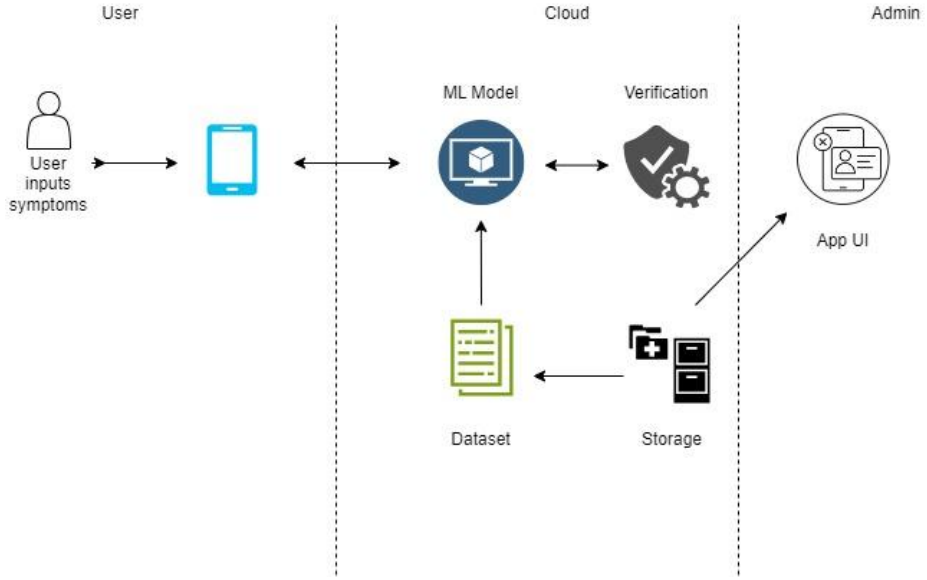
# Disease Prediction Using Machine Learning

In today's fast paced world machine learning  is crucial for addressing demand for data-driven decision-making, automation, and efficiency. ML enables real-time data analysis, predictive modeling, and automation in various domains, including finance, healthcare, e-commerce, logistics, and more. It empowers businesses to extract insights from vast datasets, automate repetitive tasks, enhance personalization, improve cybersecurity, and respond rapidly to changing circumstances. In a fast world, ML plays a vital role in staying competitive, optimizing operations, and delivering innovative solutions that meet the demands of a rapidly evolving technological landscape.

Disease prediction using machine learning is essential in healthcare for its ability to enable early diagnosis, improve preventive care, personalize treatment, manage population health, and reduce healthcare costs. By analyzing patient data, machine learning models can identify individuals at risk, offer precision medicine, and empower patients to take control of their health. This proactive approach to healthcare enhances patient outcomes and resource allocation, ultimately benefiting both individuals and healthcare systems.

## Technical Architecture



# Project Flow

User is first taken to the home page and then he enters the prediction page where they can input his specific symptoms.After that they can click the Predict button to navigate to the page where the result of the ML model will be displayed.

1)Define problem / Problem understanding
2)Data Collection and Preparation
3)Exploratory Data Analysis
4)Model Building
5)Performance Testing & Hyperparameter Tuning
6)Model Deployment

## Milestone 1:Define problem / Problem understanding

### Business Problem

Disease prediction involves the systematic identification of individuals with an elevated likelihood of developing particular health conditions, utilizing various risk factors like medical history and demographic variables. Predictive analytics and machine learning methodologies are harnessed to analyze extensive datasets, uncover patterns, and pinpoint risk elements linked to various diseases.

### Business Requirements

A disease classification project can have a variety of business requirements, depending on the specific goals and objectives of the project. Some potential requirements may include:

- Accurate and reliable information:

The case of disease prediction is critical and no false information can be tolerated since the consequences can be severe. Also the right symptoms should be linked to the right diseases so that the output is inline with all the patients health situations and variations.

- Trust:

Trust needs to be developed for the users to use the model. It is difficult to create trust among patients while dealing with a healthcare problem.

- User friendly interface:

The interface should be easy to use and understand by the user. The model should not ask inputs for which the user does not have answers.

## Literature Survey

- *Cardiovascular Diseases*: Several studies have applied machine learning techniques to predict the risk of cardiovascular diseases, such as heart attacks and strokes. Commonly used algorithms include random forests, support vector machines, and deep learning models. These studies leverage patient data, including age, blood pressure, and cholesterol levels, to achieve high accuracy in predicting disease risk.

