Disease Prediction Using Machine Learning

Project Report Format

1. INTRODUCTION

1.1 Project Overview

Machine learning has emerged as a powerful tool in the healthcare domain, offering the potential to revolutionize disease prediction and transform patient care. By analyzing vast datasets of patient records, machine learning algorithms can identify intricate patterns and correlations between symptoms, medical history, and other factors, enabling them to predict the likelihood of various diseases with remarkable accuracy. This invaluable information can then be leveraged to inform clinical decision-making, improve patient outcomes, and optimize healthcare resource allocation.

Project Objectives

The overarching goal of this project is to develop a robust machine learning model capable of accurately predicting the probability of various diseases based on a patient's symptoms and medical history. This model should be able to encompass a wide spectrum of symptoms and diseases, effectively generalizing to new data that was not part of its training dataset.

Expected Outcomes

The successful development of this machine learning model is anticipated to yield a multitude of positive outcomes:

Enhanced Disease Prediction: The model will accurately predict the likelihood of various diseases, empowering healthcare providers to make informed decisions about diagnosis, treatment, and patient management.

Improved Clinical Decision-Making: The model's insights will guide healthcare providers towards optimal clinical decisions, leading to improved patient outcomes and reduced healthcare costs.

Early Disease Detection: Accurate disease prediction at an early stage will facilitate timely interventions and better treatment outcomes, potentially saving lives.

Personalized Medicine: The model's predictions will enable personalized treatment plans, tailoring interventions to each patient's risk factors and disease profile.

1.2 PURPOSE

Machine learning has emerged as a transformative force in the realm of healthcare, revolutionizing the way we diagnose, treat, and prevent diseases. Disease prediction, a cornerstone of this revolution, utilizes machine learning algorithms to analyze vast datasets of

patient records, extracting intricate patterns and correlations between symptoms, medical history, and other factors. This enables the prediction of the likelihood of various diseases with remarkable accuracy, opening up a world of possibilities for improving patient outcomes and optimizing healthcare delivery.

Purpose 1: Early Disease Detection and Intervention

One of the most compelling purposes of disease prediction using machine learning lies in its ability to detect diseases at their earliest stages, when intervention is most effective. By identifying individuals at high risk of developing certain diseases, clinicians can proactively implement preventive measures, lifestyle changes, or early treatment strategies, potentially altering the course of the disease and improving long-term outcomes. For instance, machine learning models can analyze genetic predispositions, medical history, and environmental factors to identify individuals at risk of developing cardiovascular diseases, allowing for early interventions such as diet modifications, exercise regimens, or cholesterol-lowering medications.

Purpose 2: Personalized Medicine and Risk Stratification

Machine learning-based disease prediction empowers personalized medicine by enabling tailored treatment plans and risk stratification. By analyzing individual patient data, machine learning models can identify unique risk profiles and predict the likelihood of adverse reactions to specific medications or treatment modalities. This information can guide clinicians in selecting the most appropriate treatment approaches for each patient, maximizing efficacy and minimizing potential side effects. For example, machine learning can predict the likelihood of chemotherapy-induced cardiotoxicity in cancer patients, allowing clinicians to personalize treatment plans and mitigate adverse effects.

Purpose 3: Optimizing Healthcare Resource Allocation

Disease prediction models can play a crucial role in optimizing healthcare resource allocation by identifying individuals at high risk of requiring costly or invasive procedures. By predicting the likelihood of hospitalizations, emergency department visits, or intensive care unit admissions, healthcare providers can anticipate resource needs and allocate them effectively. This proactive approach can help reduce unnecessary healthcare expenses and improve overall patient care.

Purpose 4: Improving Clinical Decision-Making and Patient Outcomes

Machine learning-based disease prediction models serve as valuable tools for clinical decision-making, providing clinicians with data-driven insights to guide their diagnostic and treatment decisions. By analyzing patient data and predicting disease probabilities, these models can assist clinicians in identifying potential diagnoses, evaluating treatment options, and monitoring patient progress. This information can lead to more informed clinical decisions, improved patient outcomes, and reduced diagnostic errors.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Despite the significant advancements in machine learning-based disease prediction, several challenges remain:

Data Quality and Availability: The accuracy of machine learning models is heavily dependent on the quality and availability of data. Limited access to high-quality, well-curated patient data can hinder the development of robust and generalizable models.

Data Bias and Fairness: Machine learning models can inadvertently perpetuate biases present in the data they are trained on, leading to unfair or discriminatory predictions. Ensuring that data is representative and free from biases is crucial for developing fair and equitable disease prediction models.

Model Interpretability and Explainability: The decision-making process of complex machine learning models can be opaque, making it difficult for clinicians to understand the rationale behind their predictions. This lack of transparency can hinder trust and adoption in clinical settings.

Continuous Model Adaptation and Improvement: As medical knowledge and treatment approaches evolve, machine learning models need to adapt and improve continuously. This requires ongoing data collection, model retraining, and evaluation to maintain the accuracy and relevance of these models.

Integration into Clinical Workflows: Integrating machine learning-based disease prediction models into existing clinical workflows can be challenging. Clinicians may need additional training to understand and utilize these models effectively, and integrating them into healthcare systems requires careful consideration of privacy, security, and regulatory compliance.

Addressing Ethical Considerations: The use of machine learning in healthcare raises ethical concerns related to data privacy, algorithmic bias, and the potential for misdiagnosis or misinterpretation of predictions. Clear ethical guidelines and frameworks need to be established to ensure responsible and ethical use of machine learning in disease prediction.

Educating Healthcare Professionals: Healthcare professionals need to be educated and trained on the potential of machine learning in disease prediction, including its limitations and ethical considerations. This is crucial for ensuring the appropriate and effective use of these models in clinical practice.

Addressing Public Perception: Public perception and understanding of machine learning in healthcare are evolving. Addressing concerns about data privacy, algorithmic bias, and potential

misdiagnoses is essential for building trust and acceptance of these technologies among patients and the public.

2.2 References

2.3 PROBLEM STATEMENT DEFINITION

The task of predicting diseases using machine learning involves developing a robust machine learning model that can accurately determine the probability of developing various diseases for new patients based on their reported symptoms and medical history. The model should be capable of handling a diverse range of symptoms and diseases, effectively generalizing to new data that was not included in its training process. Additionally, the model should be interpretable, enabling healthcare providers to comprehend the underlying factors influencing its predictions.

This problem statement encompasses several key aspects:

Data Acquisition: Accessing a comprehensive dataset of patient records is crucial for training and evaluating the machine learning model. This dataset should include detailed information on patient symptoms, medical history, and corresponding disease diagnoses.

