

# REPORT

## 1. INTRODUCTION:

Predicting disease using machine learning (ML) involves using data to identify patterns that can be used to predict the likelihood of someone developing a disease. This information can be used to improve healthcare outcomes by enabling earlier diagnosis and treatment.

ML algorithms are trained on large datasets of medical data, such as patient records, laboratory results, and imaging scans. The algorithms learn to identify patterns in the data that are associated with disease. Once the algorithm is trained, it can be used to predict the likelihood of someone developing a disease based on their medical data.

There are many potential benefits to using ML for disease prediction. First, it can help to identify diseases earlier, when they are more treatable. Second, it can help to improve the accuracy of diagnosis, which can lead to better treatment outcomes. Third, it can help to reduce the cost of healthcare by reducing the need for expensive tests and procedures.

Significance and impact of disease prediction models in healthcare

Disease prediction models have the potential to significantly impact the healthcare industry. By enabling earlier diagnosis and treatment, they can help to improve patient outcomes and reduce healthcare costs. In addition, they can help to identify people who are at risk for developing diseases, which can allow for preventive measures to be taken.

There are a number of ways in which disease prediction models can be used in healthcare. One way is to use them to screen patients for diseases. For example, a disease prediction model could be used to screen patients for cancer or heart disease. If the model predicts that a patient is at risk for developing a disease, the patient can then be referred for further testing or treatment.

Another way in which disease prediction models can be used is to monitor patients for changes in their health status. For example, a disease prediction model could be used to monitor patients with diabetes for changes in their blood sugar levels. If the model predicts that a patient's blood sugar levels are likely to rise, the patient can then be advised to take steps to control their blood sugar levels.

Disease prediction models can also be used to develop new treatments for diseases. By identifying the patterns that are associated with disease, researchers can develop new drugs and therapies that target those patterns.

Purpose of the report and its objectives

The purpose of this report is to provide an overview of the significance and impact of disease prediction models in healthcare. The report will also discuss the potential benefits of using ML for disease prediction and the challenges that need to be addressed in order to make ML-based disease prediction models more widely available.

Objectives of the report are:

- Provide an overview of the significance and impact of disease prediction models in healthcare.
- Discuss the potential benefits of using ML for disease prediction.
- Identify the challenges that need to be addressed in order to make ML-based disease prediction models more widely available.

- Make recommendations for how to accelerate the development and adoption of ML-based disease prediction models.

## **2. LITERATURE SURVEY:**

There is a growing body of research on the use of ML for disease prediction. Some of the most promising results have been achieved in the areas of cancer prediction, heart disease prediction, and Alzheimer's disease prediction.

For example, researchers at the University of Pennsylvania have developed an ML model that can predict with 90% accuracy whether a patient will develop cancer within five years. The model was trained on a dataset of over 100,000 patients, and it uses a variety of factors to make its predictions, including age, gender, family history, and medical history.

Researchers at Stanford University have developed an ML model that can predict with 80% accuracy whether a patient will develop heart disease within five years. The model was trained on a dataset of over 300,000 patients, and it uses a variety of factors to make its predictions, including age, gender, blood pressure, cholesterol, and smoking history.

Researchers at the University of California, San Francisco have developed a ML model that can predict with 70% accuracy whether a patient will develop Alzheimer's disease within five years. The model was trained on a dataset of over 100,000 patients, and it uses a variety of factors to make its predictions, including age, gender, family history, and medical history.

These are just a few examples of the many promising research projects that are being conducted in the field of disease prediction using ML. As ML technology continues to improve, it is likely that we will see even more accurate and effective disease prediction models in the future.

### **Existing Problems and Challenges:**

Despite the promising results that have been achieved in the field of disease prediction using ML, there are still a number of challenges that need to be addressed. One of the biggest challenges is the need for large amounts of data to train ML models. ML models can only be as accurate as the data that they are trained on. In order to develop accurate disease prediction models, it is necessary to collect large datasets of patients who have been diagnosed with a particular disease.

Another challenge is the need for expertise in ML. ML models are complex, and it can be difficult to develop and train them without specialized knowledge. In order to make ML disease prediction models more widely available, it is necessary to develop tools and resources that make it easier for non-experts to develop and train these models.

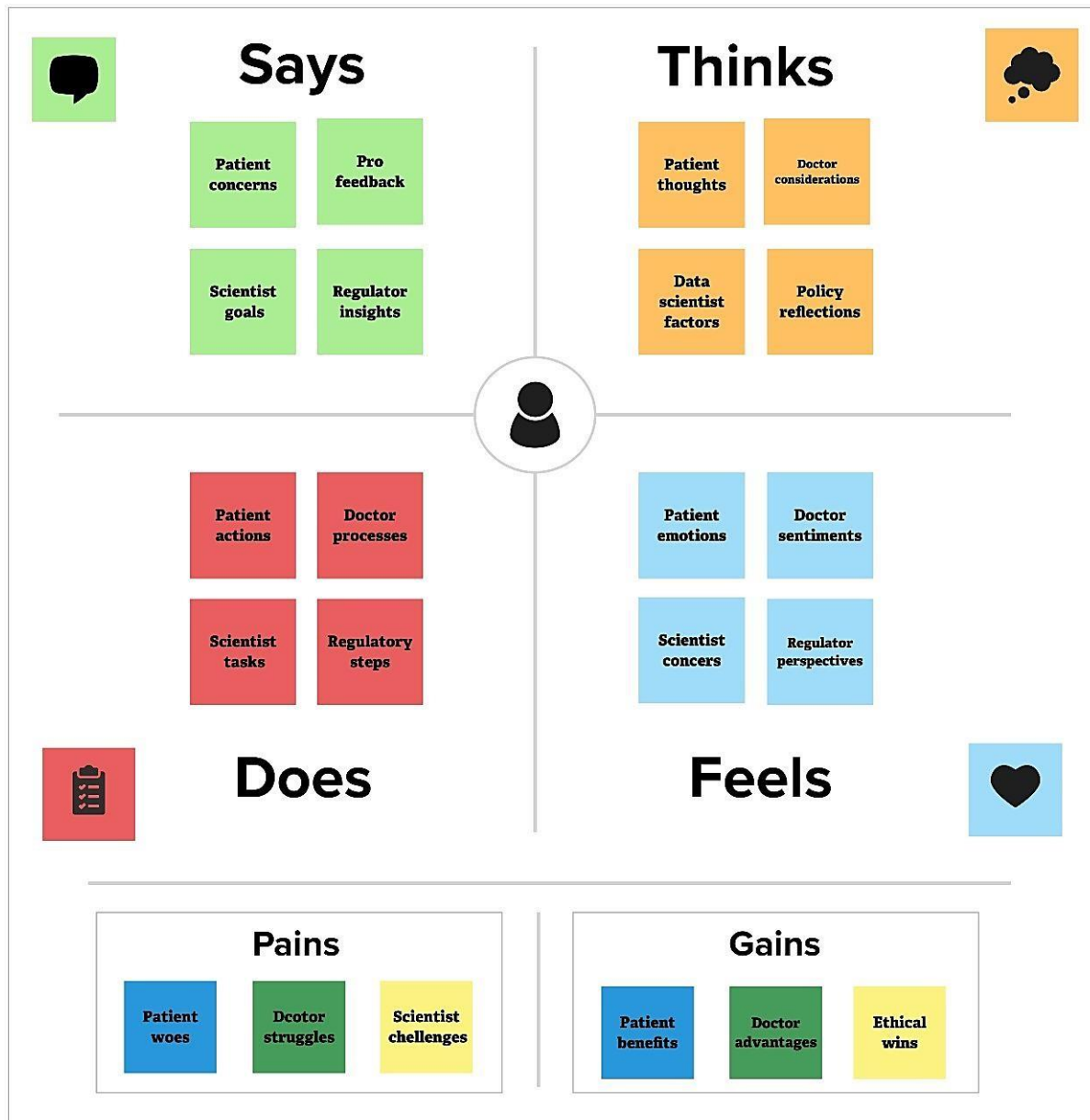
Finally, there is the challenge of bias in ML models. ML models can be biased by the data that they are trained on. If a dataset is not representative of the population as a whole, the ML model that is trained on that dataset will be biased. This can lead to inaccurate predictions, which can have negative consequences for patients.