- *Cancer Prediction*: ML-based cancer prediction models have been developed for various cancer types, such as breast, lung, and prostate cancer. These models analyze genetic data, histopathological images, and clinical data to identify individuals at high risk of developing cancer, aiding in early detection and personalized treatment.

- *Diabetes Prediction*: Machine learning is also employed in the early prediction of diabetes. Researchers have used features like body mass index (BMI), family history, and dietary habits to create predictive models. These models assist in lifestyle recommendations and targeted interventions for diabetes prevention.

- *Broader Applications*: Machine learning extends beyond specific diseases. Some studies focus on the broader application of ML in healthcare, including patient risk stratification, medical image analysis, and electronic health record analysis, contributing to disease prediction in a more holistic manner.

## Social and Business Impact

Users can see what diseases they are most likely prone to and take the appropriate action on time without waiting too long and minimizing costs.

On a business level,doctors can treat more patients more effectively using online consultations by reducing hospital rush.

# Milestone 2:Data Collection and Preparation

## Dataset

In this project we have used .csv data. This data is downloaded from kaggle.com.
https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning

## Libraries Used

```
[1]  import numpy as np
     import pandas as pd

     import matplotlib.pyplot as plt
     import seaborn as sns

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

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

     import pickle
```

## Reading the dataset

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

```
[3]  train.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

```
[5]  train.shape
```

```
(4920, 134)
```

## Data Preparation

We have to remove the redundant data and handle missing values.

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

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

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

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

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

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

     0
```

There are no missing values in this dataset.

# Milestone 3:Exploratory Data Analysis

## Descriptive Statistics

Descriptive analysis is to study the basic features of data with the statistical process. With this describe function we can understand the unique, top and frequent values of categorical features. We can also find mean, std, min, max and percentile values of continuous features.

```
train.describe()
```

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

8 rows × 132 columns

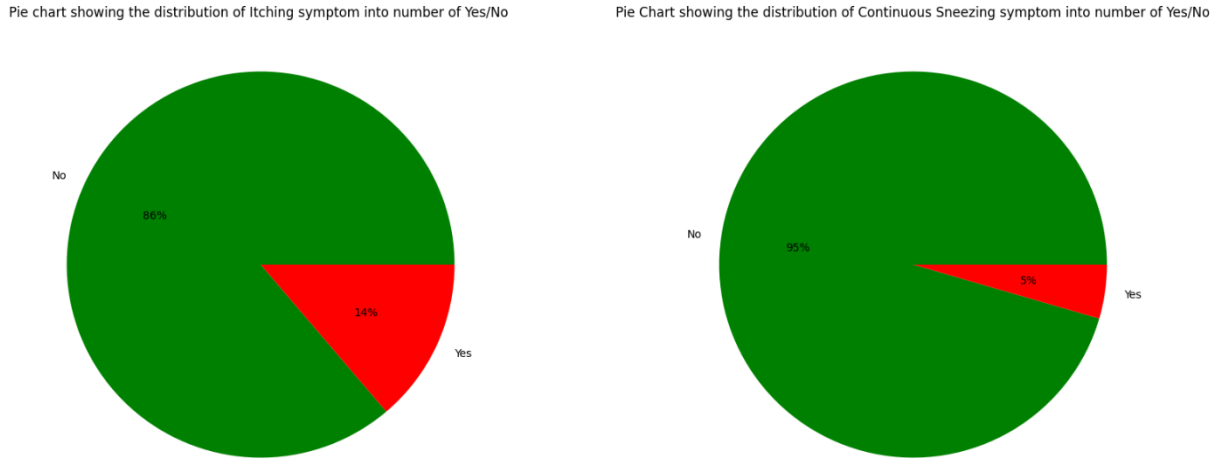
# Visualization

## 1)Univariate Analysis:

We use the matplotlib and seaborn library for visualization.

