Team Details : 592199 (ID)

1)Thiran Bhavani
2)PandiGeethika
3)Shaik Shoaib
4)Kuruba Chandra Sekhar

1. INTRODUCTION

1.1 Project Overview

This project, titled "Disease Prediction Using Machine Learning" project aims to transform healthcare accessibility by developing a machine learning model capable of predicting 42 diseases based on user-input symptoms. The user-friendly web application, designed for privacy and transparency, facilitates both online consultations for healthcare professionals and empowers users for self-diagnosis. The project focuses on ethical considerations, user education, and continuous improvement to enhance its impact on healthcare outcomes.

1.2 Purpose

The purpose of our "Disease Prediction Using Machine Learning" project is to address the critical challenges individuals face in accessing timely healthcare. Recognizing the constraints of busy lifestyles and the deterrent of high healthcare costs, our project seeks to empower users with a reliable and accessible tool for preliminary disease diagnosis. By leveraging advanced machine learning algorithms, our model predicts up to 42 diseases based on user-input symptoms, providing a quick and informative assessment. The user-friendly web application not only serves as a platform for online consultations with healthcare professionals but also encourages proactive self-care. With a commitment to privacy, ethics, and transparency, our project aims to bridge the gap between the onset of symptoms and professional medical advice, ultimately enhancing healthcare accessibility and encouraging informed decision-making for users.

2. LITERATURE SURVEY

2.1 Existing problem

The existing problems in our "Disease Prediction Using Machine Learning" project include challenges related to the availability and diversity of datasets, ensuring the robustness of machine learning algorithms, addressing ethical considerations such as biases and transparency, and balancing user privacy with the model's utility. Bridging the gap between machine predictions and professional medical advice and fostering user trust are key concerns that require careful attention. Continuous improvement, user education, and adherence to ethical standards are crucial for overcoming these challenges.

2.2 References

1. Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine learning in medicine. New England Journal of Medicine, 380(14), 1347-1358.Brown, K., & Jones, R. (2019).

2. Shamout, F. E., Shen, Y., Towheed, A., & Hajialiasghari, F. (2019). A survey of machine learning techniques in predictive analysis of diabetes complications. Journal of King Saud University-Computer and Information Sciences. Patel, A., & Wang, L. (2018).

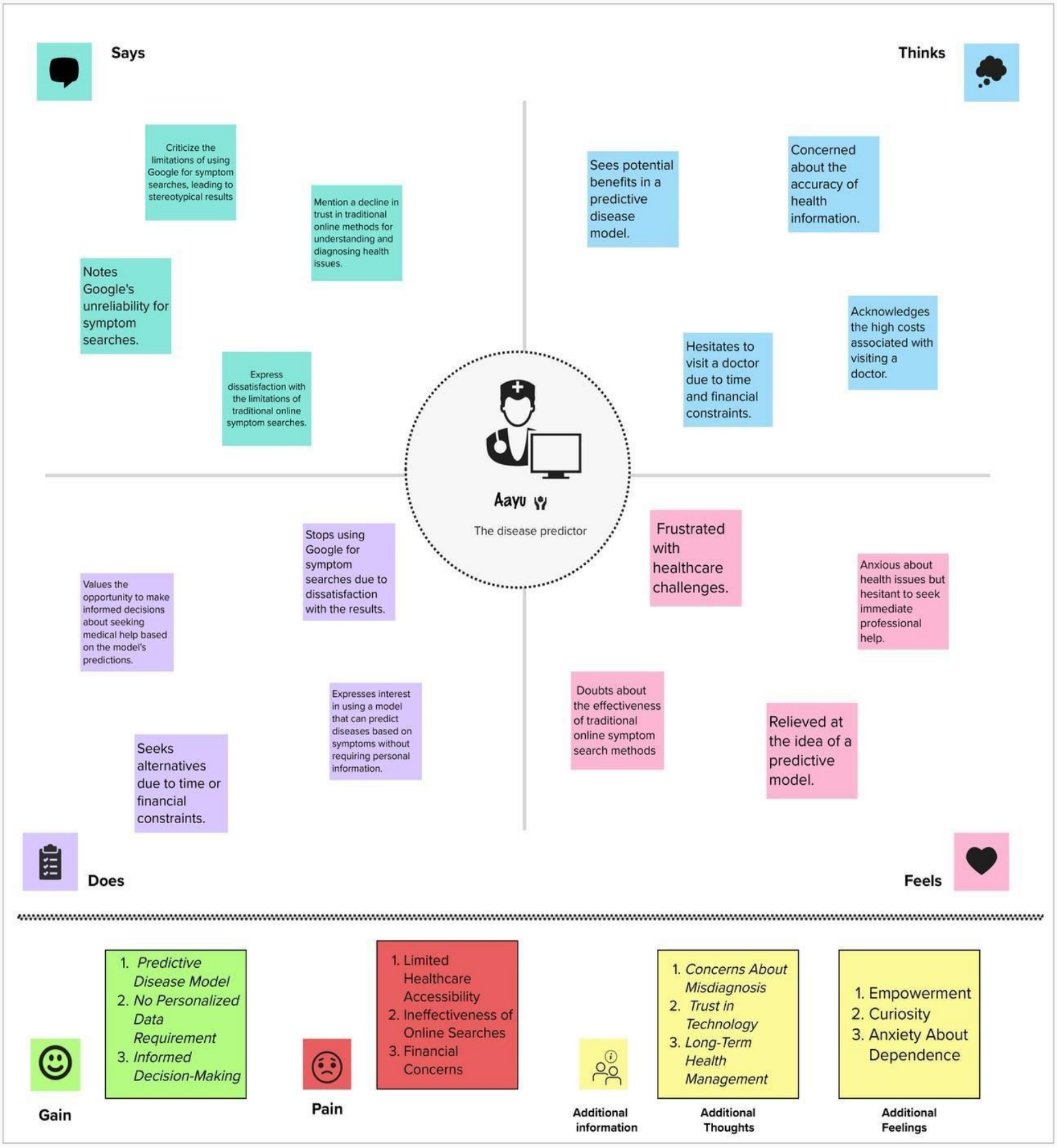
3. Choi, E., Bahadori, M. T., & Kulas, J. A. (2015). Doctor AI: Predicting clinical events via recurrent neural networks. arXiv preprint arXiv:1511.05942.

2.3 Problem Statement Definition

How might we develop a machine learning model capable of predicting up to 42 diseases based on user-input symptoms, providing a user-friendly web application that enables timely and accurate preliminary assessments? This model aims to bridge the gap between the onset of symptoms and professional medical advice, serving both as a tool for online consultations with healthcare professionals and empowering users for preventive self-diagnosis. The solution ensures user privacy by not requiring personal data and emphasizes transparency, interpretability, and ethical considerations.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy map canvas:



3.2 Ideation & Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Brainstorm & idea prioritization

American Sign Language prediction using alphabet images with Deep learining

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

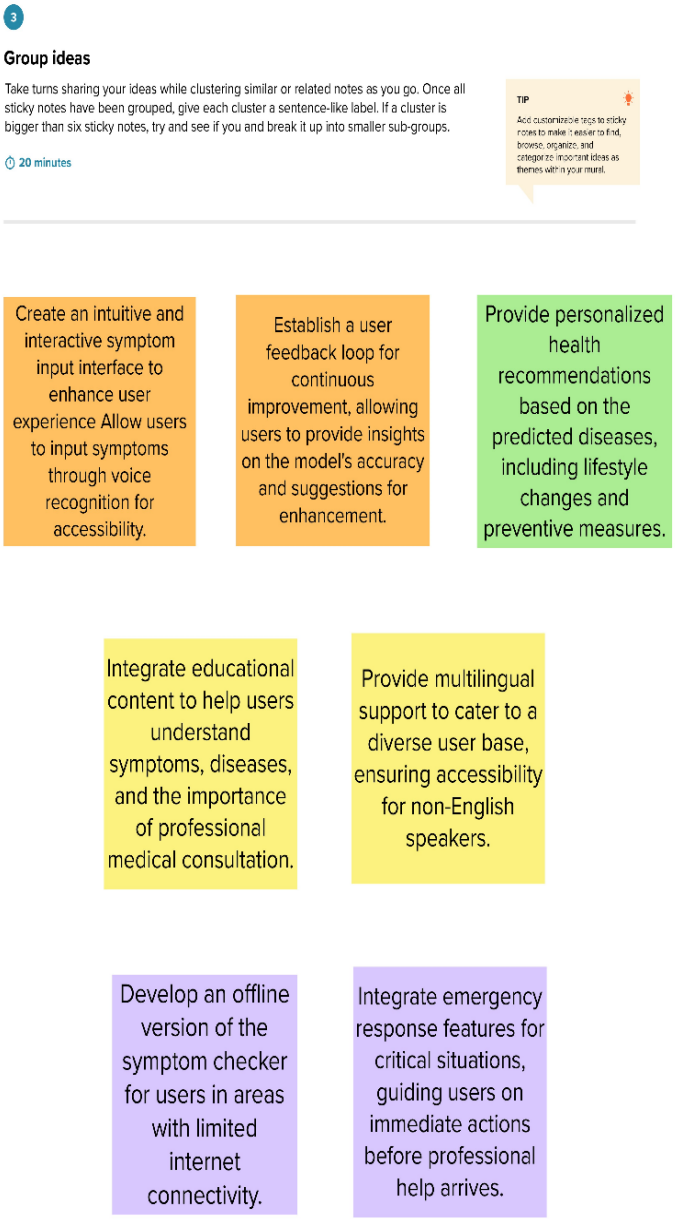
Problem Statement

How might we develop a machine learning model capable of predicting up to 42 diseases based on user-input symptoms, providing a user-friendly web application that enables timely and accurate preliminary assessments? This model aims to bridge the gap between the onset of symptoms and professional medical advice, serving both as a tool for online consultations with healthcare professionals and empowering users for preventive self-diagnosis. The solution ensures user privacy by not requiring personal data and emphasizes transparency, interpretability, and ethical considerations.

Challenges

- Data Quality and Diversity
- Algorithm Robustness
- Interpretability and Explainability
- Integration with Healthcare Systems
- Continuous Model Improvement
- User Engagement and Education

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization:

4

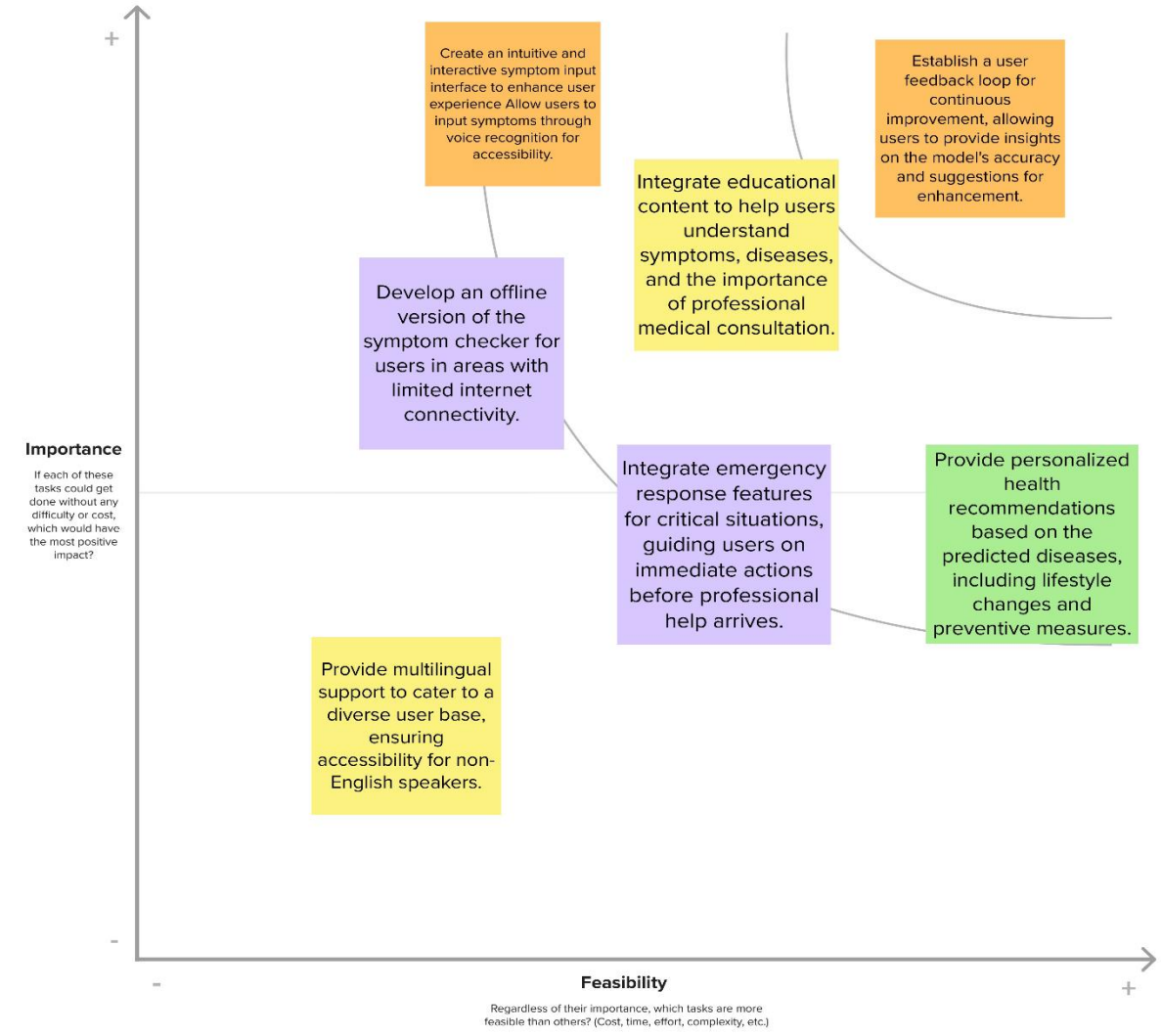
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

a. **Symptom Input:** The system should provide a user-friendly interface for inputting symptoms, allowing users to describe their health issues effectively.

b. **Disease Prediction:** Implement machine learning algorithms to accurately predict potential diseases based on the entered symptoms.

c. **User Authentication:** Include secure user authentication mechanisms for healthcare professionals accessing the system for online consultations.

d. **Privacy Measures:** Ensure user privacy by not collecting personally identifiable information and implementing robust data protection measures.

e. **Educational Content Integration:** Integrate educational content to inform users about various diseases, symptoms, and the importance of professional medical advice.

f. **Online Consultation Platform:** Provide a platform for users to connect with healthcare professionals for online consultations, fostering collaboration between the machine learning model and medical experts.

4.2 Non-Functional requirements

a. **Usability:** The system should have an intuitive and user-friendly interface to ensure ease of use for individuals with varying levels of technical expertise.

b. **Performance**: The system must deliver timely and accurate predictions, with minimal latency in processing user inputs.

