# Disease Prediction Using Machine Learning:

In our modern society, where time is often a limited resource, many individuals tend to overlook their healthcare needs until their symptoms become severe. Unfortunately, relying on Google searches for medical information frequently leads to unreliable results, often suggesting stereotypical diseases. Consequently, people have become wary of using the internet to self-diagnose their symptoms.

To address these challenges, we've developed a predictive model capable of identifying up to 42 different diseases based on symptoms alone. This model can serve a dual purpose: it can assist doctors in remote consultations with patients by providing them with valuable insights, and it can also empower individuals to engage in preventive diagnosis and self-care, especially when visiting a doctor may be cost-prohibitive. Importantly, our model does not request any personal information such as names, ages, genders, or addresses, ensuring privacy and anonymity.

You can utilize this web application whenever you have concerns about potential health issues. The model will provide you with probable disease predictions based on your symptoms. This information allows you to make an informed decision about whether it's necessary to consult a healthcare professional.

It's worth noting that our model achieves an impressive accuracy rate of 97 out of 100 times on average. However, it should be used as a tool for early intervention and preventive diagnosis by doctors and patients. It should not be viewed as a definitive diagnosis but rather as an informative resource to guide your healthcare decisions.

Technical Architecture:
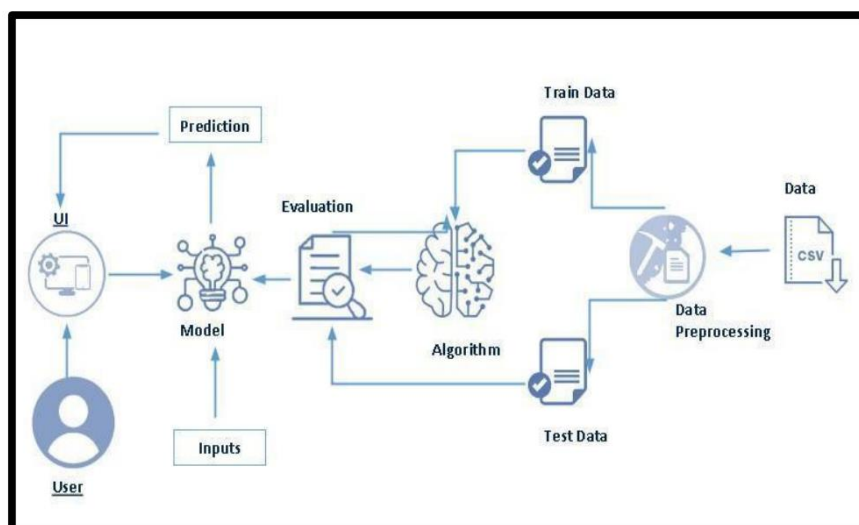
Project Flow:

- **Home Page:** When a user first visits the website, they will land on the Home page. On this page, users can explore and gather information about the platform's features and services.

- **Details Page:** On the Details page, users will be prompted to input the symptoms they are currently experiencing. After entering their symptoms, they can click on the "Predict" button to proceed

- **Predict Page:** If a user decides to proceed and get a disease prediction, they will click on the "Predict" button. This action will lead them to the Details page, where they can provide information about their symptoms.

- **Results Page:** After submitting their symptoms, the user will be redirected to the Results page. Here, our model will analyze the input provided by the user and display the prediction for the most probable disease. This information serves as a helpful reference point for the user's understanding and consideration.

To accomplish this we have to complete all the activities listed below:

- Define problem / Problem understandingo Specify the business problem
    - o Business Requirements
    - o Literature Survey
    - o Social or Business Impact
- Data Collection and Preparation
    - o Collect the dataset
    - o Data Preparation
- Exploratory Data Analysis
    - o Descriptive statistical
    - o Visual Analysis
- Model Building
    - o Creating a function for evaluation
    - o Training and testing the Models using multiple algorithms
- Performance Testing & Hyperparameter Tuning
    - o Testing model with multiple evaluation metrics
    - o Comparing model accuracy before & after applying hyperparameter tuning
    - o Comparing model accuracy for different number of features.
    - o Building model with appropriate features.
- Model Deployment
    - o Save the best model
   - o Integrate with Web Framework

Prior Knowledge:

You must have the prior knowledge of the following topics to complete this project.

