

Disease Prediction Using Machine Learning

1. INTRODUCTION

In the contemporary era of healthcare, the integration of advanced technologies has proven to be pivotal in enhancing diagnostic capabilities and improving patient outcomes. One such groundbreaking application is the utilization of Machine Learning (ML) algorithms for disease prediction. This paradigm shift from traditional diagnostic methods to predictive analytics holds the potential to revolutionize the field of medicine by enabling early detection and proactive management of diseases.

This final documentation encapsulates the comprehensive exploration, design, and implementation of a disease prediction model leveraging ML techniques. The primary objective is to harness the power of data-driven insights to predict the likelihood of a particular disease in individuals based on relevant health parameters. By employing sophisticated algorithms and statistical analysis, this project aims to contribute to the paradigm of preventive medicine, where interventions can be initiated well in advance, potentially mitigating the severity and progression of diseases.

1.1 Purpose

Here are five key use cases that highlight the purpose and potential impact of the Disease Prediction dproject:

Enhances healthcare by predicting diseases early, enabling timely intervention and mitigating the progression to severe stages.

Shifts healthcare from reactive to proactive, empowering timely actions based on predictive insights for improved health outcomes.

Maximizes efficiency in healthcare resource allocation by targeting interventions toward individuals identified as higher risk through predictive modeling.

Tailors healthcare strategies to individuals by considering unique health data, genetics, and lifestyle, improving precision in treatment approaches.

Informs public health decisions with aggregated predictive data, shaping targeted campaigns and interventions to address emerging health trends in specific populations.

2. LITERATURE SURVEY

2.1 Existing problem

Existing problems in disease prediction using machine learning include challenges related to data quality and accessibility. Limited access to high-quality, diverse healthcare data can hinder the development of robust predictive models. Biased or incomplete datasets may compromise the accuracy and generalizability of predictions, impacting the reliability of early disease detection. Additionally, the lack of interoperability and integration between different electronic health record (EHR) systems poses a barrier. Healthcare systems often use disparate platforms that may not seamlessly collaborate with machine learning tools, impeding the efficient utilization of data for predictive modeling.

Ethical considerations and privacy concerns constitute another significant challenge. Utilizing sensitive health data for predictive purposes raises ethical questions, necessitating a delicate balance between extracting valuable insights and ensuring patient confidentiality. Moreover, the issue of model interpretability arises, particularly with complex algorithms like deep learning, where the lack of transparency in decision-making processes can hinder trust from healthcare professionals and patients. The dynamic nature of health data further complicates matters, as health conditions and risk factors evolve over time due to lifestyle changes, environmental influences, and genetic variations. Adapting machine learning models to account for this dynamism and ensuring continuous accuracy is an ongoing challenge in the quest for effective disease prediction.

2.2 References

1. <https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning/>
2. <https://github.com/socksparadox/Disease-Prediction-ML-Flask>
3. <https://bootstrapmade.com/impact-bootstrap-business-website-template/>

2.3 Problem Statement Definition

The current landscape of disease prediction in healthcare faces multifaceted challenges that hinder its seamless integration and effectiveness. Insufficient access to high-quality and diverse healthcare data, interoperability issues among electronic health record (EHR) systems, ethical dilemmas surrounding patient privacy, and the interpretability of complex machine learning models all contribute to the complexity of developing reliable disease prediction systems. Furthermore, the dynamic nature of health data, influenced by factors such as lifestyle changes and genetic variations, adds an additional layer of complexity to the accurate prediction of diseases. Addressing these challenges is critical to unlocking the full potential of machine learning in disease prediction and ensuring its practical application in real-world healthcare scenarios.

Definition:

Disease prediction using machine learning involves the application of advanced computational algorithms to analyze healthcare data and forecast the likelihood of an individual developing a particular medical condition. The goal is to move beyond traditional diagnostic approaches and enable proactive healthcare management by identifying patterns and risk factors early on. This process encompasses the collection, preprocessing, and analysis of diverse health data, including electronic health records, genetic information, and lifestyle factors. The ultimate aim is to develop accurate and interpretable predictive models that contribute to early detection, personalized medicine, and optimized

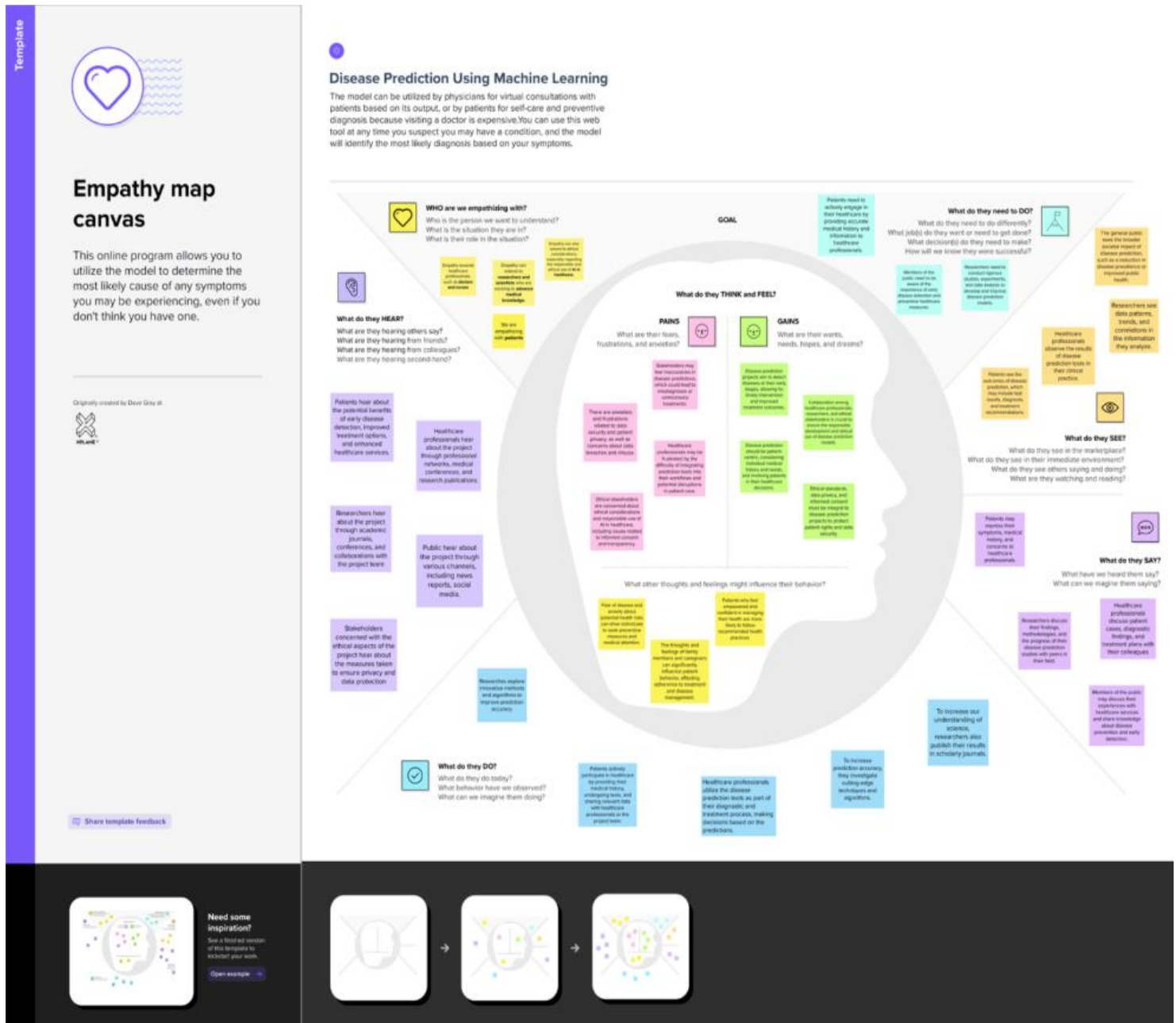
allocation of healthcare resources. In navigating the intricacies of this field, ethical considerations regarding patient privacy and the responsible use of sensitive health information play a pivotal role in shaping the development and deployment of machine learning-based disease prediction systems.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


Link:

<https://app.mural.co/t/aimlproject0788/m/aimlproject0788/1698068254092/fdc1024563a87c8bfb236172d04326f1bdfa6c64?sender=ub3d9b06ceba42c99cf2a3434>



3.2 Ideation & Brainstorming

Template



Brainstorm & idea prioritization

Disease Prediction Using Machine Learning

This online program allows you to utilize the model to determine the most likely cause of any symptoms you may be experiencing, even if you don't think you have one.

🕒 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

model

Create a disease prediction model in machine learning, addressing data challenges for accurate and reliable healthcare applications.

🧠

Key rules of brainstorming

To run an smooth and productive session

🗣️ Stay in topic.

💡 Encourage wild ideas.

🚫 Defer judgment.

👂 Listen to others.

🗣️ Go for volume.

👁️ If possible, be visual.

LINK:

<https://app.mural.co/t/aimlproject0788/m/aimlproject0788/1698075698297/89fdcd74304c0f576f5452a30e3ccdbd9601d669?sender=ub3d9b06ceba42c99cf2a3434>

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Tarun Sreenivas

Ensure that your dataset is clean, consistent, and free from missing values to prevent data-related issues that could affect the model's accuracy.