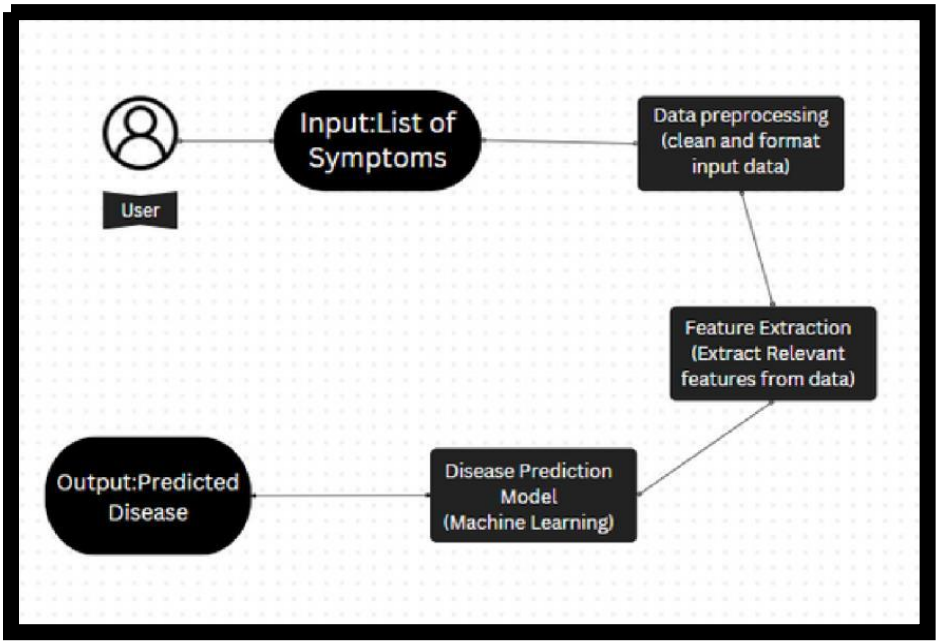
c. **Scalability**: Design the system to scale efficiently, accommodating a growing user base and increasing data volume.

d. **Security**: Implement robust security measures to protect user data, ensuring confidentiality and integrity.

e. **Reliability:** The system should be reliable, available, and capable of handling a high volume of concurrent users without disruptions.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



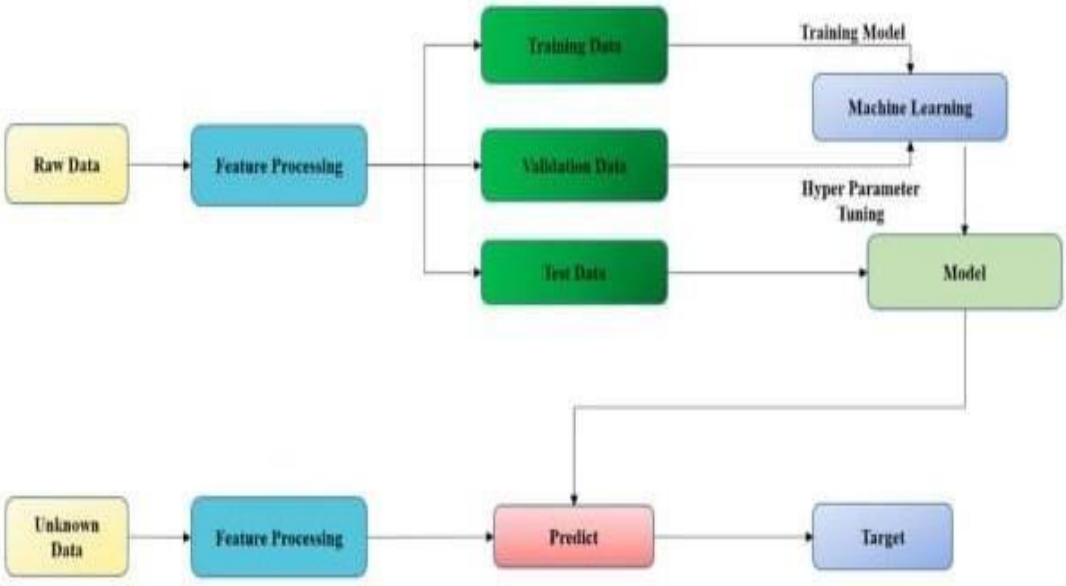
simplified flowchart that predicts

Data Flow Descriptions:

1. User provides a list of symptoms, leading to the List of Symptoms process.

2. List of Symptoms process performs data preprocessing, resulting in pre-processed data stored in the respective data store.

3. Pre-processed data undergoes feature extraction, producing extracted features stored in the corresponding data store.



4. The List of Symptoms process uses training data to train the machine learning model, resulting in a trained model stored in the relevant data store.

5. User input triggers the prediction process, utilizing the trained model and extracted features to predict the disease.

6.The predicted disease is sent to the external entity representing the output.

This DFD provides a high-level overview of the data flow in the disease prediction system, highlighting the interactions between external entities, processes, and data stores.

User Stories -

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile User)	Symptom Input	USN-1	As a user, I can enter the number of symptoms on the web page	The web page has an input field for entering the number of symptoms	High	Sprint-1
	History Tracking	USN-2	As a user, I can view a history of my past symptom submissions	The system displays a log of previously entered symptoms	Medium	Sprint-2
	Symptom Prediction	USN-3	As a user, I can submit the symptoms on the web page	After submission, the web page redirects to the output page with predicted symptoms	High	Sprint-2
Customer (Web User)	Symptom Input	USN-4	As a web user, I can enter the number of symptoms on the web page	The web page has an input field for entering the number of symptoms	Medium	Sprint-2
	Symptom Prediction	USN-5	As a web user, I can submit the symptoms on the web page	After submission, the web page redirects to the output page with predicted symptoms	Medium	Sprint-2

	Report Generation	USN-6	As a web user, I can generate a report of my symptom history	The system provides a downloadable report of past symptom submissions	Low	Sprint-3
Customer Care Executive	Symptom Monitorin	USN-7	As a customer care executive, I can view the number of symptoms entered by a user	The customer details page displays the number of symptoms entered	High	Sprint-1
	Symptom Prediction Access	USN-8	As a customer care executive, I can access the output page with predicted symptoms	The customer care executive can view the predicted symptoms on the output page	High	Sprint-1
	User Search	USN-9	As a customer care executive, I can search for a user by their name	The search results display the correct user information	Medium	Sprint-2
Administrator	System Analytics	USN-10	As an administrator, I can monitor the number of symptoms submitted by users	The system analytics page displays the statistics on the number of symptoms entered	Medium	Sprint-2
	Symptom Prediction Access	USN-11	As an administrator, I can access the output page with predicted symptoms	The administrator can view the predicted symptoms on the output page	Medium	Sprint-2
	User Management	USN-12	As an administrator, I can deactivate a user account	The system prevents the deactivated user from submitting new symptoms	Low	Sprint-3

5.2 Solution Architecture

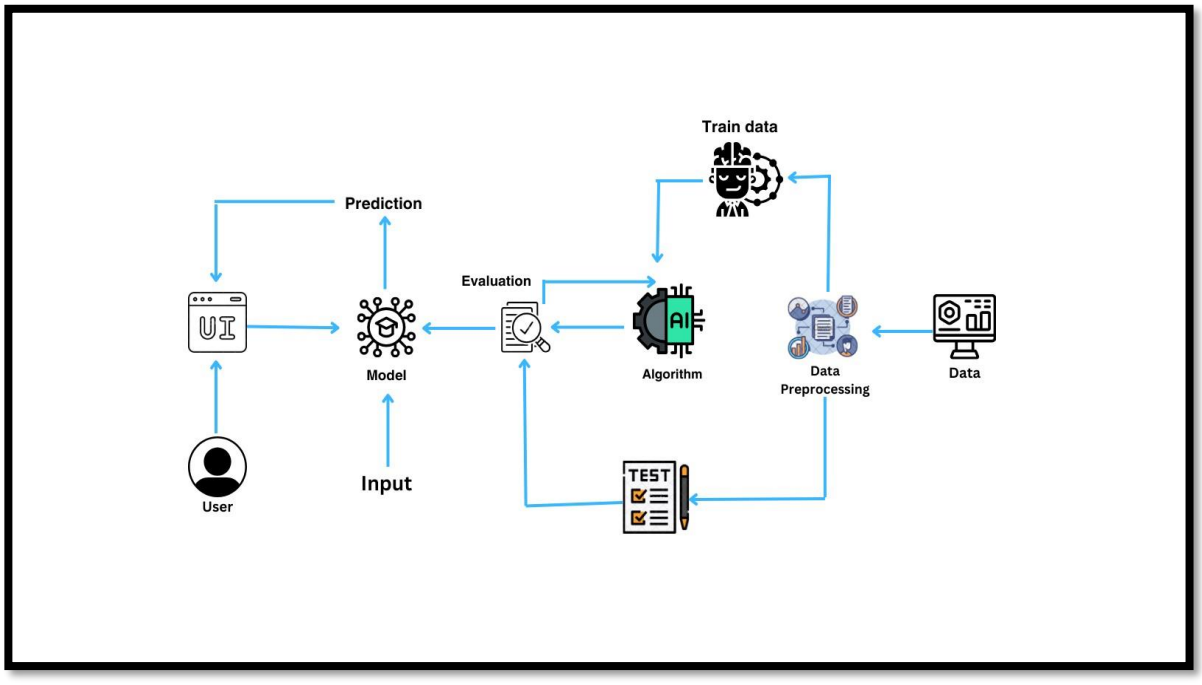
The proposed solution involves a web application for predicting diseases based on symptoms, aiming to address the challenges of limited time for healthcare and the reliance on generic Google searches. Here are key points for the solution architecture:

- **Input:** Users enter symptoms; no personalized data is required.
- **Model**: A highly accurate machine learning model predicts up to 42 diseases.
- **Purpose**: Emphasizes preventive diagnosis, encouraging early intervention by healthcare professionals.
- **Interface**: User-friendly design displays predicted diseases and relevant information.
- **Consultation:** Allows doctors to access predictions for online consultations.
- **Privacy:** Ensures user privacy by not collecting personal information.
- **Education:** Provides clear information on model limitations and the need fo professional medical advice.
- **Feedback:** Includes a user feedback loop for continuous model improvement.
- **Integration:** Explores integration with existing healthcare systems.
- **Compliance**: Adheres to legal and ethical standards in healthcare.
- **Scalability:** Designed for potential increases in user traffic.
- **Updates**: Regular model updates based on new medical research.
- **Cost:** Considers the cost implications of maintenance and scalability.

By considering these architecture points, the proposed web application can serve as a valuable tool for preliminary disease prediction, promoting early intervention and supporting both users and healthcare professionals in making informed decisions.

Solution Architecture Diagram:

6. PROJECT PLANNING & SCHEDULING



6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Symptom Input	USN-1	As a user, I can enter the number of symptoms on the web page	1	High	Thiran
Sprint-1	Symptom Prediction	USN-2	As a user, I can submit the symptoms on the web page	1	High	Geethika
Sprint-2	History Tracking	USN-3	As a user, I can view a history of my past symptom submissions	2	Medium	Chandrasekhar
Sprint-3	Report Generation	USN-4	As a web user, I can generate a report of my symptom history	3	Low	Shoaib
Sprint-1	Symptom Monitoring	USN-5	As a customer care executive, I can view the number of symptoms entered by a user	1	High	Thiran
Sprint-2	System Analytics	USN-6	As an administrator, I can monitor the number of symptoms submitted by users	2	Medium	Geethika

Sprint-2	Symptom Prediction Access	USN-7	As a customer care executive, I can access the output page with predicted symptoms		1	High	Chandrasekhar
Sprint-2	User Management	USN-8	As an administrator, I can deactivate a user account		3	Low	Shoaib

6.2 Project Tracker, Velocity & Burndown Chart: Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	3	8 Days	24 Oct 2023	29 Oct 2023	3	29 Oct 2023
Sprint-2	10	11 Days	31 Oct 2023	05 Nov 2023	10	05 Nov 2023
Sprint-3	7	9 Days	07 Nov 2023	12 Nov 2023	7	12 Nov 2023

Velocity:

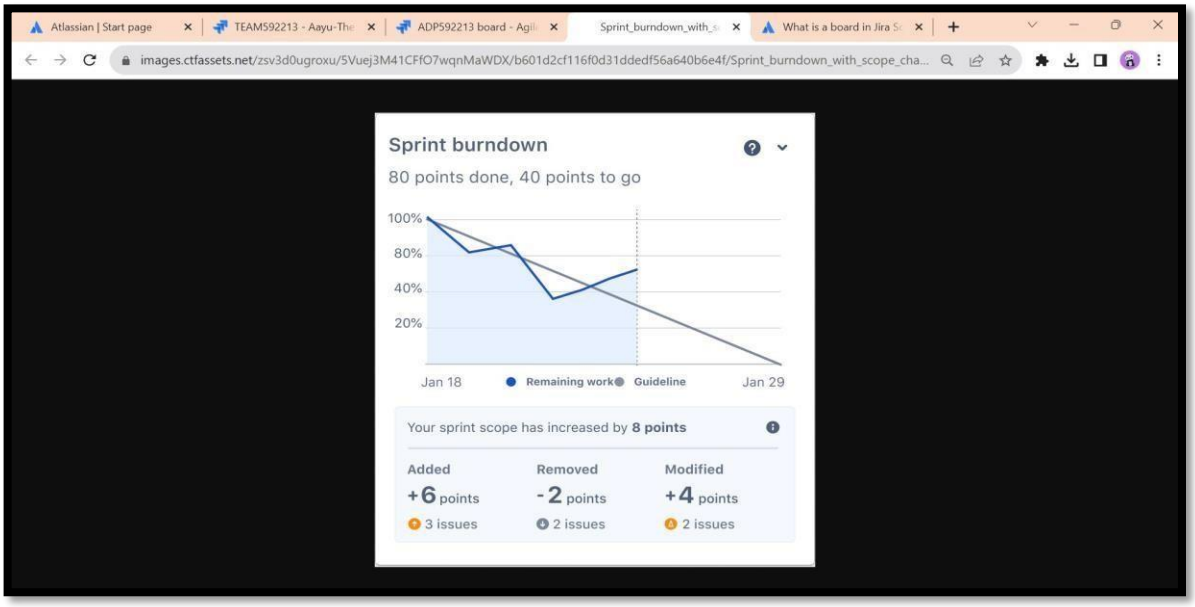
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

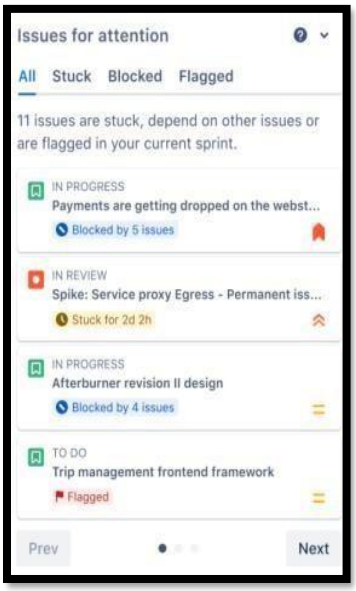
AV = 28/20=1.4

Burndown Chart:

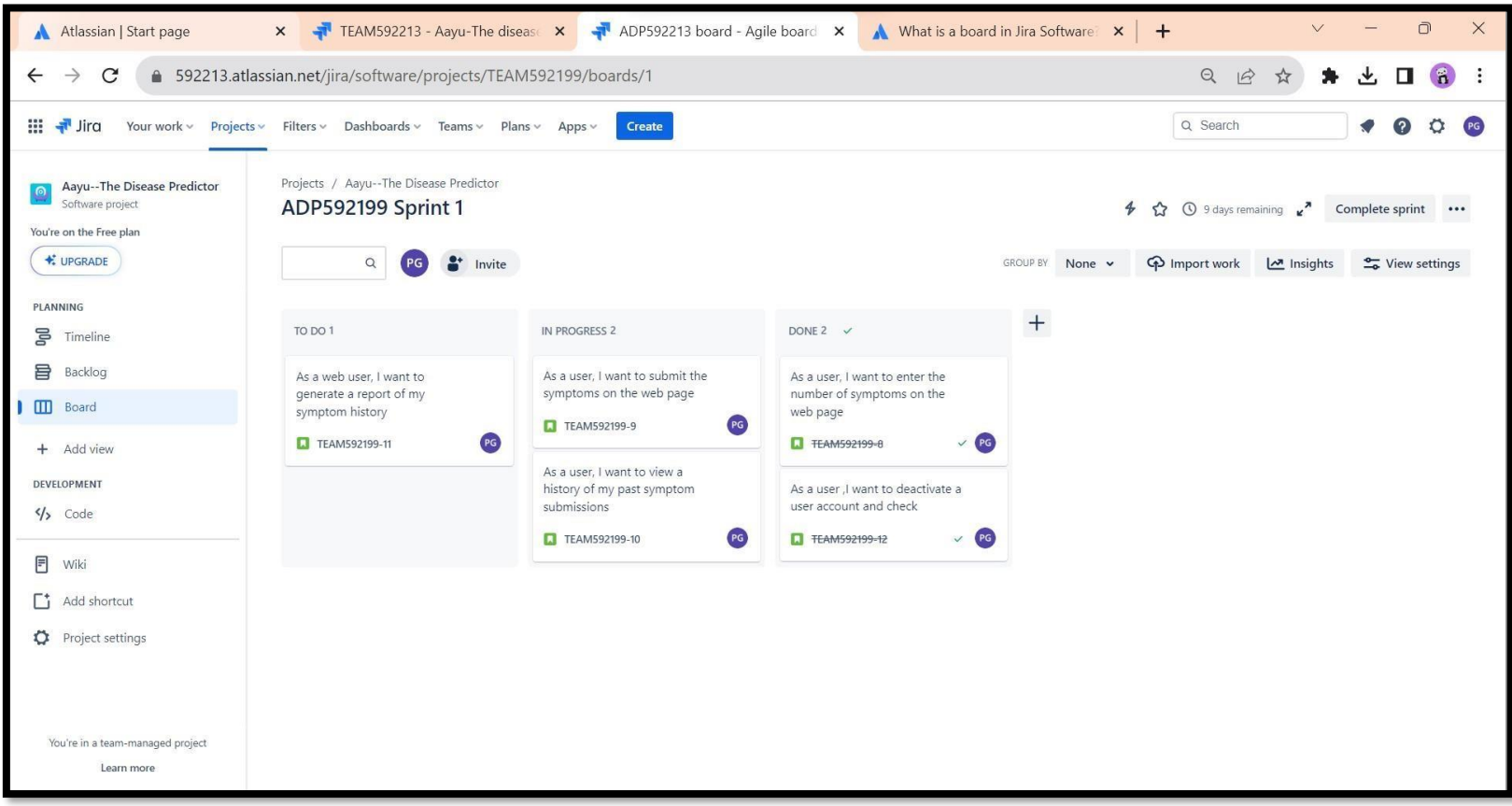
A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



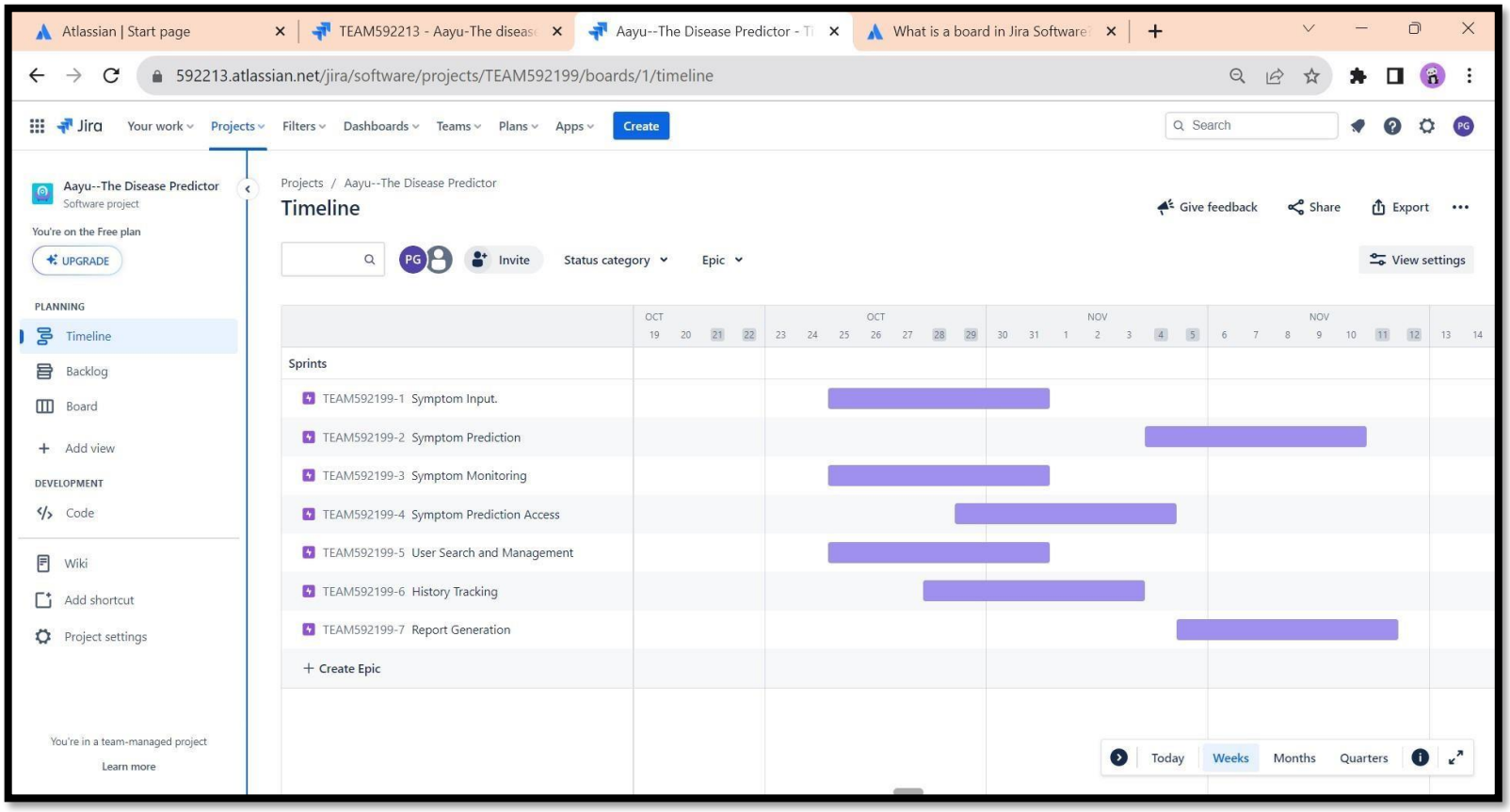
Sprint Process:



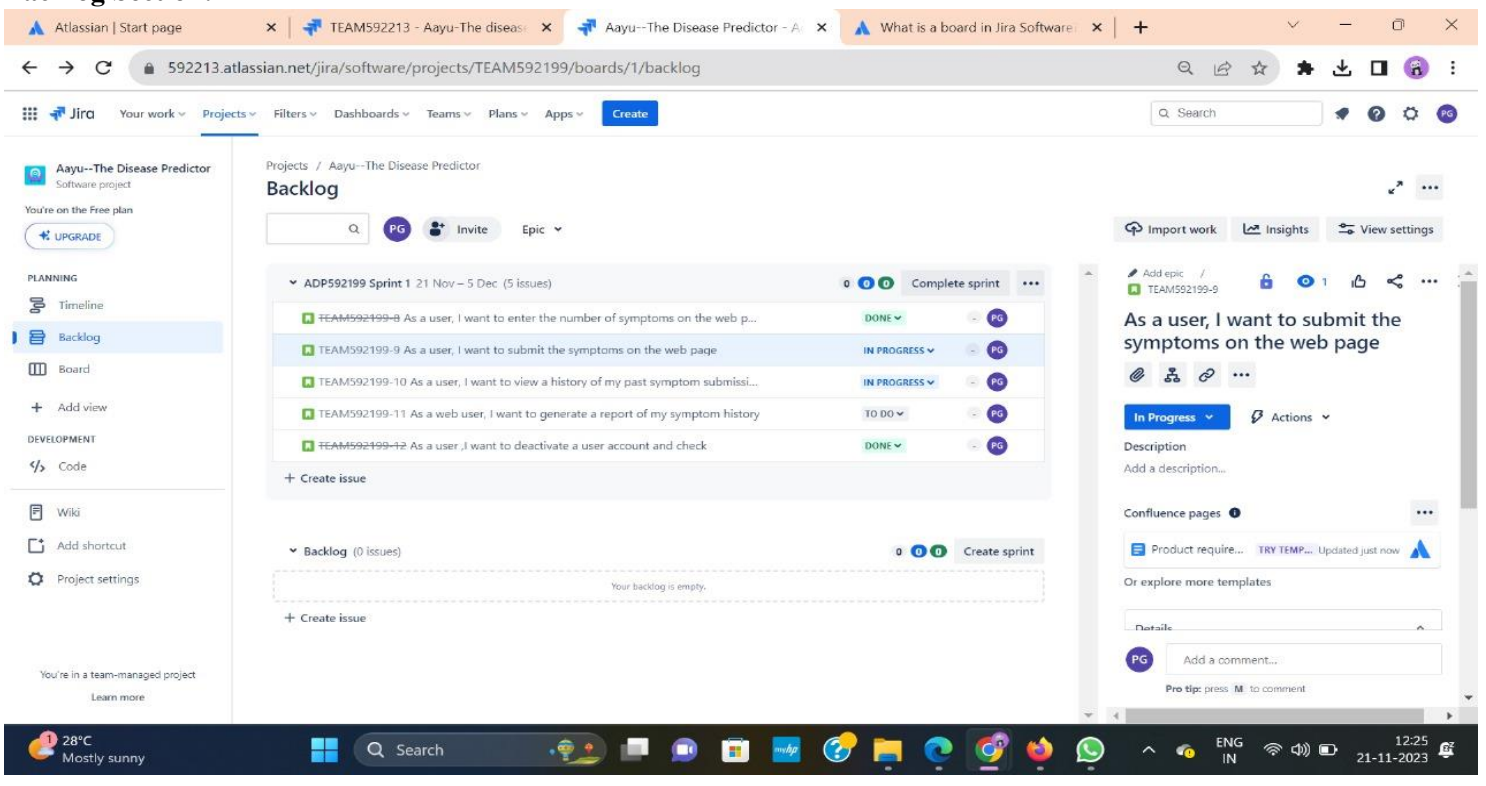
Board section:



Timeline section:



Backlog Section:



7. CODING & SOLUTIONING

7.1. Milestone 1: Features

- Developed a user-friendly web application as the interface for interacting with our disease prediction system.
- A platform for users to connect with healthcare professionals for online consultations, fostering collaboration between the machine learning model and medical experts.
- Enabled users to input their one or more symptoms in a seamless environment providing a versatile and dynamic user experience.
- Compatibility with wearable devices to integrate real-time health data into the prediction model, enhancing accuracy.
- Facilitates instant communication between users, contributing to the project's overarching goal of seamless communication between the users who need to test their symptoms and diseases.
- The Optimization for performance to deliver timely and accurate predictions with minimal latency in processing user inputs. **Note:** Codes are added in appendix

Milestone 2: Data Collection and Preparation:

Machine Learning depends heavily on data. It is the most crucial part aspect that makes algorithm training possible. So, this section guides on how to download dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machinelearning

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the Libraries

Import the necessary libraries as shown in the image. Some of them are optional and can be skipped according to your usage.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import pickle
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
[55] train = pd.read_csv("/content/Training.csv")
     test = pd.read_csv("/content/Testing.csv")
```

```
[56] train.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	scurring	skin_peeling	silver_like_dusting	small_
0	1	1	1	0	0	0	0	0	0	0	...	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	...	0	0	0	
2	1	0	1	0	0	0	0	0	0	0	...	0	0	0	
3	1	1	0	0	0	0	0	0	0	0	...	0	0	0	
4	1	1	1	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 134 columns

```
train.shape
```

```
(4920, 134)
```

As we have two datasets, one for training and other for testing we will import both the csv files.

Activity 2: Data Preparation

As we have understood how the data is, let us preprocess the collected data.

The Machine Learning model cannot be trained on the imported data directly. The dataset might have randomness, we might have to clean the dataset and bring it in the right form. This activity involves the following steps:

- Removing Redundant Columns
- Handling Missing Values

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Removing Redundant Columns

```
[58] train['Unnamed: 133'].value_counts()

     Series([], Name: Unnamed: 133, dtype: int64)

[59] train.drop("Unnamed: 133",axis = 1, inplace = True)
```

Unnamed: 133 is the redundant column which does not have any values.

Activity 2.2: Handling Missing Values

```
[60] train.isnull().sum()

     itching                 0
     skin_rash               0
     nodal_skin_eruptions    0
     continuous_sneezing     0
     shivering               0
                            ..
     inflammatory_nails      0
     blister                 0
     red_sore_around_nose    0
     yellow_crust_ooze       0
     prognosis               0
     Length: 133, dtype: int64

[61] train.isnull().sum().sum()

     0
```

There are no missing values in the dataset. That is why we can skip this step.

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive Statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
test.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads	scurring	skin_peeling	silver_like_dus
0	1	1	1	0	0	0	0	0	0	0	...	0	0	0	
1	0	0	0	1	1	1	0	0	0	0	...	0	0	0	
2	0	0	0	0	0	0	0	1	1	1	...	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	
4	1	1	0	0	0	0	0	1	0	0	...	0	0	0	

5 rows × 133 columns

Activity 2: Visual Analysis

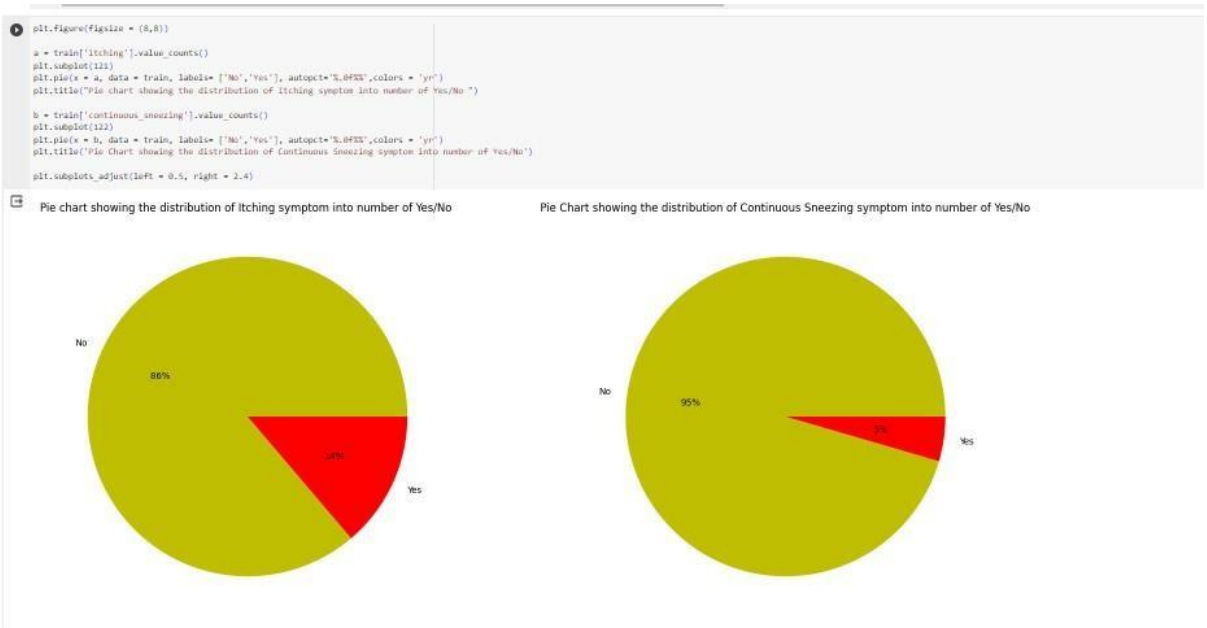
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate Analysis:

In simple words, univariate analysis is understanding the data with a single feature. We have displayed three different types of graphs and plots.