- ML Concepts:
    - o Supervised learning: https://www.javatpoint.com/supervised-machine-learning
    - o K Nearest Neighbours:
      https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - o SVM:
      https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
    - o Decision tree:
      https://www.javatpoint.com/machine-learning-decision-tree-classificationalgorithm
    - o Evaluation metrics:
      https://www.analyticsvidhya.com/blog/2019/08/11-important-modelevaluation-error-metrics/

Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create project folder which contains files as shown below:
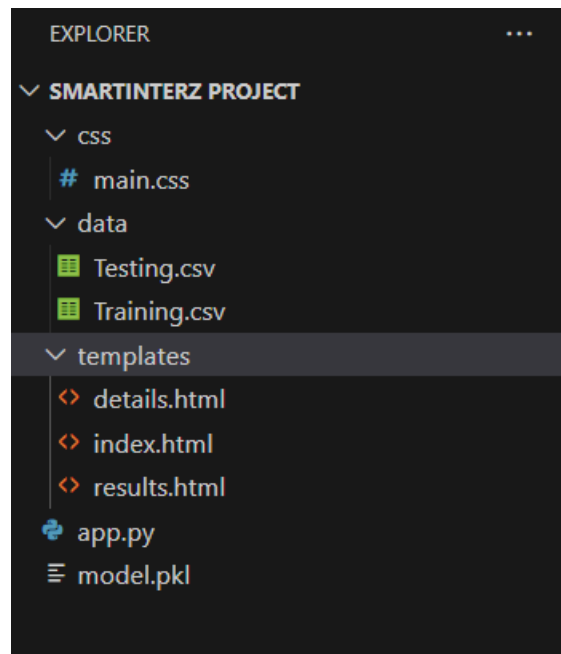
- The data obtained is in two csv files, one for training and another for testing.

- We are building a Flask application which will require the html files to be stored in the templates folder.
- The css files should be stored in the static folder.
- App.py file is used for routing purposes using scripting.
- Model.pkl is the saved model. This will further be used in the Flask integration.
- Training folder contains a model training file.

Milestone 1: Define Problem/ Problem Understanding

❖ Activity 1: Specify the Business Problem:

Disease prediction revolves around the task of identifying individuals who may be prone to developing specific medical conditions, relying on a range of risk factors including medical history and demographic variables. Predictive analytics and machine learning techniques prove invaluable in sifting through vast datasets to unearth patterns and discern the risk factors connected to various diseases.

The process of disease prediction through machine learning entails the deployment of diverse algorithms designed to scrutinize extensive datasets and pinpoint the telltale patterns and risk factors linked to these ailments. Through this analysis, machine learning algorithms excel in spotting individuals who might be at an elevated risk of developing a particular disease. This valuable insight equips healthcare professionals to offer tailored preventive care and early intervention, thereby enhancing patient outcomes.

❖ Activity 2: Business Requirements:

Accurate and Reliable Information: The utmost priority is the accuracy and reliability of the information. In disease prediction, there is zero tolerance for false information. Symptoms should be correctly associated with diseases to ensure that the output aligns with each patient's health situation and variations.

Trustworthiness: Building trust among users is a paramount concern, especially in the context of healthcare. Patients and healthcare professionals must have confidence in the model's predictions, as the consequences of errors can be severe.

Compliance: The model must adhere to all relevant laws and regulations. This includes compliance with healthcare authorities and organizations, such as the Central Drug Standard Control Organization and the Ministry of Health. Ensuring that the model operates within legal frameworks is essential.

User-Friendly Interface: The user interface should be intuitive and user-friendly. It should not require users to input information for which they may not have answers. The model should guide users through the process efficiently, making it accessible to a wide range of individuals.

Meeting these requirements is crucial for the success and acceptance of a disease classification project, as it directly impacts the quality of healthcare services and the user experience.

❖ Activity 3: Literature Survey

Conducting a literature survey for a disease prediction project entails a comprehensive review of prior research, studies, articles, and publications related to the field of disease prediction. This survey serves to accumulate insights into existing classification systems, their merits, limitations, and unexplored areas that the current project can potentially fill. Here's a breakdown of the key aspects involved:

Classification Systems Analysis: The literature review delves into the examination of current disease classification systems. It seeks to understand the strengths and weaknesses of these systems, shedding light on their effectiveness in disease prediction.

Identification of Knowledge Gaps: By scrutinizing existing literature, the survey aims to identify any gaps in knowledge within the domain of disease prediction. These gaps can serve as valuable opportunities for the current project to contribute and advance the field.