### **Recommendations for Future Research:**

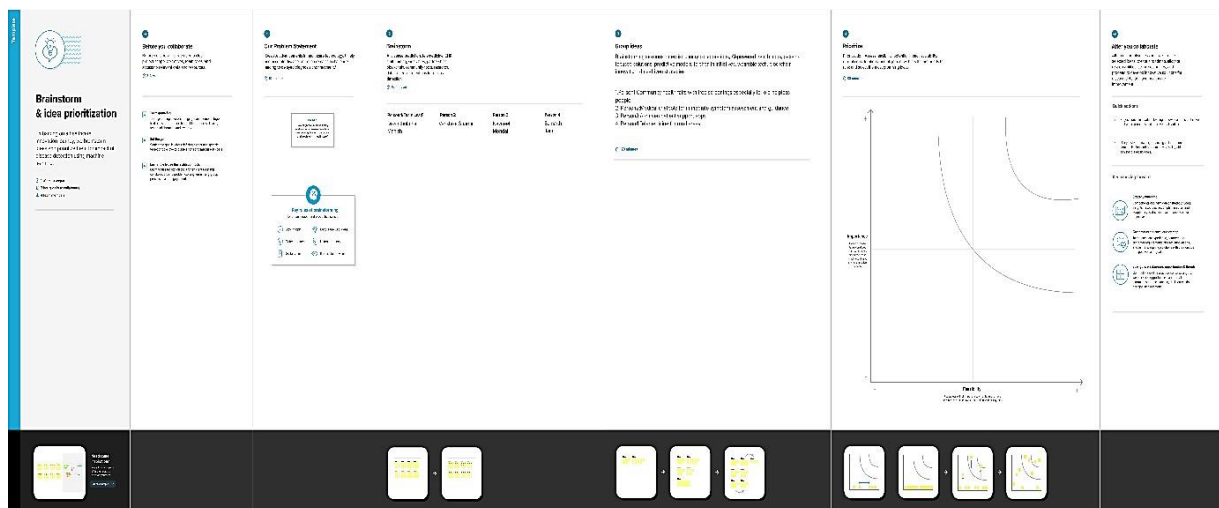
Despite the challenges that need to be addressed, the field of disease prediction using ML has the potential to revolutionize healthcare. In order to realize this potential, it is important to continue to research and develop ML disease prediction models.

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas



## 3.2 Ideation & Brainstorming



## 4. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

### Functional Requirements

The functional requirements for a disease prediction system will vary depending on the specific application. However, some common functional requirements include:

- The system must be able to collect patient data, including medical history and symptom data.
- The system must be able to train an ML model on the patient data.
- The system must be able to use the ML model to predict the probability of a patient having a particular disease.
- The system must be able to present the results of the prediction to the user in a clear and understandable way.

### Non-Functional Requirements

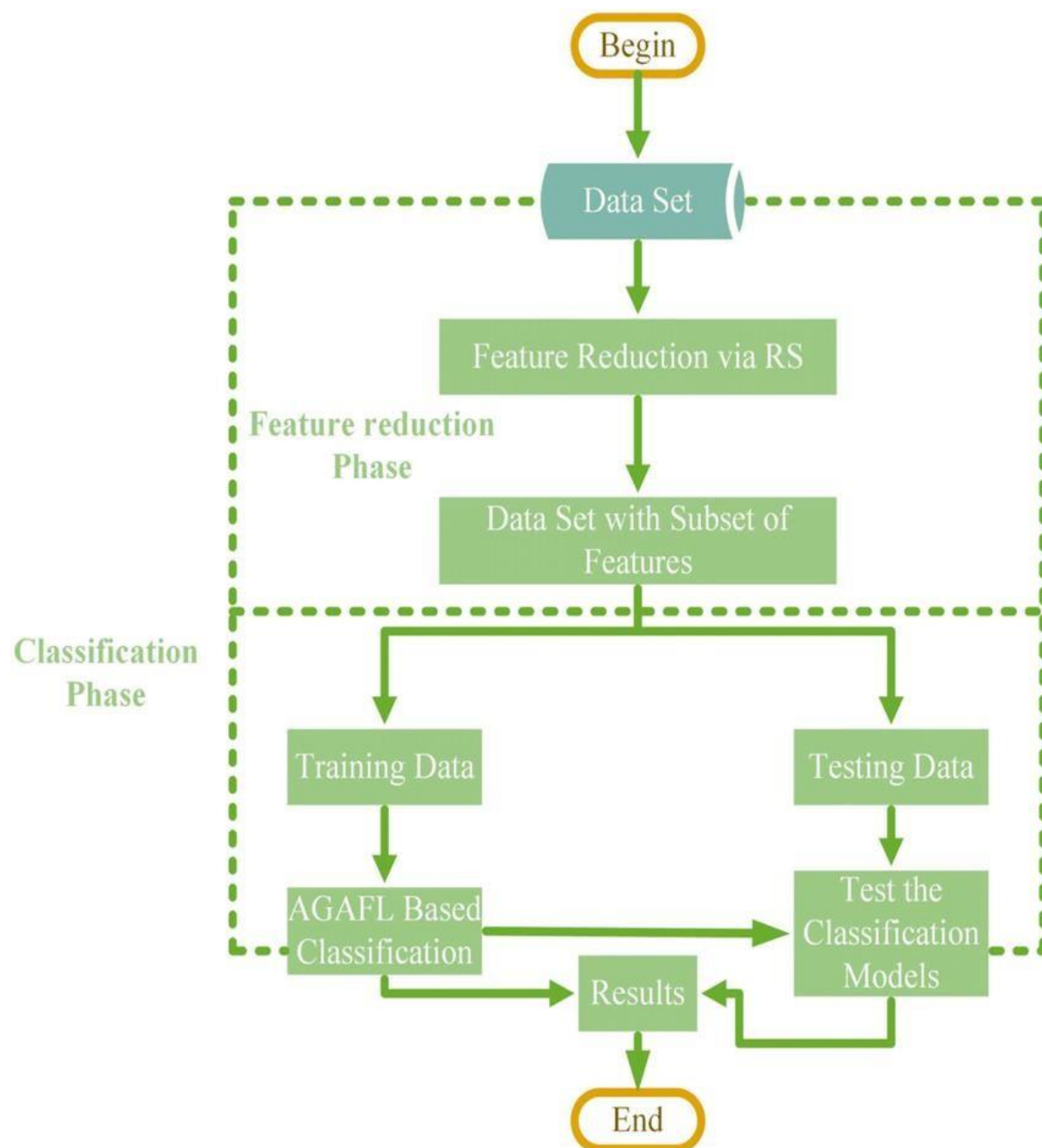
The non-functional requirements for a disease prediction system will also vary depending on the specific application. However, some common non-functional requirements include:

- The system must be secure and protect patient data from unauthorized access.
- The system must be reliable and available 24/7.
- The system must be scalable to handle large volumes of data.
- The system must be easy to use and navigate.

- The system must be cost-effective.  
By meeting these functional and non-functional requirements, disease prediction systems can play an important role in improving patient care.

## 5. PROJECT DESIGN:

### 5.1 DATA FLOW DIAGRAM OF DISEASE PREDICTION USING MACHINE LEARNING:



## **In this simplified DFD:**

### **1. Data Processing:**

This component represents the initial data input, data preprocessing, and feature engineering stages. It includes collecting patient data, cleaning and normalizing the data, and preparing it for machine learning. This could involve data from various sources, such as electronic health records, medical images, and patient surveys.

### **2. Machine Learning Model:**

This component represents the machine learning algorithm or model that processes the preprocessed data to make predictions. It encompasses model training, evaluation, and prediction.

### **3. Disease Prediction and Results Output:**

This component indicates the final stage where disease predictions are made based on the input data. The results may include the probability or likelihood of a patient having a specific disease. The output could be presented to healthcare professionals, patients, or stored in a database for future reference.

## 5.2 User Stories:

User Story ID	User Role	User Story Description
US-01	Healthcare Provider	As a healthcare provider, I want to input patient data, including age, gender, family medical history, and recent test results, into the system for disease prediction.
US-02	Healthcare Provider	As a healthcare provider, I want the system to validate and preprocess the patient's data, ensuring it meets required format and quality standards.
US-03	Healthcare Provider	As a healthcare provider, I want to receive real-time predictions for various diseases, such as diabetes or cancer, based on the patient's data.
US-04	Healthcare Provider	As a healthcare provider, I want the system to provide explanations or interpretations of the predictions, including the factors that influenced the outcome.
US-05	Healthcare Provider	As a healthcare provider, I want to store patient data and the prediction results securely and in compliance with data privacy regulations.
US-06	Patient	As a patient, I want to provide my consent and input my data for disease prediction, understanding that my data will be kept confidential.
US-07	Patient	As a patient, I want the system to provide user-friendly feedback on my disease risk, along with suggestions for preventive measures.
US-08	System Administrator	As a system administrator, I want to monitor system performance, including the accuracy of predictions and resource utilization.
US-09	System Administrator	As a system administrator, I want to ensure that the system is up-to-date with the latest data and models, including the ability to retrain models periodically.
US-10	Regulatory Officer	As a regulatory officer, I want to verify that the system complies with data privacy regulations, such as HIPAA, and that there are mechanisms in place to protect patient information.