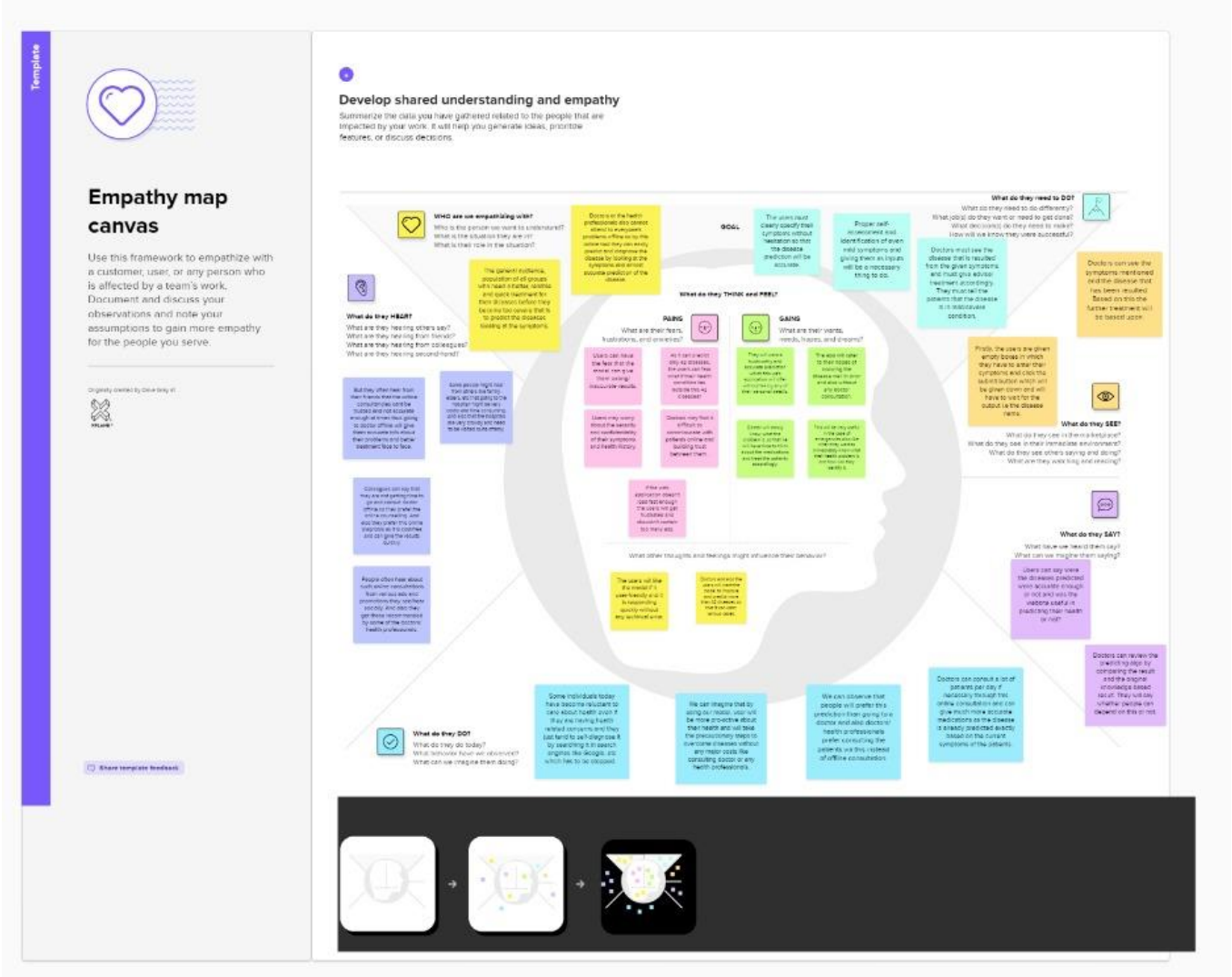
Model Development: Developing a machine learning model that can effectively identify patterns and associations within the patient data to predict disease likelihood. The model should be able to handle a variety of symptoms and diseases, ensuring its generalizability to unseen data.

Accuracy and Generalizability: The model should be able to accurately predict the likelihood of various diseases for new patients, demonstrating its effectiveness in real-world scenarios. The model should also be generalizable to new data that was not included in its training process.

Interpretability: The model's decision-making process should be transparent and understandable, allowing healthcare providers to grasp the rationale behind its predictions. This interpretability is essential for building trust and promoting the adoption of the model in clinical settings.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **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

- **Team gathering**
  Repakula Yasoda Rushitha
  Sreya Vanamadou
  Kalluri Karthikeya Gopala Abhinav
  Luv Sachdeva
- **Set the goal**
  Think about the problem you'll be focusing on solving in the brainstorming session.
- **Learn how to use the facilitation tools**
  Use the Facilitation Superpowers to run a happy and productive session.

Open article →

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

How can we predict the diseases by knowing the symptoms and this model can be helpful for both users(patients) and the doctors?

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.

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

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!

Repakula Yasoda Rushitha

Develop an user friendly interface so that the patients can easily detect their disease with their symptoms as inputs to the model.

This model will save the time of the doctor consultation and can directly get into the further treatment.

Healthcare providers, pharmaceutical companies improves their efficiency of their operations to reduce the cost by using this model.

Sreya Yerramaddu

This will help the medical sector especially the Doctors to easily detect the disease in a instance of time.

Anyone with certain symptoms can also give them as inputs to the model which gives the theory of the disease which helps the person to go for further treatment.

This model can be used to detect diseases anytime and anywhere no matter to whom it is being tested on.

Kalluri Karthikeya Gopala Abhinav

The model should cover all the mild to severe symptoms so that the prediction is accurate to their actual health problem.

The doctors must trust the model and must train the patients according to the theories that has been provided so that there won't be any confusion and time lag.

There must be proper security and confidentiality of the user's selected symptoms and their health status.

Luv Sachdeva

It can reduce the chances of misdiagnosis by providing doctors with additional data.

Models can aid in the early detection of diseases, potentially saving lives by identifying health issues before they become severe.

It will be useful to the users and the doctor if there is a database history of the usage of the model like when was the last time it was used and what was they diagnosed with.

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 and break it up into smaller sub-groups.

20 minutes

TIP

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

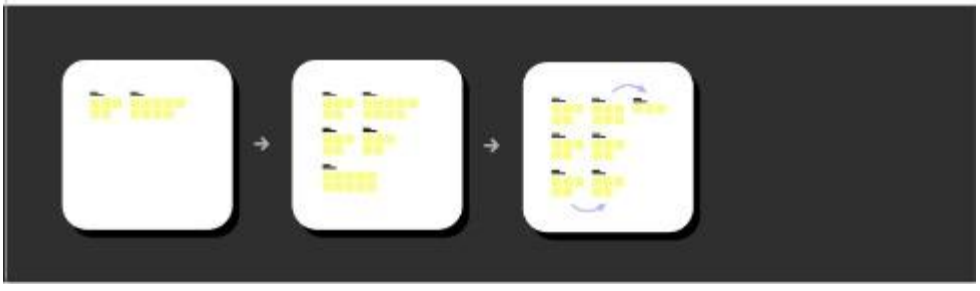
Develop an user friendly interface so that the patients can easily detect their disease with their symptoms as inputs to the model.

Anyone with certain symptoms can also give them as inputs to the model which gives the severity of the disease which helps the patient to go for further treatment.

There must be proper security and confidentiality of the user's selected symptoms and their health status.

The model should cover all the mild to severe symptoms so that the prediction is accurate to their actual health problem.

It will be useful to the users and the doctor if there is a database/ history of the usage of this model like when was the last time they used it and what was they diagnosed with.



4

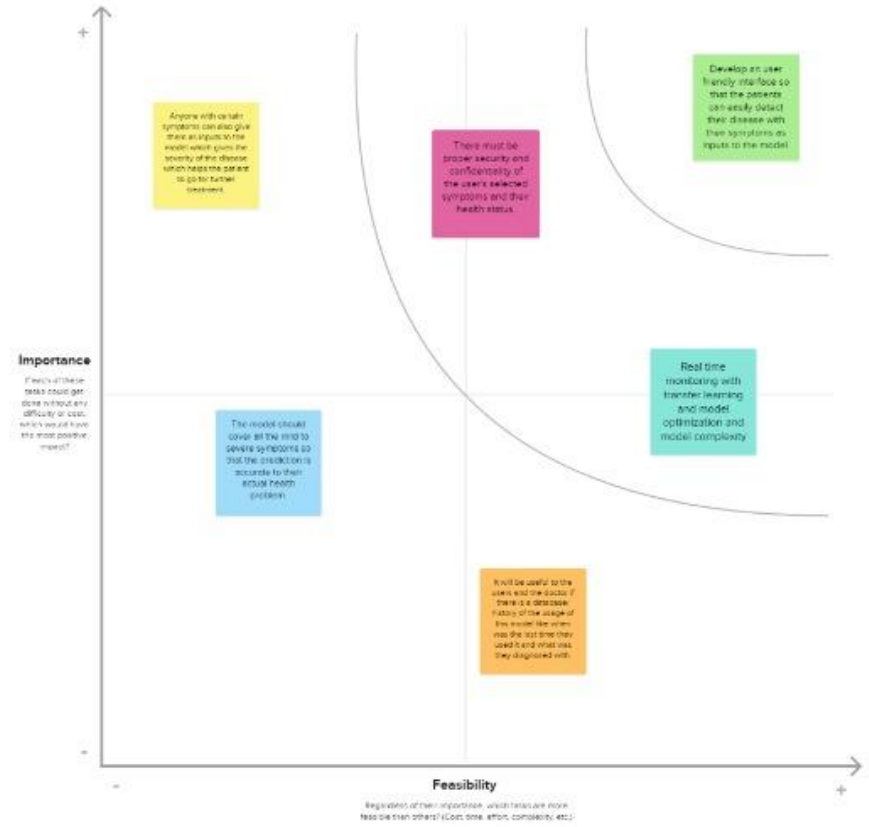
Prioritize

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.