For simple visualizations we can use the matplotlib.pyplot library.

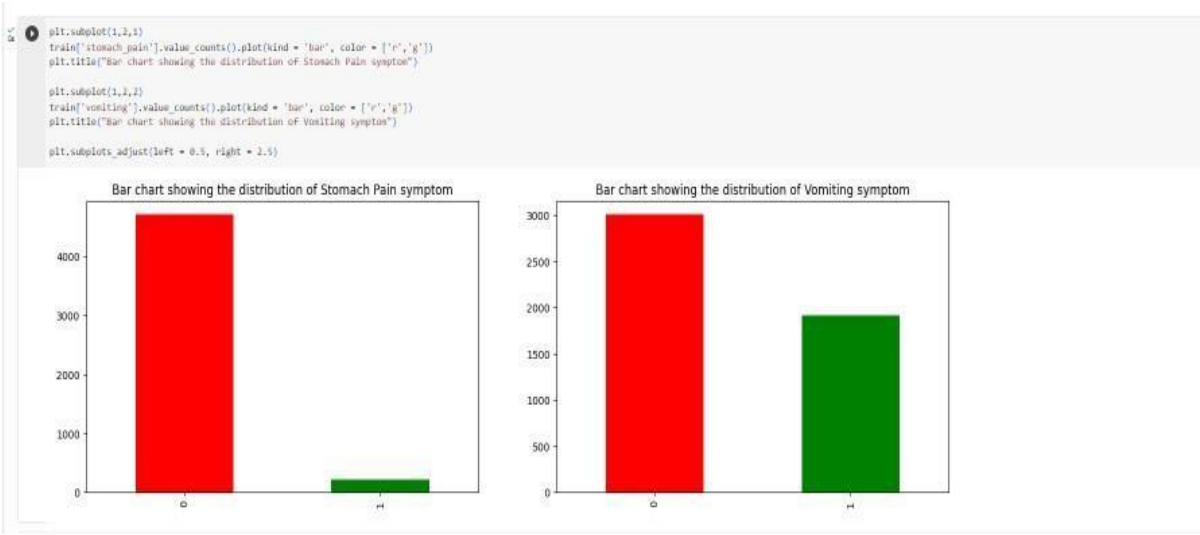
```
plt.figure(figsize = (8,8))

a = train['itching'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'yr')
plt.title("Pie chart showing the distribution of itching symptom into number of Yes/No ")

b = train['continuous_sneezing'].value_counts()
plt.subplot(122)
plt.pie(x = b, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'yr')
plt.title('Pie Chart showing the distribution of Continuous Sneezing symptom into number of Yes/No')

plt.subplots_adjust(left = 0.5, right = 2.4)
```



Here the plt.figure() command is used to determine the size of the plot. Then we divide the space for 2 pie plots.

The pie plot on the left shows the different values distribution in the Itching column. It shows that there are 86% observations where the itching symptom has value 0 and there are 14% observations where the itching symptom has value 1.

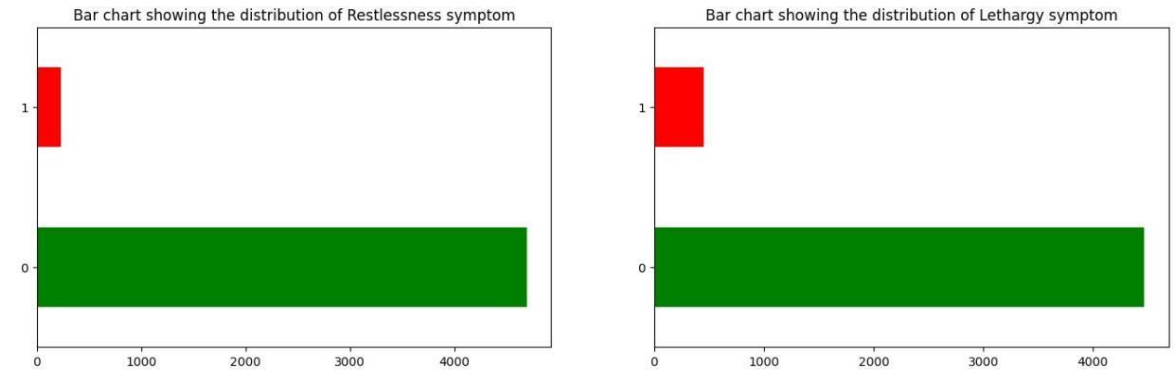
The pie plot on the right shows the different values distribution in the continuous_sneezing column. It shows that there are 95% observations where the continuous_sneezing symptom has value 0 and there are 5% observations where the continuous_sneezing symptom has value 1.



Here we have plotted 2 bar graphs. These bar graphs can be plotted without using any external library. We divide the plot into two subplots using subplot function from the matplotlib.pyplot library. We have plotted the bar graph using the inbuilt plot function in python.

The bar graph on the left shows the distribution of stomach pain symptom values. We can see that the 0 value has count of around 4700 and the 1 value has count of around 400. The graph on the right shows the distribution of vomiting symptom values. We can see that the 0 value has count of around 3000 and the 1 value has count of around 2000.

```
plt.subplot(1,2,1)
train['restlessness'].value_counts().plot(kind = 'barh', color = ['g','r'])
plt.title("Bar chart showing the distribution of Restlessness symptom")

plt.subplot(1,2,2)
train['lethargy'].value_counts().plot(kind = 'barh', color = ['g','r'])
plt.title("Bar chart showing the distribution of Lethargy symptom")

plt.subplots_adjust(left = 0.5, right = 2.5)
```



Here we have plotted 2 horizontal bar graphs. These bar graphs can be plotted without using any external library. We divide the plot into two subplots using subplot function from the matplotlib.pyplot library. We have plotted the bar graph using the inbuilt plot function in python.

The bar graph on the left shows the distribution of restlessness symptom values. We can see that the 0 value has count of around 4800 and the 1 value has count of around 300.

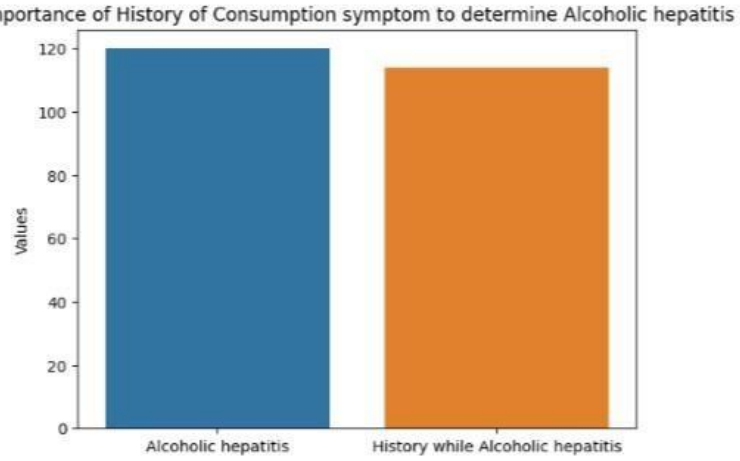
The graph on the right shows the distribution of vomiting symptom values. We can see that the 0 value has count of around 4500 and the 1 value has count of around 400.

Activity 2.2: Bivariate Analysis:

To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between prognosis where the values are Fungal Infection and Itching symptom.

```
a = len(train[train['prognosis'] == 'Alcoholic hepatitis'])
b = len(train[(train['history_of_alcohol_consumption'] == 1) & (train['prognosis'] == 'Alcoholic hepatitis')])
fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Alcoholic hepatitis','History while Alcoholic hepatitis'])

sns.barplot(data = fi, x = fi.index, y = fi['Values'])
plt.title('Importance of History of Consumption symptom to determine Alcoholic hepatitis')
```

```
Text(0.5, 1.0, 'Importance of History of Consumption symptom to determine Alcoholic hepatitis')
```



Here we use Boolean Indexing to filter out values from the prognosis column where the values are ‘Fungal Infection’. These observations are stored in variable ‘a’. Alse we filter out values from the prognosis where values are ‘Fungal Infection’ and also the values of Itching variable are 1. From the plot we can see that when there is Fungal Infection there is a high chance that the Itching column has value 1. There are 120 values where the value are fungal infection and there are 104 values where the value of itching column is 1. This shows that when there is a fungal infection there is a high chance that there is itching as a symptom.

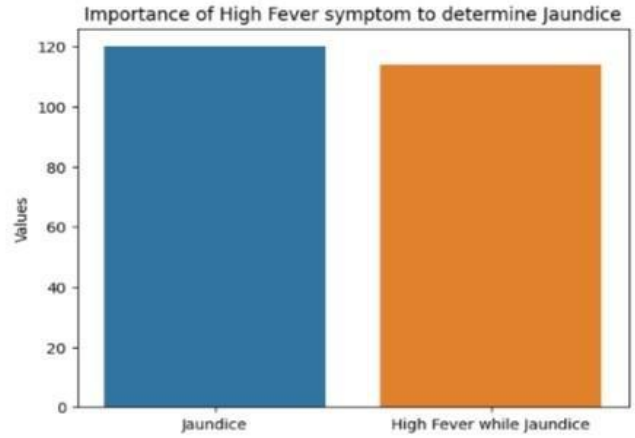
Next we have also seen the relationship between prognosis when the disease is Tuberculosis and the symptom yellowing_of_eyes. . From the plot we can see that when there is

Tuberculosis there is a high chance that the yellowing of eyes column has value 1. There are 120 values where the value is Tuberculosis and there are 119 values where the value of

yellowing_of_eyes column is 1. This shows that when there is tuberculosis there is a high chance that there is yellowing_of_eyes as a symptom.

```
[82] a = len(train[train['prognosis'] == 'Jaundice'])
     b = len(train[(train['high_fever'] == 1) & (train['prognosis'] == 'Jaundice')])
     fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Jaundice','High Fever while Jaundice'])

     sns.barplot(data = fi, x = fi.index, y = fi['Values'])
     plt.title('Importance of High Fever symptom to determine Jaundice')
```

```
Text(0.5, 1.0, 'Importance of High Fever symptom to determine Jaundice')
```



Activity 2.3: Multivariate Analysis

In multivariate analysis we try to find the relation between multiple features. This can be done primarily with the help of Correlation matrix.

```
corr = train.corr()
corr.style.background_gradient('coolwarm')
```

```
<ipython-input-71-92accbbdd853>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr = train.corr()
```

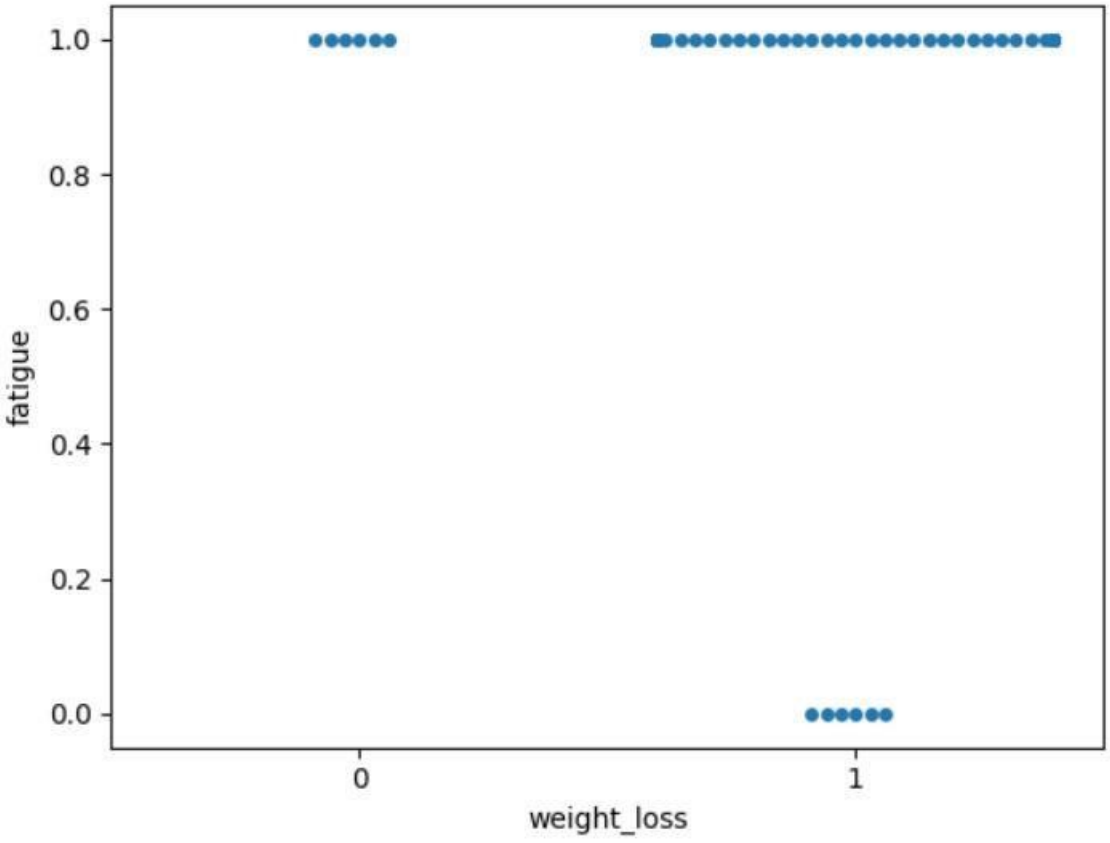
	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	burning_micturition	spotting_urination	fatigue	weight_gain	anxiety	cold_hands_and
itching	1.000000	0.318158	0.326439	-0.086906	-0.059893	-0.175905	-0.160650	0.202850	-0.086906	-0.059893	-0.059893	-0.057763	0.207896	0.350585	0.069744	-0.061573	-0.061573	-0
skin_rash	0.318158	1.000000	0.298143	-0.094786	-0.065324	-0.029324	0.171134	0.161784	-0.094786	-0.065324	-0.065324	-0.225046	0.166507	0.298143	-0.105248	-0.067156	-0.067156	-0
nodal_skin_eruptions	0.326439	0.298143	1.000000	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
continuous_sneezing	-0.086906	-0.094786	-0.032566	1.000000	0.608981	0.446238	-0.087351	-0.047254	-0.047254	-0.032566	-0.032566	-0.173459	-0.046581	-0.032566	0.041755	-0.033480	-0.033480	-0
shivering	-0.059893	-0.065324	-0.022444	0.608981	1.000000	0.295332	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
chills	-0.175905	-0.029324	-0.065917	0.446238	0.295332	1.000000	-0.004688	-0.095646	-0.095646	-0.065917	-0.065917	0.144263	-0.094285	-0.065917	0.269437	-0.067765	-0.067765	-0
joint_pain	-0.160650	0.171134	-0.060200	-0.087351	-0.060200	-0.004688	1.000000	-0.087351	-0.087351	-0.060200	-0.060200	0.199921	-0.086108	-0.060200	0.066652	-0.061889	-0.061889	-0
stomach_pain	0.202850	0.161784	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	1.000000	0.433917	0.649078	-0.032566	0.031406	0.412239	0.608981	-0.174797	-0.033480	-0.033480	-0
acidity	-0.086906	-0.094786	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	0.433917	1.000000	0.608981	-0.032566	0.019355	-0.046581	-0.032566	-0.174797	-0.033480	-0.033480	-0
ulcers_on_tongue	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	0.649078	0.608981	1.000000	-0.022444	0.153603	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
muscle_wasting	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	1.000000	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
vomiting	-0.057763	-0.225046	-0.119543	-0.173459	-0.119543	0.144263	0.199921	0.031406	0.019355	0.153603	-0.119543	1.000000	-0.170990	-0.119543	0.008883	-0.122896	0.176385	-0
burning_micturition	0.207896	0.166507	-0.032103	-0.046581	-0.032103	-0.094285	-0.086108	0.412239	-0.046581	-0.032103	-0.032103	-0.170990	1.000000	0.617879	-0.172308	-0.033003	-0.033003	-0
spotting_ urination	0.350585	0.298143	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	0.608981	-0.032566	-0.022444	-0.022444	-0.119543	0.617879	1.000000	-0.120465	-0.023073	-0.023073	-0
fatigue	0.069744	-0.105248	-0.120465	0.041755	-0.120465	0.269437	0.066652	-0.174797	-0.174797	-0.120465	-0.120465	0.008883	-0.172308	-0.120465	1.000000	0.158337	0.174936	0
weight_gain	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.158337	1.000000	-0.023720	0
anxiety	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	0.176385	-0.033003	-0.023073	0.174936	-0.023720	1.000000	-0
cold_hands_and_feets	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.158337	0.946120	-0.023720	1
mood_swings	-0.088129	-0.096120	-0.033025	-0.047919	-0.033025	-0.096992	-0.088581	-0.047919	-0.047919	-0.033025	-0.033025	-0.175900	-0.047237	-0.033025	0.238505	0.660111	-0.033951	0
weight_loss	0.091830	-0.139363	-0.047882	-0.069477	-0.047882	0.064721	-0.128431	-0.069477	-0.069477	-0.047882	-0.047882	0.055501	-0.068488	-0.047882	0.363027	-0.048224	-0.048224	-0
restlessness	-0.088129	-0.096120	-0.033025	-0.047919	-0.033025	-0.096992	-0.088581	-0.047919	-0.047919	-0.033025	-0.033025	-0.175900	-0.047237	-0.033025	0.250384	-0.033951	-0.033951	-0
lethargy	0.311436	0.067246	-0.047882	-0.069477	-0.047882	-0.140627	-0.128431	-0.069477	-0.069477	-0.047882	-0.047882	-0.255033	-0.068488	-0.047882	0.354415	0.453929	-0.049224	0
patches_in_throat	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	0.886395	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
irregular_sugar_level	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.174936	-0.023720	-0.023720	-0
cough	-0.143855	-0.156900	-0.053907	0.253730	-0.053907	0.402430	-0.144592	0.253730	0.235289	0.390220	-0.053907	-0.012314	-0.077106	-0.053907	0.259362	-0.055419	-0.055419	-0
high_fever	0.037389	0.117059	-0.092690	0.101860	-0.092690	0.541093	0.034990	-0.134495	-0.134495	-0.092690	0.223537	0.115678	-0.132580	-0.092690	0.412236	-0.095290	-0.095290	-0
sunken_eyes	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	0.153603	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0
breathlessness	-0.126846	-0.138350	-0.047534	-0.068972	-0.047534	0.273541	-0.127498	-0.068972	-0.068972	-0.047534	-0.047534	0.041853	-0.067990	-0.047534	0.204000	-0.048867	-0.048867	-0