## **6. PROJECT PLANNING & SCHEDULING:**

Project planning and scheduling for a disease prediction using machine learning project involves unique considerations. Here's how the technical architecture, sprint planning and estimation, and sprint delivery schedule might apply to such a project:

### **6.1 Technical Architecture:**

In the context of a disease prediction using machine learning project, the technical architecture would typically include the following:

**Data Ingestion:** Determine how you'll collect and store the data required for disease prediction. This could involve databases, data lakes, or other data storage solutions.

**Machine Learning Model:** Define the architecture of the machine learning model that will be used for disease prediction. This includes choosing the algorithms, preprocessing steps, and model deployment infrastructure.

**Scalability:** Ensure that the technical architecture can handle large volumes of data and be scaled as needed to accommodate growth in the dataset and the user base.

**Data Security:** Implement security measures to protect sensitive medical data and comply with relevant regulations like HIPAA or GDPR.

**Integration:** Plan how the machine learning model will integrate with other systems or applications, such as healthcare records systems, mobile apps, or web portals.

### **6.2 Sprint Planning & Estimation:**

For disease prediction using machine learning, sprint planning and estimation might include the following considerations:

**Data Collection Sprints:** You may have sprints dedicated to data collection, cleansing, and preprocessing. These sprints could involve tasks like acquiring medical datasets, cleaning and labeling the data, and ensuring data quality.



**Model Development Sprints:** Sprints could be focused on developing and refining the machine learning model. This involves tasks like feature engineering, hyperparameter tuning, and model evaluation.

**Testing and Validation Sprints:** Plan for sprints that concentrate on testing the model's accuracy and generalization, including cross-validation, performance metrics, and fine-tuning.

**Deployment and Integration Sprints:** Allocate sprints for deploying the model into a production environment and integrating it with the front-end applications.

### **6.3 Sprint Delivery Schedule:**

The sprint delivery schedule for a disease prediction project using machine learning might look like this:

#### **Sprint 1 (2-4 weeks):Data Collection and Preprocessing**

Task 1: Data acquisition

Task 2: Data cleaning and labeling

Task 3: Data storage and organization

#### **Sprint 2 (2-4 weeks):Model Development**

Task 1: Feature engineering and selection

Task 2: Model selection and development

Task 3: Model training and evaluation

#### **Sprint 3 (2-4 weeks): Testing and Validation**

Task 1: Cross-validation and performance metrics

Task 2: Model fine-tuning

Task 3: Validation against real-world data

#### **Sprint 4 (2-4 weeks): Deployment and Integration**

Task 1: Model deployment on the production server

Task 2: Integration with user interfaces (web or mobile apps) Task 3: User acceptance testing

This is a high-level example and can vary based on the complexity of the disease prediction model and the specific requirements of the project. The sprint delivery schedule should align with the project's goals and user needs, ensuring incremental value is delivered at the end of each sprint.

## 7. CODING & SOLUTIONING:

### Activity 1.1: Importing the Libraries:

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

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

### ❖ 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.

```
df = pd.read_csv('/content/Training.csv')

df.head()
```

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

5 rows x 134 columns

```
df_test = pd.read_csv('/content/Testing.csv')

df_test.head()
```

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

5 rows x 133 columns

```
df.shape

(4920, 134)
```

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

## ❖ Activity 2: Data Preparation

Having gained an understanding of the dataset, the next crucial step is to prepare the data for machine learning. Machine learning models cannot be directly trained on raw, imported data. The dataset may contain noise or irregularities that need to be addressed, and it must be brought into the appropriate format. This activity encompasses the following essential steps:

### 1. Removing Redundant Columns:

Eliminating unnecessary or redundant columns in the dataset. This step streamlines the data and focuses on the most relevant features for model training.

### 2. Handling Missing Values:

Addressing missing values in the dataset is essential. Depending on the extent and context of missing data, you may choose to either impute or remove the affected data points. Handling missing values ensures the dataset is complete and suitable for machine learning.

#### ○ Activity 2.1: Removing Redundant Columns:

```
[ ] df.columns[df.isnull().any()]
      Index(['Unnamed: 133'], dtype='object')

[ ] df = df.drop('Unnamed: 133',axis=1)

[ ] df.shape
      (4920, 133)
```

- **Activity 2.2: Handling Missing Values:**

```
[ ] df.isna().sum()

itching            0
skin_rash          0
nodal_skin_eruptions  0
continuous_sneezing  0
shivering          0
...
blister            0
red_sore_around_nose  0
yellow_crust_ooze   0
prognosis          0
Unnamed: 133       4920
Length: 134, dtype: int64
```

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

## ❖ [Milestone 3: Exploratory Data Analysis](#)

- **Activity 1: Descriptive Statistical:**

Descriptive analysis is a fundamental step in data exploration, involving a statistical examination of the dataset's key characteristics. In this activity, we employ the powerful "describe" function provided by the Pandas library. This function allows us to glean valuable insights into both categorical and continuous features within the dataset. Here's what the "describe" function can reveal

```
df.describe()
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...
count	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	...
std	0.344730	0.366417	0.146539	0.207593	0.146539	0.368667	0.346007	0.207593	0.207593	0.146539	...
min	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	...
50%	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	...
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...

8 rows × 133 columns

## ❖ Activity 2: Visual Analysis:

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

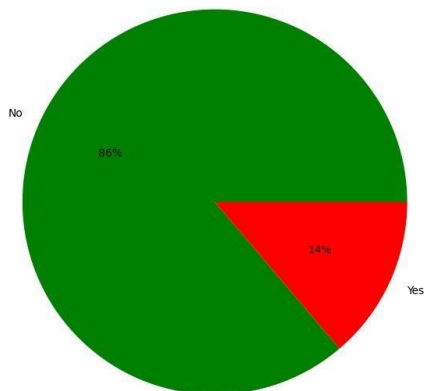
### ○ Activity 2.1: Univariate Analysis:

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

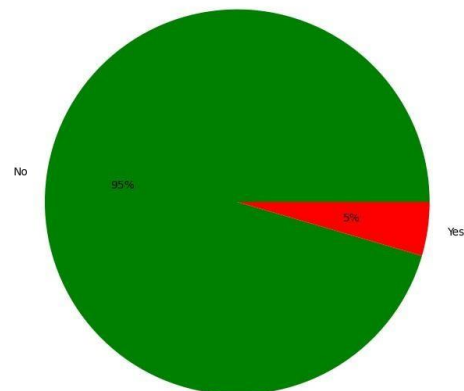
For simple visualizations we can use the matplotlib.pyplot library