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Model Interpretability: Ensure that the model's predictions can be interpreted and understood by healthcare professionals and end-users.

Explainability: Provide explanations for the model's predictions, especially for complex models like deep learning.

Data Security: Implement robust security measures to protect sensitive patient information.

User-Friendly Interface: Develop an intuitive and user-friendly interface for healthcare professionals and end-users.

Visualization: Provide visualizations that enhance the understanding of the model's predictions.

Clinical Validation: Validate the model's predictions with clinical experts to ensure its clinical relevance and utility.

Performance Metrics: Define and report relevant performance metrics, such as sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).

Algorithm Selection: Choose appropriate machine learning or statistical algorithms based on the nature of the data and the characteristics of the disease.

Training: Train the model using historical data, considering factors such as class imbalance and overfitting.

Validation: Implement validation techniques to assess the model's performance, such as cross-validation.

4.2 NON-FUNCTIONAL REQUIREMENTS

Performance Standards:
Make that the illness prediction system can process a certain throughput of prediction queries and responds to user requests in a timely manner within predetermined response times.

Scalability Conditions:
Specify the system's capacity to support an increasing number of users or requests in addition to handling increasing volumes of data for training and prediction.

Accuracy Conditions:

Indicate the lowest degree of accuracy that is acceptable for illness predictions, taking into account metrics like area under the receiver operating characteristic curve (AUC-ROC), precision, and recall.

Reliability Conditions:
Describe the system's fault tolerance, availability percentage, and capacity to carry on with proper operation even in the face of mistakes or failures.

Security requirements: include defining authentication and authorization procedures, putting safeguards in place to secure sensitive health data, and making sure data protection laws are followed.

Interoperability requirements: include compatibility with various platforms and devices, adherence to interoperability standards, and interaction with current healthcare information systems.

Usability requirements: include providing training materials and documentation to ensure effective system use and defining standards for user interface design to create a user-friendly experience.

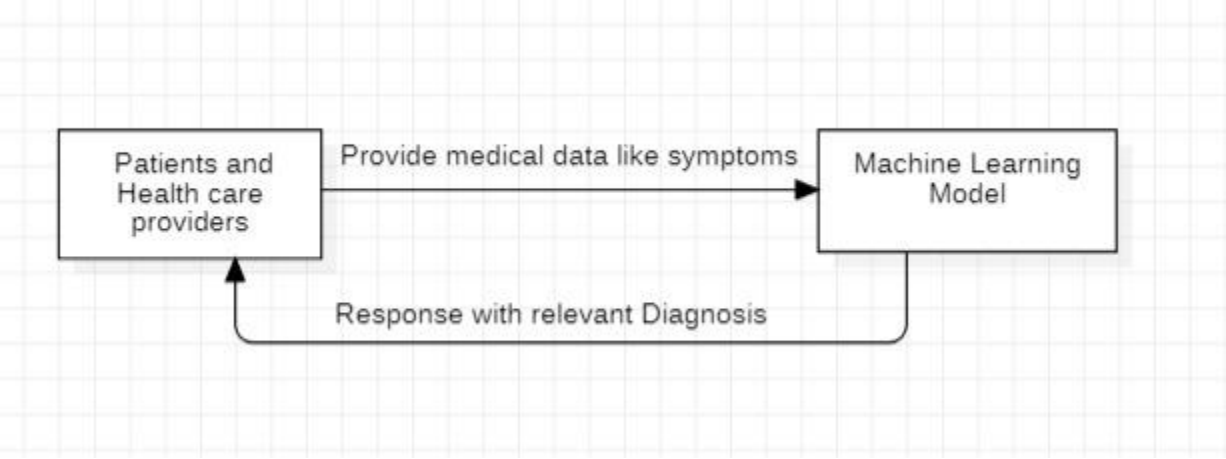
Requirements for Regulatory Compliance: Verify that the illness prediction system conforms to all applicable rules and guidelines in the healthcare industry.

Ability to Maintain Requirements: Establish coding guidelines, procedures, and documentation to make system changes and maintenance simpler.

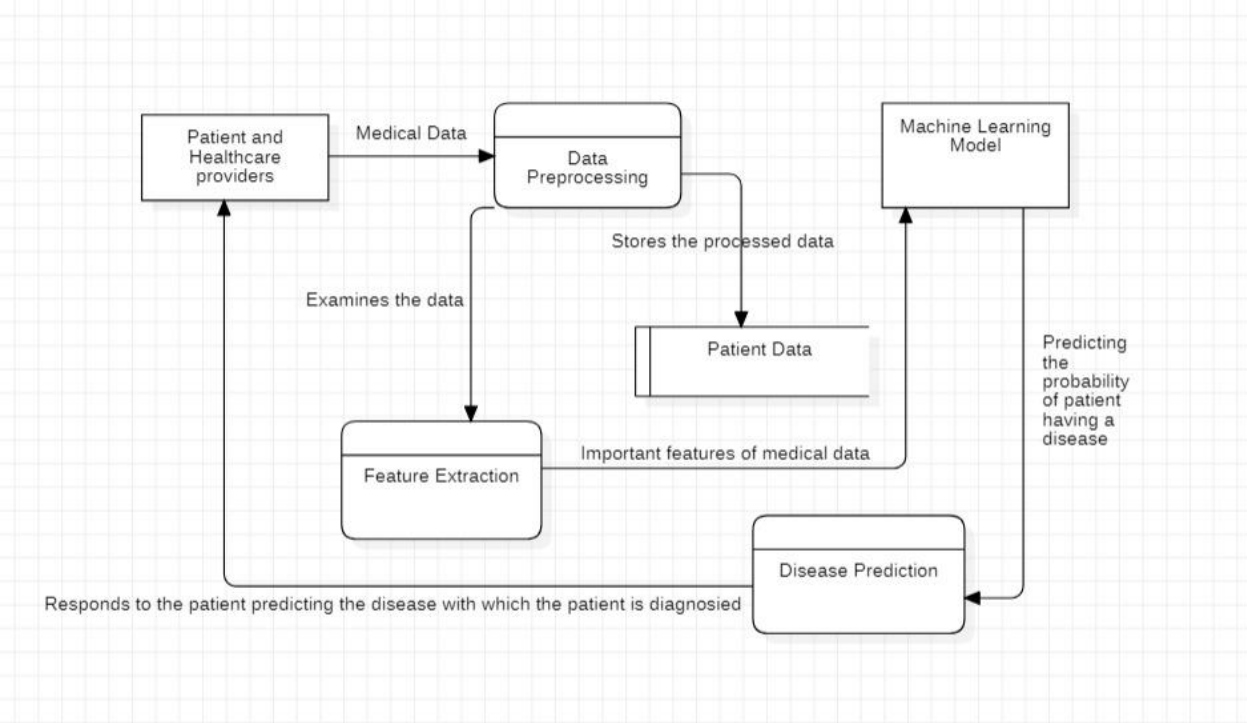
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS & USER STORIES

