 18/07/2021

Soumya Dey

Acknowledgment

First, I would like to acknowledge our supervisor DR. Biswajit Biswas for giving me the opportunity to work under his guidance and to enlighten me about this wonderful domain of Computer Science. I would like to thank our respected Principal Sw. Kamalasthananda-ji Maharaj, respected Controller of Examination Sw. Vedanuragananda-ji Maharaj and our respected Coordinator Sw. Vedavidyananda -ji Maharaj for giving me the opportunity and allowing me to explore the research field in our domain. I am overwhelmed to work with my partner in this project, Anik Ghoshal. My regards to my parents, all my superiors and my seniors.

Soumya Dey

Date: 18/07/2021 Soumya Dey

Abstract

Disease Prediction plays a pivotal role in healthcare informatics. It is crucial to diagnose the disease at an early stage. This project presents the utilization of feature selection and classification techniques for the diagnosis and prediction of diseases. The proper selection of features plays a significant role in enhancing the accuracy of classification systems. The application of classification algorithms on disease datasets yields promising results by developing adaptive, automated, and intelligent diagnostic systems for diseases. This work presents a comprehensive overview of the Multi Layer Perceptron classifier and shows how it is very effective for this purpose.

The system takes a series of symptoms passed as an array from the frontend and communicates with the integrated backend server that in turn feeds the input data of symptoms to the machine learning model to return the predicted disease along with a brief description, precautions, and severity (work in progress). The ultimate goal is to make it efficient and accurate to the extent that it can be normalized and is safe for day-to-day use.

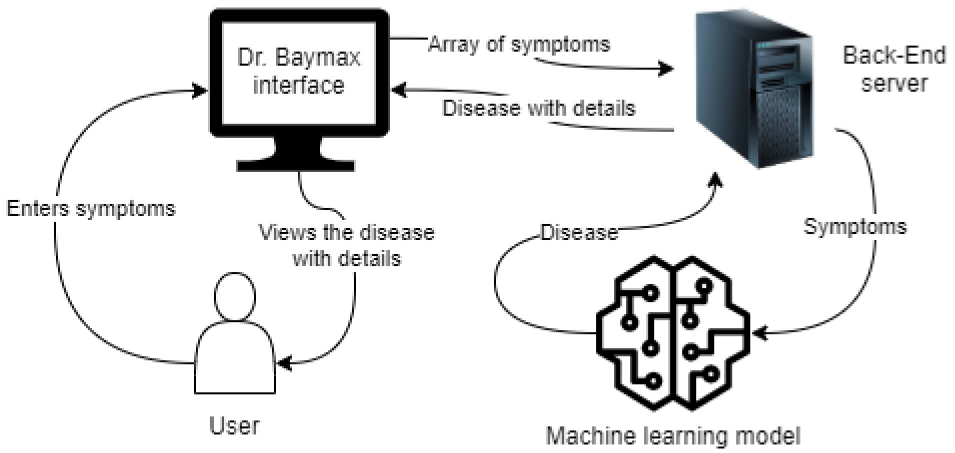

Fig 1.0 - Basic working diagram of the system

Table of Contents

Introduction

Dr. Baymax is your friendly virtual doctor. It is a pre-diagnostic disease prediction system that uses the symptoms given by the user as features and predicts the disease the user might have accordingly.
The user provides an array of symptoms to the front-end interface. Then an API call is made to the back-end server with that array of symptoms. From there, the back-end server takes the array and formats it, and builds an input vector that can be fed into the machine learning model which in this case, uses a Multi Layer Perceptron classifier to make the prediction. The model predicts and returns the output (Disease name) to the backend server and the respective disease description and precautions are generated by providing the disease name to the helper functions which are linked to respective CSV files. Then the complete JSON output (Disease name, the probability of the prediction, description and precaution of the disease) is returned to the front-end. Upon receiving the data the front-end interface (a web app or a mobile app) presents the predicted disease, its description, and precautions to the user in an easily understandable form.
In the next step of our project we added a database for users so that they can register and log in, check previous predictions, and much more. The criticality of a given disease is something that is currently a work in progress.

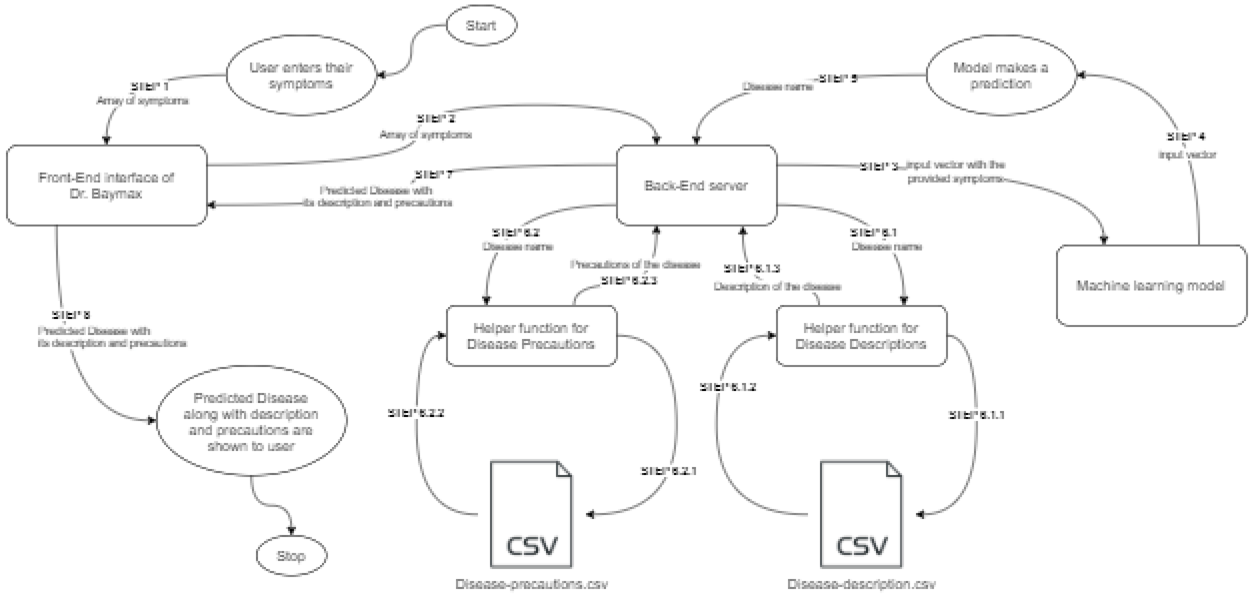
Start
User enters their symptoms
STEP 1
Array of symptoms
STEP 2
Array of symptoms
Front-End interface of Dr. Baymax
STEP 7
Predicted Disease with its description and precautions
Back-End server
STEP 5
Disease name
Model makes a prediction
STEP 4
input vector
STEP 3
input vector with the provided symptoms
Machine learning model
STEP 6.2
Disease name
Precautions of the disease
STEP 6.2.3
STEP 6.1.3
Description of the disease
STEP 6.1
Disease name
Helper function for Disease Precautions
Helper function for Disease Descriptions
STEP 8
Predicted Disease with its description and precautions
Predicted Disease along with description and precautions are shown to user
STEP 6.2.2
STEP 6.2.1
STEP 6.1.2
STEP 6.1.1
CSV
CSV
Stop
Disease-precautions.csv
Disease-description.csv