Methodologies and Techniques: The review explores the various methods and techniques employed in previous disease prediction projects. It assesses the efficacy of these approaches and identifies best practices that can inform the design and implementation of the current project.

Relevant Data and Findings: The literature survey seeks to extract pertinent data and findings from previous research efforts. These insights provide a foundation for the development of the current project, ensuring that it builds upon prior work and leverages existing knowledge.

In summary, a thorough literature survey is a fundamental step in the planning and execution of a disease prediction project. It equips researchers with the insights and expertise needed to make informed decisions, refine methodologies, and ultimately contribute to the advancement of disease prediction methods.

❖ Activity 4: Social or Business Impact

- **Social Impact**: Enhanced Preventive Diagnosis: By accurately predicting likely diseases based on symptoms, this project significantly improves preventive diagnosis. Users can swiftly identify the probable disease for their symptoms, empowering them to make informed decisions about seeking medical care. This leads to early intervention and better health outcomes.
  Improved Online Consultations: Healthcare professionals can provide more effective online consultations, as they receive well-informed patient data. This not only benefits patients but also streamlines the healthcare system by reducing unnecessary in-person visits.

o Business Impact:Expanded Patient Reach: Doctors can now extend their services to a broader spectrum of patients through online consultations. This results in a more efficient healthcare system, with a reduced burden on physical healthcare facilities.
Hospital Rush Alleviation: With a portion of patient consultations shifting online, hospitals experience decreased rush and congestion, allowing them to allocate resources more efficiently. Critical patients receive better care, and healthcare facilities operate more smoothly.

Optimized Testing: Doctors can recommend specific tests for patients before in-person visits. This targeted approach ensures that the necessary diagnostic procedures are completed before the patient arrives, enabling doctors to make faster and more accurate diagnoses.

❖ **Milestone 2: Data Collection and Preparation:**

Machine Learning heavily relies on data as it forms the fundamental component for training algorithms. This section provides guidance on how to obtain the dataset.

o **Activity 1: Collect the Dataset:**
Data for machine learning projects can be sourced from various reputable open platforms, such as Kaggle, the UCI repository, and more. In this project, we've utilized a .csv dataset downloaded from Kaggle.com. To access and obtain this dataset, please follow the link provided below:

Dataset Download Link:

https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning

Once the dataset is downloaded, the next step is to thoroughly understand the data. This involves using visualization and analysis techniques to gain insights.

Note: There are several techniques available for data understanding. In this context, we have applied a set of techniques. However, you can explore and utilize multiple methods to comprehensively grasp the dataset's characteristics.

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.

```python
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.

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

```python
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 × 134 columns

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

```python
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 × 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.

- o **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)
```

o **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

o **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 havedisplayed three different types of graphs and plots.
For simple visualizations we can use the matplotlib.pyplot library

```python
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 distributlon 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 sympton into number of yes/no")
plt.subplots_adjust(left= 0.5,right=2.4)
```

pie chart showing the distributlon of Itching syepto- into number of yes/No
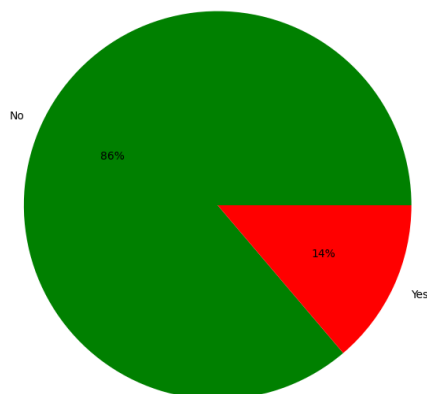
pie Chart showing the diatribution of continuous sneezing sympton into number of yes/no
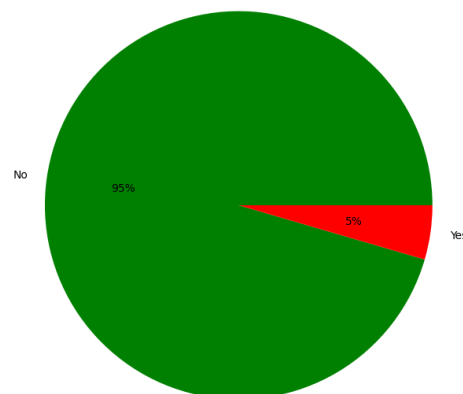
In this visualization process, we start by using the plt.figure() command to set the size of the plot. We then divide the space into two segments for two pie plots. Here's what each pie plot represents:

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.

```python
a = len(df[df['prognosis'] == 'Tuberculosis'])

b = len(df[(df['yellowing_of_eyes'] == 1) & (df['prognosis'] == "Tuberculosis")])

fi = pd.DataFrame({'Values': [a, b]}, index=['Tuberculosis', 'Yellowing of Eyes with Tuberculosis'])

sns.barplot(data=fi, x=fi.index, y=fi['Values'])

plt.title('Importance of Yellowing of Eyes Symptom to Determine Tuberculosis')

plt.show()
```


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 bedone primarily with the help of Correlation matrix.
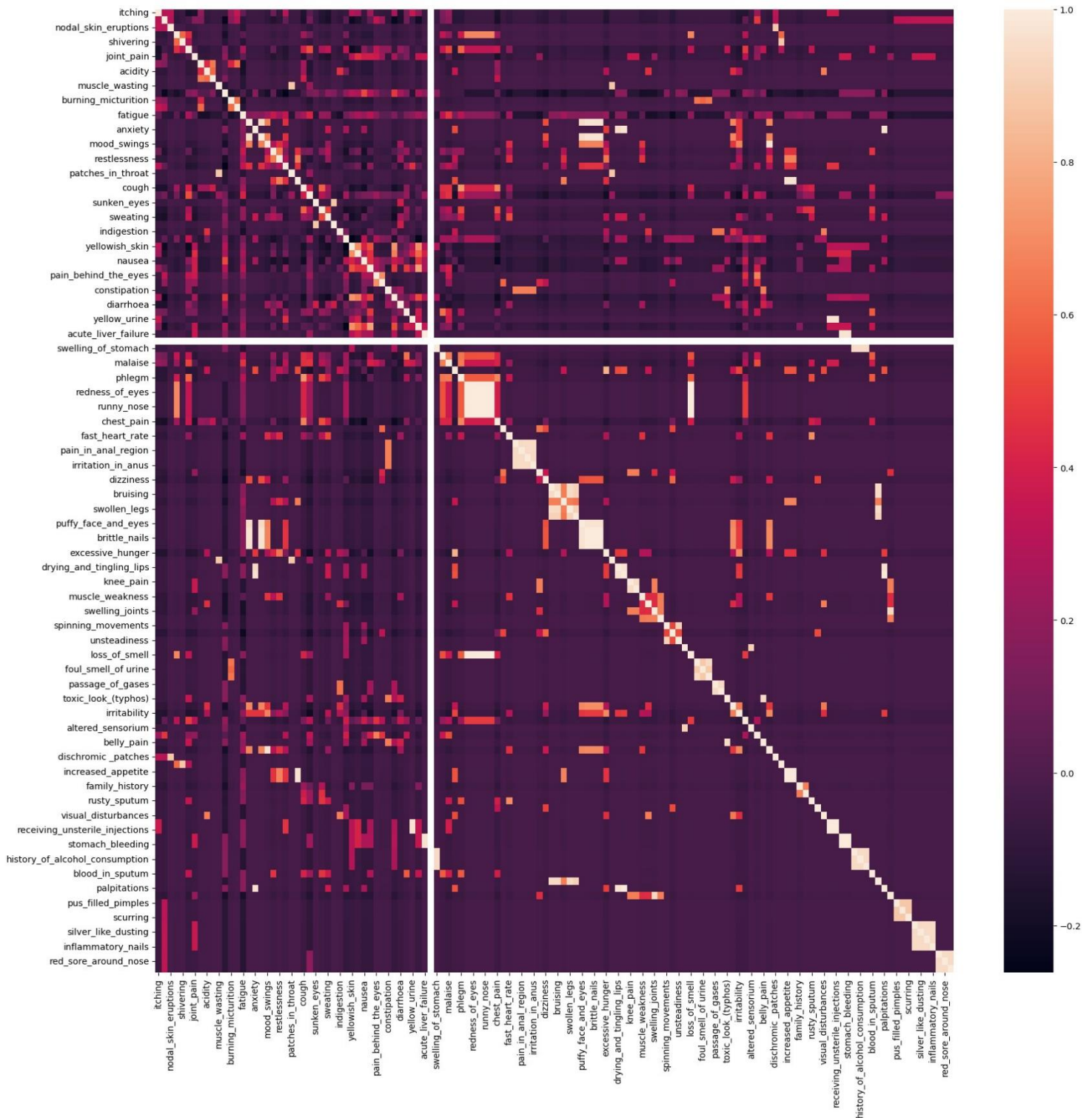
which have numerical values the correlation matrix is of dimensions 131 X