LEVEL 0-



LEVEL 1-



USER STORIES-

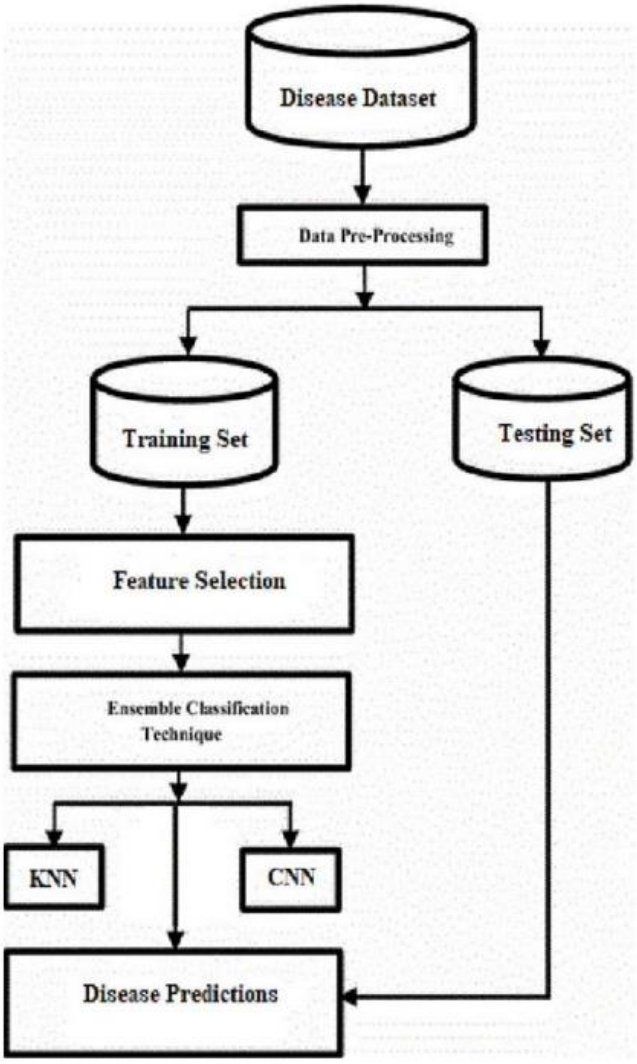
User Stories

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	Disease prediction	USN-1	As a patient, I want to be able to use a machine learning model to detect diseases based on my symptoms, so that I can get early diagnosis and treatment.	The machine learning model must achieve 90% accuracy in disease detection, be user-friendly for patients without technical background, and integrate into existing healthcare systems for efficient medical care.	High	Sprint-1
Healthcare Provider	Disease prediction	USN-2	The healthcare provider aims to utilize a machine learning model for more accurate and efficient disease diagnosis, thereby enhancing patient care.	The machine learning model should integrate with the EHR system, provide a list of possible diagnoses based on a patient's symptoms, medical history, and data, and provide information about the probability of each diagnosis, enabling more informed patient care decisions.	Medium	Sprint-1
Researcher	Disease prediction	USN-3	The researcher aims to train and evaluate machine learning models for disease detection, thereby developing improved methods for disease diagnosis.	The system should offer a platform for training and evaluating machine learning models on diverse disease datasets, provide metrics like accuracy, precision, recall, and F1-score.	High	Sprint-1
Healthcare System Administrator	Disease prediction	USN-4	As a healthcare system administrator, I aim to enhance patient care quality by deploying and managing machine learning models for disease detection.	The system should enable machine learning model deployment to production servers, monitor their performance, identify issues, and update them with new data and training.	Medium	Sprint-1
Public Health Official	Disease prediction for taking preventive steps	USN-5	As a public health official, I aim to utilize machine learning models to identify and monitor disease outbreaks, thereby preventing their spread.	The system must gather data from various sources, identify patterns, and generate alerts for public health officials to monitor and respond to disease outbreaks.	High	Sprint-2

Pharmaceutical company	Disease prediction	USN-6	The pharmaceutical company plans to utilize machine learning models to identify new drug targets and develop new treatments for disease, thereby improving patient lives.	The system must analyze large biological datasets to identify drug targets, generate new drug candidates for efficacy and safety, and predict clinical performance to make informed decisions about further drug development.	Medium	Sprint-2
Insurance Company	Disease prediction	USN-7	The insurance company aims to utilize machine learning models to predict disease risk in policyholders, thereby offering personalized and affordable insurance plans.	The system must predict disease risk with 80% accuracy, consider factors like age, gender, medical history, and lifestyle habits, and generate personalized risk assessments for policyholders.	Medium	Sprint-1
Medical Device Manufacturer	Disease prediction	USN-8	The manufacturer aims to utilize machine learning models to create advanced medical devices capable of detecting diseases earlier and more accurately, enabling timely patient treatment.	The system must develop medical devices with a 95% disease detection accuracy, be easy to use, affordable, safe, and effective for patients.	High	Sprint-1

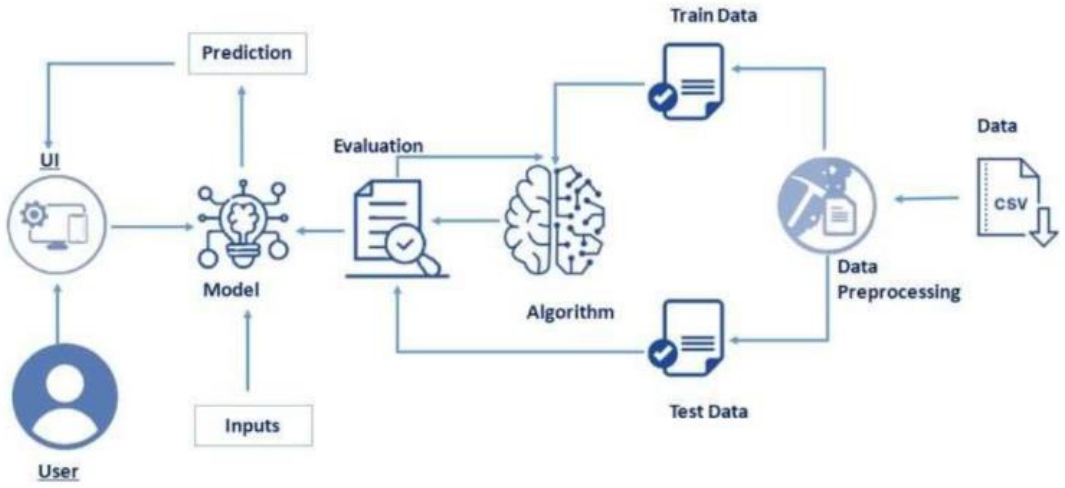
5.2 SOLUTION ARCHITECTURE



6. PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECTURE

Technical Architecture:



6.2 Sprint Planning & Estimation

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	08 November 2023
Team ID	592988
Project Name	Disease Prediction using Machine Leaning
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Disease Prediction	USN-1	The patient aims to utilize a machine learning model to identify and treat diseases based on their symptoms for early diagnosis and treatment.	2	High	Yasoda Rushitha
Sprint-1	Disease Prediction	USN-2	The healthcare provider plans to utilize a machine learning model for more precise and efficient disease diagnosis, thereby enhancing patient care.	1	Medium	Sreya Yarramaddu
Sprint-1	Disease Prediction	USN-3	The researcher aims to develop improved methods for disease diagnosis by training and evaluating machine learning models for disease detection.	2	High	Abhinav Kalluri
Sprint-2	Disease Prediction	USN-4	The public health officer aims to use machine learning models to identify and monitor disease outbreaks, thereby effectively preventing their spread.	2	High	Luv Sachdeva
Sprint-2	Disease Prediction	USN-5	Pharmaceutical company uses machine learning to identify drug targets and develop disease treatments, enhancing patient lives.	1	Medium	Sreya Yarramaddu

6.3 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	5	6 Days	01 Nov 2023	05 Nov 2023	20	05 Nov 2023
Sprint-2	5	6 Days	06 Nov 2023	11 Nov 2023		
Sprint-3	5	6 Days	12 Nov 2023	17 Nov 2023		
Sprint-4	5	6 Days	18 Nov 2023	23 Nov 2023		

Velocity:

Imagine we have a 20-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)

$$AV = \frac{Sprint\ Duration}{velocity} = \frac{24}{20} = 1.2$$

7. CODING & SOLUTIONING (EXPLAIN THE FEATURES ADDED IN THE PROJECT ALONG WITH CODE)

Activity 1: IMPORTING THE LIBRARIES:

```
0s
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

import pickle
```

Here we are importing the pandas library as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns, and various modules from scikit-learn and imbalancedlearn libraries for the tasks such as model selection, preprocessing, and machine learning.

Activity 1.1: Reading 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

1) The train.head() command is used to display the first few rows of Pandas DataFrame names train.

```
[17] train = pd.read_csv("/content/Disease-Prediction-ML-Flask/Data/Training.csv")
     test = pd.read_csv("/content/Disease-Prediction-ML-Flask/Data/Testing.csv")
```

```
train.head()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain
0	1	1	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0
2	1	0	1	0	0	0	0	0
3	1	1	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0

5 rows × 134 columns

2)train.shape is used to display the number of rows and columns.

```
[19] 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

Activity 2.1: Removing Redundant Columns

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

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

```
[21] 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 train.isnull().sum(): The train.isnull().sum() expression is used to calculate the

total number of null (missing) values in each column of the DataFrame train.

```
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
```

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

0
```

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

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.