Fig 1.1 - Data Flow Diagram

Background Research

As mentioned earlier, Disease Prediction plays a pivotal role in healthcare informatics. It is crucial to diagnose the disease at an early stage and take necessary precautions to avoid major consequences. A doctor’s appointment can sometimes be tricky to get due to various limitations, which in turn can delay the diagnosis and treatment process. Also, these days people open up more to machines than to actual human beings due to fear of judgment. There has been a hike in virtual systems in recent days due to this. In a Ted Talks session by Anne Scherer, she clearly explains why humans are more open and comfortable around machines than their counterparts. This is one of the major inspirations behind the project.

We also took inspiration for the name from the movie “Big hero 6” where the protagonist was called “Baymax” and he was a virtual assistant cum doctor. We also analyzed existing projects and systems to get a good insight. “Ada” was one of those systems and we were inspired by it. There has been a lot of thought and care put into this project, it almost feels like a part of us and we are glad to see ourselves being successful in this endeavor.

Tools used

- **Languages:**
  - Python 3.8 (For Backend API, Data cleaning, Model training)
  - JavaScript ES6, HTML 5, CSS 3 (For Frontend)

- **Libraries:**
  - Numpy
  - Pandas
  - Sklearn
  - Matplotlib
  - Seaborn
  - joblib

- **Frameworks:**
  - Back-End:
    - Flask
  - Front-End:
    - React JS
    - Redux

- **Database:**
  - MongoDB

List of Datasets

- Health Dataset: Health_dataset.csv
- Cleaned health dataset: Formatted_health_dataset.csv
- Disease Description dataset: Disease_descriptions.csv
- Disease Precaution dataset: Disease_precautions.csv
- Disease distribution in training and testing dataset: Disease_distribution.csv
- Classification report of model: mlp-classification-report.csv

Source Code

- GitHub Repository of the project: DR-BAYMAX-Disease-prediction-system.git

Chapter-1 (Dealing with the Data)

In this chapter, We will take you through the process of extracting and reforming the data to make it efficient for our model. We will also talk about its features and how they are affecting the model's performance. We will do the same for our auxiliary datasets (i.e, the Precaution table, and the Description table).

1. Understanding the Main and Auxiliary Datasets

A. Health Dataset:

Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5
Fungal infection	itching	skin_rash	dischromic _patches		
Fungal infection	itching	skin_rash	nodal_skin_eruptions		
Allergy	continuous_sneezing	shivering	chills	watering_from_eyes	
Allergy	shivering	chills	watering_from_eyes		
GERD	stomach_pain	acidity	vomiting	cough	chest_pain
GERD	stomach_pain	acidity	ulcers_on_tongue	cough	chest_pain
Chronic cholestasis	itching	vomiting	yellowish_skin	nausea	loss_of_appetite
Chronic cholestasis	itching	vomiting	yellowish_skin	nausea	loss_of_appetite
Chronic cholestasis	itching	vomiting	yellowish_skin	nausea	loss_of_appetite

I. Description

In this dataset, the first column holds the Disease and the remaining columns contain the symptoms of the respective disease where several combinations of the symptoms are spanned across the rows of the dataset.

- **Number of Rows**: 4920
- **Number of columns**: 18

II. Columns [6 of 18 columns]

1. **Disease** [Diseases that may be present]
   - Unique values: 41
   - Most common value: Fungal Infection [2%]
   - Missing values: 0 [0%]

2. **Symptom 1** [the symptoms experienced during the disease]
    - Unique values: 34
    - Most common value: vomiting [17%]
    - Missing values: 0 [0%]

3. **Symptom 2** [the symptoms experienced during the disease]
    - Unique values: 48
    - Most common value: vomiting [18%]
    - Missing values: 0 [0%]

4. **Symptom 3** [the symptoms experienced during the disease]
    - Unique values: 54
    - Most common value: fatigue [15%]
    - Missing values: 0 [0%]

5. **Symptom 4** [the symptoms experienced during the disease]
    - Unique values: 50
    - Most common value: high_fever [8%]
    - Missing values: 348 [7%]

6. **Symptom 5** [the symptoms experienced during the disease]
    - Unique values: 38
    - Most common value: headache [7%]
    - Missing values: 1206 [25%]

B. Disease Description Dataset:

Disease	Description
Drug Reaction	An adverse drug reaction (ADR) is an injury caused by taking medication. ADRs may occur following a single dose or prolonged administration of a drug or result from the combination of two or more drugs.
Malaria	An infectious disease caused by protozoan parasites from the Plasmodium family that can be transmitted by the bite of the Anopheles mosquito or by a contaminated needle or transfusion. Falciparum malaria is the most deadly type.
Allergy	An allergy is an immune system response to a foreign substance that's not typically harmful to your body.They can include certain foods, pollen, or pet dander. Your immune system's job is to keep you healthy by fighting harmful pathogens.
Hypothyroidism	Hypothyroidism, also called underactive thyroid or low thyroid, is a disorder of the endocrine system in which the thyroid gland does not produce enough thyroid hormone.
Psoriasis	Psoriasis is a common skin disorder that forms thick, red, bumpy patches covered with silvery scales. They can pop up anywhere, but most appear on the scalp, elbows, knees, and lower back. Psoriasis can't be passed from person to person. It does sometimes happen in members of the same family.