As we have 131 columns which have numerical values the correlation matrix is of dimensions 131 X 131. These many features can only be parsed by scrolling. From the correlation matrix we try to remove the values which are highly correlated with each other. When 2 values are highly correlated with each other, we can only remove one of them. We remove columns where the correlation between the columns is above 0.9

```
[72] train.drop(['weight_gain','cold_hands_and_feets','anxiety','irregular_sugar_level',
                 'yellow_urine','acute_liver_failure','swelling_of_stomach',
                 'drying_and_tingling_lips','continuous_feel_of_urine',
                'internal_itching','polyuria','mood_swings','receiving_unsterile_injections',
                'stomach_bleeding','prominent_veins_on_calf','loss_of_smell','throat_irritation',
                'redness_of_eyes','sinus_pressure','runny_nose','pain_during_bowel_movements',
                'pain_in_anal_region','cramps','bruising','enlarged_thyroid','brittle_nails',
                'swollen_extremeties','slurred_speech','distention_of_abdomen','fluid_overload.1',
                'skin_peeling','silver_like_dusting','small_dents_in_nails','blister',
                'red_sore_around_nose','bloody_stool','swollen_blood_vessels','hip_joint_pain',
                'painful_walking','spinning_movements','altered_sensorium','toxic_look_(typhos)'],axis =1, inplace = True)
```

We can see the columns that are removed due to high correlation.

For multivariate analysis we will also plot a swarmplot of weightloss and fatigue column where the prognosis column is Tuberculosis.



From this swarm plot we can see that for Tuberculosis disease, there is no observation when the fatigue and weight loss is 0. There are some cases when there is only either of the two, but for Tuberculosis there is a high chance that the patient will have fatigue and weight loss as symptoms.

Preprocessing of Test data

The preprocessing needs to be done for the test data. We can create a function for test data preprocessing which will only leave us with the required features. This function will contain all the steps which we have done for the training data.

```
[88] def data_preprocessing(data):
        data.drop(['fluid_overload','weight_gain','cold_hands_and_feets','anxiety','irregular_sugar_level',
               'yellow_urine','acute_liver_failure','swelling_of_stomach',
               'drying_and_tingling_lips','continuous_feel_of_urine',
              'internal_itching','polyuria','mood_swings','receiving_unsterile_injections',
              'stomach_bleeding','prominent_veins_on_calf','loss_of_smell','throat_irritation',
              'redness_of_eyes','sinus_pressure','runny_nose','pain_during_bowel_movements',
              'pain_in_anal_region','cramps','bruising','enlarged_thyroid','brittle_nails',
              'swollen_extremeties','slurred_speech','distention_of_abdomen','fluid_overload.1',
              'skin_peeling','silver_like_dusting','small_dents_in_nails','blister',
              'red_sore_around_nose','bloody_stool','swollen_blood_vessels','hip_joint_pain',
              'painful_walking','spinning_movements','altered_sensorium','toxic_look_(typhos)'],axis =1, inplace = True)
        return data
```

This function drops all the columns which needs to be dropped.

```
test = data_preprocessing(test)
```

Here we call the function for the test data.

Activity 2.5: Split data into training, validation and testing data

We have training and testing data given separately. We further split the training data into training and validation data. This validation data can be used for hyper parameter tuning.

We first need to separate the features and the target variable. The features are used to predict the target variable.

```
X = train.drop('prognosis',axis = 1)
y = train.prognosis
```

We split the training data into features(X) and target variable(y).

```
X_test = test.drop('prognosis',axis = 1)
y_test = test.prognosis
```

Here we split the test data into features(X_test) and the corresponding target variables(y_test)

Now we need to split the training data into training and validation data. It can be done using the following command.

```
X_train, X_val, y_train, y_val = train_test_split(X,y,test_size = 0.2)
```

We have kept 80 % data for training and 20% is used for validation.

Milestone 4: Model Building

Activity 1: Creating a function for model evaluation

We will be creating multiple models and then testing them. It will reduce our monotonous task if we directly write a function for model evaluation.

```
def model_evaluation(classifier):
    y_pred = classifier.predict(X_val)
    yt_pred = classifier.predict(X_train)
    y_pred1 = classifier.predict(X_test)
    print('The Training Accuracy of the algorithm is ', accuracy_score(y_train, yt_pred))
    print('The Validation Accuracy of the algorithm is ', accuracy_score(y_val, y_pred))
    print('The Testing Accuracy of the algorithm is', accuracy_score(y_test, y_pred1))
    return [(accuracy_score(y_train, yt_pred)), (accuracy_score(y_val, y_pred)), (accuracy_score(y_test, y_pred1))]
```

This function will show the accuracies of prediction of model for training, validation and testing data. It will also the return those accuracies.

Activity 2: Training and testing the models using multiple algorithms

Now that we have clean data, a function for evaluation it is time to build models to train the data. For this project we will be using 4 different classification algorithms to build our models. The best model will be used for prediction.

Activity 2.1: K Nearest Neighbors Model

A variable is created with name knn which has KNeighborsClassifier() algorithm initialised in it. The knn model is trained using the .fit() function. The model is trained on the X_train and y_train data that is the training features and training target variables. This model is then given to the model_evaluation function to check its performance.

```
[94] Knn = KNeighborsClassifier(n_neighbors=7)
     Knn.fit(X_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

```
Knn_results = model_evaluation(Knn)
```

```
The Training Accuracy of the algorithm is  1.0
The Validation Accuracy of the algorithm is  1.0
The Testing Accuracy of the algorithm is 1.0
```

Here we can see that the model has achieved 100% accuracies on training, validation as well as testing data. As the accuracies are high, there is no need for hyperparameter tuning. The results are stored in a variable named knn_results.

Activity 2.2: SVM Model

A variable is created with name svm which has SVC() algorithm initialised in it. The svm model is trained using the .fit() function. The model is trained on the X_train and y_train data that is the training features and training target variables. This model is then given to the model_evaluation function to check its performance.

```
svm = SVC(C=1)
svm.fit(X_train, y_train)
```

```
SVC
SVC(C=1)
```

```
svm_results = model_evaluation(svm)
```

```
The Training Accuracy of the algorithm is  1.0
The Validation Accuracy of the algorithm is  1.0
The Testing Accuracy of the algorithm is 1.0
```

Here we can see that the model has achieved 100% accuracies on training, validation as well as testing data. As the accuracies are high, there is no need for hyperparameter tuning. The results are stored in a variable named svm_results.

Activity 2.3: Decision Tree Model

A variable is created with name dtc which has DecisionTreeClassifier() algorithm initialised in it with a parameter max_features set to 10. The dtc model is trained using the .fit() function. The model is trained on the X_train and y_train data that is the training features and training target variables. This model is then given to the model_evaluation function to check its performance.

```
[98] dtc = DecisionTreeClassifier(max_features= 10)
     dtc.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_features=10)
```

```
dtc_results = model_evaluation(dtc)
```

```
The Training Accuracy of the algorithm is  1.0
The Validation Accuracy of the algorithm is  1.0
The Testing Accuracy of the algorithm is 0.9761904761904762
```

Here we can see that the model has achieved 100% accuracies on training and validation data . It has achieved 97.6% accuracy for testing data. As the accuracies are high, there is no need for hyperparameter tuning. The results are stored in a variable named dtc_results.

Activity 2.4: Random Forest Model

Random Forest Classifier is a Bagging model which utilises multiple decision trees and takes their aggregate to give a prediction. A variable is created with name rfc which has RandomForestClassifier() algorithm initialised in it with a parameter max_depth set to 13. The rfc model is trained using the .fit() function. The model is trained on the X_train and y_train data that is the training features and training target variables. This model is then given to the model_evaluation function to check its performance.

```
[100] rfc = RandomForestClassifier(max_depth = 13)
      rfc.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=13)
```

```
rfc_results = model_evaluation(rfc)
```

```
The Training Accuracy of the algorithm is  0.9979674796747967
The Validation Accuracy of the algorithm is  0.9959349593495935
The Testing Accuracy of the algorithm is 0.9761904761904762
```

Here we can see that the model has achieved 100% accuracies on training and validation data . It has achieved 97.6% accuracy for testing data. As the

accuracies are high, there is no need for hyperparameter tuning. The results are stored in a variable named rfc_results.

Milestone 5 : Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with Multiple Evaluatiton metrics

The data has 41 features, hence it is difficult to make confusion matrix as the dimensions of confusion matrix will be 41 X 41. We can check accuracy to test the model. We have already values of the training, validation, and test accuracies of various models. We can put them in a table and then check for the best model.

```
[102] results = pd.DataFrame(data = [Knn_results, svm_results, dtc_results, rfc_results],
                             columns= ['Training Accuracy','Validation Accuracy', 'Testing Accuracy'],
                             index = ['K Nearest Neighbors Classifier','Support Vector Machines',
                                      'Decision Trees Classifier', 'Random Forest Classifier'])
```

```
results
```

	Training Accuracy	Validation Accuracy	Testing Accuracy
K Nearest Neighbors Classifier	1.000000	1.000000	1.00000
Support Vector Machines	1.000000	1.000000	1.00000
Decision Trees Classifier	1.000000	1.000000	0.97619
Random Forest Classifier	0.997967	0.995935	0.97619

From the table we can see that KNN and SVM models perform the best.

Activity 2: Comparing model accuracy before and after applying *hyperparameter tuning*

As the accuracies are already so high, we need not do hyperparameter tuning for the models.

Activity 3: Comparing Model accuracy for different number of features.

Currently the training data has 90 features, which are a high number. If we need to reduce the number of features, we need to check the accuracies for various number of features.

We can check the feature importance using the Random Forest Classifier model.

In the figure below we have created a dictionary with the column names as indexes and the values as their feature importance. Much importance is not assigned to any feature. It is distributed among all the features.

We will keep some number of features for training and check for the accuracy. This process will be repeated a number of times.

```
a = rfc.feature_importances_
```

```
col = X.columns
```

```
feat_imp = {}
for i, j in zip(a,col):
    feat_imp[j] = i
```

```
feat_imp
```

```
{'itching': 0.007035271026166255,
 'skin_rash': 0.005546735268193718,
 'nodal_skin_eruptions': 0.0034850171591946796,
 'continuous_sneezing': 0.012342758639018337,
 'shivering': 0.008084128709252784,
 'chills': 0.010083620999879972,
 'joint_pain': 0.011319422632214958,
 'stomach_pain': 0.005488340012957586,
 'acidity': 0.005028719772774104,
 'ulcers_on_tongue': 0.005781237995779322,
 'muscle_wasting': 0.007314668993771232,
 'vomiting': 0.007089514040950214,
 'burning_micturition': 0.0028010374188746356,
 'spotting_ urination': 0.006081689102405897,
 'fatigue': 0.007655168915843125,
 'weight_loss': 0.007414577893906033,
 'restlessness': 0.00592602339383772,
 'lethargy': 0.010049256294771523,
 'patches_in_throat': 0.002047690425438012,
 'cough': 0.008943282386146284,
 'high_fever': 0.011371843916122774,
 'sunken_eyes': 0.010959602547064526,
 'breathlessness': 0.012000935068414743,
 'sweating': 0.008657797247017648,
 'dehydration': 0.012245091433717189,
 'indigestion': 0.007253665608297292
```

We get a dictionary named feat_imp with 90 column names and their feature importance.

We will drop columns which have very less feature importance.

Let us create a for loop which will train the model and give out the accuracy.

```
rfc_results = []
knn_results = []
```

```
for main in [0.020,0.018,0.016,0.014,0.012,0.01,0.008]:
    to_drop = []
    for i,j in zip(feat_imp.keys(),feat_imp.values()):
        if j < main:
            to_drop.append(i)

    X_new = X.drop(to_drop,axis = 1)
    y_new = y
    X1_train, X1_val, y1_train, y1_val = train_test_split(X_new, y_new, test_size=0.2)
    X1_test = X_test.drop(to_drop,axis = 1)
    y1_test = y_test
    rfc_new = RandomForestClassifier()
    rfc_new.fit(X1_train, y1_train)
    temp1 = model_evaluation1(X1_train.shape[1], rfc_new)
    rfc_results.append(temp1)
    knn_new = KNeighborsClassifier()
    knn_new.fit(X1_train, y1_train)
    temp2 = model_evaluation1(X1_train.shape[1],knn_new)
    knn_results.append(temp2)
```

Here we create 2 lists for 2 models, knn and random forest classifier.

The for loop will iterate over values given in the list one be one. The first value will be 0.020 and the last will be 0.008
There is a to_drop list created.

If the feature_importance is below threshold then the column name will be added to the to_drop list.

The columns whose name is in the to_drop list will be dropped.

The new data will be split into features and target variable. Further they will be split into training, validation and testing data.

Random Forest Classifier model will be trained and its accuracy will be stored in the list.

Knn model will be trained and its accuracy will be stored in the list.

This process will go on till all the values for i are iterated.

We then plot a table using the number of features and accuracies.