Establish a system to monitor and update your model over time to adapt to changing

Develop an intuitive and user-friendly interface for healthcare professionals and end-users to interact with the disease prediction system easily.

Anish Nair

Develop an User friendly interface

Consider techniques for handling missing data, outlier detection, and data normalization.

Make sure your dataset is error-free, consistent, and free of missing values to avoid problems with the data that could compromise the accuracy of the model.

Perisetla Sri Sai Yagnik

Gather feedback from users to enhance the model's real-world applicability.

Define a strategy for model updates and retraining based on changing disease patterns and the degradation of model performance over time

Choose and utilize appropriate evaluation metrics that align with the nature of the disease and your project goals.

Amada Karthikeya

Regularly update and retrain the model using the latest data and collaborate with experts to adapt to changing disease patterns.

Invest time in feature selection and engineering to improve the model's predictive accuracy.

Get user feedback so that the model can be more practically improved

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Develop an
User friendly
interface

Establish a
system to monitor
and update your
model over time
to adapt to
changing

Gather feedback
from users to
enhance the
model's real-
world
applicability.

Invest time in
feature selection
and engineering to
improve the
model's predictive
accuracy.

4

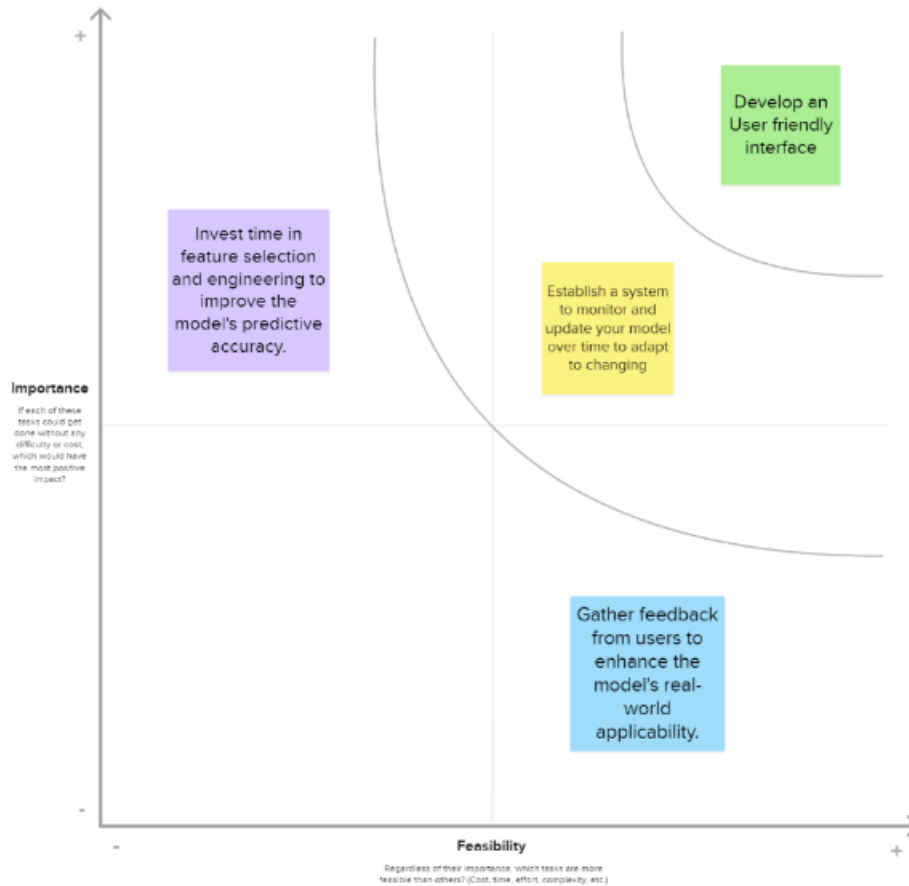
Prioritize

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

The functional requirements for a disease prediction system using machine learning encompass a spectrum of crucial capabilities. First and foremost, the system should proficiently collect and integrate a diverse array of healthcare data, ranging from electronic health records to genetic information and lifestyle factors. This mandates a robust data collection and integration module that ensures the creation of a comprehensive and representative dataset. Furthermore, the system must be equipped with sophisticated preprocessing and cleaning functionalities, capable of addressing challenges such as missing values, outliers, and data inconsistencies. This ensures the quality and reliability of the input features, laying a solid foundation for accurate predictions.

Algorithm selection and training constitute another vital functional requirement, necessitating the platform's ability to support a variety of machine learning algorithms. This includes traditional statistical models and advanced techniques like deep learning, with the flexibility to adapt to the specific characteristics of the healthcare data at hand. To ascertain the effectiveness of the predictive models, the system should feature robust model evaluation and validation mechanisms, employing metrics such as accuracy, precision, recall, and F1 score. This ensures that the disease prediction models meet the desired standards of reliability and performance. Moreover, a critical aspect is the incorporation of features that enhance the interpretability and explainability of the machine learning models. This not only fosters trust among healthcare professionals and end-users but also facilitates the translation of predictive insights into actionable healthcare interventions. In essence, these functional requirements collectively constitute a comprehensive framework for the successful development and deployment of a disease prediction system using machine learning in real-world healthcare scenarios.

4.2 Non-Functional requirements

The system must scale seamlessly to handle varying volumes of healthcare data and user loads while maintaining consistent performance.

Low-latency response times are crucial for timely delivery of predictions, addressing the urgent nature of healthcare decision-making.

Robust security measures, including encryption, access controls, and auditing features, must be implemented to safeguard sensitive healthcare data and ensure compliance with relevant data protection regulations.

The system should exhibit high reliability, minimizing downtime and ensuring continuous access for healthcare professionals who rely on timely disease predictions.

Seamless integration with existing healthcare infrastructure, including electronic health record (EHR) platforms, is essential for interoperability and efficient data exchange.

The user interface should be intuitive and user-friendly to accommodate healthcare professionals with varying technical expertise, enhancing the overall usability of the system.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Healthcare Data Repository:

Central repository for diverse healthcare data, including electronic health records, genetic information, and lifestyle factors.

Preprocessing Module:

Cleans and normalizes the healthcare data to ensure data quality and reliability for subsequent analysis.

Algorithm Selection and Training:

Utilizes machine learning algorithms to train predictive models based on the characteristics of the preprocessed healthcare data.

Predictive Model Deployment:

Deploys trained predictive models to make accurate predictions about the likelihood of individuals developing specific medical conditions.

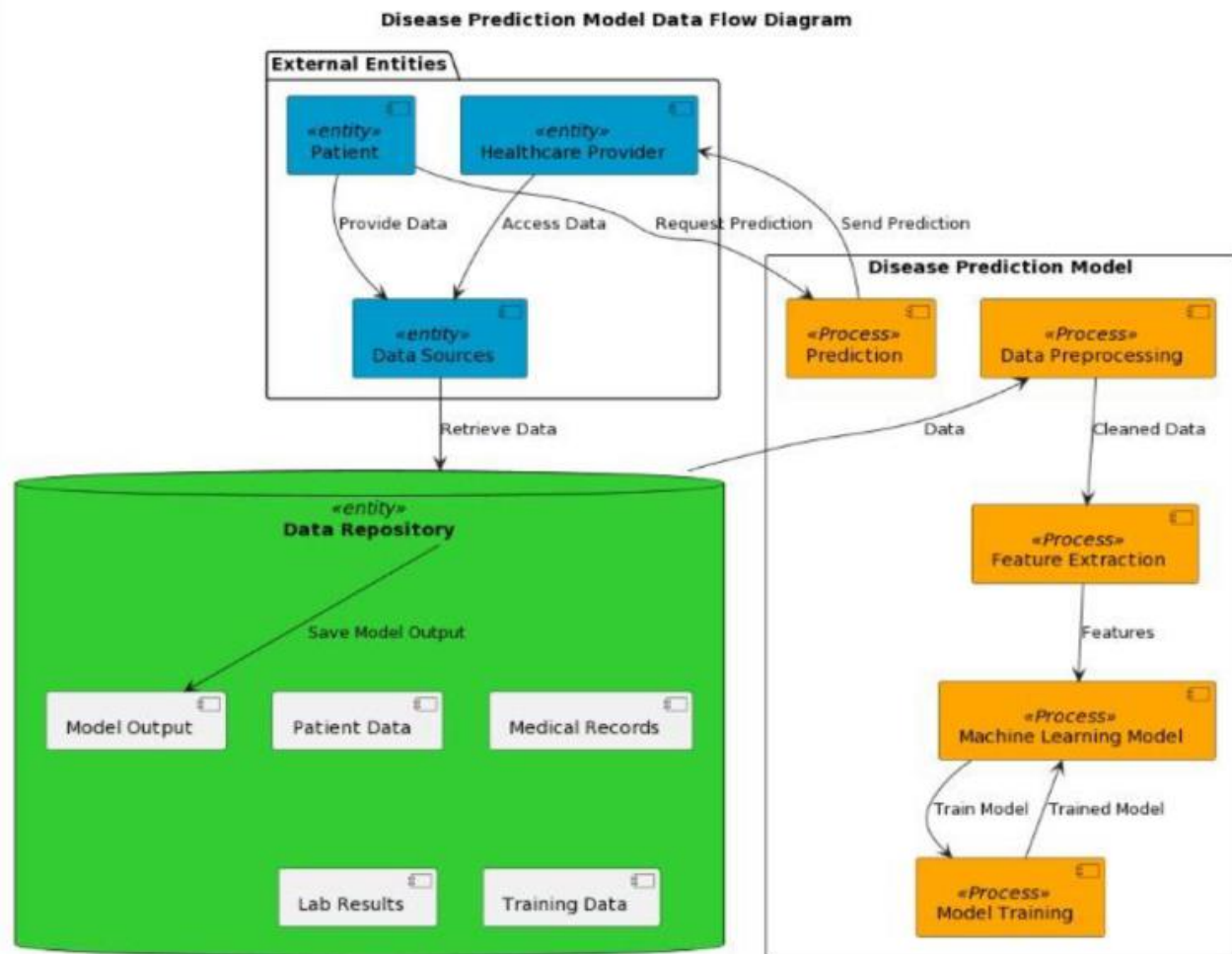
User Interface:

Provides an intuitive user interface for healthcare professionals to input patient data and receive clear and actionable disease predictions.