The train.describe() is a Pandas DataFrame method that provides statistical summaries of the numerical columns in the DataFrame.

```
train.describe()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pair
count	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
std	0.344730	0.366417	0.146539	0.207593	0.146539	0.368667	0.346007
min	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
50%	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
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 132 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:

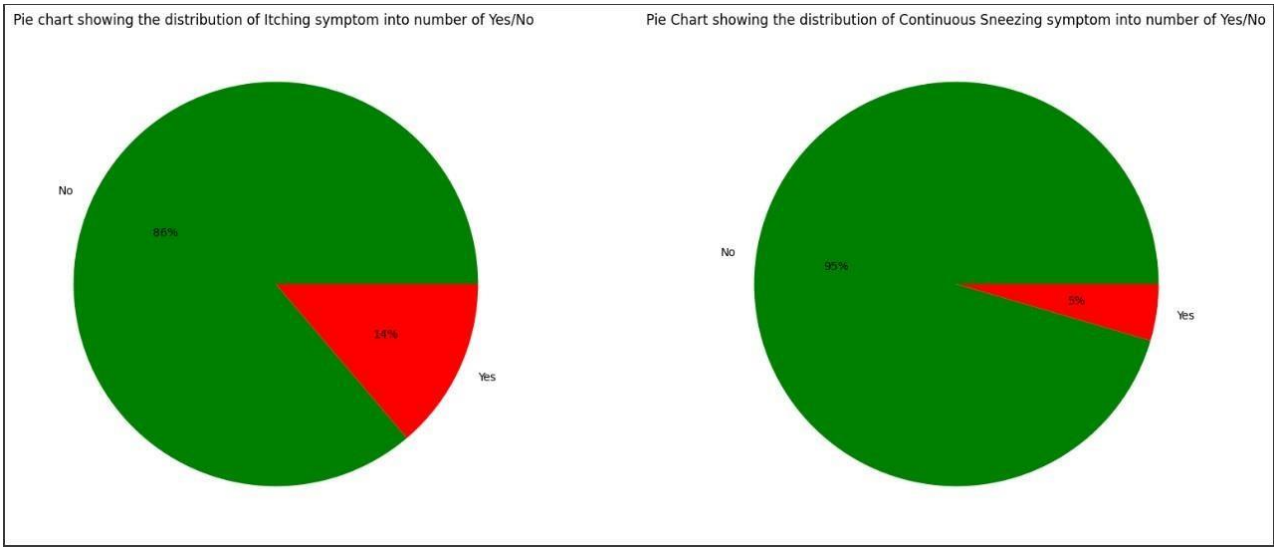
In simple words, univariate analysis is understanding the data with a single feature. We have displayed three different types of graphs and plots. For simple visualizations we can use the matplotlib.pyplot library

```
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 = '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)
```



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

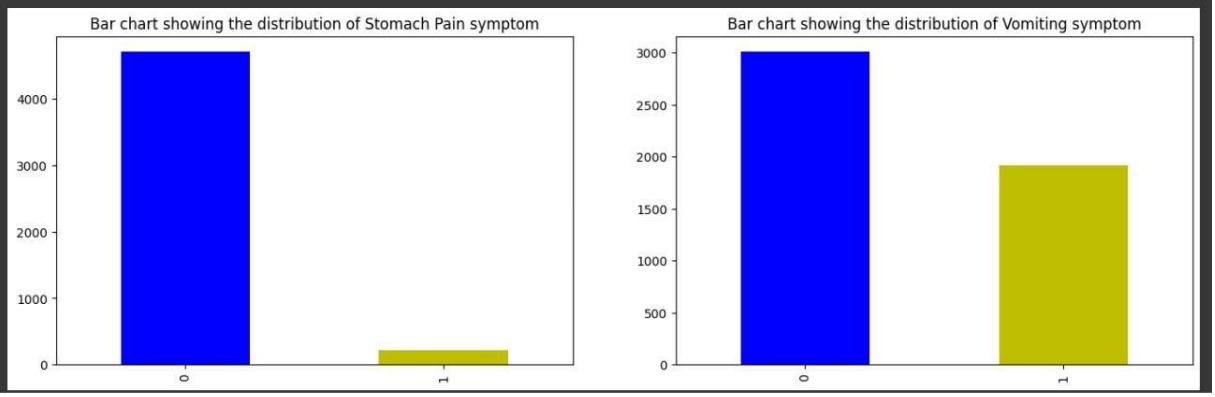
The pie plot on the left shows the different values distribution in the Itching column. It shows that there are 86% observations where the itching symptom has value 0 and there are 14% observations where the itching symptom has value 1.

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.

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

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

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



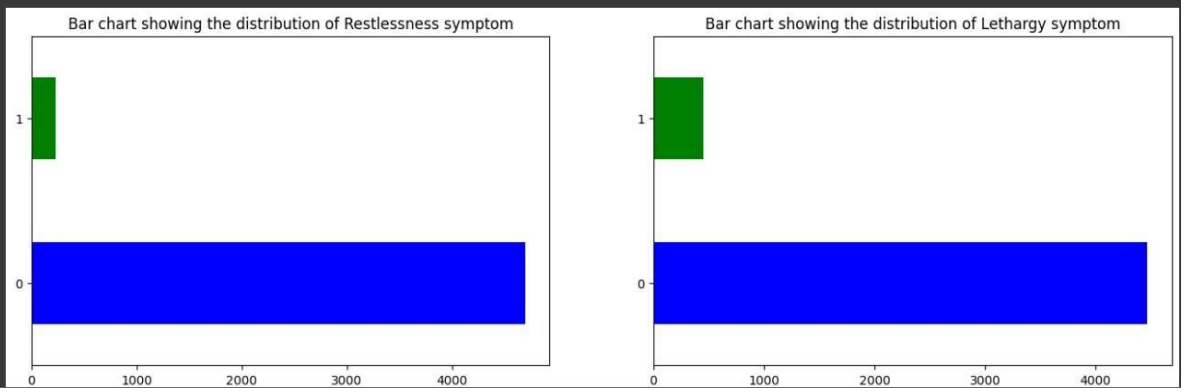
Here we have plotted 2 bar graphs. These bar graphs can be plotted without using any external library. We divide the plot into two subplots using subplot function from the matplotlib.pyplot library. We have plotted the bar graph using the inbuilt plot function in python.

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 = ['b','g'])
plt.title("Bar chart showing the distribution of Restlessness symptom")

plt.subplot(1,2,2)
train['lethargy'].value_counts().plot(kind = 'barh', color = ['b','g'])
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.

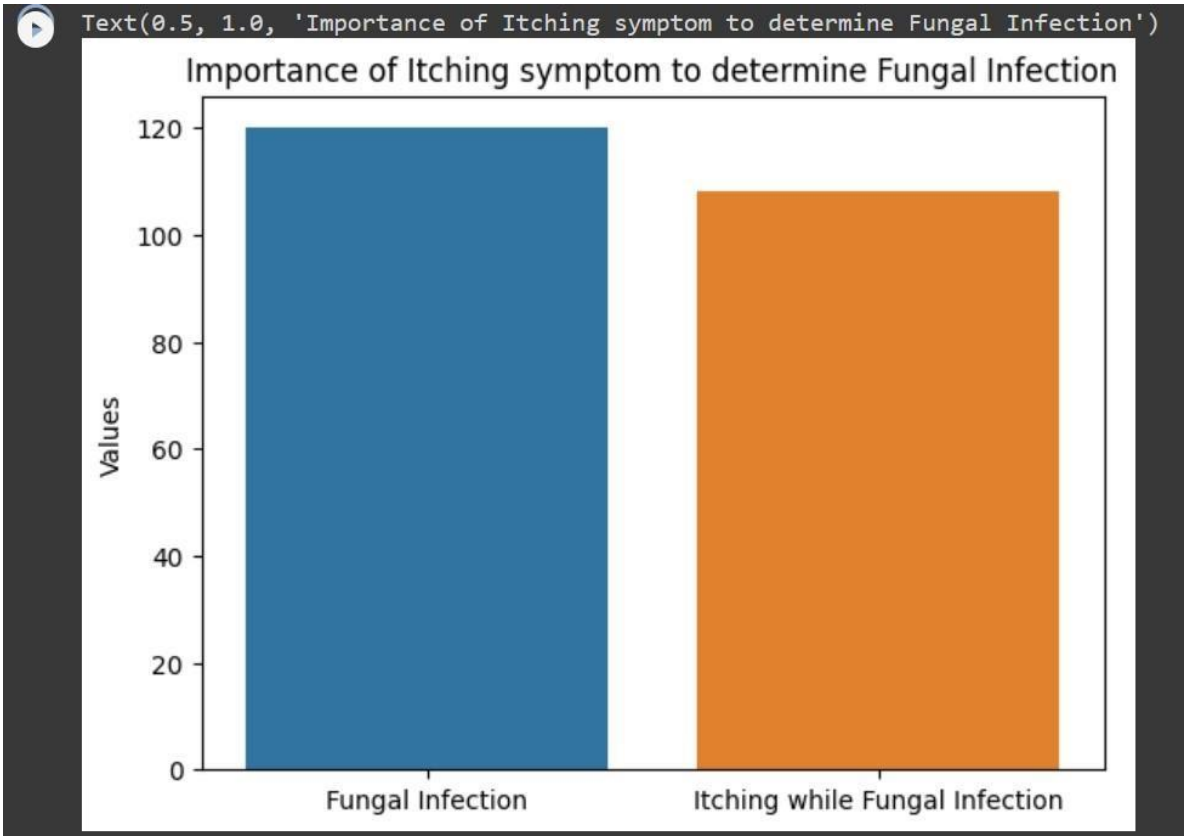
Activity 2.2: Bivariate Analysis:

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'] == '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 Infectio

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')
```