I. Description

In this dataset, the first column holds the Disease and the remaining columns contain the description of the respective disease.

- **Number of Rows**: 41
- **Number of columns**: 2

II. Columns [2 of 2 columns]

1. **Disease** [Diseases experienced]
    - Unique values: 41
    - Missing values: 0 [0%]

2. **Description** [Description About the disease]
    - Unique values: 41
    - Missing values: 0 [0%]

C. Disease Precaution Dataset:

Disease	Precaution_1	Precaution_2	Precaution_3	Precaution_4
Drug Reaction	stop irritation	consult nearest hospital	stop taking drug	follow up
Malaria	Consult nearest hospital	avoid oily food	avoid non veg food	keep mosquitos out
Allergy	apply calamine	cover area with bandage		use ice to compress itching
Hypothyroidism	reduce stress	exercise	eat healthy	get proper sleep
Psoriasis	wash hands with warm soapy water	stop bleeding using pressure	consult doctor	salt baths
GERD	avoid fatty spicy food	avoid lying down after eating	maintain healthy weight	exercise
Chronic cholestasis	cold baths	anti itch medicine	consult doctor	eat healthy
hepatitis A	Consult nearest hospital	wash hands through	avoid fatty spicy food	medication
Osteoarthritis	acetaminophen	consult nearest hospital	follow up	salt baths
Hypoglycemia	lie down on side	check in pulse	drink sugary drinks	consult doctor

I. Description

In this dataset, the first column holds the Disease and the remaining columns contain the precautions to be taken for the respective disease.

- **Number of Rows**: 41
- **Number of columns**: 5

II. Columns [5 of 5 columns]

1. **Disease** [Diseases experienced]
    - Unique values: 41
    - Missing values: 0 [0%]

2. **Precaution 1** [precaution to take]
    - Unique values: 32
    - Most common value: Consult nearest hospital [7%]
    - Missing values: 0 [0%]

3. **Precaution 2** [precaution to take]
    - Unique values: 34
    - Most common value: exercise [7%]
    - Missing values: 0 [0%]

4. **Precaution 3** [precaution to take]
    - Unique values: 30
    - Most common value: consult doctor [15%]
    - Missing values: 0 [0%]

5. **Precaution 4** [precaution to take]
    - Unique values: 24
    - Most common value: follow up [15%]
    - Missing values: 0 [0%]

2. Reforming and Cleaning the Dataset

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct.

We have gone through the following steps for cleaning and properly formatting the "Health dataset" so that it becomes suitable for training and testing the model.

1. Removing any duplicate feature values.
2. Removing irrelevant features.
3. Fixing some structural errors such as inconsistent naming conventions, inconsistent null or missing values. For example: "N/A", "null" and "blank characters", all were present for representing non-existent symptoms in the dataset in some samples.

After performing the above mentioned cleanups and encoding the "Symptom" values as 0 and 1, where 0 indicates non-existent while 1 indicates existent we get the following dataset.

disease	itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity
Fungal infection	1	1	0	0	0	0
Fungal infection	0	1	0	0	0	0
Allergy	0	0	1	1	0	0
Allergy	0	0	0	1	0	0
GERD	0	0	0	0	1	1
GERD	0	0	0	0	1	0
Chronic cholestasis	1	0	0	0	0	0
Drug Reaction	1	1	0	0	1	0
Drug Reaction	1	0	0	0	1	0
Peptic ulcer disease	0	0	0	0	0	0
Peptic ulcer disease	0	0	0	0	0	0

Chapter-2 (Data models and Classifiers)

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn. And a classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "classes." So, In this chapter, we'll take you through the models used and their accuracy, and why we chose a certain model, and its advantages.

1. Models used and their accuracy

The two major models used with their accuracy:

Model	Accuracy score
Gaussian Naive Bayes Model	1.0 [100%]
Multi Layer Perceptron Model	0.9871 [98.71%]

Let's look into them in detail,

A. Gaussian Naive Bayes:

One type of Naive Bayes algorithm. Very easy to implement but very powerful as a classifier. Here we assume that there is no covariance between features i.e. the features are naturally independent.

- ➔ **Problem faced**:

The main problem we faced while working with the model was the problem of overfitting because while testing the model for every sample of the test dataset the probability of the prediction was exactly 100% which can generate erroneous results at times.

B. Multi Layer Perceptron:

A multi-layered perceptron (MLP) is one of the most common neural network models used in the field of deep learning. Often referred to as a “vanilla” neural network, an MLP is simpler than the complex models of today’s era. However, the techniques it introduced have paved the way for further advanced neural networks.

- ➔ **Advantage**:

Neural networks are capable of generalisation, that is, they classify an unknown pattern with other known patterns that share the same distinguishing features. This means noisy or incomplete inputs will be classified because of their similarity with pure and complete inputs.