Interoperability Interface:

Interfaces with external healthcare systems, such as electronic health record platforms, for seamless data exchange and enhanced interoperability.

Data Flow Diagram:



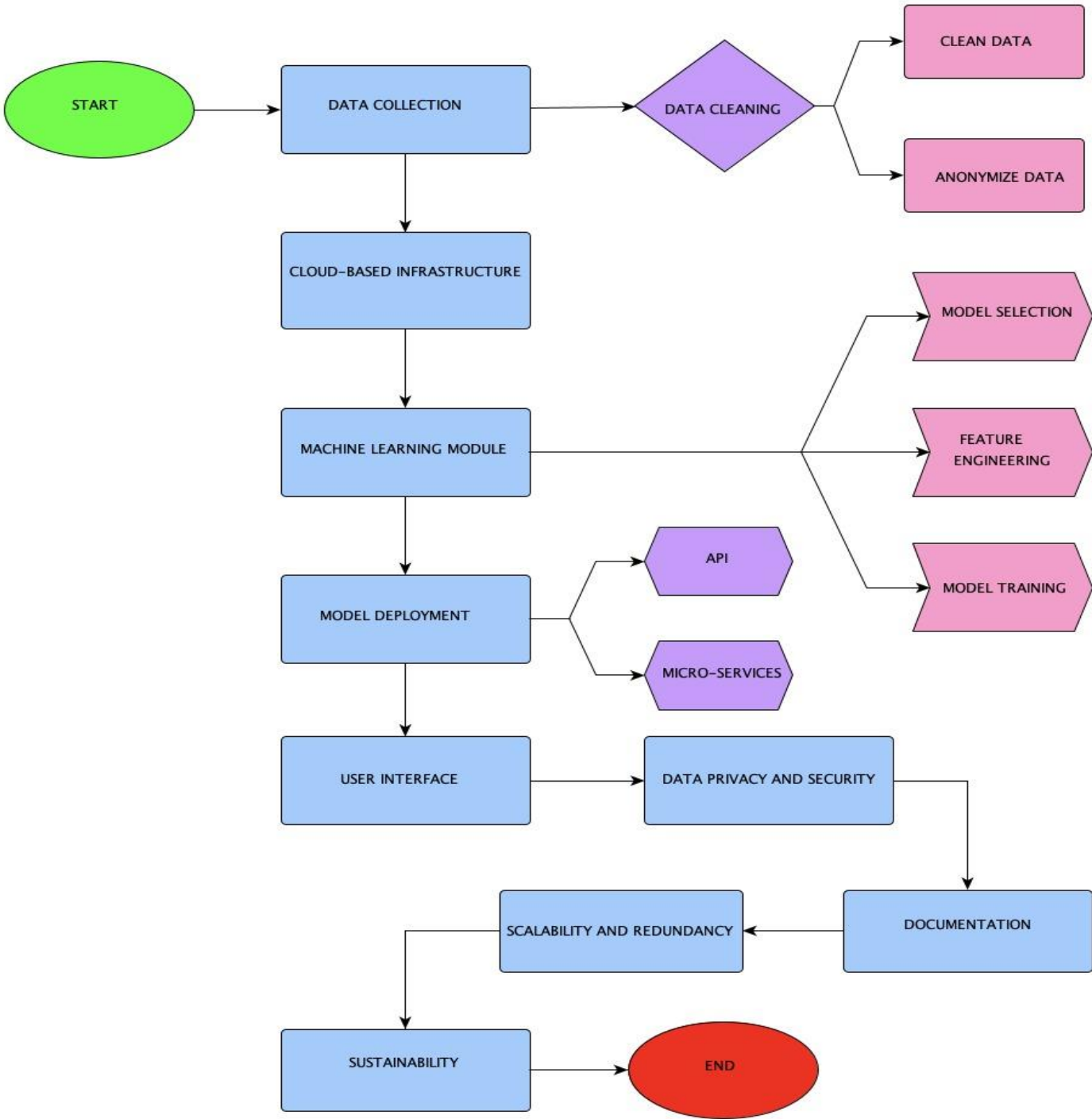
User Stories:

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Patient	Registration	US001	As a patient, I want to register for an account, so I can access my health information.	The registration form should include fields for name, email, and password. Upon successful registration, the system should send a confirmation email.	High	Release 1
Physician	View Patient Records	US002	As a physician, I want to view the medical records of my patients.	I should be able to search for patients by name or patient ID. Access to records should be secure and require authentication.	High	Release 1
Admin	User Management	US003	As an admin, I want to manage user accounts to ensure data security.	I should be able to add, delete, or modify user accounts. Admin actions should be logged for auditing.	High	Release 1
Patient	Request Appointment	US004	As a patient, I want to request an appointment with my physician.	I should be able to select a preferred date and time. The system should notify me of appointment confirmation or rejection.	Medium	Release 2
Physician	Record Patient Notes	US005	As a physician, I want to record notes about a patient's visit.	I should be able to attach notes to a patient's profile. Notes should include date, time, and details of	Medium	Release 2

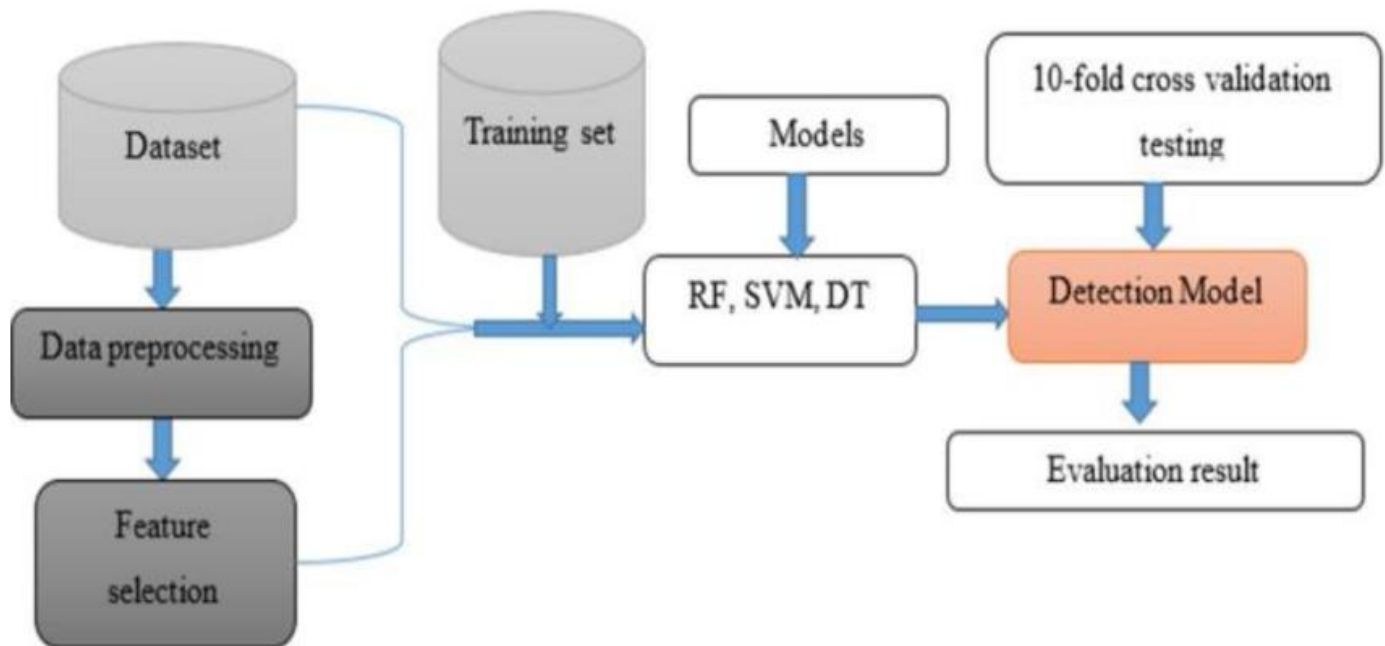
				the visit.		
Admin	Generate Reports	US006	As an admin, I want to generate usage reports for system performance analysis.	Reports should include the number of registered users, usage statistics, and peak usage times. Reports should be exportable in CSV format.	Low	Release 3

Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a user, I can collect patient data for training	5	High	Tarun Sreenivas
Sprint-1	Feature Selection	USN-2	As a user, I can identify relevant features	3	High	Tarun Sreenivas
Sprint-2	Model Development	USN-3	As a user, I can build machine learning model	8	High	Attada Karthikeya
Sprint-2	Testing	USN-4	As a user, I can evaluate model performance	5	Medium	Anish Narla
Sprint-3	Integration	USN-5	As a user, I can integrate model into application	5	Medium	P Sri Sai Yagnik
Sprint-3	User Interface	USN-6	Design UI for the prediction system	8	High	P Sri Sai Yagnik
Sprint-4	Documentation	USN-7	Create project documentation	3	Low	Anish Narla
Sprint-4	Optimization	USN-8	Optimize model for performance	3	Low	Attada Karthikeya

6.3 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	20	6 Days	24 Oct 2023	29 Oct 2023	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2023	04 Nov 2023	20	04 Nov 2023
Sprint-3	20	6 Days	05 Nov 2023	10 Nov 2023	20	10 Nov 2023
Sprint-4	20	6 Days	05 Nov 2023	10 Nov 2023	20	10 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)

$$AV = \text{sprint duration} / \text{velocity} = 80/4 = 20$$

Coding and Solutioning:

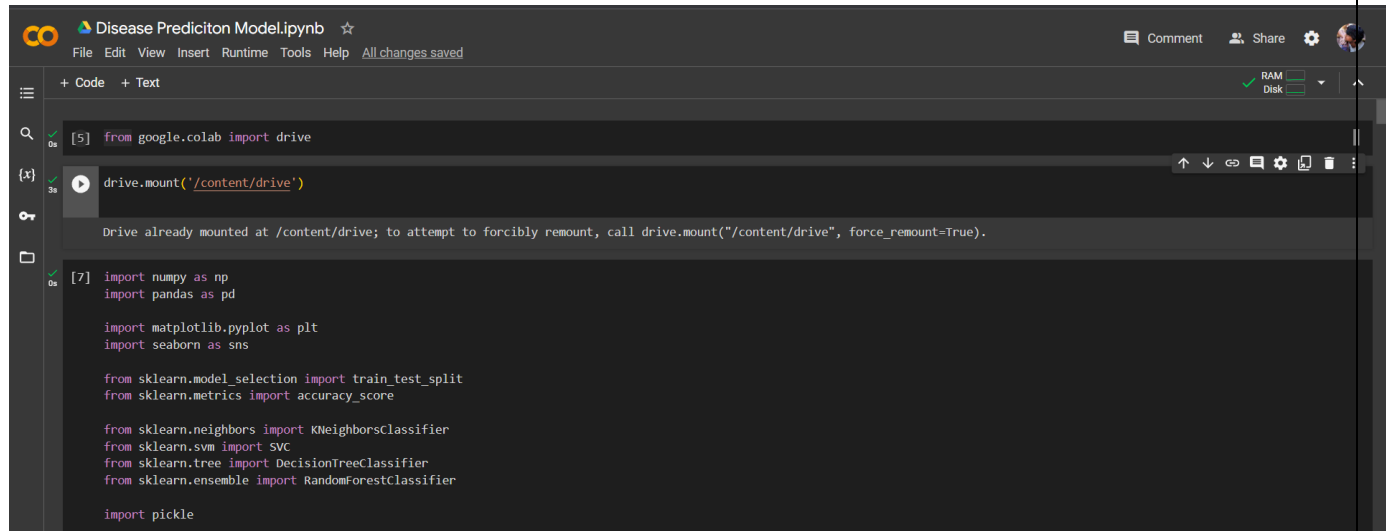
Project Structure:

Create project folder which contains files as shown below:

Name	Date Modified
> Data	21-11-2023 15:46
> Flask	21-11-2023 15:46
> static	21-11-2023 15:46
> css	21-11-2023 15:46
</> main.css	15-04-2023 01:32
> img	21-11-2023 15:46
> js	21-11-2023 15:46
.js main.js	18-09-2023 09:58
> scss	21-11-2023 15:46
> vendor	21-11-2023 15:46
> aos	21-11-2023 15:46
> bootstrap	21-11-2023 15:46
> bootstrap-icons	21-11-2023 15:46
> glightbox	21-11-2023 15:46
> isotope-layout	21-11-2023 15:46
> php-email-form	21-11-2023 15:46
> purecounter	21-11-2023 15:46
> swiper	21-11-2023 15:46
.DS_Store	21-11-2023 15:26
> templates	21-11-2023 16:06
.DS_Store	21-11-2023 15:26
app.py	15-04-2023 01:32
model.pkl	15-04-2023 01:32
.DS_Store	21-11-2023 15:46
Disease Prediciton Model.ipynb	15-04-2023 01:32
Disease Prediction Using Machine Learning.pdf	15-04-2023 01:32

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.



The screenshot shows a Jupyter Notebook titled "Disease Prediction Model.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar (All changes saved). The notebook has two code cells. The first cell, labeled [5], contains the code `from google.colab import drive` and `drive.mount('/content/drive')`. Below the code, a message states: "Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount(\"/content/drive\", force_remount=True).". The second cell, labeled [7], contains a series of import statements for various libraries: `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`, and `import pickle`. The notebook interface also shows a sidebar with icons for file explorer, search, and other functions, and a top right corner with RAM and Disk usage indicators.

```
[5] from google.colab import drive

drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[7] 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.


```
[8] train = pd.read_csv("/content/drive/MyDrive/Training.csv")
test = pd.read_csv("/content/drive/MyDrive/Testing (1).csv")

train.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	scurrying	skin_peeling	silver_like_dusting
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 x 134 columns

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

Activity 2.1: Removing Redundant Columns

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

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

[12] 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

```
✓ 0s train.isnull().sum()

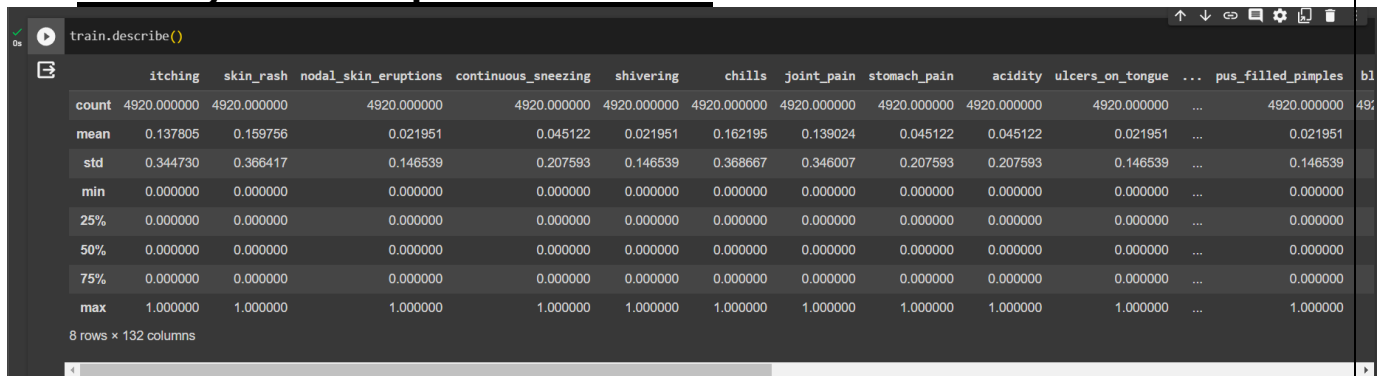
⇨ itching      0
  skin_rash    0
  nodal_skin_eruptions  0
  continuous_sneezing  0
  shivering    0
  ..
  inflammatory_nails  0
  blister         0
  red_sore_around_nose  0
  yellow_crust_ooze  0
  prognosis       0
  Length: 133, dtype: int64

✓ 0s [14] train.isnull().sum().sum()

0
```

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

Activity 1: Descriptive Statistical

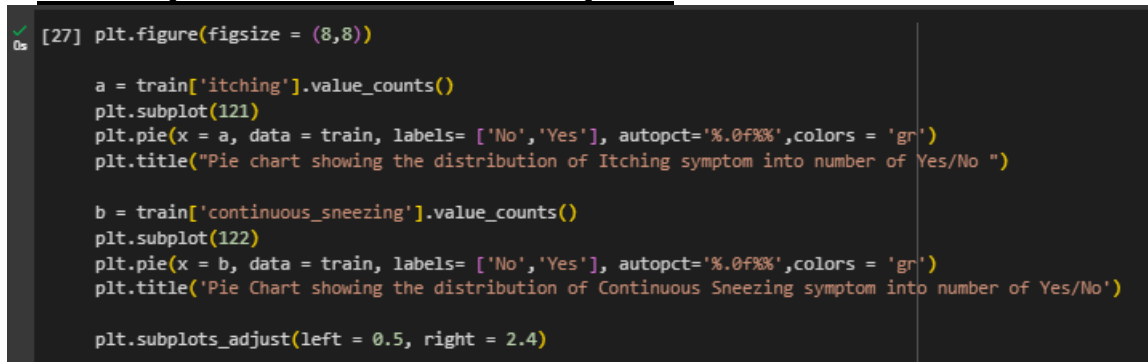


	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	pus_filled_pimples	bl
count	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	...	4920.000000	4920.000000
mean	0.137805	0.159756	0.021951	0.045122	0.021951	0.162195	0.139024	0.045122	0.045122	0.021951	...	0.021951	0.021951
std	0.344730	0.366417	0.146539	0.207593	0.146539	0.368667	0.346007	0.207593	0.207593	0.146539	...	0.146539	0.146539
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000