### Pie Chart

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

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

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

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

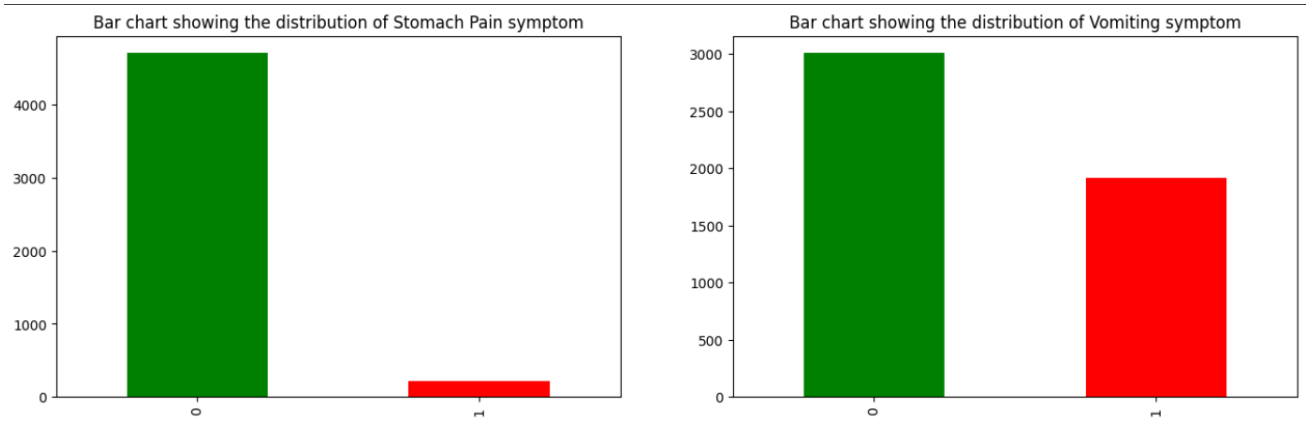



86% of diseases in this dataset do not have itching as a symptom.95% of the diseases in this

dataset also do not have continuous sneezing as a symptom.

**Bar Chart**

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

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

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



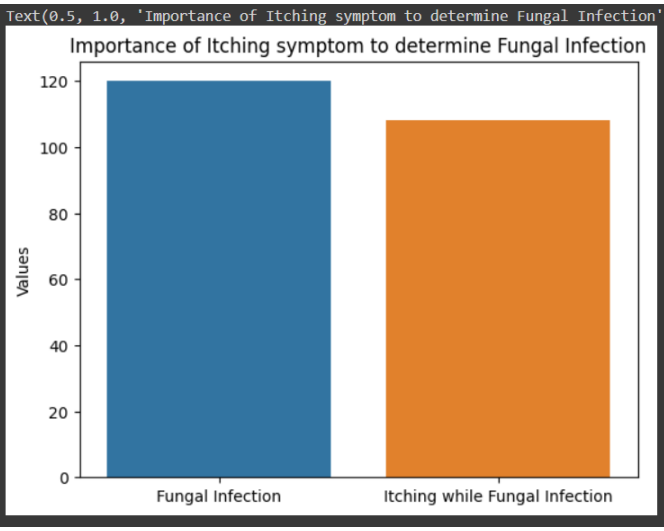

The bar graph on the left shows the distribution of stomach pain symptom values. We can see that the 0 value has a count of around 4700 and the 1 value has a count of around 400.
The graph on the right shows the distribution of vomiting symptom values. We can see that the 0 value has a count of around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

**2)Bivariate Analysis**

Itching with respect to fungal infection

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

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

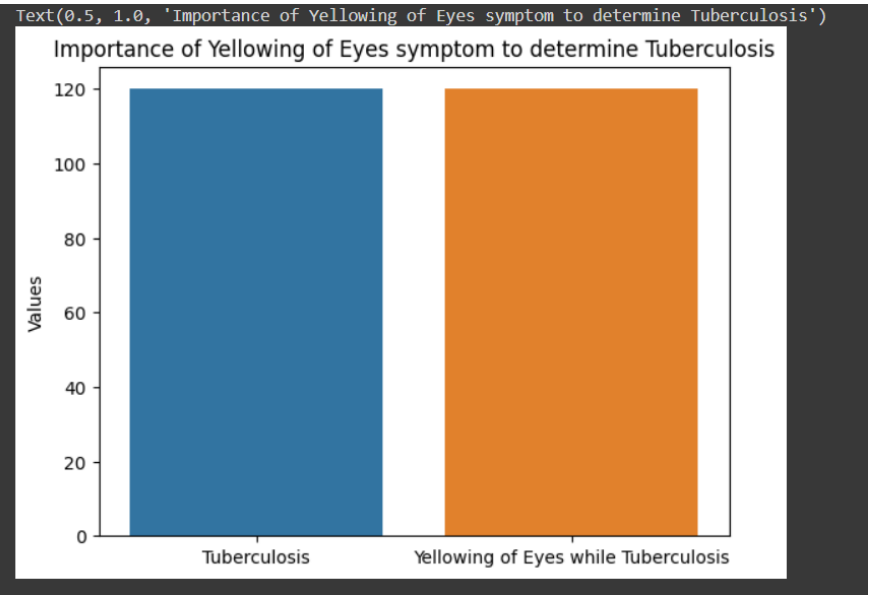
```
Text(0.5, 1.0, 'Importance of Itching symptom to determine Fungal Infection')
```