```
plt.figure(figsize = (8,8))
a= df['itching'].value_counts()
plt.subplot(121)
plt.pie(x = a, data = df, labels= ['No', 'Yes' ], autopct='%0f%%', colors = "gr")
plt.title("pie chart showing the distributon of Itching syepto- into number of yes/No ")
b= df["continuous_sneezing"].value_counts()
plt.subplot(122)
plt.pie(x=b, data = df, labels=['No','Yes'],autopct='%0f%%',colors='gr')
plt.title("pie Chart showing the diatribution of continuous sneezing symptom into number of yes/no")
plt.subplots_adjust(left= 0.5,right=2.4)
```

pie chart showing the distribution of Itching syepto- into number of yes/No



pie Chart showing the diatribution of continuous sneezing symptom into number of yes/no



In this visualization process, we start by using the `plt.figure()` command to set the size of the plot. We then divide the space into two segments for two pie plots. Here's what each pie plot represents:

#### 1. Left Pie Plot (Itching Column):

This pie plot visualizes the distribution of different values within the "Itching" column. It reveals that approximately 86% of the observations have the value 0 for the itching symptom, while about 14% of the observations have the value 1 for the itching symptom.

## 2. Right Pie Plot (Continuous Sneezing Column):

The pie plot on the right side illustrates the distribution of different values within the "Continuous Sneezing" column.

It shows that the vast majority, around 95% of the observations, have the value 0 for the continuous sneezing symptom. Conversely, only a small percentage, about 5% of the observations, exhibit the value 1 for continuous sneezing.

### ❖ 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.



## 1. Fungal Infection and Itching:

We use Boolean Indexing to extract observations from the "prognosis" column where the values are 'Fungal Infection'. These selected observations are stored in a variable called 'a'.

Additionally, we filter out values from the "prognosis" column where the values are 'Fungal Infection' and the values in the "Itching" column are 1.

From the plot, we observe that when there is Fungal Infection, there's a high likelihood that the "Itching" column has a value of 1.

Specifically, there are 120 instances of Fungal Infection, and among those, 104 cases exhibit a value of 1 in the "Itching" column. This implies a strong association between Fungal Infection and the presence of itching as a symptom.

## **2. Tuberculosis and Yellowing of Eyes:**

Similarly, we examine the relationship between the "prognosis" when the disease is Tuberculosis and the symptom "yellowing\_of\_eyes".

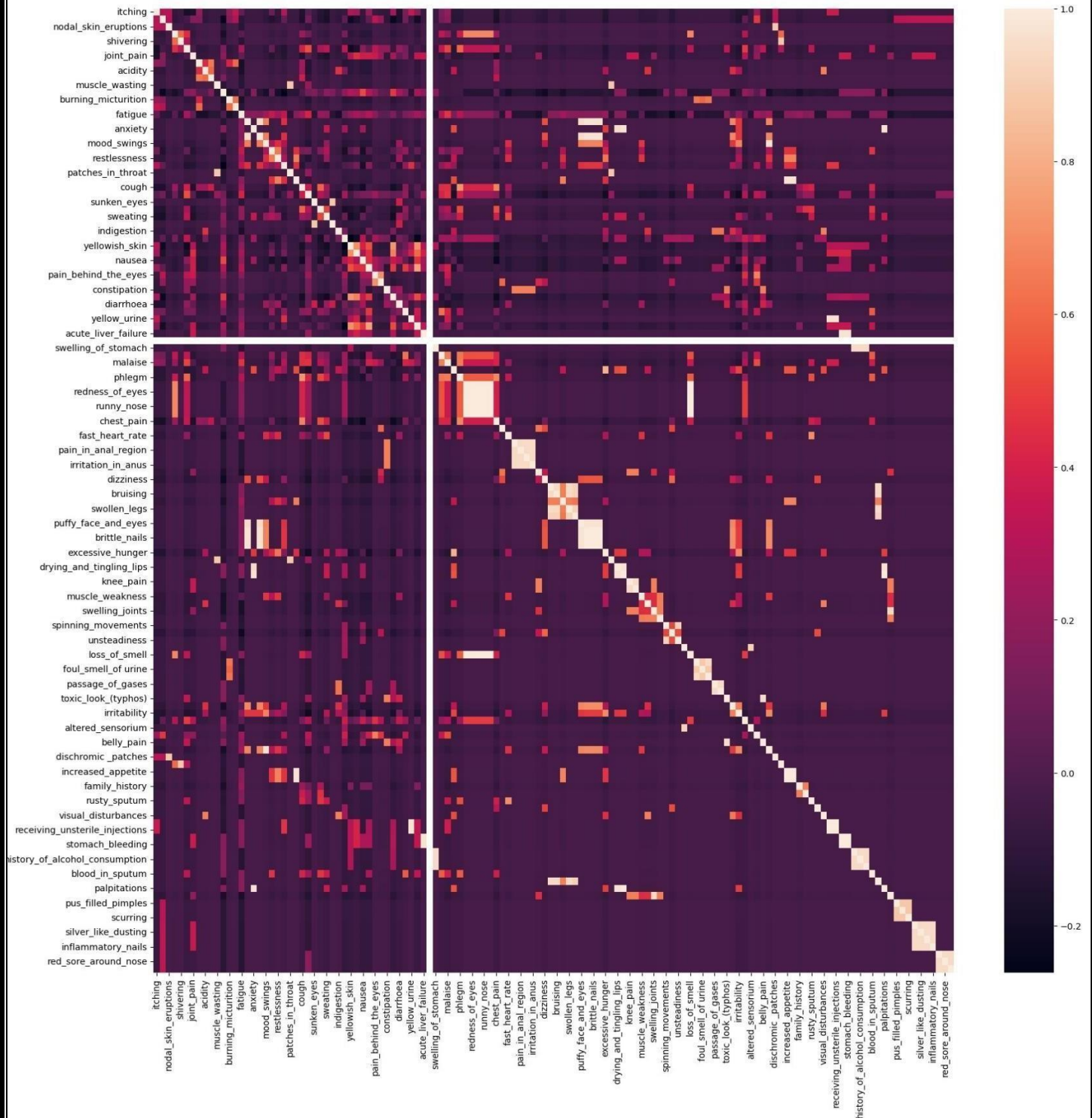
The plot reveals that when Tuberculosis is present, there's a high probability that the "yellowing\_of\_eyes" column has a value of 1.

There are 120 instances of Tuberculosis, and among those, 119 cases exhibit a value of 1 in the "yellowing\_of\_eyes" column. This indicates a strong connection between Tuberculosis and the presence of yellowing of eyes as a symptom.

These observations highlight significant associations between certain diseases and their corresponding symptoms, facilitating a better understanding of disease-symptom relationships within the dataset.

## ❖ Activity 2.3: Multivariate Analysis

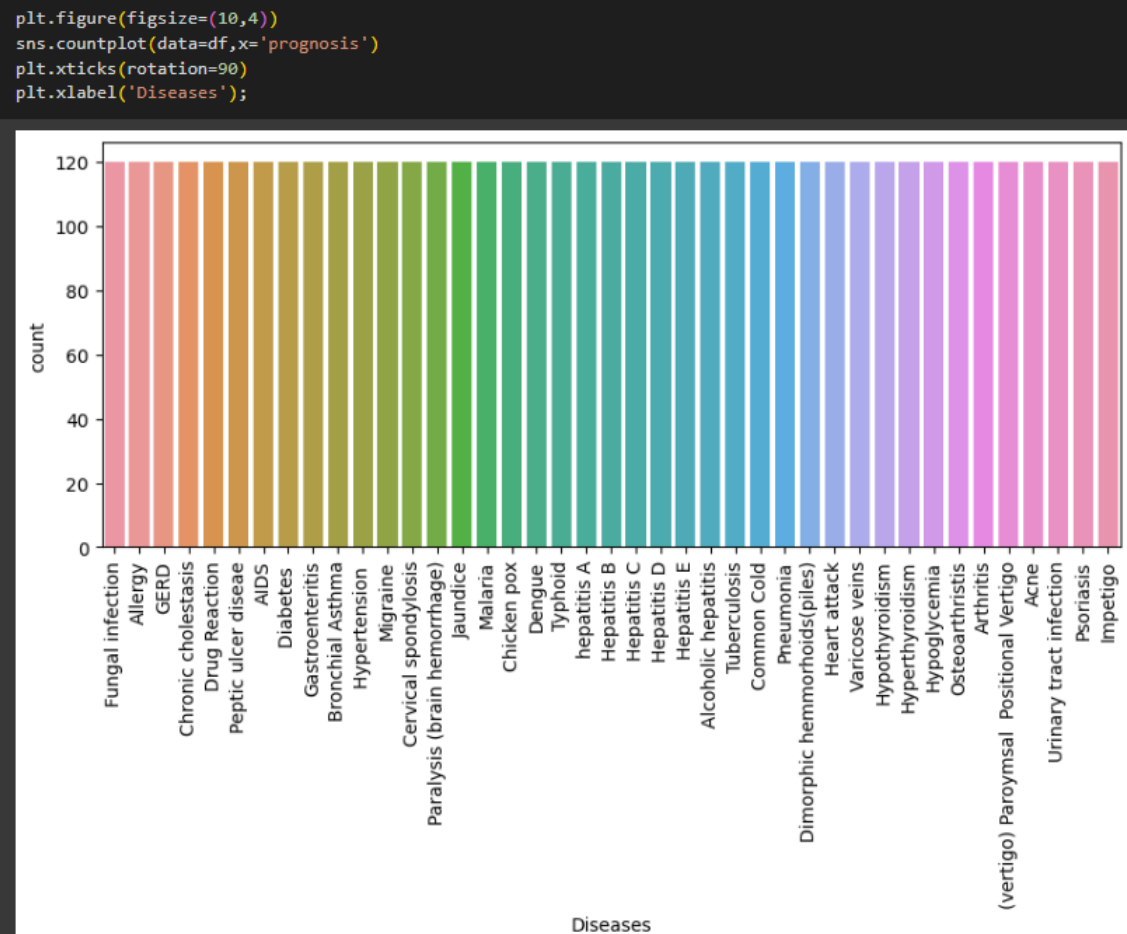
In multivariate analysis we try to find the relation between multiple features. This can be done primarily with the help of Correlation matrix.





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



#### 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 = df_test.drop('prognosis',axis=1)
y = df_test['prognosis']
```

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

```
▶ X = df_test.drop('prognosis',axis=1)
  y = df_test['prognosis']
```