8 rows x 132 columns

Activity 2: Visual Analysis

Activity 2.1: Univariate Analysis:

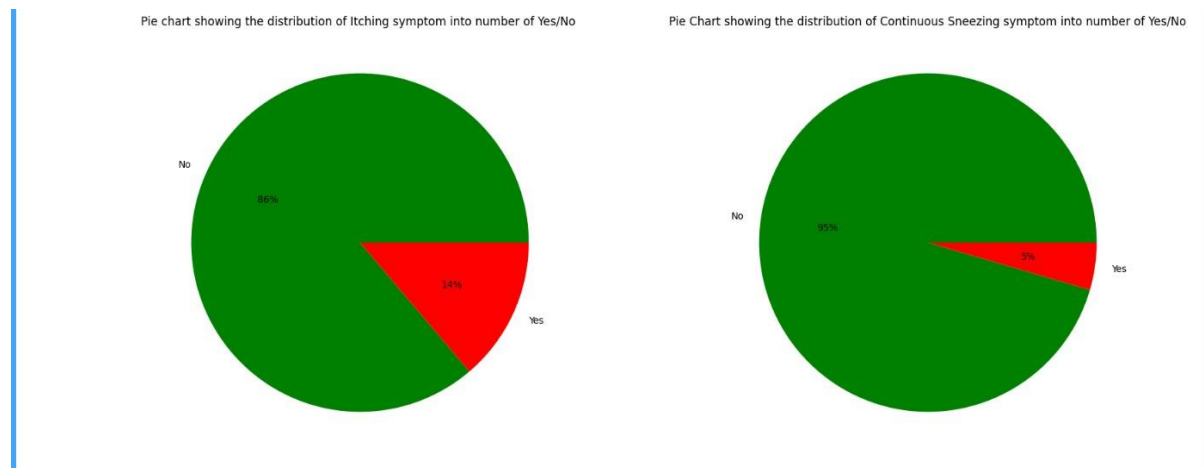


```
[27] plt.figure(figsize = (8,8))

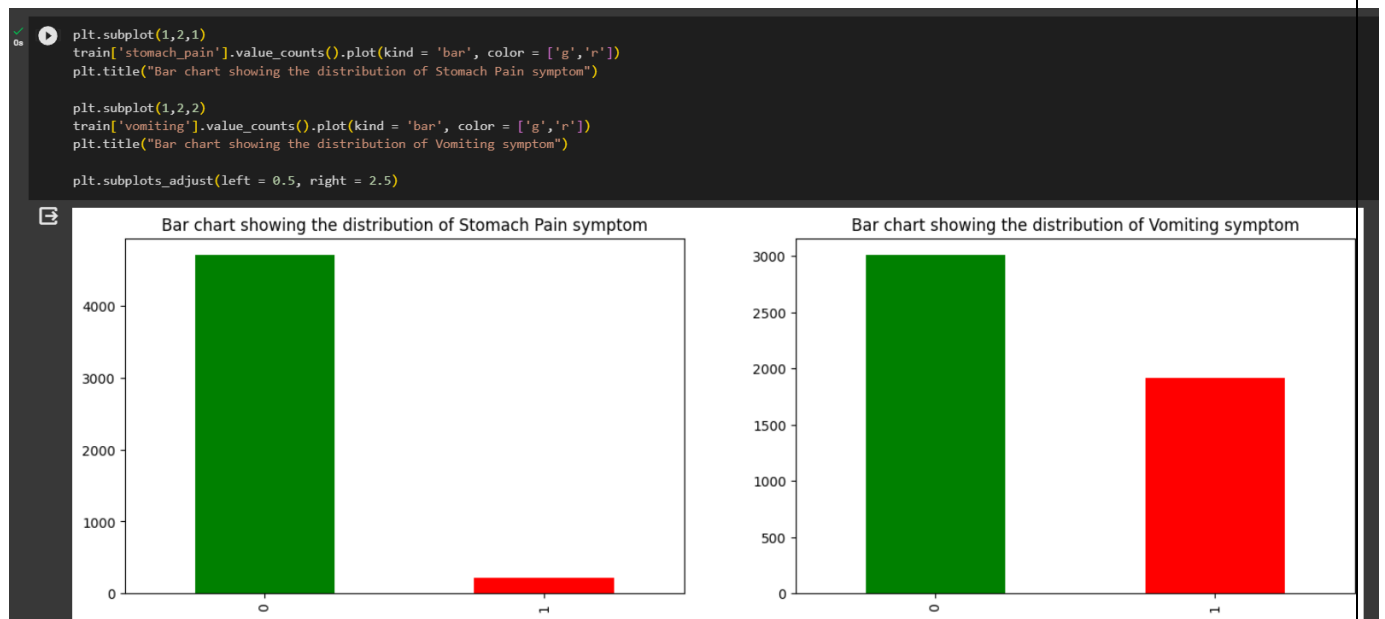
a = train['itching'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = train, labels= ['No','Yes'], autopct='%0.0f%%', colors = 'gr')
plt.title("Pie chart showing the distribution of Itching symptom into number of Yes/No ")

b = train['continuous_sneezing'].value_counts()
plt.subplot(122)
plt.pie(x = b, data = train, labels= ['No','Yes'], autopct='%0.0f%%', 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)
```



1.





Activity 2.2: Bivariate Analysis:

✓
1s

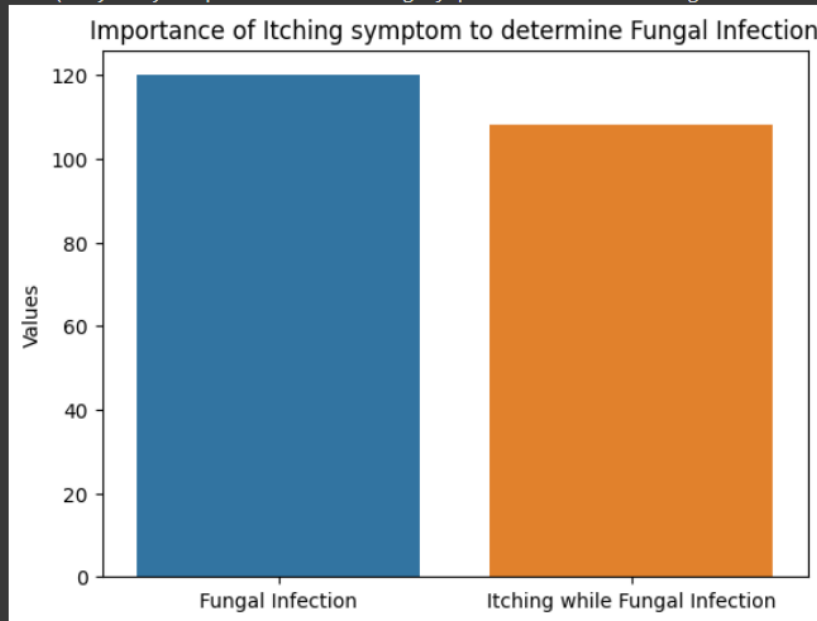


```
a = len(train[train['prognosis'] == 'Fungal infection'])
b = len(train[(train['itching'] == 1) & (train['prognosis'] == 'Fungal infection')])
fi = pd.DataFrame(data = [a,b], columns=['Values'],index = ['Fungal Infection','Itching while Fungal Infection'])

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



Text(0.5, 1.0, 'Importance of Itching symptom to determine Fungal Infection')





Activity 2.3: Multivariate Analysis

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

<ipython-input-24-92accbbd8d53>: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.

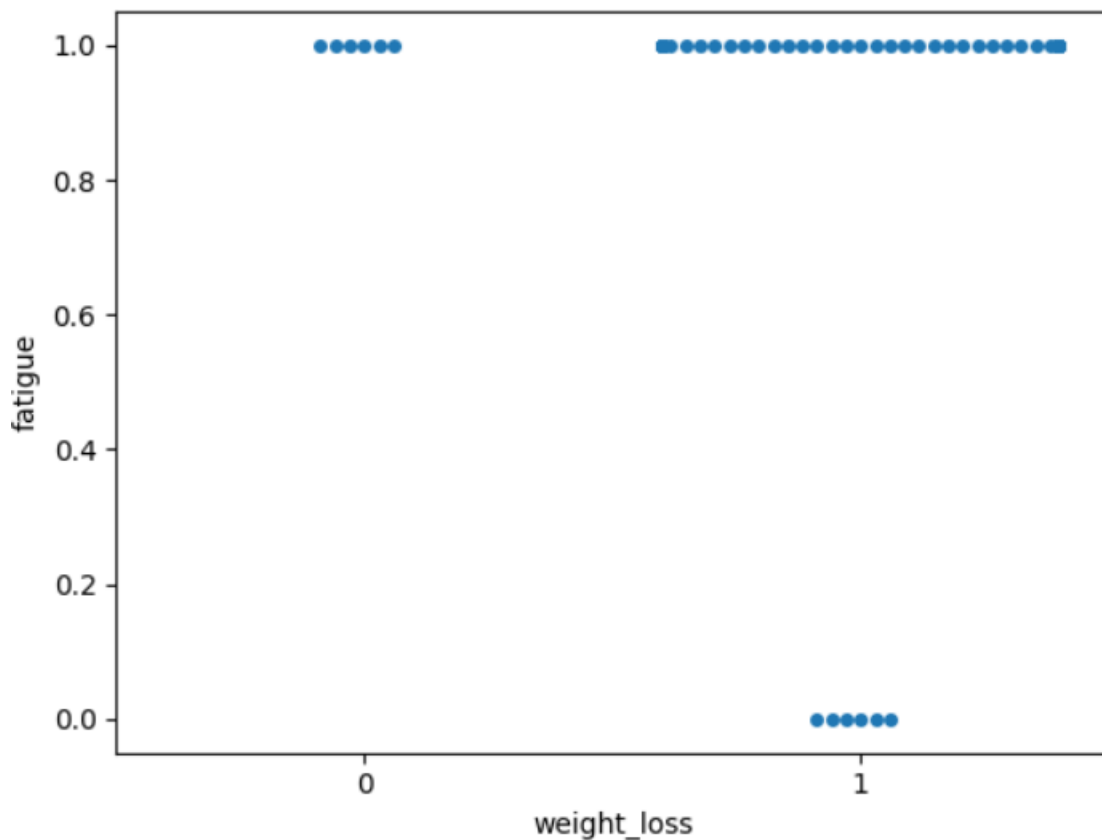
corr = train.corr()

	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
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
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	-0.061573
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	-0.021073
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	-0.031480
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	-0.021073
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	-0.061765
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
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	-0.031480
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	-0.031480
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	-0.021073
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	-0.021073
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	-0.121896
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	-0.031003
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	-0.021073
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	0.151337
weight_gain	-0.061573	-0.061765	-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
anxiety	-0.061573	-0.061765	-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	-0.021720
cold_hands_and_feets	-0.061573	-0.061765	-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	0.941120
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	0.661111
weight_loss	-0.091839	-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	-0.041224
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	-0.031951
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	0.451929