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.

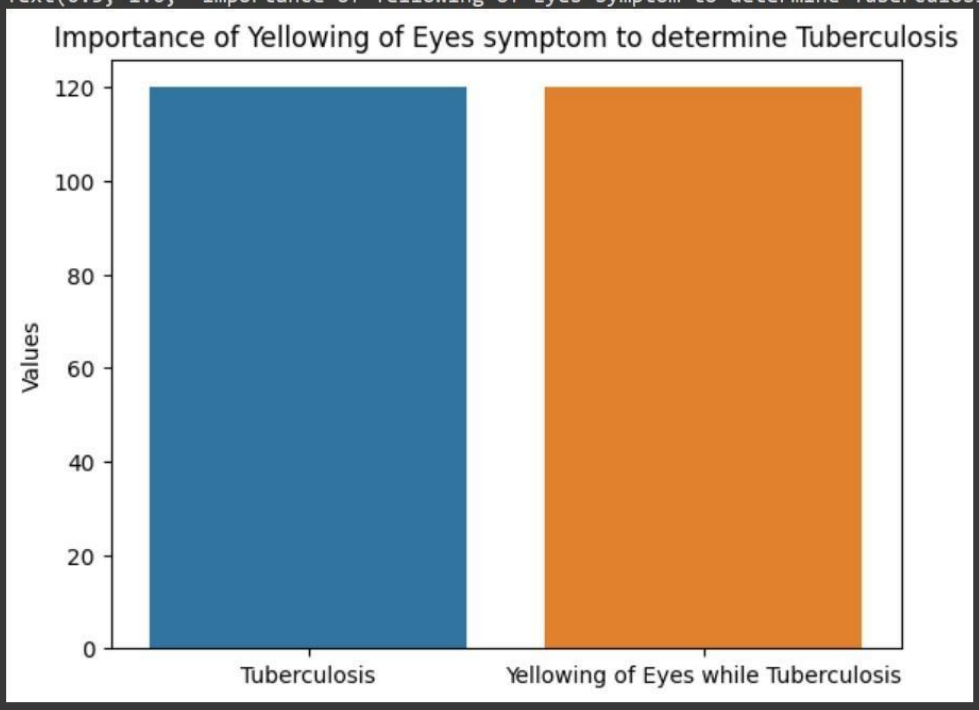
Next we have also seen the relationship between prognosis when the disease is Tuberculosis and the symptom yellowing_of_eyes. . From the plot we can see that when there is Tuberculosis there is a high chance that the yellowing of eyes column has value 1. There are 120 values where the value is Tuberculosis 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

```
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 Tuberculo

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

```
Text(0.5, 1.0, 'Importance of Yellowing of Eyes symptom to determine Tuberculosis')
```



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')
```

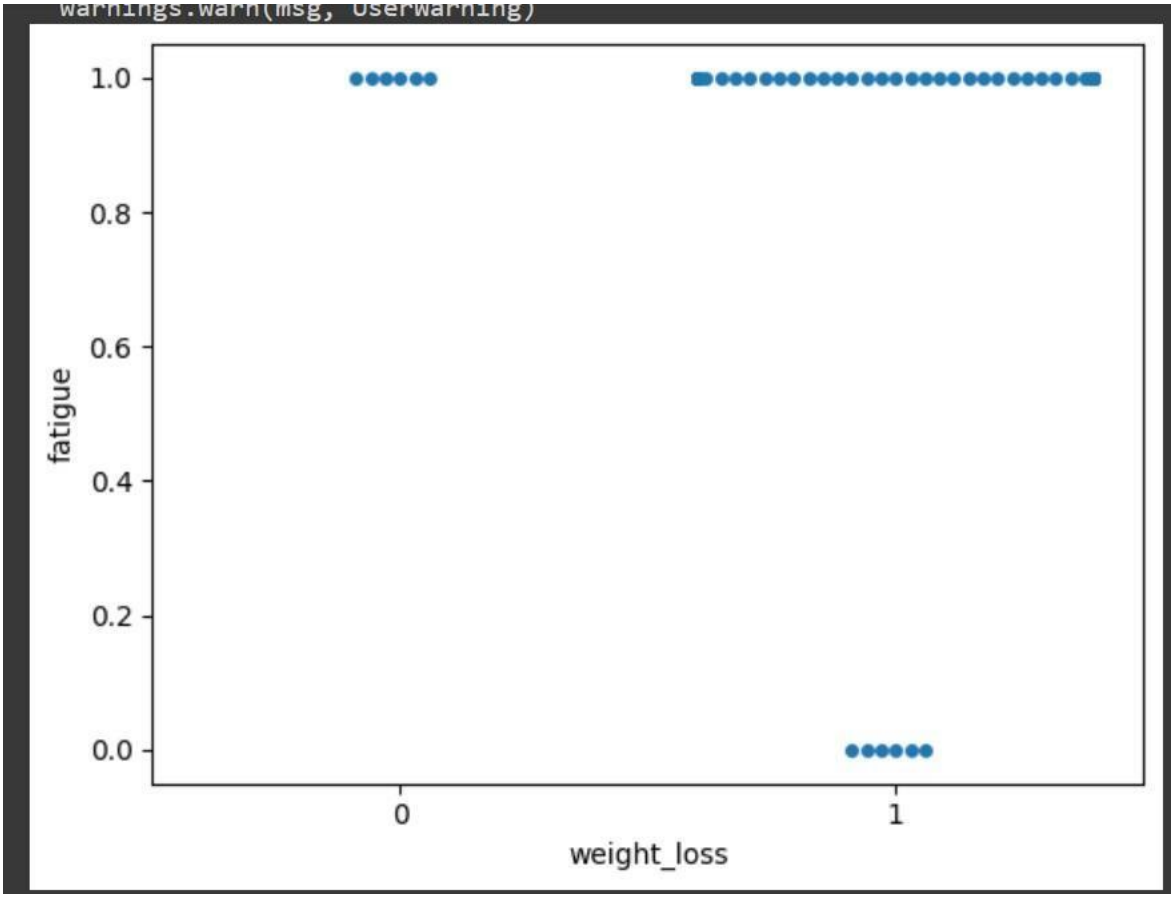
itching	1.000000	0.318158	0.326439	-0.086906	-0.059893	-0.175905	-0.160650	0.202850	-0.086
skin_rash	0.318158	1.000000	0.298143	-0.094786	-0.065324	-0.029324	0.171134	0.161784	-0.094
nodal_skin_eruptions	0.326439	0.298143	1.000000	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032
continuous_sneezing	-0.086906	-0.094786	-0.032566	1.000000	0.608981	0.446238	-0.087351	-0.047254	-0.047
shivering	-0.059893	-0.065324	-0.022444	0.608981	1.000000	0.295332	-0.060200	-0.032566	-0.032
chills	-0.175905	-0.029324	-0.065917	0.446238	0.295332	1.000000	-0.004688	-0.095646	-0.095
joint_pain	-0.160650	0.171134	-0.060200	-0.087351	-0.060200	-0.004688	1.000000	-0.087351	-0.087
stomach_pain	0.202850	0.161784	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	1.000000	0.433
acidity	-0.086906	-0.094786	-0.032566	-0.047254	-0.032566	-0.095646	-0.087351	0.433917	1.000
ulcers_on_tongue	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	0.649078	0.608
muscle_wasting	-0.059893	-0.065324	-0.022444	-0.032566	-0.022444	-0.065917	-0.060200	-0.032566	-0.032
vomiting	-0.057763	-0.225046	-0.119543	-0.173459	-0.119543	0.144263	0.199921	0.031406	0.019
burning_micturition	0.207896	0.166507	-0.032103	-0.046581	-0.032103	-0.094285	-0.086108	0.412239	-0.046

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

```
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

```
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.

```
[53] 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.

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

```
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: 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.