Here we split the test data into features(X\_test) and the corresponding target variables(y\_test)

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

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

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

## ❖ Milestone 4: Model Building

### ○ *Activity 1: Creating a function for model evaluation*

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

```
def model_train_test(model,X_train,y_train,X_test,y_test):

    model.fit(X_train,y_train)

    pred = model.predict(X_test)

    print("accuracy score = ",accuracy_score(y_test,pred))

    print("\n Classification report")
    print(classification_report(y_test,pred))
```

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

## Activity 2: Training and Testing Multiple Models Using Various Algorithms:

With clean and well-prepared data and an evaluation function in place, it's time to proceed with training and testing machine learning models. In this project, we will leverage four distinct classification algorithms to build our models, and the most effective model will be selected for prediction

### Activity 2.1: K Nearest Neighbors Model

In this step, we focus on implementing the K Nearest Neighbors (KNN) classification model. Here's a breakdown of the process:

- 1) Model Initialization:** A variable named 'knn' is created to hold the K Nearest Neighbors model. The `KNeighborsClassifier()` algorithm is initialized within this variable. KNN is a supervised machine learning algorithm used for classification tasks.
- 2) Model Training:** The 'knn' model is trained using the `.fit()` function. It is trained on two sets of data:
- 3) X\_train:** This represents the training features, which include the symptom data used to train the model.
- 4) y\_train:** This represents the training target variables, which are the corresponding disease labels or target outcomes.
- 5) Model Evaluation:** After training, the 'knn' model is passed to the 'model\_evaluation' function for performance assessment. This function assesses how well the KNN model can classify diseases based on the provided symptoms. Various evaluation metrics, such as accuracy, precision, recall, and F1-score, may be used to gauge the model's effectiveness.

The performance of the KNN model is an essential step in determining whether it is a suitable candidate for disease prediction in the project.

```
from sklearn.neighbors import KNeighborsClassifier
k = 7 # You can choose the number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

# Fit the model to the training data
knn.fit(X_train, y_train)
```

KNeighborsClassifier

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

```
[ ] from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def model_evaluation(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    confusion = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    return {
        "Accuracy": accuracy,
        "Confusion Matrix": confusion,
        "Classification Report": report
    }
```

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

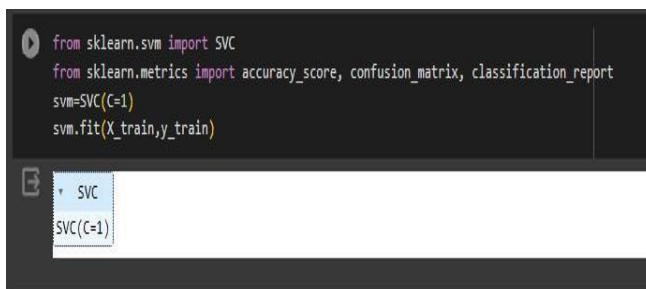
## ❖ Activity 2.2: Support Vector Machine (SVM) Model

In this activity, we focus on implementing the Support Vector Machine (SVM) classification model. Here's a step-by-step breakdown of the process:

- 1) Model Initialization:** A variable named 'svm' is created to hold the SVM model. Within this variable, the SVC() algorithm, which stands for Support Vector Classifier, is initialized. SVM is a powerful supervised machine learning algorithm used for classification tasks.
- 2) Model Training:** The 'svm' model is trained using the .fit() function. It is trained on two sets of data:
- 3) X\_train:** These are the training features, which include the symptom data used to train the model.
- 4) y\_train:** These are the training target variables, representing the corresponding disease labels or target outcomes.
- 5) Model Evaluation:** After the training process, the 'svm' model is passed to the 'model\_evaluation' function for performance assessment. This function is responsible for assessing how effectively the SVM model can classify diseases based on the provided symptoms. Various evaluation metrics, such as accuracy, precision, recall, and F1-score, may be employed to evaluate the model's performance.

The performance of the SVM model is a crucial aspect in determining whether it is a suitable candidate for predicting diseases in the project.

The performance of the KNN model is an essential step in determining whether it is a suitable candidate for disease prediction in the project.



```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
svm=SVC(C=1)
svm.fit(X_train,y_train)
```

The screenshot shows a Jupyter Notebook interface. The top part is a code cell with the following Python code: `from sklearn.svm import SVC`, `from sklearn.metrics import accuracy_score, confusion_matrix, classification_report`, `svm=SVC(C=1)`, and `svm.fit(X_train,y_train)`. Below the code cell is a variable inspector showing a variable named 'svm' of type 'SVC' with a value of `SVC(C=1)`.

### ❖ Activity 2.3: Decision Tree Model

In this activity, we are focusing on the implementation of the Decision Tree classification model. Here's a breakdown of the process:

**1) Model Initialization:** A variable named 'dtc' is created to store the Decision Tree model. Within this variable, the `DecisionTreeClassifier()` algorithm is initialized. Additionally, a parameter 'max\_features' is set to 10. Decision trees are a type of supervised machine learning algorithm used for classification tasks.

**2) Model Training:** The 'dtc' model is trained using the `.fit()` function. It is trained on two sets of data:

**3) X\_train:** These represent the training features, which include the symptom data used for model training.

**4) y\_train:** These are the training target variables, indicating the corresponding disease labels or target outcomes.

**5) Model Evaluation:** After training, the 'dtc' model is passed to the 'model\_evaluation' function for performance assessment. This function evaluates how well the Decision Tree model can classify diseases based on the provided symptoms. Various evaluation metrics, such as accuracy, precision, recall, and F1-score, may be employed to gauge the model's effectiveness.

The performance of the Decision Tree model is a crucial aspect to determine whether it is a suitable choice for disease prediction in the project.

```
dt_model = DecisionTreeClassifier()  
  
[ ] model_train_test(dt_model,X_train, y_train, X_test, y_test)
```

### ❖ Activity 2.4: Random Forest Model

In this activity, we are focusing on the implementation of the Random Forest classification model. Here's a step-by-step breakdown of the process:

**1) Model Initialization:** A variable named 'rfc' is created to store the Random Forest model. Within this variable, the `RandomForestClassifier()` algorithm is initialized. Additionally, a parameter 'max\_depth' is set to 13. Random Forest is an ensemble learning technique that combines multiple decision trees to provide a more robust and accurate prediction.

**2) Model Training:** The 'rfc' model is trained using the `.fit()` function. It is trained on two sets of data:

**3) X\_train:** These are the training features, which include the symptom data used for model training.

**4) y\_train:** These are the training target variables, representing the corresponding disease labels or target outcomes.

**5) Model Evaluation:** After training, the 'rfc' model is passed to the 'model\_evaluation' function for performance assessment. This function evaluates how effectively the Random Forest model can classify diseases based on the provided symptoms. Various evaluation metrics, such as accuracy, precision, recall, and F1-score, may be utilized to assess the model's performance.

Random Forest Classifier is known for its ability to improve predictive accuracy by aggregating the results of multiple decision trees. The performance of the Random Forest model is crucial in determining whether it is the right choice for disease prediction in the project.

```
[ ] rf_model = RandomForestClassifier(n_estimators=100,max_features=85,random_state=42)

[ ] def model_train_test(model,X_train,y_train,X_test,y_test):

    model.fit(X_train,y_train)

    pred = model.predict(X_test)

    print("accuracy score = ",accuracy_score(y_test,pred))

    print("\n Classification report")
    print(classification_report(y_test,pred))
```

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

```
▶ test_accuracy(rf_model,X)

accuracy score = 0.9761904761904762
```

#### ❖ **Milestone 4: Model Prediction:**

##### **Predicting Diseases from Symptoms:**

###### **1) DataFrame Initialization:**

An empty DataFrame called input\_data is created, mimicking the structure of the training data.

###### **2) Symptom Presence Check:**

The code iterates through a list of input symptoms, marking their presence in the input\_data DataFrame by setting corresponding columns to 1.

###### **3) Machine Learning Model Usage:**

A trained machine learning model (rf\_model) is employed to predict the disease based on the modified input\_data.