2. Multilayer perceptron model

A Multilayer perceptron consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

A. Activation function:

We have used **ReLU** as the non-linear activation function in the neural network. ReLU (Rectified Linear Unit) activation function is an activation function defined as the positive part of its argument. ReLU has advantages over logistic sigmoid or tanh functions because of its efficient computation using only comparison, addition and multiplication.

$$f(x) = x^{+} = max(0, x)$$

B. Loss curve in training:

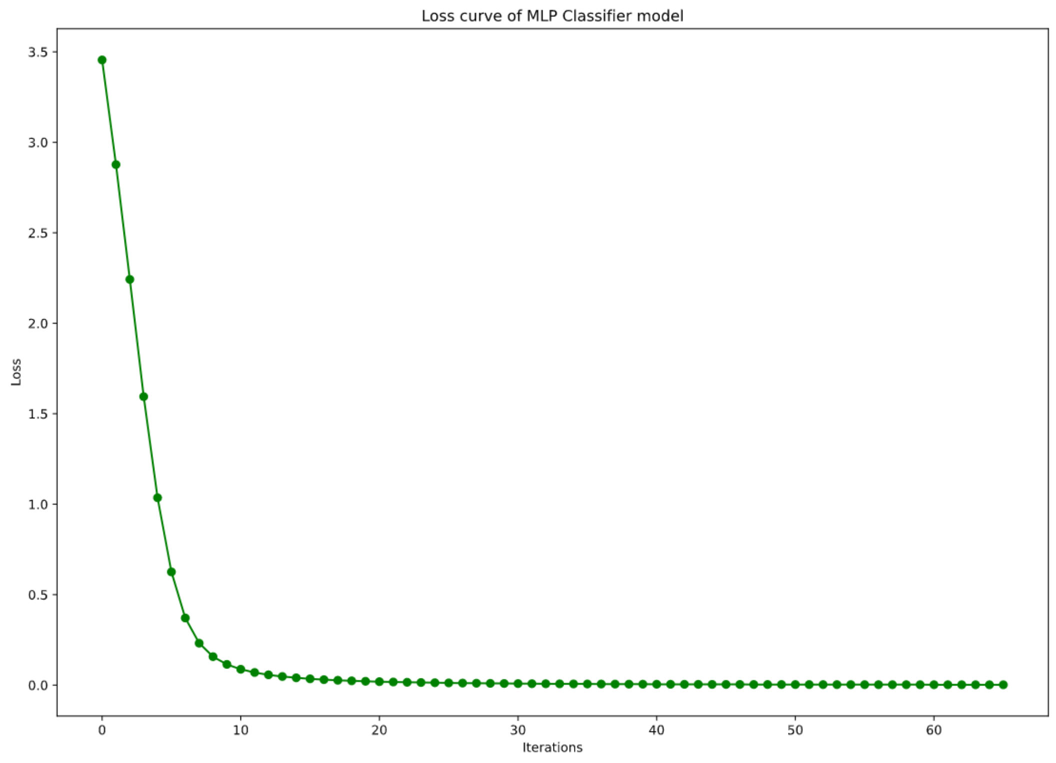

Fig 2.0 - Loss curve of MLP Classifier model

Chapter-3 (Data Visualization)

Data visualization is the process of transforming large data sets into a statistical and graphical representation to make data less confusing and more accessible. This can be helpful when exploring and getting to know a dataset.
Visualization and plots also helps a lot to understand the inner workings machine learning model in a more readable way. For example, confusion matrix visualization gives a much better idea of the performance of the ML model than just the accuracy score.

1. Confusion matrix visualization

A Confusion matrix is an $N \times N$ matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

For a 2 class classifier the following 4 are the basic terminology in understanding confusion matrix.

- **True Positives (TP)**: When the actual value is Positive and predicted value is also Positive.
- **True Negatives (TN)**: When the actual value is Negative and prediction is also Negative.
- **False Positives (FP)**: When the actual is negative but prediction is Positive. Also known as the Type 1 error.
- **False Negatives (FN)**: When the actual is Positive but the prediction is Negative. Also known as the Type 2 error.

Confusion matrices are widely used because they give a better idea of a model's performance than classification accuracy does.

For example, in classification accuracy, there is no information about the number of misclassified instances. Imagine that your data has two classes where 85% of the data belongs to class A, and 15% belongs to class B. Also, assume that the classification model correctly classified all the instances of class A, and misclassified all the instances of class B. In this case, the model is 85% accurate. However, class B is misclassified, which is undesirable. The confusion matrix, on the other hand, displays the correctly and incorrectly classified instances for all the classes and will, therefore, give a better insight into the performance of your classifier.

Confusion matrix with heatmap for the Multi Layer Perceptron model on the testing dataset:

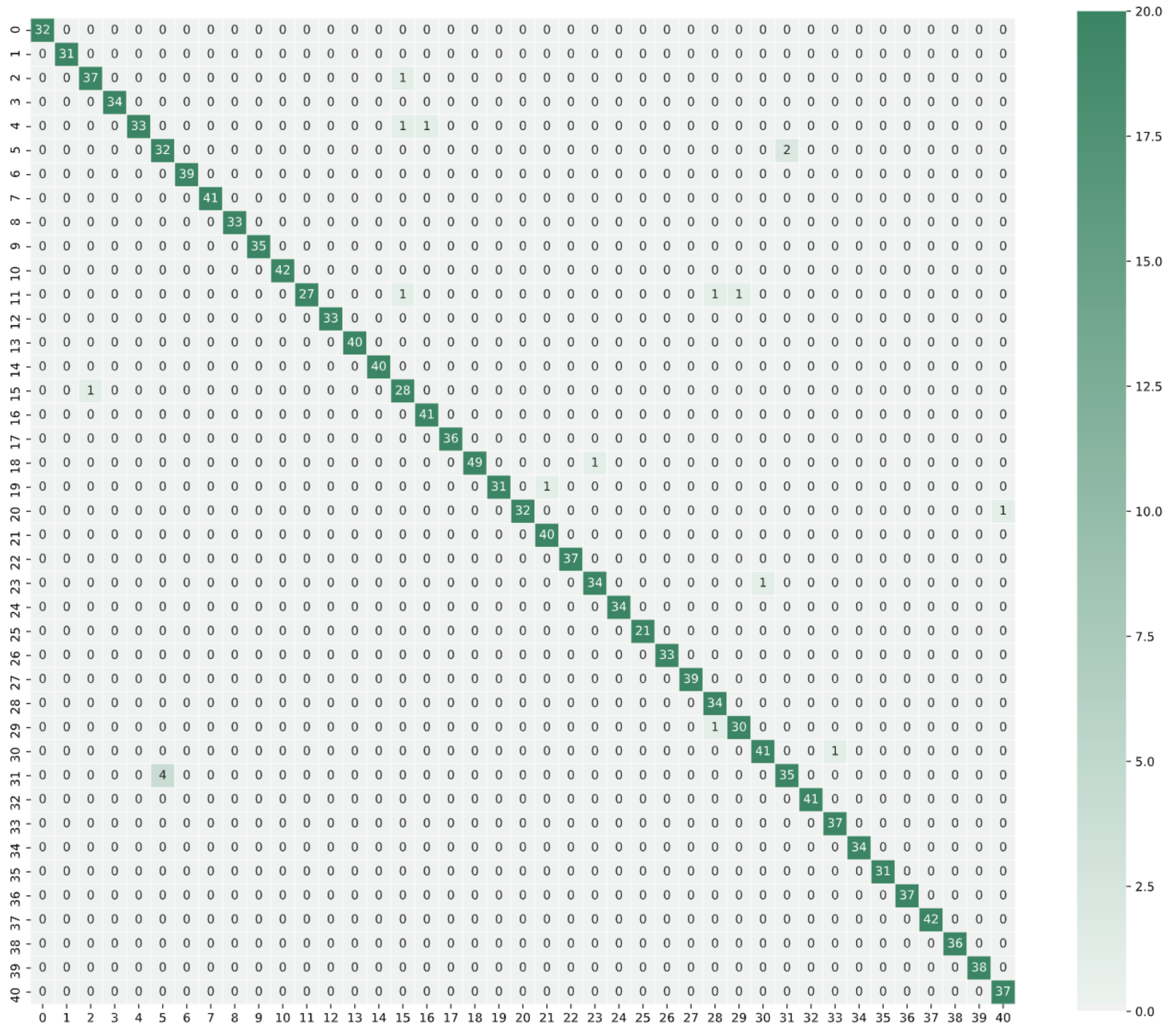

Fig 2.1 - 41×41 Confusion matrix of MLP model

2. Classification report

A Classification report is used to measure the quality of predictions from a classification algorithm. Only seeing the accuracy score of any model may sometimes fail to show actual performance of the model.
The following are the measures which will help us in determining the metrics to understand the performance of our model better.

- $Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{Correct\ predictions}{Total\ Predictions}$
- $Precision = \frac{TP}{(TP+FP)} = \frac{Predictions\ actually\ positive}{Total\ predicted\ positive}$
- $Recall = \frac{TP}{(TP+FN)} = \frac{Predictions\ actually\ positive}{Total\ actual\ positive}$
- $F1\ score = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)}$

Classification report of Multi Layer Perceptron model on testing dataset:

Disease	precision	recall	f1-score	support
(vertigo) Paroxysmal Positional Vertigo	1.00	1.00	1.00	32
AIDS	1.00	1.00	1.00	31
Acne	0.97	0.97	0.97	38
Alcoholic hepatitis	1.00	1.00	1.00	34
Allergy	1.00	0.94	0.97	35
Arthritis	0.89	0.94	0.91	34
Bronchial Asthma	1.00	1.00	1.00	39
Cervical spondylosis	1.00	1.00	1.00	41
Chicken pox	1.00	1.00	1.00	33
Chronic cholestasis	1.00	1.00	1.00	35
Common Cold	1.00	1.00	1.00	42

Dengue	1.00	0.90	0.95	30
Diabetes	1.00	1.00	1.00	33
Dimorphic hemorrhoids(piles)	1.00	1.00	1.00	40
Drug Reaction	1.00	1.00	1.00	40
Fungal infection	0.9	0.97	0.93	29
GERD	0.98	1.00	0.99	41
Gastroenteritis	1.00	1.00	1.00	36
Heart attack	1.00	0.98	0.99	50
Hepatitis B	1.00	0.97	0.98	32
Hepatitis C	1.00	0.97	0.98	33
Hepatitis D	0.98	1.00	0.99	40
Hepatitis E	1.00	1.00	1.00	37
Hypertension	0.97	0.97	0.97	35
Hyperthyroidism	1.00	1.00	1.00	34
Hypoglycemia	1.00	1.00	1.00	21
Hypothyroidism	1.00	1.00	1.00	33
Impetigo	1.00	1.00	1.00	39
Jaundice	0.94	1.00	0.97	34
Malaria	0.97	0.97	0.97	31
Migraine	0.98	0.98	0.98	42
Osteoarthritis	0.95	0.90	0.92	39
Paralysis (brain hemorrhage)	1.00	1.00	1.00	41
Peptic ulcer disease	0.97	1.00	0.99	37
Pneumonia	1.00	1.00	1.00	34
Psoriasis	1.00	1.00	1.00	31
Tuberculosis	1.00	1.00	1.00	37
Typhoid	1.00	1.00	1.00	42
Urinary tract infection	1.00	1.00	1.00	36
Varicose veins	1.00	1.00	1.00	38
hepatitis A	0.97	1.00	0.99	37
accuracy			0.99	1,476

Chapter-4 (Back-End Development)

Back end Development refers to the server side of development where you are primarily focused on how the site works. Making updates and changes to the data and different integrations in addition to monitoring functionality of the site will be your primary responsibility. This type of web development usually consists of three parts: a server, an application, and a database. Code written by back-end developers is what communicates the database information to the front-end i.e. the client's browser. Anything you can't see easily with the eye such as databases and servers falls under back-end development.

In this chapter, We'll discuss how the backend server of Dr. Baymax application is set up, and how it is integrated with the database. We will also talk about it's features and how it works alongside the machine learning model.

1. Back-End API Development with Flask

A. What is Flask:

Flask is a free and open-source web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco.