```

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)

```



Preprocessing of Test data


```

0s 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.

```

0s [44] test = data_preprocessing(test)

```

Here we call the function for the test data.

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

```

0s [42] x = train.drop('prognosis',axis = 1)
    y = train.prognosis

```

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

```

0s [45] 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)

```

0s [43] x_train, x_val, y_train, y_val = train_test_split(X,y,test_size = 0.2)

```

Activity 1: Creating a function for model evaluation

```
0s ✓ ▶ 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))]
```

Activity 2.1: K Nearest Neighbors Model

```
0s ✓ ▶ knn = KNeighborsClassifier(n_neighbors=7)  
    knn.fit(X_train,y_train)
```

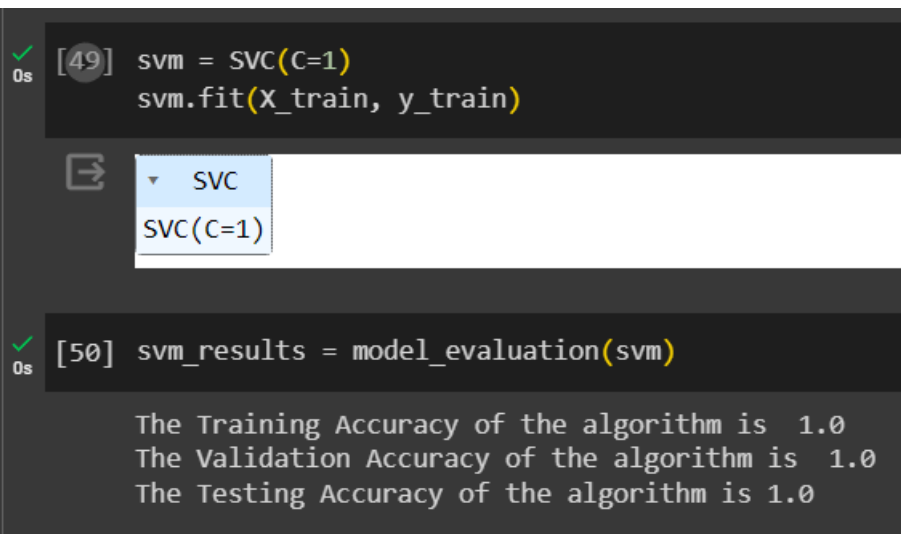
→ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)

```
1s ✓ ▶ 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
```

Activity 2.2: SVM Model

```
✓ [49] svm = SVC(C=1)
0s      svm.fit(x_train, y_train)
```



```
✓ [50] svm_results = model_evaluation(svm)
0s
```

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

Activity 2.3: Decision Tree Model

```
✓ 0s ▶ dtc = DecisionTreeClassifier(max_features= 10)
      dtc.fit(X_train,y_train)
```

DecisionTreeClassifier
DecisionTreeClassifier(max_features=10)

```
✓ 0s ▶ 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

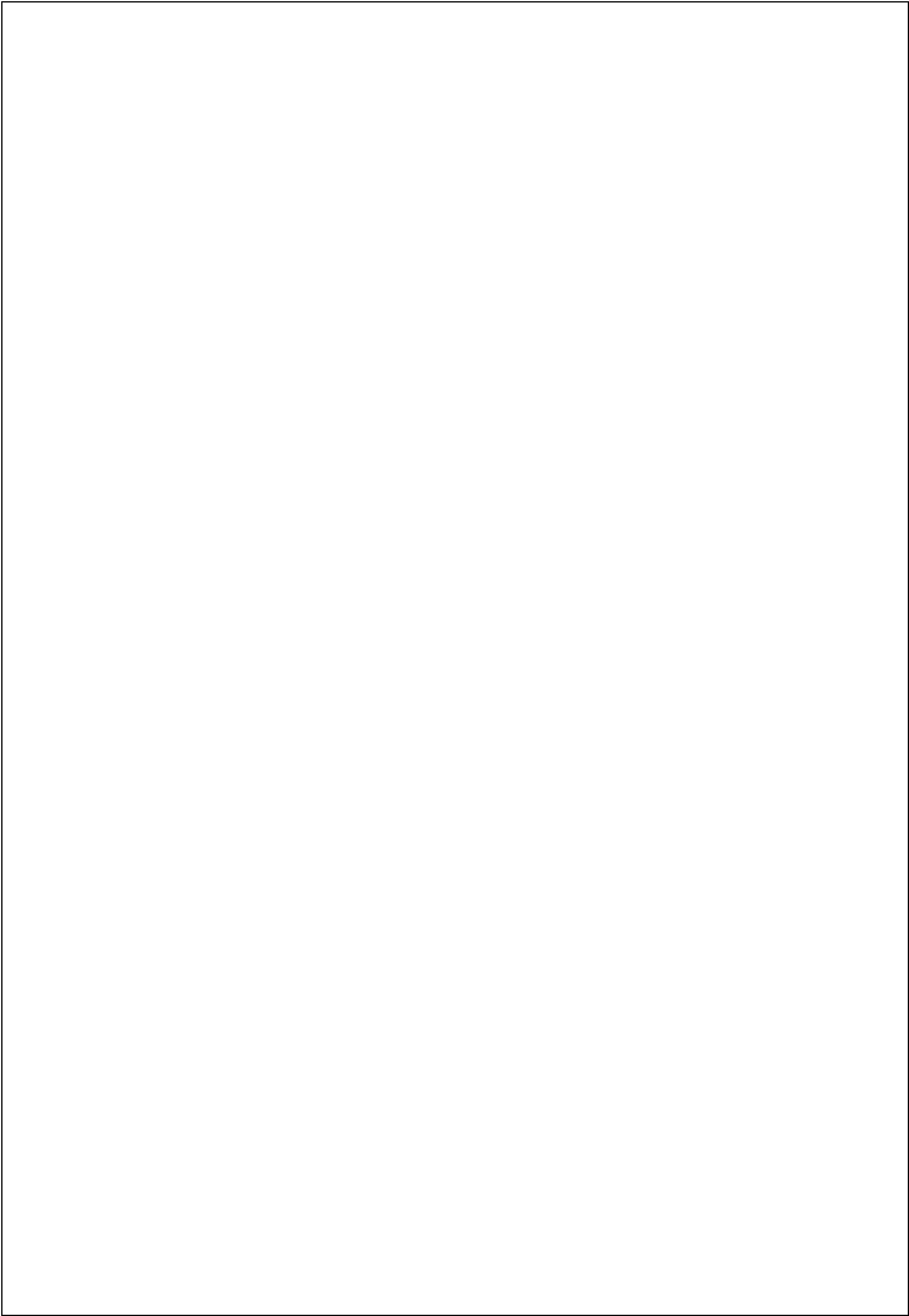
Activity 2.4: Random Forest Model

```
✓ 0s ▶ rfc = RandomForestClassifier(max_depth = 13)
      rfc.fit(X_train, y_train)
```