Yellowing of eyes with respect to tuberculosis

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

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

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




\

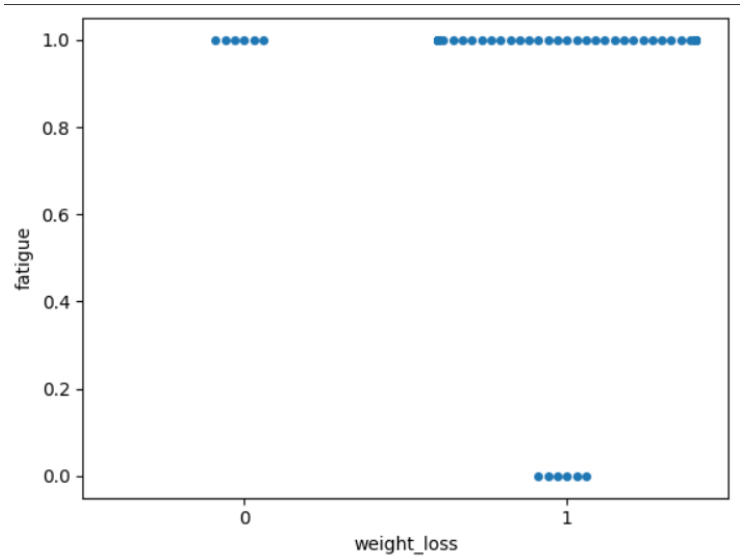
### 3)Multivariate Analysis

Correlation of all symptoms to each other

```
[16] corr = train.corr()
     corr.style.background_gradient('coolwarm')
```

```
<ipython-input-16-92accbbd8d53>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it wi
  corr = train.corr()
```

| | itching | skin_rash | nodal_skin_eruptions | continuous_sneezing | shivering | chills | joint_pain | stomach_pain | |
|---|---|---|---|---|---|---|---|---|---|
| itching | 1.000000 | 0.318158 | 0.326439 | -0.086906 | -0.059893 | -0.175905 | -0.160650 | 0.202850 | -( |
| skin_rash | 0.318158 | 1.000000 | 0.298143 | -0.094786 | -0.065324 | -0.029324 | 0.171134 | 0.161784 | -( |
| nodal_skin_eruptions | 0.326439 | 0.298143 | 1.000000 | -0.032566 | -0.022444 | -0.065917 | -0.060200 | -0.032566 | -( |
| continuous_sneezing | -0.086906 | -0.094786 | -0.032566 | 1.000000 | 0.608981 | 0.446238 | -0.087351 | -0.047254 | -( |
| shivering | -0.059893 | -0.065324 | -0.022444 | 0.608981 | 1.000000 | 0.295332 | -0.060200 | -0.032566 | -( |
| chills | -0.175905 | -0.029324 | -0.065917 | 0.446238 | 0.295332 | 1.000000 | -0.004688 | -0.095646 | -( |
| joint_pain | -0.160650 | 0.171134 | -0.060200 | -0.087351 | -0.060200 | -0.004688 | 1.000000 | -0.087351 | -( |
| stomach_pain | 0.202850 | 0.161784 | -0.032566 | -0.047254 | -0.032566 | -0.095646 | -0.087351 | 1.000000 | ( |
| acidity | -0.086906 | -0.094786 | -0.032566 | -0.047254 | -0.032566 | -0.095646 | -0.087351 | 0.433917 | |
| ulcers_on_tongue | -0.059893 | -0.065324 | -0.022444 | -0.032566 | -0.022444 | -0.065917 | -0.060200 | 0.649078 | ( |
| muscle_wasting | -0.059893 | -0.065324 | -0.022444 | -0.032566 | -0.022444 | -0.065917 | -0.060200 | -0.032566 | -( |
| vomiting | -0.057763 | -0.225046 | -0.119543 | -0.173459 | -0.119543 | 0.144263 | 0.199921 | 0.031406 | ( |
| burning_micturition | 0.207896 | 0.166507 | -0.032103 | -0.046581 | -0.032103 | -0.094285 | -0.086108 | 0.412239 | -( |
| spotting_ urination | 0.350585 | 0.298143 | -0.022444 | -0.032566 | -0.022444 | -0.065917 | -0.060200 | 0.608981 | -( |




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

### Preprocessing test data

This function is used to drop features that are not required for the test data.