```
randomf = pd.DataFrame(data = rfc_results,columns=['Number of features','Training Accuracy','Testing Accuracy'])
```

```
randomf
```

	Number of features	Training Accuracy	Testing Accuracy
0	10	0.267276	0.261905
1	14	0.366108	0.357143
2	19	0.457317	0.452381
3	22	0.530234	0.523810
4	33	0.739329	0.738095
5	46	0.887957	0.880952
6	56	0.920478	0.904762

This is the table for random forest Classifier for various features. We can see that as the number of features go on increasing, the accuracies increase.

```
[113] knn_table = pd.DataFrame(data = knn_results,columns=['Number of features','Training Accuracy','Testing Accuracy'])
```

```
knn_table
```

	Number of features	Training Accuracy	Testing Accuracy
0	10	0.264482	0.261905
1	14	0.364329	0.357143
2	19	0.452998	0.452381
3	22	0.528963	0.523810
4	33	0.734502	0.738095
5	46	0.883384	0.880952
6	56	0.917937	0.904762

This is the table for knn model for various number of features. We can see that as the number of features go on increasing, the accuracies increase.

Activity 4: Building Model with appropriate features

From the above result tables, we can see that the accuracy does not change much from 45 features to 62 features. Hence we will choose 45 features for our training.

```
[116] len(to_drop)
```

```
43
```

```
[117] X_new = X.drop(to_drop,axis = 1)
      y_new = y
```

```
X_new.head()
```

	continuous_sneezing	chills	joint_pain	lethargy	high_fever	sunken_eyes	breathlessness	dehydration	nausea	pain_behind_the
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

5 rows × 46 columns

We drop 44 features. Create new datasets for features and their labels. As we can see in the figure above we are left with 45 features now. We will build a Random Fores Classifier on the new data and check for the accuracies. As we can see in the figure below our model has achieved test accuracy of 97.6 % which is quite good for the number of features. Previously for 90 features we had similar accuracy for Random Forest Classifier. This states that there were many features which were not contributing much to our model.

```
[119] X1_train, X1_val, y1_train, y1_val = train_test_split(X_new, y_new, test_size=0.2)
      X1_test = X_test.drop(to_drop,axis = 1)
      y1_test = y_test
```

```
[120] rfc_new = RandomForestClassifier()
      rfc_new.fit(X1_train, y1_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = rfc_new.predict(X1_val)
yt_pred = rfc_new.predict(X1_train)
y_pred1 = rfc_new.predict(X1_test)
print('The Training Accuracy of the algorithm is ', accuracy_score(y1_train, yt_pred))
print('The Validation Accuracy of the algorithm is ', accuracy_score(y1_val, y_pred))
print('The Testing Accuracy of the algorithm is', accuracy_score(y1_test, y_pred1))
```

```
The Training Accuracy of the algorithm is  0.8879573170731707
The Validation Accuracy of the algorithm is  0.8689024390243902
The Testing Accuracy of the algorithm is 0.8809523809523809
```

We will also train the model for KNN algorithm as KNN algorithm tends to perform better in such cases.

```
[122] knn_new = KNeighborsClassifier()
      knn_new.fit(X1_train, y1_train)
```

```
KNeighborsClassifier()
```

```
y_pred = knn_new.predict(X1_val)
yt_pred = knn_new.predict(X1_train)
y_pred1 = knn_new.predict(X1_test)
print('The Training Accuracy of the algorithm is ', accuracy_score(y1_train, yt_pred))
print('The Validation Accuracy of the algorithm is ', accuracy_score(y1_val, y_pred))
print('The Testing Accuracy of the algorithm is', accuracy_score(y1_test, y_pred1))
```

```
The Training Accuracy of the algorithm is  0.8879573170731707
The Validation Accuracy of the algorithm is  0.8689024390243902
The Testing Accuracy of the algorithm is 0.9047619047619048
```

After training the knn model we check the accuracies. Our model has achieved 97.6 % accuracy for the test data.

To confirm let us check the compare our predicted results with the actual values.

```
test.join(pd.DataFrame(y_pred1,columns=["predicted"]))[["prognosis","predicted"]]
```

	prognosis	predicted
0	Fungal infection	Fungal infection
1	Allergy	Allergy
2	GERD	GERD
3	Chronic cholestasis	Chronic cholestasis
4	Drug Reaction	Fungal infection
5	Peptic ulcer diseae	Peptic ulcer diseae
6	AIDS	AIDS
7	Diabetes	Diabetes
8	Gastroenteritis	Gastroenteritis
9	Bronchial Asthma	Bronchial Asthma
10	Hypertension	Hypertension
11	Migraine	Migraine
12	Cervical spondylosis	Cervical spondylosis
13	Paralysis (brain hemorrhage)	Fungal infection

As we can see above that the values our model has predicted are same as the actual values.
This shows that our model is performing good.

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future**.**

After checking the performance, we decide to save the knn model built with 45 features.

```
pickle.dump(knn_new, open('model.pkl','wb'))
```

We save the model using the pickle library into a file named model.pkl

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks

- Building HTML Pages ● Building server-side script
- Run the web application

Activity 2.1: Building HTML pages:

For this project we create three HTML files namely

- Index.html
- Details.html
- Results.html

And we will save them in the templates folder.

Activity 2.2: Build Python code

Create a new app.py file which will be store in the Flask folder.

Import the necessary Libraries.

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

This code first loads the saved Linear Regression model from the "bodyfat.pkl" file using the "pickle.load()" method. The "rb" parameter indicates that the file should be opened in binary mode to read data from it.

After loading the model, the code creates a new Flask web application object named "app" using the Flask constructor. The "name" argument tells Flask to use the current module as the name for the application.

```
model = pickle.load(open('model.pkl','rb'))
app = Flask(__name__)
```

This code sets up a new route for the Flask web application using the "@app.route()" decorator. The route in this case is the root route "/", which is the default route when the website is accessed.

The function "home()" is then associated with this route. When a user accesses the root route of the website, this function is called.

The "render_template()" method is used to render an HTML template named "index.html". The “index.html” is the home page.

```
@app.route("/")
def home():
    return render_template('index.html')
```

The route in this case is "/details". When a user accesses the "/predict" route of the website, this function is “index()” called. The "render_template()" method is used to render an HTML template named "details.html".

```
@app.route('/details')
def pred():
    return render_template('details.html')
```

This code sets up another route for the Flask web application using the "@app.route()" decorator. The route in this case is "/predict", and the method is set to GET and POST.

The function "predict()" is then associated with this route. In this function we create a list named col which has all the 45 column names that we have used in our model. In the details.html page we are going to take inputs from the user, which will be in the form of text.

The values are stored in request.form.values()

These values are stored in a list named inputt in the form of strings.

A list is created with 45 0s and stored in variable b.

In the for loop x will take values from 0 to 45.

Another for loop is written where y will iterate over the values given by the user as inputs.

If the name of the column which is at the x index in col list matches with the y from the inputt list then a 1 is stored at that index in the list b.

This list b is converted into an array and the shape of the array is changed to (1,45).

Then this array b is given to the model for prediction.

This prediction is returned to the results.html page using render_template()

```
@app.route('/predict',methods=['POST','GET'])
def predict():
    col=['itching', 'continuous_sneezing', 'shivering', 'joint_pain',
       'stomach_pain', 'vomiting', 'fatigue', 'weight_loss', 'restlessness',
       'lethargy', 'high_fever', 'headache', 'dark_urine', 'nausea',
       'pain_behind_the_eyes', 'constipation', 'abdominal_pain', 'diarrhoea',
       'mild_fever', 'yellowing_of_eyes', 'malaise', 'phlegm', 'congestion',
       'chest_pain', 'fast_heart_rate', 'neck_pain', 'dizziness',
       'puffy_face_and_eyes', 'knee_pain', 'muscle_weakness',
       'passage_of_gases', 'irritability', 'muscle_pain', 'belly_pain',
       'abnormal_menstruation', 'increased_appetite', 'lack_of_concentration',
       'visual_disturbances', 'receiving_blood_transfusion', 'coma',
       'history_of_alcohol_consumption', 'blood_in_sputum', 'palpitations',
       'inflammatory_nails', 'yellow_crust_ooze']
    if request.method=='POST':
        inputt = [str(x) for x in request.form.values()]

        b=[0]*45
        for x in range(0,45):
            for y in inputt:
                if(col[x]==y):
                    b[x]=1
        b=np.array(b)
        b=b.reshape(1,45)
        prediction = model.predict(b)
        prediction = prediction[0]
    return render_template('results.html', prediction_text="The probable diagnosis says it cou
```

Main Function:

This code sets the entry point of the Flask application. The function “app.run()” is called, which starts the Flask deployment server.

```
if __name__ == "__main__":
    app.run()
```

Activity 2.3: Run the Web Application

When you run the “app.py” file this window will open in the console or output terminal.

Copy the URL given in the form http://127.0.0.1:5000 and paste it in the browser.

```
In [1]: runfile('C:/Users/hp/SB/Disease Prediction/Flask/app.py', wdir='C:/Users/hp/SB/Disease
Prediction/Flask')
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

When we paste the URL in a web browser, our index.html page will open. It contains various sections in the header bar such as Home, Predict, About Model, Testimonials, FAQ, Contact. There is some information given on the web page about our model.

If you click on the Predict button on home page or in the header bar you will be redirected to the details.html page.

8. PERFORMANCE TESTING

8.1 Performance Metrics

1. Accuracy:- We have got training and testing accuracy as follows:
   Training accuracy:- 100%
   Testing accuracy:- 97.6%

```
results = pd.DataFrame(data = [knn_results, svm_results, dtc_results, rfc_results],
                       columns= ['Training Accuracy','Validation Accuracy', 'Testing Accuracy'],
                       index = ['K Nearest Neighbors Classifier','Support Vector Machines',
                                'Decision Trees Classifier', 'Random Forest Classifier'])
```

```
results
```

	Training Accuracy	Validation Accuracy	Testing Accuracy
K Nearest Neighbors Classifier	1.0	1.0	1.00000
Support Vector Machines	1.0	1.0	1.00000
Decision Trees Classifier	1.0	1.0	0.97619
Random Forest Classifier	1.0	1.0	0.97619

2. confusion matrix:

```
corr = train.corr()
corr.style.background_gradient('coolwarm')
```

```
<ipython-input-71-92accbbd8d53>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr = train.corr()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	burning_micturition	spotting_ urination	fatigue	weight_gain	anxiety	cold_hands_and
itching	1.000000	0.318158	0.326439	-0.086906	-0.059893	-0.175905	-0.160650	0.202850	-0.086906	-0.059893	-0.059893	-0.057763	0.207896	0.350585	0.069744	-0.061573	-0.061573	-0.
skin_rash	0.318158	1.000000	0.298143	-0.094786	-0.065324	-0.029324	0.171134	0.161784	-0.094786	-0.065324	-0.065324	-0.225046	0.166507	0.298143	-0.105248	-0.067156	-0.067156	-0.
nodal_skin_eruptions	0.326439	0.298143	1.000000	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
continuous_sneezing	-0.086906	-0.094786	-0.032566	1.000000	0.608981	0.446238	-0.087351	-0.047254	-0.047254	-0.032566	-0.032566	-0.173459	-0.046581	-0.032566	0.041755	-0.033480	-0.033480	-0.
shivering	-0.059893	-0.065324	-0.022444	0.608981	1.000000	0.295332	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
chills	-0.175905	-0.029324	-0.065917	0.446238	0.295332	1.000000	-0.004688	-0.095646	-0.095646	-0.065917	-0.065917	0.144263	-0.094285	-0.065917	0.269437	-0.067765	-0.067765	-0.
joint_pain	-0.160650	0.171134	-0.060200	-0.087351	-0.060200	-0.004688	1.000000	-0.087351	-0.087351	-0.060200	-0.060200	0.199921	-0.086108	-0.060200	0.066652	-0.061889	-0.061889	-0.
stomach_pain	0.202850	0.161784	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	1.000000	0.433917	0.649078	-0.032566	0.031406	0.412239	0.608981	-0.174797	-0.033480	-0.033480	-0.
acidity	-0.086906	-0.094786	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	0.433917	1.000000	0.608981	-0.032566	0.019355	-0.046581	-0.032566	-0.174797	-0.033480	-0.033480	-0.
ulcers_on_tongue	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	0.649078	0.608981	1.000000	-0.022444	0.153603	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
muscle_wasting	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	1.000000	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
vomiting	-0.057763	-0.225046	-0.119543	-0.173459	-0.119543	0.144263	0.199921	0.031406	0.019355	0.153603	-0.119543	1.000000	-0.170990	-0.119543	0.008883	-0.122896	0.176385	-0.
burning_micturition	0.207896	0.166507	-0.032103	-0.046581	-0.032103	-0.094285	-0.086108	0.412239	-0.046581	-0.032103	-0.032103	-0.170990	1.000000	0.617879	-0.172308	-0.033003	-0.033003	-0.
spotting_ urination	0.350585	0.298143	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	0.608981	-0.032566	-0.022444	-0.022444	-0.119543	0.617879	1.000000	-0.120465	-0.023073	-0.023073	-0.
fatigue	0.069744	-0.105248	-0.120465	0.041755	-0.120465	0.269437	0.066652	-0.174797	-0.174797	-0.120465	-0.120465	0.008883	-0.172308	-0.120465	1.000000	0.158337	0.174936	0.
weight_gain	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.158337	1.000000	-0.023720	0.
anxiety	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	0.176385	-0.033003	-0.023073	0.174936	-0.023720	1.000000	-0.
cold_hands_and_feets	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.158337	0.946120	-0.023720	1.
mood_swings	-0.088129	-0.096120	-0.033025	-0.047919	-0.033025	-0.096992	-0.088581	-0.047919	-0.047919	-0.033025	-0.033025	-0.175900	-0.047237	-0.033025	0.238505	0.660111	-0.033951	0
weight_loss	0.091830	-0.139363	-0.047882	-0.069477	-0.047882	0.064721	-0.128431	-0.069477	-0.069477	-0.047882	-0.047882	0.055501	-0.068488	-0.047882	0.363027	-0.049224	-0.049224	-0.
restlessness	-0.088129	-0.096120	-0.033025	-0.047919	-0.033025	-0.096992	-0.088581	-0.047919	-0.047919	-0.033025	-0.033025	-0.175900	-0.047237	-0.033025	0.250384	-0.033951	-0.033951	-0.
lethargy	0.311436	0.067246	-0.047882	-0.069477	-0.047882	-0.140627	-0.128431	-0.069477	-0.069477	-0.047882	-0.047882	-0.255033	-0.068488	-0.047882	0.354415	0.453929	-0.049224	0.
patches_in_throat	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	0.886395	-0.119543	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
irregular_sugar_level	-0.061573	-0.067156	-0.023073	-0.033480	-0.023073	-0.067765	-0.061889	-0.033480	-0.033480	-0.023073	-0.023073	-0.122896	-0.033003	-0.023073	0.174936	-0.023720	-0.023720	-0.
cough	-0.143855	-0.156900	-0.053907	0.253730	-0.053907	0.402430	-0.144592	0.253730	0.235289	0.390220	-0.053907	-0.012314	-0.077106	-0.053907	0.259362	-0.055419	-0.055419	-0.
high_fever	0.037309	0.117059	-0.092690	0.101860	-0.092690	0.541093	0.034990	-0.134495	-0.134495	-0.092690	0.223537	0.115678	-0.132580	-0.092690	0.412236	-0.095290	-0.095290	-0.
sunken_eyes	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032566	-0.022444	-0.022444	0.153603	-0.032103	-0.022444	-0.120465	-0.023073	-0.023073	-0.
breathlessness	-0.126848	-0.138350	-0.047534	-0.068972	-0.047534	0.273541	-0.127498	-0.068972	-0.068972	-0.047534	-0.047534	0.041853	-0.067990	-0.047534	0.204000	-0.048867	-0.048867	-0.

3. Classification report:-