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. Weremove columns where the correlation between the columns is above 0.9

```python
plt.figure(figsize=(10,4))
sns.countplot(data=df,x='prognosis')
plt.xticks(rotation=90)
plt.xlabel('Diseases');
```
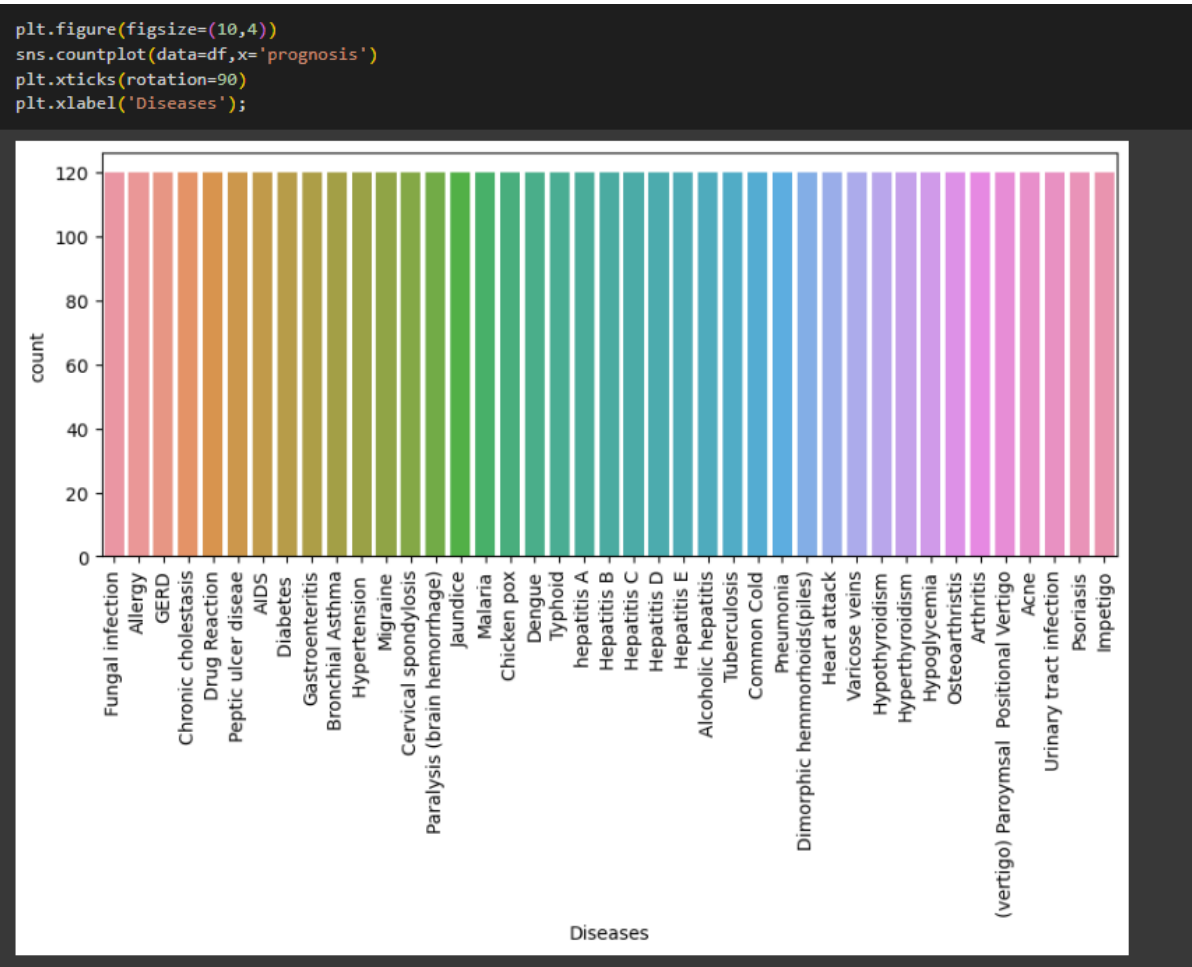

Activity 2.5: Split data into training, validation and testing data

We have training and testing data given separately. We further split the training data intotraining 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 predictthe target variable.

```python
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 usingthe 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

o Activity 1: Creating a function for model evaluation

We will be creating multiple models and then testing them. It will reduce our monotonoustask 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 andtesting data. It will also the 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.