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

```
[57] 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.

8. PERFORMANCE TESTING

8.1 Performace Metrics

Milestone 5 : Performance Testing & Hyperparameter Tuning

```
a = rfc.feature_importances_
```

```
[68] col = X.columns
```

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

```
[70] feat_imp
```

```
 'pain_behind_the_eyes': 0.01319824851771783,
 'back_pain': 0.01689300502268126,
 'constipation': 0.012293307899228953,
 'abdominal_pain': 0.007626042803453611,
 'diarrhoea': 0.009261739845267866,
 'mild_fever': 0.01985800908553994,
 'yellowing_of_eyes': 0.016565298512947028,
 'swelled_lymph_nodes': 0.01657807305537706,
 'malaise': 0.007187502891558174,
 'blurred_and_distorted_vision': 0.008410478000376021,
 'phlegm': 0.006391219937020358,
 'congestion': 0.022700557966486682,
 'chest_pain': 0.00952889415930193,
 'weakness_in_limbs': 0.007195358230920688
```

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.

```
[72] knn_results = []

[97] 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
         knn_new = KNeighborsClassifier()
         knn_new.fit(X1_train, y1_train)
         temp1 = model_evaluation1(X1_train.shape[1],knn_new)
         knn_results.append(temp1)
```

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.

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.

```
[98] 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	0.168953	0.166667
1	10	0.312500	0.309524
2	14	0.387449	0.380952
3	26	0.713669	0.714286
4	36	0.839939	0.833333
5	46	0.911585	0.904762
6	64	0.990600	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.

```
len(to_drop)
```

```
43
```

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

```
[81] X_new.head()
```

	joint_pain	vomiting	fatigue	weight_loss	high_fever	headache	yellowish_skin	dark_urine	nausea	pain_behind_the_eyes	...	rusty_sputum	lack_
0	0	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 46 columns

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

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

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

+ Code + Text

```
[101] 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  0.9923780487804879
The Validation Accuracy of the algorithm is  0.9939024390243902
The Testing Accuracy of the algorithm is 0.9761904761904762
```

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	Drug Reaction
5	Peptic ulcer diseae	Peptic ulcer diseae
6	AIDS	AIDS
7	Diabetes	Diabetes

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.pk

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 ●

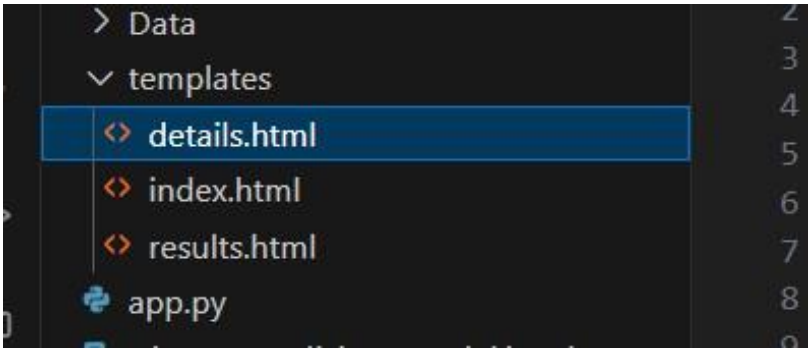
Run the web application

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.

```
11
12  model = pickle.load(open('model.pkl','rb'))
13  app = Flask(__name__)
14
```

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.

```
14
15  @app.route("/")
16  def home():
17      return render_template('index.html')
18
```

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".

```
18
19  @app.route('/details')
20  def pred():
21      return render_template('details.html')
22
```

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_

```
@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))

```

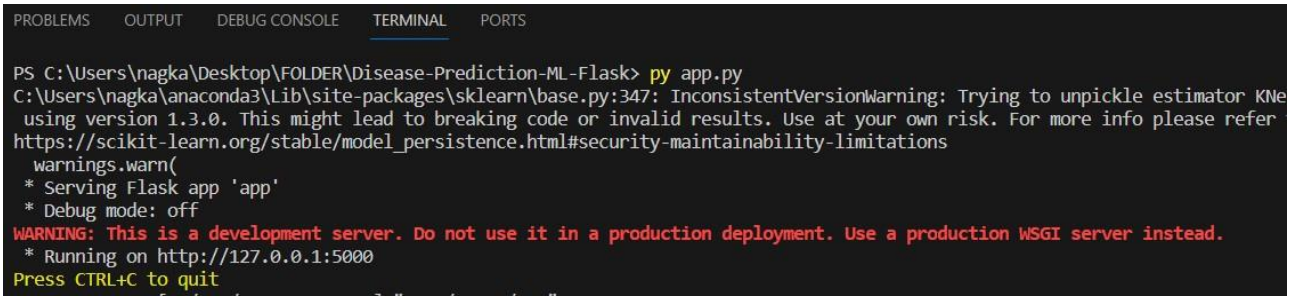
Main Function:

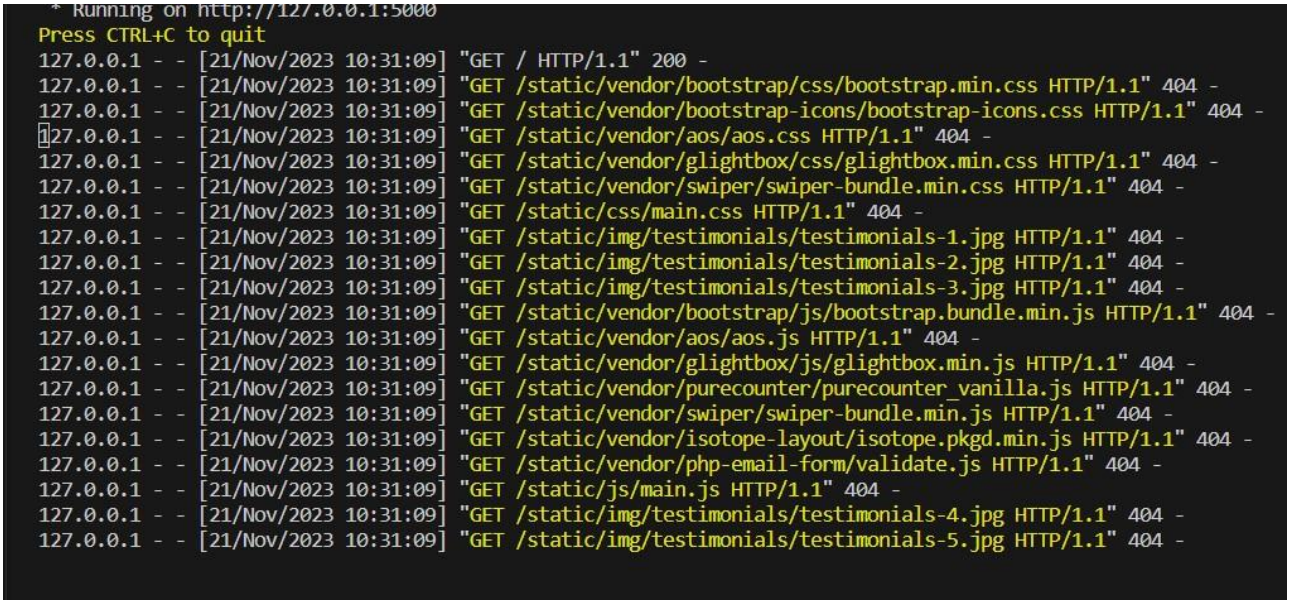
This code sets the entry point of the Flask application. The function “app.run()” is called, which starts the Flask deployment server.

```
51
52  if __name__ == "__main__":
53      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.





9. RESULTS

9.1 OUTPUT SCREENSHOTS

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 (homedoc.html)

Our details.html will be shown as(symptoms.html)

File | C:/Users/nagka/Desktop/MLWEBSITE/symptoms.html

DISEASE PREDICTION

Home Predict About Model Testimonals FAQ Contact

You will have the input box below where you can select your symptoms.
You can input the number of symptoms you have and leave the others blank.
This is the list of the symptoms. If you have symptoms which are from this list please enter the symptom in the same form as shown below.

Symptoms Column-1	Symptoms Column-2	Symptoms Column-3
itching	constipation	irritability
muscle_pain	abdominal_pain	continuous_sneezing
shivering	diarrhoea	puffy_face_and_eyes
joint_pain	mild_fever	abnormal_menstruation
stomach_pain	malaise	increased_appetite
vomiting	phlegm	lack_of_concentration
fatigue	congestion	visual_disturbances
weight_loss	chest_pain	receiving_blood_transfusion
restlessness	fast_heart_rate	pain_behind_the_eyes
lethargy	neck_pain	history_of_alcohol_consumption
high_fever	dizziness	blood_in_sputum
headache	belly_pain	yellowing_of_eyespalpitations

File | C:/Users/nagka/Downloads/doctor.html

DISEASE PREDICTION

Home Predict About Model Testimonals FAQ Contact

Disease prediction is a critical aspect of healthcare that involves using various techniques and tools to identify the likelihood of a person developing a particular medical condition. Predictive modeling in healthcare can be approached in different ways, incorporating clinical data, genetic information, lifestyle factors and more.

Symptom-1 Enter Symptom-1 Symptom-2 Enter Symptom-2 Symptom-3 Enter Symptom-3

Symptom-4 Enter Symptom-4 Symptom-5 Enter Symptom-5 Symptom-6 Enter Symptom-6

Symptom-7 Enter Symptom-7 Symptom-8 Enter Symptom-8 Symptom-9 Enter Symptom-9

Predict

Input-1:

Output-1:

The probable diagnosis says it could be Chicken pox

Input-2:

Output-2:

The probable diagnosis says it could be Dimorphic hemmorhoids(piles)

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES-

1) Early Detection: Machine learning makes it possible to analyze large datasets and identify patterns and subtle signs early on, which helps identify diseases in the early stages of development.

2) Customized medicine: Machine learning algorithms are capable of analyzing patient data to create customized treatment regimens that recommend actions based on environmental, lifestyle, and genetic factors.

3) Increased Accuracy: When compared to conventional techniques, machine learning models produce more accurate diagnoses and predictions because they are able to examine intricate correlations inside datasets.

4) Effective Data Analysis: Machine learning algorithms are capable of processing and analyzing vast amounts of heterogeneous healthcare data, such as medical records, photos, and genetic data, in a timely and effective manner.

5) Cost-Effective Healthcare: By lowering the need for pricey treatments and hospital stays, early disease prediction and prevention can reduce costs and increase the sustainability of healthcare.

6) Enhanced Decision Support: By helping with diagnosis, treatment planning, and prognostic evaluations, machine learning models give medical personnel useful tools for decision support.

7) Population Health Management: ML can be used to identify at-risk groups, assess trends in population health, and carry out focused disease prevention actions on a larger scale.

8) Continuous Monitoring: By evaluating real-time data and sending out timely alarms for possible health problems, machine learning algorithms can allow for the continuous monitoring of patients.

9) Data-Driven Insights: Machine learning advances knowledge of disease causes, risk factors, and treatment outcomes by gleaning insights from a variety of healthcare data sources.

10) Decreased Human mistake: The use of automation in disease prediction lowers the

possibility of human mistake that comes with manual diagnosis, producing more consistent and dependable outcomes.

DISADVANTAGES-

1) Data quality issues: The representativeness and quality of training data are crucial for machine learning models, and biased or lacking datasets might produce predictions that are not correct.

2) Difficulties with Interpretability: Complex machine learning models may not be comprehensible, which makes it hard for medical professionals to comprehend and have faith in the decision-making process.

3) Risks associated with overfitting: ML models may overfit to certain datasets, which would limit its applicability to novel or varied patient populations.

4) Ethical and Privacy Issues: When handling sensitive data, there may be questions regarding patient privacy and the moral use of healthcare data.

5) Dependency on Technology: If healthcare personnel lack the necessary training or if there are technological problems, relying too much on machine learning technology could present problems.

6) Resource Intensiveness: ML model development and maintenance can demand a lot of resources, including knowledge, processing power, and regular updates.

7) Limited Understanding of Causation: ML models frequently find correlations rather than causes, which makes it difficult to comprehend the fundamental mechanisms driving illness states.

8) Security Vulnerabilities: Using machine learning in healthcare comes with additional security risks, like the possibility of malevolent attacks on patient information and model databases.

9) Algorithm Bias: Machine learning methods may reinforce pre-existing biases in training data, resulting in differences in predictions among various demographic groups.

10) Limited Data Availability: The creation of precise and trustworthy prediction models may occasionally be hampered by the absence of adequate data for particular diseases.

11. CONCLUSION

In conclusion, our machine learning project for disease prediction has great potential to transform healthcare. Early detection, tailored treatment, and affordable healthcare are just a few benefits that show promise for greatly enhancing patient outcomes and general well-being. Still, we have to face the difficulties head-on. Prioritizing data quality and representativeness, resolving interpretability issues, and handling ethical issues are crucial. Maintaining the human touch while utilizing cutting-edge technology in healthcare delivery is essential, underscoring the necessity of continuing education for medical staff.

While the project speeds up medical research, improves decision support, and helps to population health management, it necessitates careful consideration of privacy and security concerns. Overcoming regulatory compliance difficulties and promoting widespread usage necessitate stakeholder collaboration. Because health data is dynamic, it necessitates continual model refining and adaption. As we manage these issues, our commitment to ethical AI methods, openness, and diversity will be critical in reaching the full promise of machine learning for disease prediction, ushering in a new era of preventive and personalized healthcare.

12. FUTURE SCOPE

1) Integration with Wearable Devices: In the future, data from wearable devices will be incorporated, enabling for real-time monitoring of vital signs and lifestyle aspects. This integration can improve forecast accuracy and provide a more thorough view of individual health.

2) Expanded Disease Coverage: By using breakthroughs in machine learning techniques and an ever-growing pool of healthcare data, the project can be expanded to forecast a broader spectrum of diseases. This broadening could include both common and rare disorders, resulting in a more comprehensive healthcare approach.

3) Explainable AI in Healthcare: Incorporating explainable AI techniques into the project can address interpretability issues, making forecasts more transparent and intelligible for healthcare professionals and patients. This is critical for establishing trust and general acceptance.

4) Continuous Learning and Adaptive Models: Using models that can constantly learn and adapt to changing health patterns guarantees that the system remains relevant throughout time. Regular updates based on growing medical information and shifting disease trends will improve prediction accuracy and effectiveness.

5) Telehealth Integration: As telehealth becomes more prevalent, including disease prediction systems into telehealth platforms can help with remote monitoring and early

intervention. This has the potential to improve healthcare accessibility, particularly in areas with little healthcare infrastructure.

6) AI-Driven Public Health Strategies: In the future, the project's scope will include providing insights for public health planning. Machine learning can help in the development of focused public health policies, the optimization of resource allocation, and the prediction and prevention of disease epidemics on a broader scale.

13. APPENDIX

GitHub & Project Demo Link

Link of the Project & Website Demonstration-

https://drive.google.com/file/d/1vsD0QGsu15-2ZtQQZRiJjYKDFeShrzRq/view?usp=sharing

Project GitHub Link-

https://github.com/smartinternz02/SI-GuidedProject-609651-1698733060

Notebook-

https://github.com/smartinternz02/SI-GuidedProject-609651-1698733060/blob/main/Project%20Development%20Phase/Disease_Prediciton_Model.ipynb