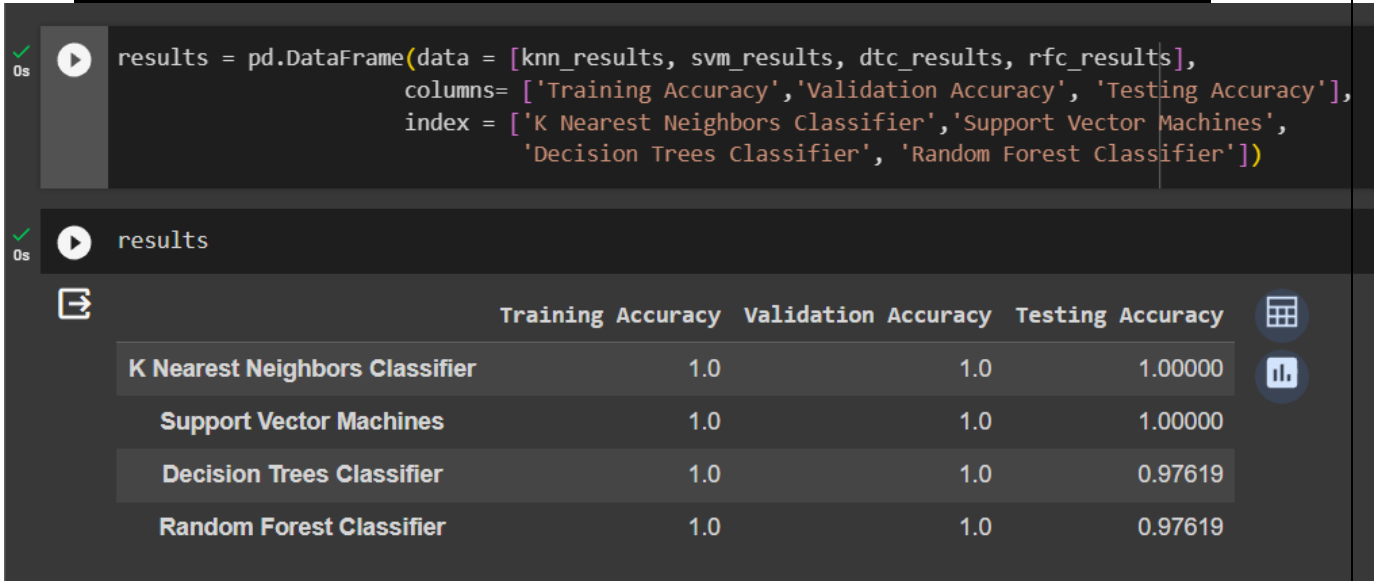
RandomForestClassifier
RandomForestClassifier(max_depth=13)

```
✓ 0s [54] rfc_results = model_evaluation(rfc)
```

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



Activity 1: Testing model with Multiple Evaluation metrics



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.

✓ 0s [57] a = rfc.feature_importances_

✓ 0s [58] col = X.columns

✓ 0s [59] feat_imp = {}
for i, j in zip(a,col):
 feat_imp[j] = i

✓ 0s  feat_imp

 {'itching': 0.00637397372259958,
'skin_rash': 0.008519439365783138,
'nodal_skin_eruptions': 0.009549262899606057,
'continuous_sneezing': 0.008626620888667925,
'shivering': 0.003919396712971001,
'chills': 0.012137802362256356,
'joint_pain': 0.013519771114257641,
'stomach_pain': 0.00918139430300248,
'acidity': 0.004368667057111305,
'ulcers_on_tongue': 0.00480266899429282,
'muscle_wasting': 0.00790611607210145,
'vomiting': 0.01223386911022589,
'burning_micturition': 0.003045690358981559,
'spotting_urination': 0.0057233252756286705,
'fatigue': 0.015317989905595474,
'weight_loss': 0.017722995399831284,
'restlessness': 0.007082547681380771,
'lethargy': 0.007606323176478849,
'patches_in_throat': 0.007283863352965087,
'cough': 0.008617594575101773,
'high_fever': 0.013597738613822798,
'sunken_eyes': 0.006591391294148255,

```

✓ 0s [62] rfc_results = []
      knn_results = []

✓ 10s [▶] 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 by one. The first value will be 0.020 and the last will be 0.008

0s [64] randomf = pd.DataFrame(data = rfc_results,columns=['Number of features','Training Accuracy','Testing Accuracy'])

0s randomf

	Number of features	Training Accuracy	Testing Accuracy
0	6	0.170732	0.166667
1	10	0.264990	0.261905
2	15	0.431402	0.428571
3	23	0.598831	0.595238
4	36	0.854929	0.857143
5	49	0.934959	0.928571
6	61	0.992124	0.976190

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.

```
[66] 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	6	0.166667	0.166667
1	10	0.261941	0.261905
2	15	0.430640	0.452381
3	23	0.594512	0.595238
4	36	0.854675	0.857143
5	49	0.934451	0.928571
6	61	0.992124	0.976190

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.

```
[69] len(to_drop)
```

```
40
```

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

```
X_new.head()
```

	chills	joint_pain	vomiting	fatigue	weight_loss	high_fever	breathlessness	sweating	headache	dark_urine	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...

5 rows × 49 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 Forest 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.

```
✓ [72] X1_train, X1_val, y1_train, y1_val = train_test_split(X_new, y_new, test_size=0.2)
0s      X1_test = X_test.drop(to_drop,axis = 1)
      y1_test = y_test

✓ [73] rfc_new = RandomForestClassifier()
0s      rfc_new.fit(X1_train, y1_train)

      ▾ RandomForestClassifier
      RandomForestClassifier()

✓ [74] y_pred = rfc_new.predict(X1_val)
0s      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  0.9352134146341463
The Validation Accuracy of the algorithm is  0.9359756097560976
The Testing Accuracy of the algorithm is 0.9523809523809523
```

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

```
✓ [75] knn_new = KNeighborsClassifier()
0s      knn_new.fit(X1_train, y1_train)

      ▾ KNeighborsClassifier
      KNeighborsClassifier()

✓ [76] y_pred = knn_new.predict(X1_val)
0s      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  0.9347052845528455
The Validation Accuracy of the algorithm is  0.9380081300813008
The Testing Accuracy of the algorithm is 0.9285714285714286
```

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 disease	Peptic ulcer disease
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)	Paralysis (brain hemorrhage)

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.

Outlier Detection:

✓
0s

▶

```
symptom_columns = train.columns[:-2]
z_scores = np.abs(zscore(train[symptom_columns]))
outlier_threshold = 3
outliers_df = pd.DataFrame(z_scores > outlier_threshold, columns=symptom_columns)
outlier_rows = train[outliers_df.any(axis=1)]
print(outlier_rows)
```

📄

4915	0	0	0	0	0	0
4916	0	0	0	0	0	0
4917	0	0	0	0	0	0
4918	0	0	1	0	0	0

...

coma

history_of_alcohol_consumption

blood_in_sputum

\

0

...

0

0

0

1

...

0

0

0

2

...

0

0

0

3

...

0

0

0

4

...

0

0

0

...

...

...

...

...

4914

...

0

0

0

4915

...

0

0

0

4916

...

0

0

0

4917

...

0

0

0

4918

...

0

0

0

0

palpitations

pus_filled_pimples

blackheads

scurring

\

1

0

0

0

0

2

0

0

0

0

3

0

0

0

0

4

0

0

0

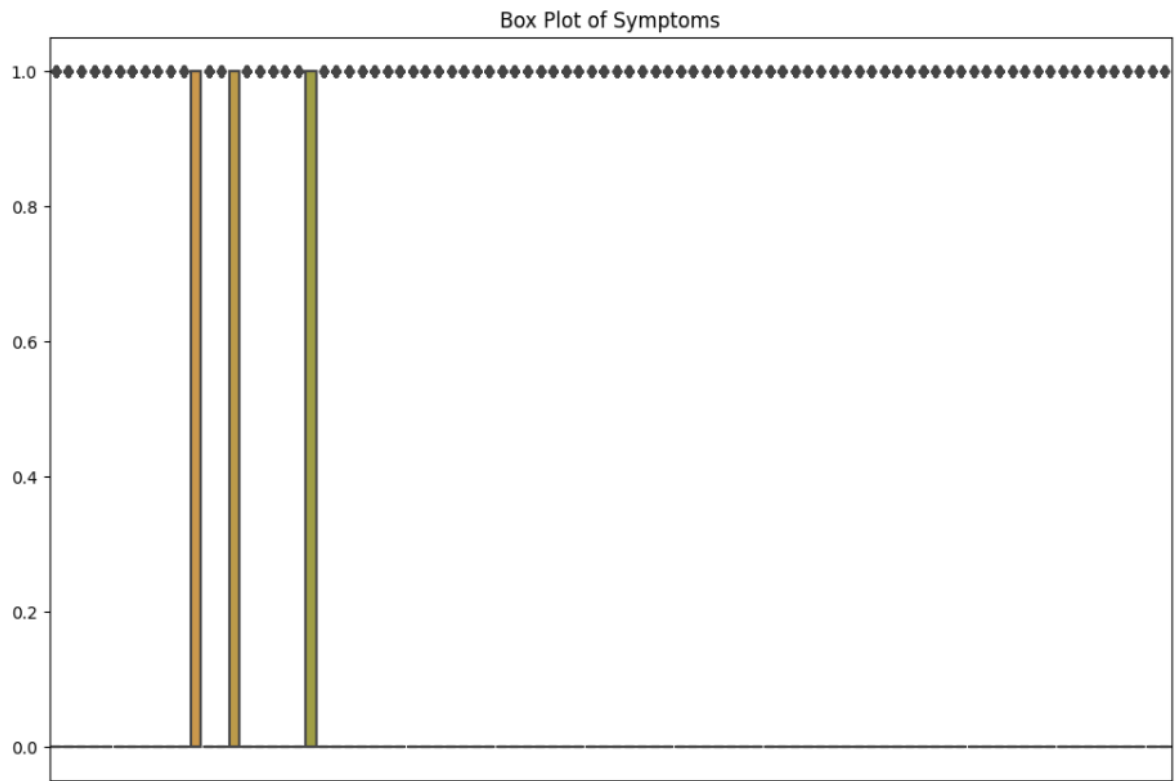
0

Outlier visualization:

Box Plot:

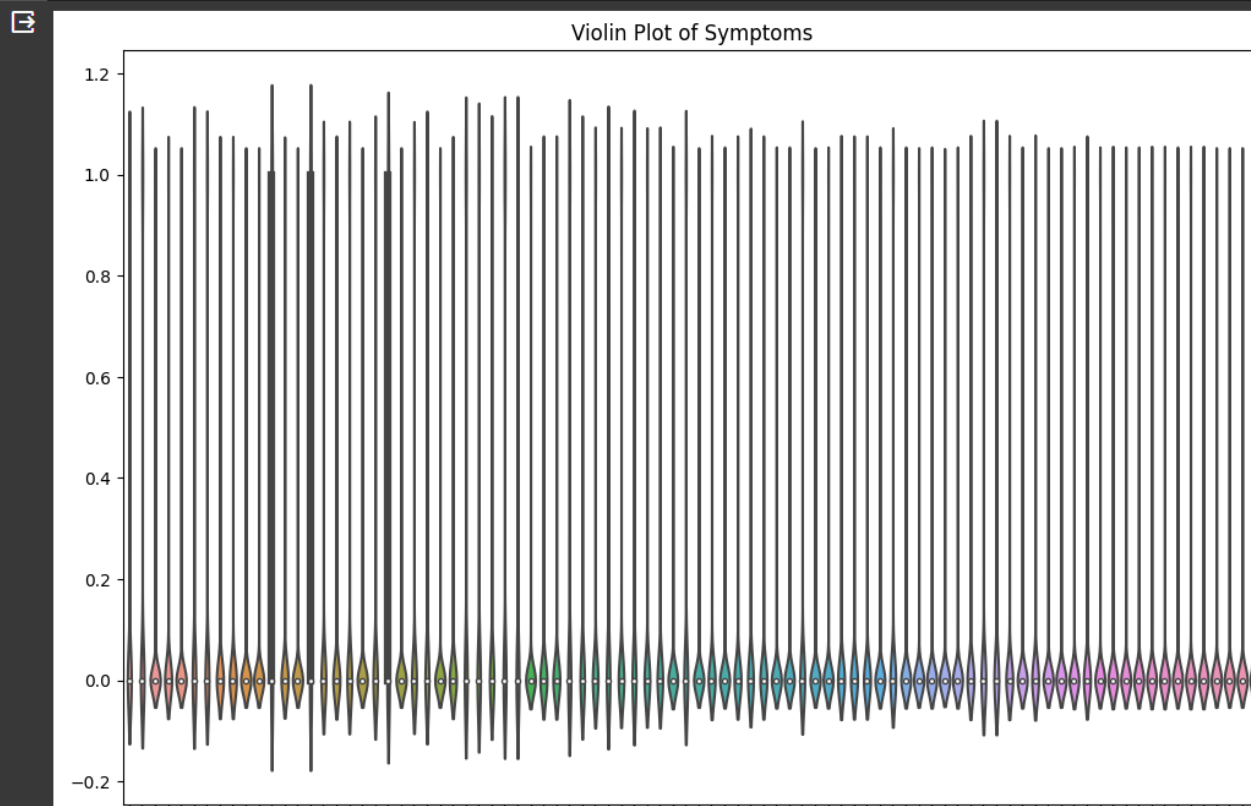
```
symptom_columns = train.columns[:-2]

plt.figure(figsize=(12, 8))
sns.boxplot(data=train[symptom_columns])
plt.title('Box Plot of Symptoms')
plt.show()
```



Violin Plot:

```
plt.figure(figsize=(12, 8))
sns.violinplot(data=train[symptom_columns])
plt.title('Violin Plot of Symptoms')
plt.show()
```



Milestone 6: Model Deployment

Activity 1: Save the best model

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

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

10.ADVANTAGES & DISADVANTAGES

Advantages:

The implementation of disease prediction using machine learning offers several significant advantages in the realm of healthcare:

Early Detection and Intervention:

One of the primary advantages is the ability to detect potential health issues at an early stage. Machine learning algorithms can analyze large datasets to identify patterns and risk factors, enabling early intervention and treatment.

Precision and Personalization:

Machine learning models can take into account a myriad of factors, including individual health data, genetics, and lifestyle choices. This allows for more precise and personalized predictions, tailoring healthcare strategies to the unique characteristics of each individual.

Optimized Resource Allocation:

Predictive models contribute to resource optimization in healthcare settings. By identifying individuals at higher risk, healthcare resources can be targeted more effectively, ensuring that interventions are directed towards those who are most likely to benefit.

Improved Patient Outcomes:

Timely and accurate disease predictions contribute to improved patient outcomes. Early detection, coupled with personalized interventions, can lead to better management of health conditions, reduced severity of diseases, and enhanced overall well-being.

Enhanced Public Health Strategies:

The insights generated by disease prediction models can inform public health strategies. Aggregated data can identify emerging health trends, guide policy decisions, and facilitate the implementation of targeted interventions at a broader population level.

Cost-Efficiency:

Early intervention and targeted resource allocation result in cost savings for healthcare systems. By preventing the progression of diseases to advanced stages, the financial burden associated with extensive treatments and hospitalizations can be mitigated.

Data-Driven Decision-Making:

Machine learning empowers healthcare professionals with data-driven insights, aiding them in making more informed and evidence-based decisions. This shift towards data-driven decision-making enhances the quality and efficiency of healthcare delivery.

Disadvantages:

Data Quality and Bias:

The accuracy of machine learning models heavily relies on the quality and representativeness of the training data. Biases present in historical healthcare data can lead to biased predictions, affecting certain demographic groups disproportionately.

Interpretability and Explainability:

Complex machine learning models, especially deep learning algorithms, are often challenging to interpret and explain. Lack of transparency in decision-making processes may limit the trust that healthcare professionals and patients place in the predictive models.

Ethical Concerns and Privacy Issues:

The use of sensitive health data raises ethical concerns, including issues related to patient consent, data ownership, and privacy. Ensuring compliance with ethical standards and regulations is crucial to maintaining trust in the healthcare system.

Over-Reliance on Technology:

Over-reliance on machine learning predictions without adequate clinical validation and human judgment may lead to inappropriate interventions or neglect of critical aspects of patient care. Human oversight is essential to ensure responsible and ethical use of predictive models.

Dynamic Nature of Health Conditions:

Health conditions and risk factors are dynamic, influenced by various factors such as lifestyle changes, environmental factors, and genetic variations. Machine learning models may struggle to adapt to these changes and require constant updating.

Understanding and addressing these disadvantages is essential to ensure the responsible and effective implementation of machine learning in disease prediction. Balancing the benefits of predictive analytics with ethical considerations, privacy protection, and ongoing validation is crucial for the success of these technologies in the healthcare domain.

11.CONCLUSION

In conclusion, the integration of machine learning in disease prediction represents a transformative paradigm for healthcare, offering a multitude of advantages while posing certain challenges that demand careful consideration. The ability to detect diseases at early stages, personalize interventions, optimize resource allocation, and inform public health strategies underscores the potential of machine learning to significantly improve patient outcomes and healthcare efficiency. However, the reliance on historical data introduces challenges related to biases, interpretability, and ethical concerns, necessitating a balanced approach that combines technological advancements with ethical considerations and human oversight.

12.FUTURE SCOPE

The future scope of disease prediction using machine learning holds tremendous potential for further advancements and positive impacts on healthcare. Several key areas present exciting opportunities for development and exploration:

Integration of Multi-Modal Data:

Future efforts can focus on integrating diverse datasets, including genetic information, wearable device data, and social determinants of health. This holistic approach can provide a more comprehensive understanding of an individual's health, enhancing the accuracy and scope of disease prediction models.

Explainable AI and Model Transparency:

Addressing the interpretability challenge is crucial. Future research can focus on developing machine learning models with increased transparency and explainability, allowing healthcare professionals and patients to better understand and trust the predictions, ultimately improving the acceptance and adoption of these technologies.

Continuous Learning and Adaptive Models:

Implementing machine learning models that can adapt and learn from real-time patient data updates presents a promising avenue. Continuous learning models can dynamically adjust to changing health conditions, ensuring ongoing accuracy and relevance in predicting disease risks.

Precision Medicine Advances:

Further advancements in precision medicine can be achieved by tailoring disease prediction models to individual genetic profiles, lifestyle choices, and environmental exposures. This approach can lead to more personalized and effective preventive strategies and treatments.

Population Health Management:

Disease prediction models can be integrated into broader population health management strategies. Predictive analytics can help healthcare systems identify and address health disparities, allocate resources efficiently, and implement targeted interventions at the population level.

13.APPENDIX:

The drive link for Website Files:

<https://drive.google.com/drive/u/1/folders/1-yCmF7KkJyZxzWWXlPi9cMHQz2MtVnv?authuser=0>

The github link :

<https://github.com/PSriSaiYagnik/SI-GuidedProject-603168-1697561363>

Demo Link:

<https://drive.google.com/file/d/1gvGjIDSmzuYhAfjuaetFQgL54B9UaYHk/view?usp=sharing>