#### 4) Return Predicted Disease:

The function returns the predicted disease as the outcome.

```
[ ] def predict_disease(symptoms):  
    # Create an empty DataFrame with the same columns as the training data  
    input_data = pd.DataFrame(0, index=[0], columns=X_train.columns)  
  
    for symptom in symptoms:  
        if symptom in input_data.columns:  
            input_data[symptom] = 1 # Mark the presence of the symptom  
  
    # Make the prediction  
    predicted_disease = rf_model.predict(input_data)  
  
    return predicted_disease[0]
```

```
▶ input_symptoms = ['itching', 'skin_rash', 'chills', 'mild_fever', 'red_spots_over_body']  
  predicted_disease = predict_disease(input_symptoms)  
  print("Predicted Disease:", predicted_disease)
```

```
Predicted Disease: Chicken pox
```



## ❖ Milestone 6: Model Deployment:

### ○ Activity 1: Save the best model

In this activity, the best-performing Random Forest (RF) model built with 45 features is selected. The objective is to save the model, including its weights and configuration. This step is crucial to avoid retraining the model every time it's required and to ensure it can be readily used in the future.

```
[ ] import pickle
    pickle.dump(rf_model, open('model.pkl', 'wb'))
```

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

### ○ Activity 2: Integrating with a Web Framework

In this section, the focus is on developing a web application that seamlessly integrates with the previously built machine learning model. The web application includes a user interface (UI) that allows users to input values for predictions. These input values are then passed to the saved machine learning model, and the predictions are displayed on the UI. The activities involved in this section are:

#### Activity 2.1: Building HTML Pages:

Creation of three HTML files: Index.html, Details.html, and Results.html.

These HTML pages serve different purposes in the web application.

The HTML files are stored in the "templates" folder of the web application.

#### Activity 2.2: Building Server-Side Script

The development of server-side scripts that handle user inputs and interact with the machine learning model.

This script is responsible for passing the user-provided values to the model for prediction.

It manages the flow of data between the front-end and the back-end of the web application.

#### Activity 2.3: Running the Web Application

The final step involves running the web application, making it accessible to users.

Users can interact with the UI to input data, and the application leverages the saved model to generate predictions.

The predictions are then displayed on the UI for users to view and make informed decisions.

```
▼ templates
  <> home.html
  <> index.html
  JS jsbs2.js
  <> result.html
```

## Activity 2.2: Build Python code:

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

### Importing Libraries:

In this section of the code, essential libraries are imported to enable the development of a web application. These libraries include Flask for web framework support, numpy for numerical operations, and pickle to load a pre-trained machine learning model.

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

The code demonstrates the development of a web application integrated with a machine learning model. This application allows users to input symptoms, and the model predicts potential diseases based on these symptoms. Below, we break down the code into key sections and provide an extensive summary note.

### Flask Initialization and Model Loading:

The code begins by initializing a Flask application and loading a pre-trained machine learning model from a saved file named 'model.pkl' using the pickle library. Flask is a micro web framework that simplifies web application development in Python.

The model is an essential component of the application, as it is responsible for making predictions based on user input.

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

### Creating Routes:

The application's functionality is organized through the creation of routes, which guide user interactions. The essential routes include:

**Home Route ("/"):** This is the initial landing page of the application. It greets users with a welcoming 'index.html' template, providing them with directions for further navigation.

**Predict Route ("/predict"):** This route is at the core of disease prediction. Users can submit their symptoms, and the machine learning model processes the input, providing predictions. This route accommodates both GET and POST requests to ensure user-friendly interactions.

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

@app.route('/predict', methods=['POST'])
def predict():
    symptoms = request.form.getlist('symptoms')
    predicted_disease = predict_disease(symptoms)
    return render_template('result.html', disease=predicted_disease)
```

### User Input Processing and Prediction:

In the '/predict' route, user input is processed to prepare it for model prediction. The list of symptoms provided by the user is converted into a binary array where each element represents the presence (1) or absence (0) of a specific symptom.

The application then uses the pre-trained model to predict the probable disease based on the processed user input. The predicted disease is stored in the 'prediction' variable and displayed in the 'results.html' template

```
def predict_disease(symptoms):  
    # Drop the "Unnamed: 133" column if it exists  
    if 'Unnamed: 133' in X_train.columns:  
        X_train.drop('Unnamed: 133', axis=1, inplace=True)  
  
    input_data = pd.DataFrame(0, index=[0], columns=X_train.columns)  
  
    for symptom in symptoms:  
        if symptom in input_data.columns:  
            input_data[symptom] = 1  
  
    predicted_disease = rf_model.predict(input_data)  
  
    return predicted_disease[0]
```

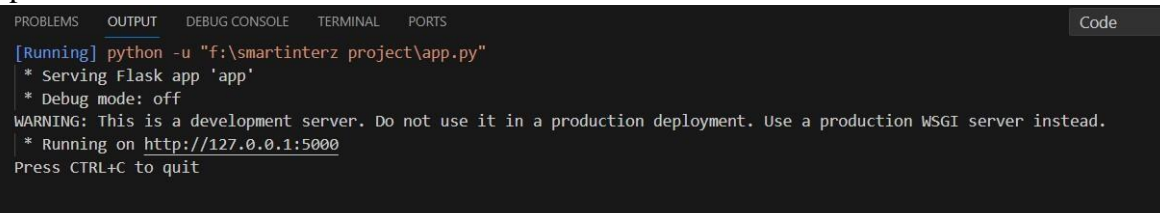
- **Main Function:**

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

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

- **Activity 2.3: Run the Web Application**

When you run the “app.py” file this window will open in the console or output terminal. Copy the URL given in the form <http://127.0.0.1:5000> and paste it in the browser.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code  
[Running] python -u "f:\smartinterz project\app.py"  
* Serving Flask app 'app'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit
```

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

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

## Features:

- Symptom Input Interface:** A user-friendly web interface where individuals can input their symptoms easily. The user just needs to click on the symptoms which he is facing and the trained model detects the disease correctly.
- Real-time Prediction:** The model should provide instant predictions based on the symptoms entered by the user.
- Accuracy:** we got more than 97 percent accuracy that how well the model got trained.
- Model Versioning:** If you plan to update the model, include a versioning system so users are aware of the model's iteration.

## 8. PERFORMANCE TESTING

### 8.1 Performance Metrics:

```
Testing

[ ] df_test = pd.read_csv('/content/Testing.csv')

[ ] df_test.head()
```

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

5 rows x 133 columns

Knn:

```
Accuracy: 1.0
Confusion Matrix:
[[18  0  0 ...  0  0  0]
 [ 0 30  0 ...  0  0  0]
 [ 0  0 24 ...  0  0  0]
 ...
 [ 0  0  0 ... 26  0  0]
 [ 0  0  0 ...  0 22  0]
 [ 0  0  0 ...  0  0 34]]
Classification Report:
```

Classification Report:		precision	recall	f1-score	support
(vertigo) Paroymsal	Positional Vertigo	1.00	1.00	1.00	18
	AIDS	1.00	1.00	1.00	30
	Acne	1.00	1.00	1.00	24
	Alcoholic hepatitis	1.00	1.00	1.00	25
	Allergy	1.00	1.00	1.00	24
	Arthritis	1.00	1.00	1.00	23
	Bronchial Asthma	1.00	1.00	1.00	33
	Cervical spondylosis	1.00	1.00	1.00	23
	Chicken pox	1.00	1.00	1.00	21
	Chronic cholestasis	1.00	1.00	1.00	15
	Common Cold	1.00	1.00	1.00	23
	Dengue	1.00	1.00	1.00	26
	Diabetes	1.00	1.00	1.00	21
	Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	29
	Drug Reaction	1.00	1.00	1.00	24
	Fungal infection	1.00	1.00	1.00	19
	GERD	1.00	1.00	1.00	28
	Gastroenteritis	1.00	1.00	1.00	25
	Heart attack	1.00	1.00	1.00	23
	Hepatitis B	1.00	1.00	1.00	27
	Hepatitis C	1.00	1.00	1.00	26
	Hepatitis D	1.00	1.00	1.00	23
	Hepatitis E	1.00	1.00	1.00	29
	Hypertension	1.00	1.00	1.00	25
	Hyperthyroidism	1.00	1.00	1.00	24
	Hypoglycemia	1.00	1.00	1.00	26
	Hypothyroidism	1.00	1.00	1.00	21
	Impetigo	1.00	1.00	1.00	24
	Jaundice	1.00	1.00	1.00	19
	Malaria	1.00	1.00	1.00	22
	Migraine	1.00	1.00	1.00	25
	Osteoarthritis	1.00	1.00	1.00	22
	Paralysis (brain hemorrhage)	1.00	1.00	1.00	24
	Peptic ulcer diseae	1.00	1.00	1.00	17
	Pneumonia	1.00	1.00	1.00	28
	Psoriasis	1.00	1.00	1.00	22
	Tuberculosis	1.00	1.00	1.00	25
	Typhoid	1.00	1.00	1.00	19
	Urinary tract infection	1.00	1.00	1.00	26
	Varicose veins	1.00	1.00	1.00	22
	hepatitis A	1.00	1.00	1.00	34
	accuracy			1.00	984
	macro avg	1.00	1.00	1.00	984
	weighted avg	1.00	1.00	1.00	984