```python
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)
```

```python
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
    }
```

```
kn_results = model_evaluation(knn, X_test, y_test)
print("Accuracy:", evaluation_results["Accuracy"])
print("Confusion Matrix:\n", evaluation_results["Confusion Matrix"])
print("Classification Report:\n", evaluation_results["Classification Report"])
```

```
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) 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
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
Osteoarthristis	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

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 hyperparametertuning. 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.

```python
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)
```

```
▾ SVC
SVC(C=1)
```

```python
svm_res=model_evaluation(svm, X_test, y_test)
svm_res
```

```
{'Accuracy': 1.0,
 'Confusion Matrix': array([[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\n\n(vertigo) Paroymsal  Positional Vertigo     1.00      1.00      1.00        18\n                     AIDS       1.00
          1.00      1.00        30\n                              Acne       1.00      1.00      1.00        24\n         Alcoholic hepatitis     1.00      1.00      1.00        25\n                Allergy       1.00      1.00
          1.00        24\n                         Arthritis       1.00      1.00      1.00        23\n            Bronchial Asthma     1.00      1.00      1.00        33\n        Cervical spondylosis     1.00      1.00      1.00
23\n                       Chicken pox       1.00      1.00      1.00        21\n         Chronic cholestasis     1.00      1.00      1.00        15\n            Common Cold     1.00      1.00      1.00        23\n
Dengue         1.00      1.00      1.00        26\n                           Diabetes       1.00      1.00      1.00        21\n   Dimorphic hemmorhoids(piles)     1.00      1.00      1.00        29\n                   Drug
Reaction       1.00      1.00      1.00        24\n                  Fungal infection       1.00      1.00      1.00        19\n                         GERD     1.00      1.00      1.00        28\n
Gastroenteritis       1.00      1.00      1.00        25\n                       Heart attack       1.00      1.00      1.00        23\n                  Hepatitis B     1.00      1.00      1.00        27\n
Hepatitis C       1.00      1.00      1.00        26\n                        Hepatitis D       1.00      1.00      1.00        23\n                  Hepatitis E     1.00      1.00      1.00        29\n
Hypertension       1.00      1.00      1.00        25\n                   Hyperthyroidism       1.00      1.00      1.00        24\n                  Hypoglycemia     1.00      1.00      1.00        26\n
Hypothyroidism       1.00      1.00      1.00        21\n                          Impetigo       1.00      1.00      1.00        24\n                     Jaundice     1.00      1.00      1.00        19\n
Malaria       1.00      1.00      1.00        22\n                           Migraine       1.00      1.00      1.00        25\n                Osteoarthristis     1.00      1.00      1.00        22\n        Paralysis (brain
hemorrhage)       1.00      1.00      1.00        24\n            Peptic ulcer diseae       1.00      1.00      1.00        17\n                  Pneumonia     1.00      1.00      1.00        28\n
Psoriasis       1.00      1.00      1.00        22\n                       Tuberculosis       1.00      1.00      1.00        25\n                     Typhoid     1.00      1.00      1.00        19\n                Urinary tract
infection       1.00      1.00      1.00        26\n                      Varicose veins       1.00      1.00      1.00        22\n                  hepatitis A     1.00      1.00      1.00        34\n\n
accuracy                            1.00       984\n                         macro avg       1.00      1.00      1.00       984\n                 weighted avg     1.00      1.00      1.00       984\n'}
```

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 hyperparametertuning. The results are stored in a variable named svm_results.

❖ 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)
```

```
accuracy score =  1.0

 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
```

```
            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
           Osteoarthristis      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
```

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 arehigh, there is no need for hyperparameter tuning. The results are stored in a variable named dtc_results.

❖ 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))
```

```
accuracy score =  1.0

 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
                         Osteoarthristis      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
```

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 arehigh, 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.

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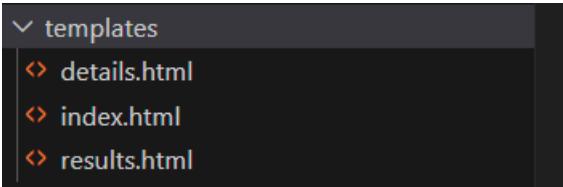
It manages the flow of data between the front-end and the back-end of the web application.