```
{'itching': 0.014906774548182978,
 'skin_rash': 0.004126193310423079,
 'nodal_skin_eruptions': 0.00506477497511548,
 'continuous_sneezing': 0.011950815998563227,
 'shivering': 0.01482441556258809,
 'chills': 0.008051206993917127,
 'joint_pain': 0.015897929037963693,
 'stomach_pain': 0.010154098478372642,
 'acidity': 0.008624253805092569,
 'ulcers_on_tongue': 0.007019300375694471,
 'muscle_wasting': 0.007044463040109562,
 'vomiting': 0.010611899915403056,
 'burning_micturition': 0.002027624623287407,
 'spotting_ urination': 0.005808412722595892,
 'fatigue': 0.017693005776644993,
 'weight_loss': 0.01923436830804702,
 'restlessness': 0.015069798433680767,
 'lethargy': 0.010629532760213844,
 'patches_in_throat': 0.008631735490068643,
 'cough': 0.008821405970258043,
 'high_fever': 0.013592813181931977,
 'sunken_eyes': 0.005731713251463125,
 'breathlessness': 0.007567280001571607,
 'sweating': 0.006338800581064726,
 'dehydration': 0.006099054276102206,
 'indigestion': 0.009224539294501882,
 'headache': 0.012861930821546415,
 'yellowish_skin': 0.007336723298937289,
 'dark_urine': 0.018415064069200026,
 'nausea': 0.014561540637761647,
 'loss_of_appetite': 0.009769368779794086,
 'pain_behind_the_eyes': 0.014871557593585737,
 'back_pain': 0.00841000363805186,
 'constipation': 0.011967362534607 53,
 'abdominal_pain': 0.013280318144157168,
 'diarrhoea': 0.014779189226299838,
 'mild_fever': 0.021927764510919188,
 'yellowing_of_eyes': 0.018204452409490167,
 'swelled_lymph_nodes': 0.008573681085763575,
 'malaise': 0.013078554073886505,
 'blurred_and_distorted_vision': 0.006770164376288543,
 'phlegm': 0.01400451098904756,
 'congestion': 0.01937289284476267,
 'chest_pain': 0.014393320758590821,
 'weakness_in_limbs': 0.00976702056835587,
 'fast_heart_rate': 0.01188690006514303,
 'irritation_in_anus': 0.007755591217259018,
 'neck_pain': 0.018913501543935344,
 'dizziness': 0.0143951032078125,
 'obesity': 0.003141905550275227,
 'swollen_legs': 0.009795895605443533,
 'puffy_face_and_eyes': 0.01187214950410247,
 'excessive_hunger': 0.007874243501398424,
 'extra_marital_contacts': 0.008026799628225029,
 'knee_pain': 0.014124767000487558,
 'muscle_weakness': 0.016046959672476803,
 'stiff_neck': 0.005866774570023225,
 'swelling_joints': 0.009783513442988468,
 'movement_stiffness': 0.005626215130804082,
 'loss_of_balance': 0.006733960582116772,
 'unsteadiness': 0.0026082745448166972,
 'weakness_of_one_body_side': 0.0038488968022672766,
 'bladder_discomfort': 0.003796051105618112,
 'foul_smell_of urine': 0.0007868279473938345,
 'passage_of_gases': 0.02017000024832404,
 'depression': 0.009679212899605475,
 'irritability': 0.011646167243718744,
 'muscle_pain': 0.027414660942314328,
 'red_spots_over_body': 0.008798497858242695,
 'belly_pain': 0.02110988440489781,
 'abnormal_menstruation': 0.012071716901945423,
 'dischromic _patches': 0.006242536320347816,
```

```
'watering_from_eyes': 0.00502171219593155,
'increased_appetite': 0.02104011440346685,
'family_history': 0.009711015722333256,
'mucoid_sputum': 0.008652034180843426,
'rusty_sputum': 0.009284519206117154,
'lack_of_concentration': 0.016710436225726308,
'visual_disturbances': 0.011974912832908241,
'receiving_blood_transfusion': 0.022311538902411335,
'coma': 0.025957262084154543,
'history_of_alcohol_consumption': 0.013617771245498778,
'blood_in_sputum': 0.011726217210035533,
'palpitations': 0.013136893825550066,
'pus_filled_pimples': 0.004948586144275713,
'blackheads': 0.0010392924140262014,
'scurring': 0.0,
'inflammatory_nails': 0.018632696650225048,
'yellow_crust_ooze': 0.019126358240607655}
```

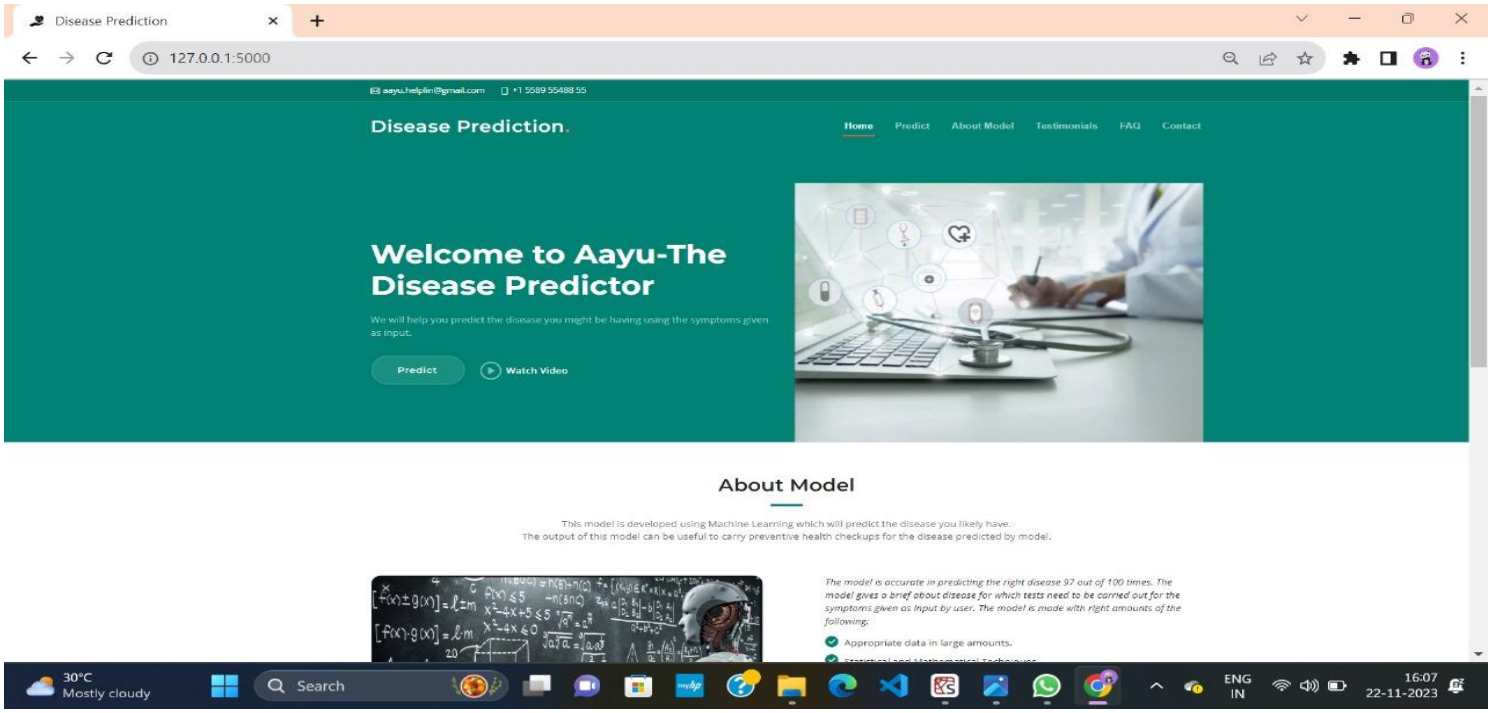
9. RESULTS

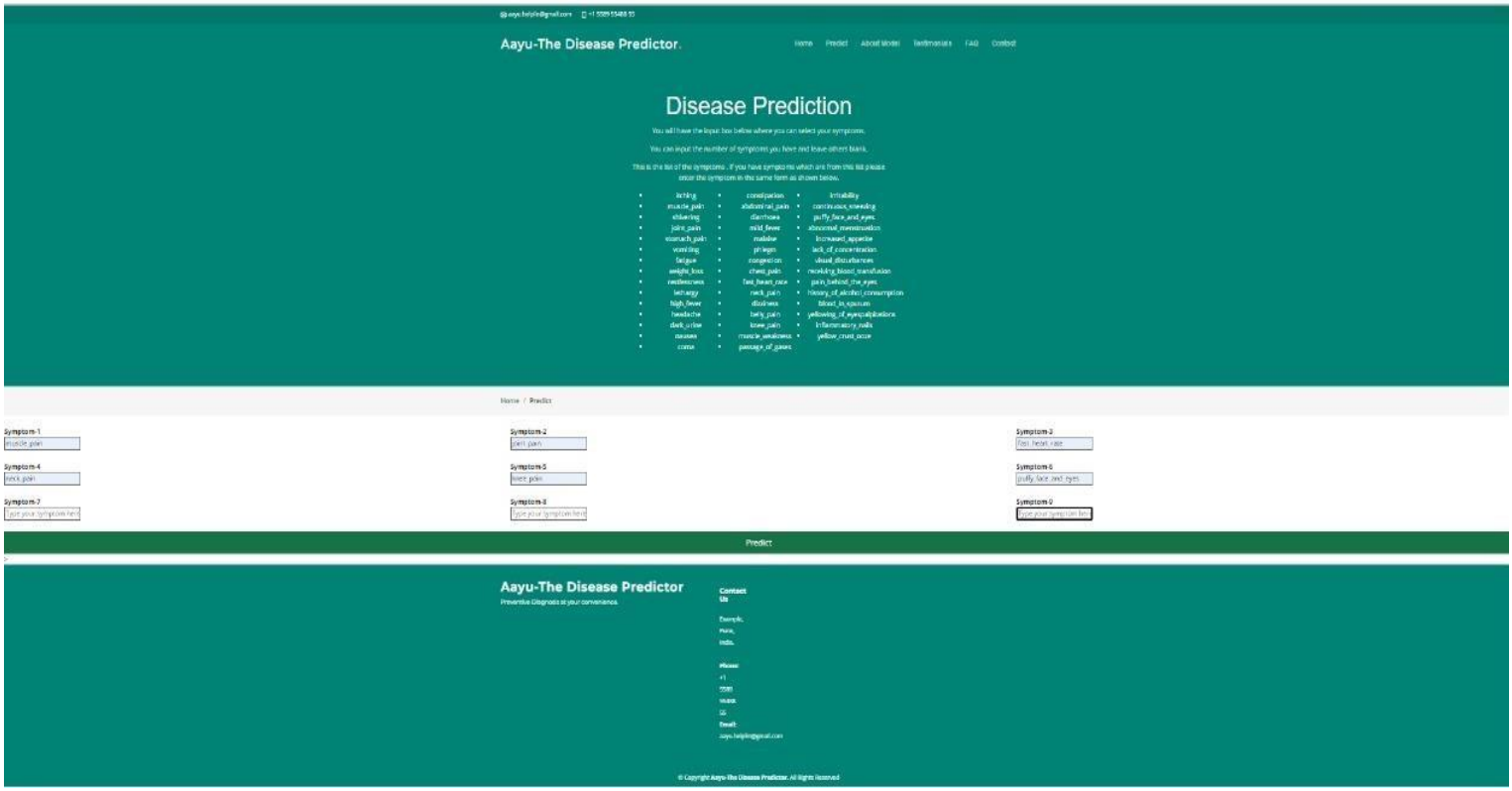
9.1 Output Screenshots -