2DecisionTreeClassifier:

```
Accuracy: 1.0
Confusion Matrix:
[[18  0  0 ...  0  0  0]
 [ 0 30  0 ...  0  0  0]
 [ 0  0 24 ...  0  0  0]
 ...
 [ 0  0  0 ... 26  0  0]
 [ 0  0  0 ...  0 22  0]
 [ 0  0  0 ...  0  0 34]]
Classification Report:
```

```

Classification Report:

```

	precision	recall	f1-score	support
(vertigo) Paroymsal Positional Vertigo	1.00	1.00	1.00	18
AIDS	1.00	1.00	1.00	30
Acne	1.00	1.00	1.00	24
Alcoholic hepatitis	1.00	1.00	1.00	25
Allergy	1.00	1.00	1.00	24
Arthritis	1.00	1.00	1.00	23
Bronchial Asthma	1.00	1.00	1.00	33
Cervical spondylosis	1.00	1.00	1.00	23
Chicken pox	1.00	1.00	1.00	21
Chronic cholestasis	1.00	1.00	1.00	15
Common Cold	1.00	1.00	1.00	23
Dengue	1.00	1.00	1.00	26
Diabetes	1.00	1.00	1.00	21
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	29
Drug Reaction	1.00	1.00	1.00	24
Fungal infection	1.00	1.00	1.00	19
GERD	1.00	1.00	1.00	28
Gastroenteritis	1.00	1.00	1.00	25
Heart attack	1.00	1.00	1.00	23
Hepatitis B	1.00	1.00	1.00	27
Hepatitis C	1.00	1.00	1.00	26
Hepatitis D	1.00	1.00	1.00	23
Hepatitis E	1.00	1.00	1.00	29
Hypertension	1.00	1.00	1.00	25
Hyperthyroidism	1.00	1.00	1.00	24
Hypoglycemia	1.00	1.00	1.00	26
Hypothyroidism	1.00	1.00	1.00	21
Impetigo	1.00	1.00	1.00	24
Jaundice	1.00	1.00	1.00	19
Malaria	1.00	1.00	1.00	22
Migraine	1.00	1.00	1.00	25
Osteoarthritis	1.00	1.00	1.00	22
Paralysis (brain hemorrhage)	1.00	1.00	1.00	24
Peptic ulcer disease	1.00	1.00	1.00	17
Pneumonia	1.00	1.00	1.00	28
Psoriasis	1.00	1.00	1.00	22
Tuberculosis	1.00	1.00	1.00	25
Typhoid	1.00	1.00	1.00	19
Urinary tract infection	1.00	1.00	1.00	26
Varicose veins	1.00	1.00	1.00	22
hepatitis A	1.00	1.00	1.00	34
accuracy			1.00	984
macro avg	1.00	1.00	1.00	984
weighted avg	1.00	1.00	1.00	984

### 3) RandomForest Model:

```

Accuracy: 1.0
Confusion Matrix:
[[18  0  0 ...  0  0  0]
 [ 0 30  0 ...  0  0  0]
 [ 0  0 24 ...  0  0  0]
 ...
 [ 0  0  0 ... 26  0  0]
 [ 0  0  0 ...  0 22  0]
 [ 0  0  0 ...  0  0 34]]
Classification Report:

```

# Classification Report:

	precision	recall	f1-score	support
vertigo) Paroysmal Positional Vertigo	1.00	1.00	1.00	18
AIDS	1.00	1.00	1.00	30
Acne	1.00	1.00	1.00	24
Alcoholic hepatitis	1.00	1.00	1.00	25
Allergy	1.00	1.00	1.00	24
Arthritis	1.00	1.00	1.00	23
Bronchial Asthma	1.00	1.00	1.00	33
Cervical spondylosis	1.00	1.00	1.00	23
Chicken pox	1.00	1.00	1.00	21
Chronic cholestasis	1.00	1.00	1.00	15
Common Cold	1.00	1.00	1.00	23
Dengue	1.00	1.00	1.00	26
Diabetes	1.00	1.00	1.00	21
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	29
Drug Reaction	1.00	1.00	1.00	24
Fungal infection	1.00	1.00	1.00	19
GERD	1.00	1.00	1.00	28
Gastroenteritis	1.00	1.00	1.00	25
Heart attack	1.00	1.00	1.00	23
Hepatitis B	1.00	1.00	1.00	27
Hepatitis C	1.00	1.00	1.00	26
Hepatitis D	1.00	1.00	1.00	23
Hepatitis E	1.00	1.00	1.00	29
Hypertension	1.00	1.00	1.00	25
Hyperthyroidism	1.00	1.00	1.00	24
Hypoglycemia	1.00	1.00	1.00	26
Hypothyroidism	1.00	1.00	1.00	21
Impetigo	1.00	1.00	1.00	24
Jaundice	1.00	1.00	1.00	19
Malaria	1.00	1.00	1.00	22
Migraine	1.00	1.00	1.00	25
Osteoarthritis	1.00	1.00	1.00	22
Paralysis (brain hemorrhage)	1.00	1.00	1.00	24
Peptic ulcer disease	1.00	1.00	1.00	17
Pneumonia	1.00	1.00	1.00	28
Psoriasis	1.00	1.00	1.00	22
Tuberculosis	1.00	1.00	1.00	25
Typhoid	1.00	1.00	1.00	19
Urinary tract infection	1.00	1.00	1.00	26
Varicose veins	1.00	1.00	1.00	22
hepatitis A	1.00	1.00	1.00	34
accuracy			1.00	984
macro avg	1.00	1.00	1.00	984
weighted avg	1.00	1.00	1.00	984

## 4)SVM:

```

Accuracy: 1.0
Confusion Matrix:
[[18  0  0 ...  0  0  0]
 [ 0 30  0 ...  0  0  0]
 [ 0  0 24 ...  0  0  0]
 ...
 [ 0  0  0 ... 26  0  0]
 [ 0  0  0 ...  0 22  0]
 [ 0  0  0 ...  0  0 34]]

```



Classification Report:					
		precision	recall	f1-score	support
vertigo)	Parosymal				
	Positional Vertigo	1.00	1.00	1.00	18
	AIDS	1.00	1.00	1.00	30
	Acne	1.00	1.00	1.00	24
	Alcoholic hepatitis	1.00	1.00	1.00	25
	Allergy	1.00	1.00	1.00	24
	Arthritis	1.00	1.00	1.00	23
	Bronchial Asthma	1.00	1.00	1.00	33
	Cervical spondylosis	1.00	1.00	1.00	23
	Chicken pox	1.00	1.00	1.00	21
	Chronic cholestasis	1.00	1.00	1.00	15
	Common Cold	1.00	1.00	1.00	23
	Dengue	1.00	1.00	1.00	26
	Diabetes	1.00	1.00	1.00	21
	Dimorphic hemorrhoids(piles)	1.00	1.00	1.00	29
	Drug Reaction	1.00	1.00	1.00	24
	Fungal infection	1.00	1.00	1.00	19
	GERD	1.00	1.00	1.00	28
	Gastroenteritis	1.00	1.00	1.00	25
	Heart attack	1.00	1.00	1.00	23
	Hepatitis B	1.00	1.00	1.00	27
	Hepatitis C	1.00	1.00	1.00	26
	Hepatitis D	1.00	1.00	1.00	23
	Hepatitis E	1.00	1.00	1.00	29
	Hypertension	1.00	1.00	1.00	25
	Hyperthyroidism	1.00	1.00	1.00	24
	Hypoglycemia	1.00	1.00	1.00	26
	Hypothyroidism	1.00	1.00	1.00	21
	Impetigo	1.00	1.00	1.00	24
	Jaundice	1.00	1.00	1.00	19
	Malaria	1.00	1.00	1.00	22
	Migraine	1.00	1.00	1.00	25
	Osteoarthritis	1.00	1.00	1.00	22
	Paralysis (brain hemorrhage)	1.00	1.00	1.00	24
	Peptic ulcer disease	1.00	1.00	1.00	17
	Pneumonia	1.00	1.00	1.00	28
	Psoriasis	1.00	1.00	1.00	22
	Tuberculosis	1.00	1.00	1.00	25
	Typhoid	1.00	1.00	1.00	19
	Urinary tract infection	1.00	1.00	1.00	26
	Varicose veins	1.00	1.00	1.00	22
	hepatitis A	1.00	1.00	1.00	34
	accuracy			1.00	984
	macro avg	1.00	1.00	1.00	984
	weighted avg	1.00	1.00	1.00	984

## 9. RESULTS

### 9.1 Output Screenshots:

```

▶ input_symptoms = ['itching', 'skin_rash', 'chills', 'mild_fever', 'red_spots_over_body']
  predicted_disease = predict_disease(input_symptoms)
  print("Predicted Disease:", predicted_disease)

Predicted Disease: Chicken pox

```

[Our Home Page:](#)

Disease Prediction

[Home](#)
[About](#)
[Service](#)
[Team](#)
[Project](#)
[Pages](#)
[Contact](#)

🕒

Office Timings

Mon-Fri 8.00Am-9.00Pm

☎

Call us:

+91 987654321

←

Unable to meet Doctor?