B. Reasons for choosing Flask as our back-end framework:

1. Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable. Instead of enforcing dependencies or project layout, it let's the user take control of the web development.
2. Flask supports various extensions to add many complex capabilities to the application.
3. Flask is very lightweight as a web framework.

C. Server API Endpoints:

- **Root route:** Base route to get the status of the server API.
    - **Endpoint**: "/"
    - **Method**: GET
    - **Response**:
        - <u>If successful</u>: Status code 200

```
{
    "msg": "Server running on => http://localhost:5000"
}
```

- <u>If unsuccessful</u>: Status code 500

```
{
        "errors": [{"msg": "Server error occurred. Please try again!"}]
}
```

- **Login route:** Route for logging in and authenticating users with a token.
  - **Endpoint**: “/login”
  - **Method**: POST
  - **Request**:
    - Body: Email and Password
    - Headers: None
  - **Response:**
    - If successful: Status code 200

```
{
    "token": "<authentication token with encoded user details>"
}
```

    - If unsuccessful: Status code 400

```
{
    "errors": [{"msg": "Email or Password incorrect!"}]
}
```

- **Register route:** Route for registering and authenticating users with a token.
  - **Endpoint**: “/register”
  - **Method**: POST
  - **Request**:
    - Body: Name, Email and Password
    - Headers: None
  - **Response:**
    - If successful: Status code 200

```
{
    "token": "<authentication token with encoded user details>"
}
```

            - If unsuccessful: Status code 400

```
{
    "errors": [{"msg": "<Error message here>"}]
}
```

- **Auth route:** Route for getting the saved data of a currently logged in user.
    - **Endpoint**: “/auth”
    - **Method**: GET or POST
    - **Request**:
        - Body: None
        - Headers: Authentication token provided during login/register
    - **Response:**
        - If successful: Status code 200

```
{
    "name": "<Name of the user>",
    "email": "<Email of the user>",
    "created": "<UTC Timestamp of the time when the account was created>"
}
```

        - If unsuccessful: Status code 400

```
{
    "errors": [{"msg": "<Error message here>"}]
}
```

- **Prediction route:** Route for making a disease prediction.
    - **Endpoint**: “/predict”
    - **Method**: POST

- **Request**:
    - Body: Array of the symptoms
    - Headers: Authentication token provided during login/register
- **Response:**
    - If successful: Status code 200

```
{
    // prediction object
    "prediction": {
        "user": {<user object>},
        "symptoms": [<array of the symptoms provided>],
        "disease": "Predicted disease by the machine learning
model",
        "probability": <Probability of the prediction>,
        "created": "<UTC Timestamp of the time when the account
was created>"
    },
    // prediction object end
    "description": "<Description of the disease predicted by the
machine learning model>",
    "precautions": [<array of the precautions to be taken for
the predicted disease by the machine learning model>]
}
```

- If unsuccessful: Status code 400

```
{
    "errors": [{"msg": "<Error message here>"}]
}
```

- **Reports route:** Route for getting all the previous predictions of a user.
    - **Endpoint**: “/report”
    - **Method**: GET

- **Request**:
    - Body: None
    - Headers: Authentication token provided during login/register
- **Response:**
    - If successful: Status code 200

```
{
    "count": <number of previous predictions>,
    "reports": [<array of previous predictions where each
element of the array is a prediction object>]
}
```

- If unsuccessful: Status code 400

```
{
    "errors": [{"msg": "<Error message here>"}]
}
```

D. Helper functions:

- **getDescription(disease):** For providing the description of a disease using Disease-Description dataset
    - Arguments:
        - Name of the Disease: *string*
    - Returns:
        - Description of the disease: *string*
- **getPrecautions(disease):** For providing the precautions of a disease using Disease-Precaution dataset.
    - Arguments:
        - Name of the Disease: *string*
    - Returns:
        - Precautions of the disease: *list*

2. Integrating Database with Back-End server

We have chosen MongoDB as our database for storing User data and Prediction data.

A. What is MongoDB:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

- Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.

B. Reasons for using MongoDB as our database:

1. MongoDB uses JSON-like documents that means it offers a lot of flexibility.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages.
3. MongoDB also supports schemas, hence it also offers control over the structure of the data while providing flexibility.
4. We can nest JSON to store complex data objects.
5. MongoDB Atlas, a Database as a service can be set up on the cloud for free.

C. Database schema structure of Dr. Baymax application:

- **Users Collection:** Collection to store user data.
    - **Schema:**
        - **_id**: ObjectId field [auto generated hex id by mongodb]
        - **name**: String field [max length = 30]
        - **email:** Email field [required = true, unique = true]
        - **password:** String field [required = true]
        - **created:** DateTime field [default = current UTC timestamp]

- **Sample of an User document:**

```
{
    "_id" : ObjectId("60e7262432c3d24ccce694a9"),
    "email" : "abc@xyz.com",
    "password" : "$2b$12$tbkzFhqVdvBCMZ6yebE0ZOyoL4C9qR6/3Vp816xVhS",
    "name" : "Dr Baymax User",
    "created" : ISODate("2021-07-08T16:22:06.484Z")
}
```

- **Predictions Collection:** Collection to store prediction data.
    - **Schema:**
        - **_id**: ObjectId field [auto generated hex id by mongodb]
        - **user**: Reference ObjectId field [referencing to the _id field of the respective user document]
        - **symptoms:** List field
        - **disease:** String field
        - **probability:** Float field
        - **created:** DateTime field [default = current UTC timestamp]
    - **Sample of an User document:**

```
{
    "_id" : ObjectId("60ec8c412546633e738a6cc7"),
    "user" : ObjectId("60e7262432c3d24ccce694a9"),
    "symptoms" : [
        "itching",
        "skin_rash",
        "continuous_sneezing",
        "shivering"
    ],
    "disease" : "Allergy",
    "probability" : 0.830390228837533,
    "created" : ISODate("2021-07-12T18:38:58.915Z")
}
```

Chapter-5 (Front-End Development)

Front-end web development, also known as client-side development, is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly. The challenge associated with front end development is that the tools and techniques used to create the front end of a website change constantly and so the developer needs to constantly be aware of how the field is developing.