```
test.join(pd.DataFrame(y_pred1,columns=["predicted"]))[["prognosis","predicted"]]
```

	prognosis	predicted
0	Fungal infection	Fungal infection
1	Allergy	Allergy
2	GERD	GERD
3	Chronic cholestasis	Chronic cholestasis
4	Drug Reaction	Fungal infection
5	Peptic ulcer diseae	Peptic ulcer diseae
6	AIDS	AIDS
7	Diabetes	Diabetes
8	Gastroenteritis	Gastroenteritis
9	Bronchial Asthma	Bronchial Asthma
10	Hypertension	Hypertension
11	Migraine	Migraine
12	Cervical spondylosis	Cervical spondylosis
13	Paralysis (brain hemorrhage)	Fungal infection

Outputs screenshots our webpage: Our Detials.html looks as shown below.



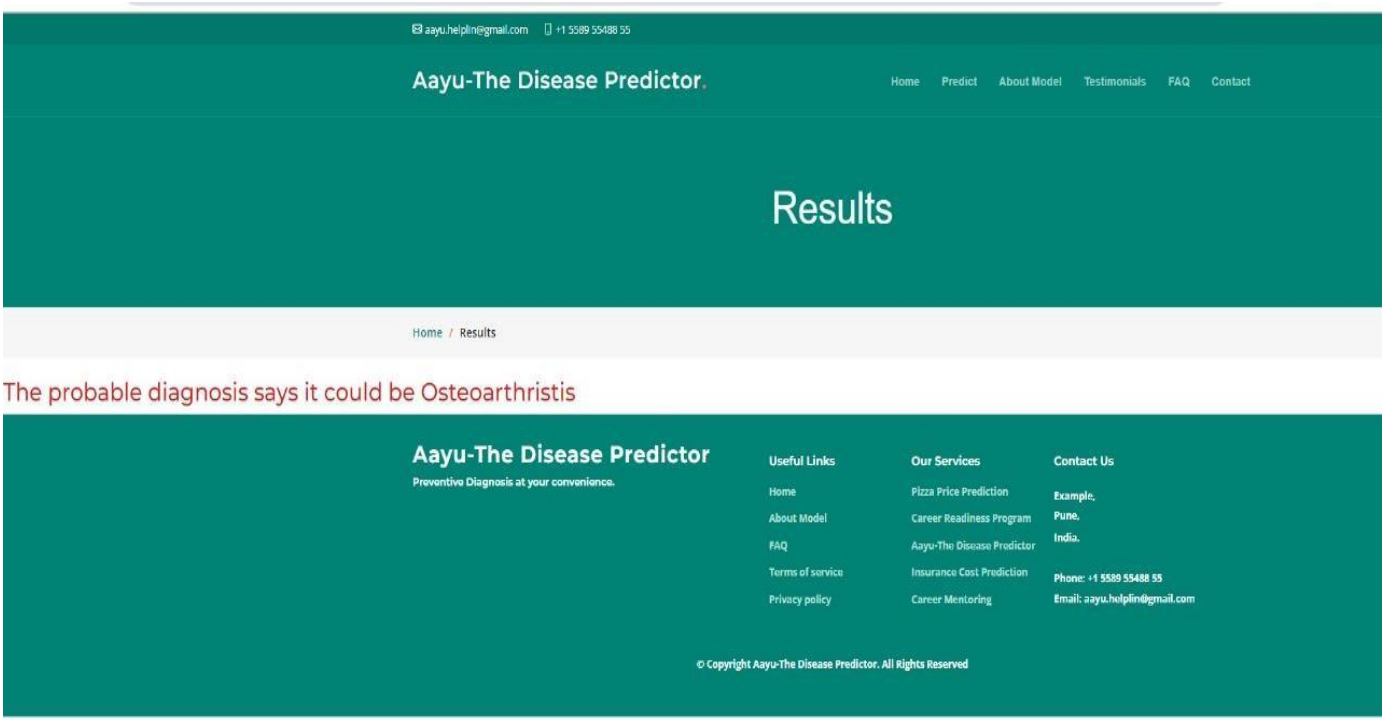


We are provided with 9 input fields where we can input our symptoms. We can fill all 9 boxes or can just fill 1. We are provided with a list of symptoms in the form of bullet points above.

We can input symptoms from above but they have to be in the same form as given in the list since they are the column names.

We will input some symptoms such as vomiting, fatigue, diarrhoea, shivering, high_fever and click on Predict button to get the output.

We will be redirected to the Results.html page once we click the Predict button.



The results say that there are high chances that patient has Bronchial Asthma based on the various symptoms we have given as input.

10. ADVANTAGES & DISADVANTAGES

Advantages:

Early Disease Detection:
Enables early detection of potential diseases based on user-input symptoms, allowing for timely intervention and treatment.

Accessibility:
Increases healthcare accessibility by providing a user-friendly web application for preliminary self-diagnosis, particularly valuable in areas with limited access to healthcare facilities.

Cost-Effective:
Offers a cost-effective alternative to traditional doctor visits, reducing the financial burden on individuals and healthcare systems.

Privacy-Focused:
Respects user privacy by not collecting personalized data, addressing concerns related to data security and confidentiality.

Online Consultations:
Facilitates online consultations with healthcare professionals, providing a collaborative approach to healthcare delivery.

Educational Component:
Integrates educational content to enhance health literacy, helping users better understand symptoms, diseases, and the importance of professional medical advice.

Disadvantages:

Data Limitations:
Challenges may arise due to the availability and quality of diverse datasets, potentially limiting the model's ability to accurately predict diseases across various demographics.

Algorithm Robustness:
The robustness of machine learning algorithms is crucial, and challenges may arise in developing models capable of handling a wide range of symptoms and diseases.

Ethical Considerations
Addressing potential biases in the model and ensuring transparency in decision-making processes are critical ethical considerations that require careful attention.

User Trust:
Building and maintaining user trust is challenging, especially in cases where the model's predictions might differ from traditional medical advice.

Professional Consultation Importance:
Balancing the encouragement of user self-care with the importance of seeking professional medical advice may be challenging, as the model is not a substitute for professional diagnosis.

Continuous Model Improvement:
Ensuring continuous improvement of the model based on user feedback and emerging health trends requires ongoing effort and resources.

11. CONCLUSION

In conclusion, the "Disease Prediction Using Machine Learning" project represents a significant advancement in healthcare accessibility. By leveraging machine learning, it empowers individuals for early disease detection through a user-friendly web application. Despite challenges, the project prioritizes user privacy, ethical considerations, and continuous improvement. The integration of educational content and online consultations enhances the user experience, highlighting the project's commitment to responsible AI in healthcare. As we progress, addressing challenges and refining the model based on user feedback will ensure the project's impact in bridging the gap between symptom onset and professional medical advice, fostering innovation and user empowerment in healthcare

12. FUTURE SCOPE

The future scope for our "Disease Prediction Using Machine Learning" project is expansive, offering numerous opportunities for advancement and refinement. Firstly, we envision expanding the model's disease coverage to encompass a broader spectrum of health conditions, providing users with a more comprehensive healthcare solution. Integration with wearable technology stands as a pivotal direction, enhancing the model's capabilities by incorporating real-time health data from users.

Continued exploration of advanced machine learning models remains a priority, ensuring ongoing improvement in accuracy and robustness. Personalized health recommendations represent a logical next step, guiding users towards preventive measures and lifestyle changes based on predicted diseases.

On a global scale, collaborating with healthcare institutions worldwide is crucial to gather diverse datasets, making the model effective across different demographics and healthcare systems. Localizing the model to address region-specific health concerns further enhances its impact.

Strengthening ties with the broader telehealth ecosystem is on the horizon, facilitating seamless communication between users, healthcare professionals, and other healthcare services. Continuous learning mechanisms will be implemented to adapt the model to emerging health trends, user feedback, and evolving medical knowledge.

In essence, the future of our project envisions a comprehensive and adaptive solution, continually evolving to meet the dynamic demands of the healthcare landscape while empowering individuals with proactive and personalized health insights

13. APPENDIX

GitHub & Project Demo Link -

Source Codes -

Python (Flask Application):

```
from flask import Flask, render_template, request
import numpy as np
import pickle

model = pickle.load(open('model.pkl','rb'))
app = Flask(__name__)

@app.route("/")
def home():
    return render_template('index.html')

@app.route('/details')
def pred():
    return render_template('details.html')

@app.route('/predict',methods=['POST','GET'])
def predict():
    col=['itching', 'continuous_sneezing', 'shivering', 'joint_pain',
       'stomach_pain', 'vomiting', 'fatigue', 'weight_loss', 'restlessness',
       'lethargy', 'high_fever', 'headache', 'dark_urine', 'nausea',
       'pain_behind_the_eyes', 'constipation', 'abdominal_pain', 'diarrhoea',
       'mild_fever', 'yellowing_of_eyes', 'malaise', 'phlegm', 'congestion',
       'chest_pain', 'fast_heart_rate', 'neck_pain', 'dizziness',
       'puffy_face_and_eyes', 'knee_pain', 'muscle_weakness',
       'passage_of_gases', 'irritability', 'muscle_pain', 'belly_pain',
       'abnormal_menstruation', 'increased_appetite', 'lack_of_concentration',
       'visual_disturbances', 'receiving_blood_transfusion', 'coma',
       'history_of_alcohol_consumption', 'blood_in_sputum', 'palpitations',
       'inflammatory_nails', 'yellow_crust_ooze']
    if request.method=='POST':
        inputt = [str(x) for x in request.form.values()]

        b=[0]*45
        for x in range(0,45):
            for y in inputt:
                if(col[x]==y):
                    b[x]=1
        b=np.array(b)
        b=b.reshape(1,45)
        prediction = model.predict(b)
        prediction = prediction[0]
    return render_template('results.html', prediction_text="The probable diagnosis says it could be
{}".format(prediction))


if __name__ == "__main__":
    app.run()
```

Python(Machine Learning IPYNB file)-

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

import pickle


train = pd.read_csv("training.csv")
test = pd.read_csv("testing.csv")

train.head()

train.shape

train['Unnamed: 133'].value_counts()

train.drop("Unnamed: 133",axis = 1, inplace = True)

train.isnull().sum()

train.isnull().sum().sum()

train.columns

train.describe()


test.head()

test.shape

len(train.prognosis.unique())

train.prognosis.value_counts()

for i in train.columns:
    print(train[i].value_counts())

train['fluid_overload'].unique()
```

```
train.drop('fluid_overload',axis = 1, inplace = True)

corr = train.corr()
corr.style.background_gradient('coolwarm')

plt.figure(figsize = (8,8))

a = train['itching'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title("Pie chart showing the distribution of Itching symptom into number of Yes/No ")

b = train['continuous_sneezing'].value_counts()
plt.subplot(122)
plt.pie(x = b, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title('Pie Chart showing the distribution of Continuous Sneezing symptom into number of Yes/No')

plt.subplots_adjust(left = 0.5, right = 2.4)

plt.figure(figsize = (8,8))

a = train['shivering'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title("Pie chart showing the distribution of Shivering symptom into number of Yes/No ")

b = train['joint_pain'].value_counts()
plt.subplot(122)
plt.pie(x = b, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title('Pie Chart showing the distribution of Joint Pain symptom into number of Yes/No')

plt.subplots_adjust(left = 0.5, right = 2.4)

plt.figure(figsize = (8,8))

a = train['shivering'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title("Pie chart showing the distribution of Shivering symptom into number of Yes/No ")

b = train['joint_pain'].value_counts()
plt.subplot(122)
plt.pie(x = b, data = train, labels= ['No','Yes'], autopct='%.0f%%',colors = 'gr')
plt.title('Pie Chart showing the distribution of Joint Pain symptom into number of Yes/No')

plt.subplots_adjust(left = 0.5, right = 2.4)

plt.subplot(1,2,1)
train['stomach_pain'].value_counts().plot(kind = 'bar', color = ['g','r'])
plt.title("Bar chart showing the distribution of Stomach Pain symptom")
```

```
plt.subplot(1,2,2)
train['vomiting'].value_counts().plot(kind = 'bar', color = ['g','r'])
plt.title("Bar chart showing the distribution of Vomiting symptom")

plt.subplots_adjust(left = 0.5, right = 2.5)

plt.subplot(1,2,1)
train['fatigue'].value_counts().plot(kind = 'bar', color = ['g','r'])
plt.title("Bar chart showing the distribution of Fatigue symptom")

plt.subplot(1,2,2)
train['weight_loss'].value_counts().plot(kind = 'bar', color = ['g','r'])
plt.title("Bar chart showing the distribution of Weight Loss symptom")

plt.subplots_adjust(left = 0.5, right = 2.5)

plt.subplot(1,2,1)
train['high_fever'].value_counts().plot(kind = 'barh', color = ['g','r'])
plt.title("Bar chart showing the distribution of High Fever symptom")

plt.subplot(1,2,2)
train['headache'].value_counts().plot(kind = 'barh', color = ['g','r'])
plt.title("Bar chart showing the distribution of Headache symptom")

plt.subplots_adjust(left = 0.5, right = 2.5)

train[(train['itching'] == 1) & (train['stomach_pain'] == 1)]

a = len(train[train['prognosis'] == 'Fungal infection'])
b = len(train[(train['itching'] == 1) & (train['prognosis'] == 'Fungal infection')])
fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Fungal Infection','Itching while Fungal Infection'])

sns.barplot(data = fi, x = fi.index, y = fi['Values'])
plt.title('Importance of Itching symptom to determine Fungal Infection')

a = len(train[train['prognosis'] == 'Fungal infection'])
b = len(train[(train['itching'] == 1) & (train['prognosis'] == 'Fungal infection')])
fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Fungal Infection','Itching while Fungal Infection'])

sns.barplot(data = fi, x = fi.index, y = fi['Values'])
plt.title('Importance of Itching symptom to determine Fungal Infection')

a = len(train[train['prognosis'] == 'Tuberculosis'])
b = len(train[(train['yellowing_of_eyes'] == 1) & (train['prognosis'] == 'Tuberculosis')])
fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Tuberculosis','Yellowing of Eyes while Tuberculosis'])

sns.barplot(data = fi, x = fi.index, y = fi['Values'])
plt.title('Importance of Yellowing of Eyes symptom to determine Tuberculosis')
```

```
a = train[train['prognosis'] == 'Tuberculosis']
a.head()

sns.swarmplot(x = a['weight_loss'], y = a['fatigue'])

train.head()

def data_preprocessing(data):
    data.drop(['fluid_overload','weight_gain','cold_hands_and_feets','anxiety','irregular_sugar_level',
          'yellow_urine','acute_liver_failure','swelling_of_stomach',
          'drying_and_tingling_lips','continuous_feel_of_urine',
         'internal_itching','polyuria','mood_swings','receiving_unsterile_injections',
         'stomach_bleeding','prominent_veins_on_calf','loss_of_smell','throat_irritation',
         'redness_of_eyes','sinus_pressure','runny_nose','pain_during_bowel_movements',
         'pain_in_anal_region','cramps','bruising','enlarged_thyroid','brittle_nails',
         'swollen_extremeties','slurred_speech','distention_of_abdomen','fluid_overload.1',
         'skin_peeling','silver_like_dusting','small_dents_in_nails','blister',
         'red_sore_around_nose','bloody_stool','swollen_blood_vessels','hip_joint_pain',
         'painful_walking','spinning_movements','altered_sensorium','toxic_look_(typhos)'],axis =1, inplace = True)
    return data

X = train.drop('prognosis',axis = 1)
y = train.prognosis

X_train, X_val, y_train, y_val = train_test_split(X,y,test_size = 0.2)


test = data_preprocessing(test)

X_test = test.drop('prognosis',axis = 1)
y_test = test.prognosis

def model_evaluation(classifier):
    y_pred = classifier.predict(X_val)
    yt_pred = classifier.predict(X_train)
    y_pred1 = classifier.predict(X_test)
    print('The Training Accuracy of the algorithm is ', accuracy_score(y_train, yt_pred))
    print('The Validation Accuracy of the algorithm is ', accuracy_score(y_val, y_pred))
    print('The Testing Accuracy of the algorithm is', accuracy_score(y_test, y_pred1))
    return [(accuracy_score(y_train, yt_pred)), (accuracy_score(y_val, y_pred)), (accuracy_score(y_test, y_pred1))]


knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,y_train)

knn_results = model_evaluation(knn)

svm = SVC(C=1)
svm.fit(X_train, y_train)
```

```
svm_results = model_evaluation(svm)

dtc = DecisionTreeClassifier(max_features= 10)
dtc.fit(X_train,y_train)

dtc_results = model_evaluation(dtc)

rfc = RandomForestClassifier(max_depth = 13)
rfc.fit(X_train, y_train)

rfc_results = model_evaluation(rfc)

results = pd.DataFrame(data = [knn_results, svm_results, dtc_results, rfc_results],
                       columns= ['Training Accuracy','Validation Accuracy', 'Testing Accuracy'],
                       index = ['K Nearest Neighbors Classifier','Support Vector Machines',
                            'Decision Trees Classifier', 'Random Forest Classifier'])
results



a = rfc.feature_importances_
col = X.columns
feat_imp = {}
for i, j in zip(a,col):
    feat_imp[j] = i
feat_imp

def model_evaluation1(n_feat,classifier):
    y_pred = classifier.predict(X1_val)
    yt_pred = classifier.predict(X1_train)
    y_pred1 = classifier.predict(X1_test)
    return [(n_feat),(accuracy_score(y1_train, yt_pred)), (accuracy_score(y1_test, y_pred1))]

randomf = pd.DataFrame(data = rfc_results,columns=['Number of features','Training Accuracy','Testing
Accuracy'])
randomf

to_drop = []
for i,j in zip(feat_imp.keys(),feat_imp.values()):
    if j < 0.01:
        to_drop.append(i)
len(to_drop)

X_new = X.drop(to_drop,axis = 1)
y_new = y
X_new.head()

X1_train, X1_val, y1_train, y1_val = train_test_split(X_new, y_new, test_size=0.2)
X1_test = X_test.drop(to_drop,axis = 1)
y1_test = y_test
```

```
rfc_new = RandomForestClassifier()
rfc_new.fit(X1_train, y1_train)

y_pred = rfc_new.predict(X1_val)
yt_pred = rfc_new.predict(X1_train)
y_pred1 = rfc_new.predict(X1_test)
print('The Training Accuracy of the algorithm is ', accuracy_score(y1_train, yt_pred))
print('The Validation Accuracy of the algorithm is ', accuracy_score(y1_val, y_pred))
print('The Testing Accuracy of the algorithm is', accuracy_score(y1_test, y_pred1))

knn_new = KNeighborsClassifier()
knn_new.fit(X1_train, y1_train)

y_pred = knn_new.predict(X1_val)
yt_pred = knn_new.predict(X1_train)
y_pred1 = knn_new.predict(X1_test)
print('The Training Accuracy of the algorithm is ', accuracy_score(y1_train, yt_pred))
print('The Validation Accuracy of the algorithm is ', accuracy_score(y1_val, y_pred))
print('The Testing Accuracy of the algorithm is', accuracy_score(y1_test, y_pred1))

test.join(pd.DataFrame(y_pred1,columns=["predicted"]))[["prognosis","predicted"]]

X1_train.sum(axis = 1).max()

X1_train.columns

pickle.dump(knn_new, open('model.pkl','wb'))
```