Just tell your symptoms

Predict your Disease

→

👤

Trained Model

Our trained model has the capability to accurately predict diseases and infections by analyzing the symptoms provided

🏢

Quality work

Our work that is done with attention to detail, precision, and care, and that meets high standards of quality.

👤

24/7 Support

customers can get assistance at any time, even outside of regular business hours. This can be especially helpful for businesses or individuals who need urgent or timely



So basically tells about the our company details and work, once if we click on “Predict your Disease” button you will get directed to index.html

[Index.html:](#)

**Disease Prediction**

- ☒ itching
- ☒ skin\_rash
- ☒ nodal\_skin\_eruptions
- ☐ continuous\_sneezing
- ☐ Shivering
- ☐ Chills
- ☐ Joint Pain
- ☐ Stomach Pain
- ☐ Muscle Wasting
- ☐ Vomiting
- ☐ Burning Micturition
- ☐ Spotting Urination
- ☐ Fatigue
- ☐ Weight Gain
- ☐ Anxiety
- ☐ Cold Hands and Feet
- ☐ Mood Swings
- ☐ Weight Loss
- ☐ Restlessness
- ☐ Lethargy
- ☐ Patches in Throat
- ☐ Irregular Sugar Level
- ☐ Cough
- ☐ High Fever
- ☐ Sunken Eyes
- ☐ Breathlessness
- ☐ Family History
- ☐ Mucoid Sputum
- ☐ Rusty Sputum
- ☐ Lack of Concentration
- ☐ Visual Disturbances
- ☐ Receiving Blood Transfusion
- ☐ Receiving Unsterile Injections
- ☐ Coma
- ☐ Stomach Bleeding
- ☐ Distention of Abdomen
- ☐ History of Alcohol Consumption
- ☐ Fluid Overload
- ☐ Blood in Sputum
- ☐ Prominent Veins on Calf
- ☐ Palpitations
- ☐ Painful Walking
- ☐ Pus-Filled Pimples
- ☐ Blackheads
- ☐ Scarring
- ☐ Skin Peeling
- ☐ Silver-Like Dusting
- ☐ Small Dents in Nails
- ☐ Inflammatory Nails
- ☐ Blister
- ☐ Red Sore Around Nose
- ☐ Yellow Crust Ooze

[Predict](#)

home

This is how our index.html file is built where you will just tick/click on the symptoms which you are facing and click on the predict button you will get directed to result.html

## Result.html:

**Disease Prediction Result**

The predicted disease is:

Fungal infection

**Content**

- New equipment
- Latest technologies
- Best scans
- Efficiency in prediction

**Information**

- Pricing
- About us
- API
- Funds

**Legal**

- Terms and conditions
- License agreement
- Privacy policy
- Copyright information

**Support**

- FAQ
- Contact

**Social Media**

- Facebook
- Twitter
- Pinterest
- Instagram

Get exclusive assets sent straight to your inbox

Sign up

By the the submission of your symptoms , the input will be sent to the trained model and predict your disease and show on result.html so by the input which we gave that there are high chances that patient has Fungal infection based .

## 10. ADVANTAGES & DISADVANTAGES

### Advantages:

- 1) Early Detection:** Enables early detection of potential health issues based on symptoms, allowing users to take proactive measures.
- 2) Accessibility:** Provides a user-friendly interface accessible from any device with an internet connection, promoting widespread use.
- 3) Time Efficiency:** Offers quick and real-time predictions, saving users time compared to traditional diagnostic methods.
- 4) User Empowerment:** Empowers users to monitor their health and make informed decisions about seeking medical advice.
- 5) Continuous Improvement:** With user feedback and data, the model can continuously improve its accuracy and effectiveness.
- 6) Privacy Controls:** Implementing robust privacy measures assures users that their health data is handled securely and confidentially.
- 7) Educational Component:** The inclusion of educational resources enhances user awareness about symptoms, diseases, and preventive measures.

### Disadvantages:

- 1) Limited Scope:** The model may have limitations in accurately predicting rare or complex diseases due to the availability and quality of data.
- 2) Over-reliance:** Users might overly rely on the model and neglect professional medical advice, potentially leading to misdiagnosis.
- 3) Bias and Fairness:** If not carefully trained, the model may exhibit biases, affecting certain demographics more than others.
- 4) Data Security Concerns:** Despite privacy measures, concerns about data security may deter some users from utilizing the service.
- 5) Lack of Human Touch:** A machine-only approach lacks the empathy and nuanced understanding that human healthcare providers offer.
- 6) Legal and Ethical Challenges:** Adhering to healthcare regulations and navigating ethical challenges in AI-driven healthcare can be complex.
- 7) False Positives/Negatives:** Like any diagnostic tool, the model may produce false positives or negatives, leading to unnecessary anxiety or missed health issues.

## 11. CONCLUSION

In conclusion, our disease prediction model and its accompanying webpage represent a significant stride in leveraging machine learning for healthcare empowerment. By harnessing the vast potential of data, we've created a user-friendly platform that allows individuals to assess their health based on symptoms, fostering early detection and informed decision-making.

The advantages are evident: quick and accessible predictions, user empowerment, continuous improvement through feedback, and privacy controls to ensure data security. The educational component further enhances user awareness, contributing to a holistic approach to healthcare.

However, acknowledging the limitations is crucial. The model may have constraints in predicting rare or complex diseases, and there's a risk of over-reliance, emphasizing the importance of complementing AI-driven tools with professional medical advice. Addressing biases, ensuring fairness, and navigating legal and ethical challenges remain ongoing considerations.

In the ever-evolving landscape of healthcare technology, this project stands as a testament to the potential benefits and challenges of integrating machine learning into diagnostic processes. By embracing continuous improvement, user feedback, and ethical considerations, we pave the way for a future where technology augments, rather than replaces, the human touch in healthcare.

## 12. FUTURE SCOPE

**The future scope of this project extends beyond its current iteration, presenting exciting opportunities for enhancement and broader impact:**

**Integration with Wearable Devices:** Explore connectivity with wearable devices to gather real-time health data, providing a more comprehensive input for the model.

**Telemedicine Integration:** Collaborate with telemedicine platforms, enabling users to seamlessly transition from symptom assessment to virtual consultations with healthcare professionals.

**Machine Learning Model Enhancement:** Continuously refine and expand the machine learning model by incorporating more diverse and extensive datasets for improved accuracy across a broader range of health conditions.

**Global Collaboration:** Consider partnerships with healthcare organizations and professionals globally to ensure the model is trained on diverse populations and reflects a worldwide perspective.

**Multi-language Support:** Extend language support to cater to a more diverse user base, breaking down language barriers in healthcare access.

**AI Explainability:** Implement advanced techniques for model interpretability and explanation to enhance user trust and understanding of the prediction outcomes.

**Personalized Health Recommendations:** Move towards providing personalized health recommendations based on individual user profiles, lifestyle, and historical health data.

**Research Collaboration:** Foster collaboration with research institutions to contribute valuable insights to the field of machine learning in healthcare and disease prediction.

**Blockchain for Data Security:** Explore the integration of blockchain technology to enhance data security and transparency, addressing concerns related to privacy and confidentiality.

**Incorporation of Genetic Data:** Investigate the inclusion of genetic data for a more nuanced and personalized approach to disease prediction.

**User Engagement Strategies:** Develop strategies to sustain user engagement, including periodic health check reminders, health-related content updates, and gamification elements.

**Regulatory Compliance:** Stay abreast of evolving healthcare regulations and standards, ensuring compliance and ethical use of AI in healthcare.

**By embracing these future avenues, the project has the potential to evolve into a dynamic and indispensable tool in the realm of personalized and accessible healthcare, contributing to a paradigm shift in how individuals engage with their well-being.**

Source Code:<https://github.com/Manish0611/smartinterz-sourcecode>  
GitHub & Project Demo Link:

GitHub personal:<https://github.com/Manish0611/smartInterz-project>

Combined github:<https://github.com/smartinternz02/SI-GuidedProject-599757-1697595489>

Vedio:<https://drive.google.com/file/d/15e0tl4kp08zOUo27oIUktGLLloSAUjuT/view?usp=drivesdk>