The objective of designing a site is to ensure that when the users open up the site they see the information in a format that is easy to read and relevant. In this chapter, We'll take you through our process and the entire idea behind our frontend interface.

1. UI Design

For designing the User interface(UI) “Figma Design Tool” has been used. Figma is a vector graphics editor and prototyping tool which is primarily web-based. We have designed the main interface and the logo of our application using this tool.

A. Logo Design:

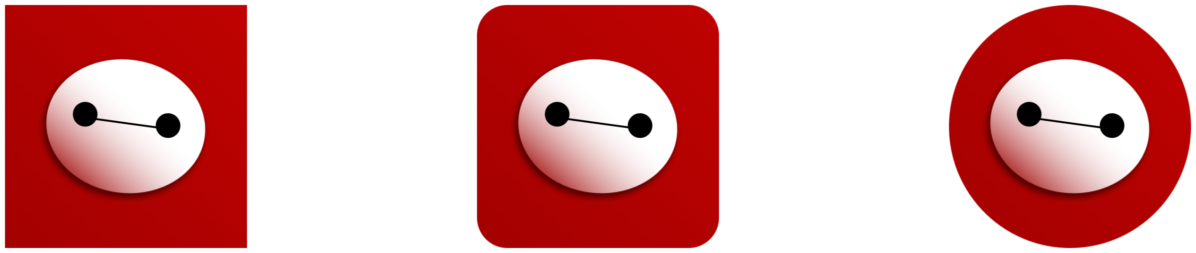

Fig 3.0 - Logo designs for Dr. Baymax application

B. Web Interface for laptops and desktops:

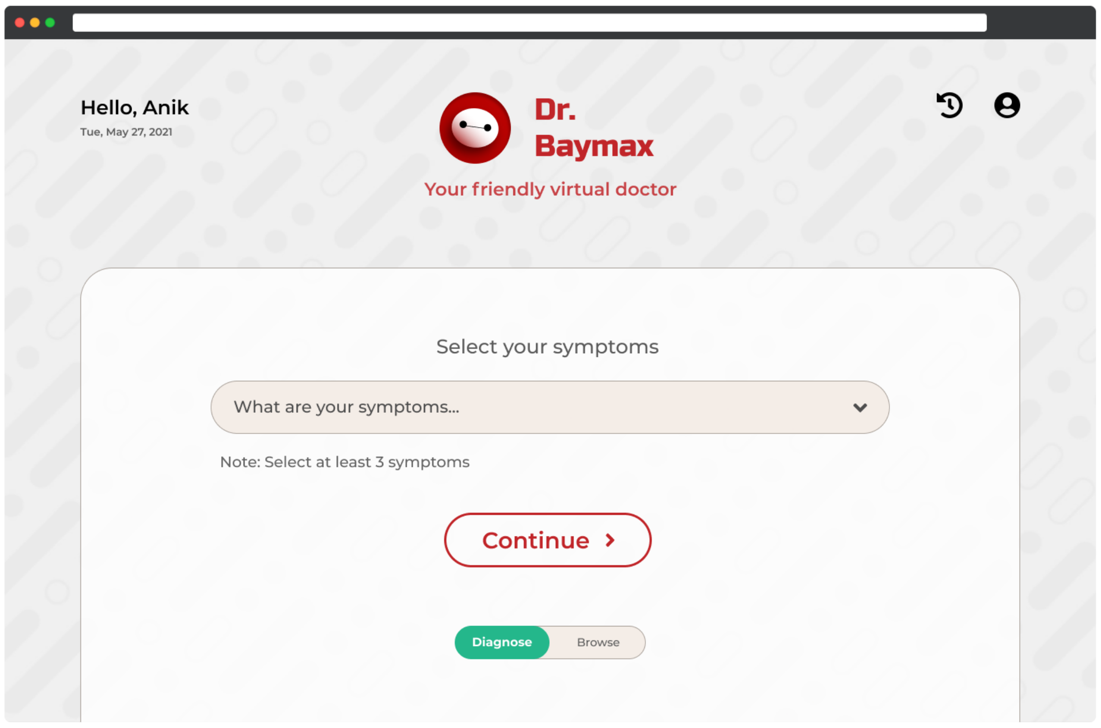
Hello, Anik
Tue, May 27, 2021
Dr. Baymax
Your friendly virtual doctor
Select your symptoms
What are your symptoms...
Note: Select at least 3 symptoms
Continue
Diagnose
Browse

Fig 3.1 - Web UI for Dr. Baymax application

2. Front-End Development with React JS

A. What is React JS:

React JS is a free and open-source JavaScript library for building user interfaces based on components created by Facebook.

B. Reasons for choosing React JS as our front-end framework:

6. React JS uses Virtual DOM, the virtual representation of DOM, to compare previous and recent versions of an application and only re-renders the parts of the interface that got changed.
7. React JS code is easier to update and manage due to its modular structure.
8. With React JS, the complex UI is broken down into smaller, reusable components that can be reused to build another application having the same functionality.
9. React uses an XML-like syntax called JSX for template creation that accepts HTML quoting and makes it easier for subcomponents to render.

C. Styling the elements of the Application:

For styling the different elements of the application CSS (Cascading Style Sheets) has been used. CSS is a language for specifying how documents are presented to users. CSS is a rule-based language — we define rules specifying groups of styles that should be applied to particular elements or groups of elements on your web page.

D. Component structure of Dr. Baymax React Application:

1. **Layouts** [Components used across the whole application]:
    a. **LandingPage.js** [The first page a user will see when visiting the application]
    b. **Alert.js** [Separate component for showing error messages]
    c. **NotFound.js** [The Page shown when a user tries to visit some page that is not available in the application]

2. **Auth**:
    a. **Login.js** [The component through which the Login process is done]
    b. **Register.js** [The component through which the Registration process is done]

3. **Dashboard**:
    a. **Dashboard.js** [The component page shown after successful login or registration]
        i. **DashboardTopBar.js** [Component for top part of Dashboard]
        ii. **DashboardInput.js** [Component for taking the symptom as inputs]
        iii. **Result.js** [Component for showing the predicted result to user]

4. **Report**:
    a. **Report.js** [The component page for showing the previous predictions]
        i. **ReportItem.js** [Separate component for showing each of the previous prediction]
        ii. **Pagination.js** [Separate component for handling the pagination for showing previous predictions]

3. Integration with Backend API

For integrating the Front-End interface with Back-End API and managing the state of the application Redux state container has been used.