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
  <> details.html
  <> index.html
  <> results.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.

```python
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.

```python
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.

```python
@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

```python
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]
```
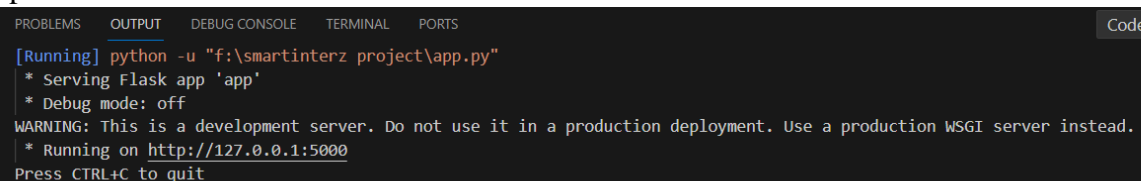
Main Function:

This code sets the entry point of the Flask application. The function "app.run()" iscalled, which starts the Flask deployment server.

```python
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 varioussections 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 tothe details.html page.

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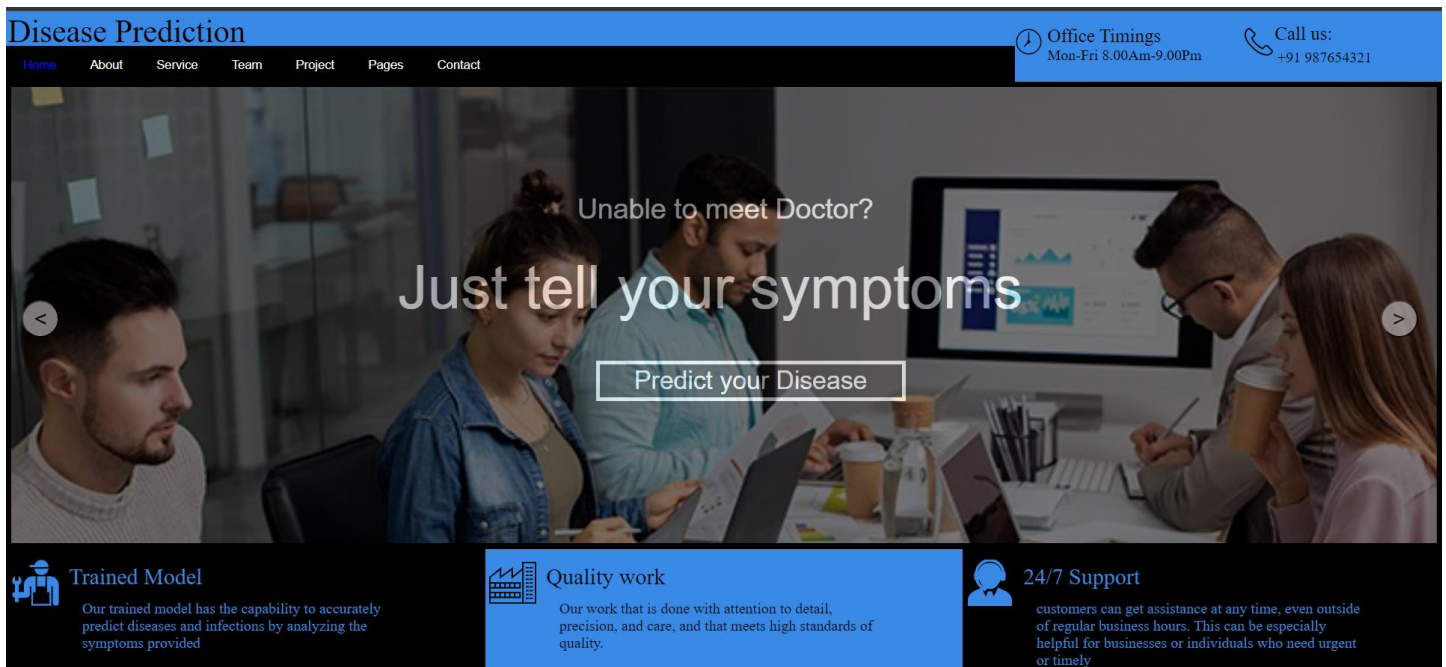
So basically tells about the our company details and work, once if we click on "Predict your Disease" button you will get directed to index.html

Index.html:

○ 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 bult where you will just tick/click on the symtoms 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

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 .