A. What is Redux:

Redux is a global store to store the state of the variables in our app. Redux creates a process and procedures to interact with the store so that components will not just update or read the store randomly.

The 4 main concepts in Redux state management are:

1. **View**: It is the front-end of the application i.e. different components of the application discussed in the previous section.
2. **Store**: It is the place where all of the state management data resides.
3. **Actions**: Actions are responsible for performing some change by making requests to the back-end server and then it ***"dispatches"*** that action to the reducers so that the state management can be done and the new change can be reflected on the View.
4. **Reducers**: Reducers are responsible for making changes to the state object in the Store, to be reflected in the View.

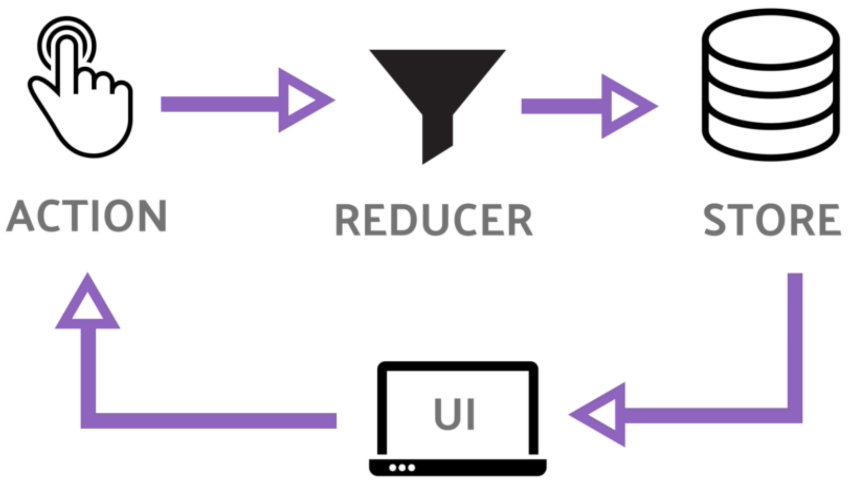

Fig 4.0 - Redux state management overview

B. State management structure in Dr. Baymax Application:

1. **Store:**
    a. **Auth** [global store to authentication and user data]
        i. Authentication status
        ii. User details
    b. **Prediction** [global store for current prediction and previous predictions]
        i. Current prediction
        ii. History of previous predictions
    c. **Alert** [global store for alert and error messages]
        i. Success alert
        ii. Error alert

2. **Actions:**
    a. **auth.js** [responsible for making request to backend server regarding login and registration process]
    b. **prediction.js** [responsible for making request to backend server regarding disease prediction process]
    c. **alert.js** [responsible for dispatching error messages received from backend server]

3. **Reducers:**
    a. **auth.js** [synchronizes the changes dispatched by auth action module and updates the auth store]
    b. **prediction.js** [synchronizes the changes dispatched by prediction action module and updates the prediction store]
    c. **alert.js** [synchronizes the changes dispatched by alert action module and updates the alert store]

Future prospects

As this project comes to an end, We look beyond this and have great ambitions attached to it. In this section we'll talk about ultimate goals and aspirations regarding this project.

- **Reliable pre-diagnostic system:** A reliable system is desirable when it comes to healthcare and especially in disease prediction. Our goal is to make the system more reliable so that it can be used as a legit preliminary diagnosis system in a hospital setting.

- **Android and IOS app:** Next obvious step is to build a react native application for android and IOS devices and make it accessible to a larger audience, And it gives a new taste to the entire model.

- **UI for smart watches:** Seeing the recent hike in smart watches, It'll be very innovative to have an UI for a smart watch which will be connected via the mobile app.

- **Doctor's availability:** This is one of the innovations on the auxiliary part of the project. We are planning to add a list of doctors that are available nearby according to that disease, Which can make the post diagnosis process very smooth.

- **Prediction stations:** The last piece to the puzzle is adding disease prediction systems, Which are like these small machines installed in the busiest roads and stations. It can be very helpful for people who are always in a rush and don't have time for actually visiting a doctor. They can get a pre diagnostic report and later can consult a doctor regarding the issue.

References

- https://www.kaggle.com/itachi9604/disease-symptom-description-dataset/
- https://ada.com/about/
- https://iq.opengenus.org/gaussian-naive-bayes/
- https://machinelearningmastery.com/neural-networks-crash-course/
- https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
- https://towardsdatascience.com/tagged/heatmap/
- https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5
- https://scikit-learn.org/sklearn.naive_bayes.GaussianNB.html/
- https://scikit-learn.org/sklearn.neural_network.MLPClassifier.html/
- https://flask.palletsprojects.com/en/2.0.x/

Conclusion

Recent methods of data interpretation have made it easy for a model to work efficiently and accurately. With this project, we were not only exposed to new types of models, classifiers, integration tools but also the medical aspect of it. We wanted to do something that can potentially be a change and at the same time help people. For us, this was the perfect opportunity and we are happy with our end result. Said that We know there’s a lot of potential waiting to be unlocked. We'll try our best to make sure that we have a part in that. As students, Our role is to shape the world and bring innovations to change it as well. The things we mentioned in our prospects may seem outrageous to some, But for us, It’s a gateway to the future and we’ll be more than happy to take that extra step.

As this project comes to an end, We’d like to thank our teachers and supervisor for allowing us to work on this wonderful project. We learned a lot throughout the process and we have discovered a new spark regarding this topic and we will try to continue to explore more possibilities.