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

```
[34] test = data_preprocessing(test)
```

### Data Splitting

We further split the training data into training and validation data. Validation data is used for hyper parameter tuning in the later stage.

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

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

Splitting the training data into x and y tests.

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

Further splitting into training and validation data.

# Milestone 4:Model Building

## Function for model evaluation

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

We can use this function for testing the different accuracies of the different algorithms we use.

## 1)SVM Model

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

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

```
[40] svm_results = model_evaluation(svm)
```

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

We first create a variable to initialize the svm function and use the .fit to train our data.Then using the earlier evaluation function we check the accuracy of the model.
The accuracy of this model is 1 meaning no hyper parameter testing is required.

## 2)Random Forest Model

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

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

```
[44] rfc_results = model_evaluation(rfc)
```

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

Again we initialize the random forest classifier algorithm to a variable rfc and use the .fit function to train the model.Then we test the accuracies and find that the accuracies are pretty high.

## 3)Decision Tree Model

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

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

```
[42] dtc_results = model_evaluation(dtc)
```

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

Here also we use a variable dtc to store the decision tree classifier algorithm and then use the .fit method to train the mode.Upon testing the accuracies of this model we see that the accuracy is high.

## 4)KNN Model

```
[37] knn = KNeighborsClassifier(n_neighbors=7)
     knn.fit(X_train,y_train)
```

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

```
[38] knn_results = model_evaluation(knn)
```

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

The K nearest neighbors classifiers model is initialized to a variable knn with n value =7. The .fit variable is used to train the model.The accuracies of these are checked using the function.

# Milestone 5:Performance Testing & Hyperparameter Tuning

Our training dataset has 90 features making it not optimal to create a model.To find the optimal number of features we need to find what is the perfect number of features according to their respective accuracies.

```
[47] a = rfc.feature_importances_

[48] col = X.columns

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

feat_imp is a dictionary which stores 90 columns and their respective importance.

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

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

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

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

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

[55] knn_table

| | Number of features | Training Accuracy | Testing Accuracy |
|---|---|---|---|
| 0 | 6 | 0.219004 | 0.214286 |
| 1 | 11 | 0.335874 | 0.333333 |
| 2 | 16 | 0.452490 | 0.452381 |
| 3 | 25 | 0.685722 | 0.690476 |
| 4 | 36 | 0.794207 | 0.809524 |
| 5 | 49 | 0.866362 | 0.880952 |
| 6 | 62 | 0.989837 | 0.976190 |

[69] randomf

| | Number of features | Training Accuracy | Testing Accuracy |
|---|---|---|---|
| 0 | 6 | 0.225102 | 0.214286 |
| 1 | 11 | 0.337144 | 0.333333 |
| 2 | 16 | 0.455030 | 0.452381 |
| 3 | 25 | 0.690803 | 0.690476 |
| 4 | 36 | 0.795478 | 0.785714 |
| 5 | 49 | 0.893801 | 0.880952 |
| 6 | 62 | 0.990091 | 0.976190 |

We see that the accuracies don't change much after 45 features meaning that we can use 49 attributes for our optimal model and drop the rest.

```
[57] len(to_drop)
```

```
40
```

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

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

| | itching | joint_pain | vomiting | spotting_ urination | fatigue | weight_loss | high_fever | sunken_eyes | dark_urine | nausea | ... | family_history | mucoid_sputum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 |

5 rows × 49 columns

We drop 40 features. As we can see in the figure above we are left with 49 features now. We will build a KNN classifier based on the new data and check for the accuracies.

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

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

```
KNeighborsClassifier
KNeighborsClassifier()
```

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

     The Training Accuracy of the algorithm is  0.9639227642276422
     The Validation Accuracy of the algorithm is  0.9613821138211383
     The Testing Accuracy of the algorithm is 0.9523809523809523
```

Our model has achieved 96.3 % accuracy for the test data.

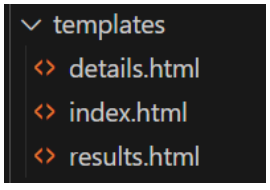
# Milestone 6: Model Deployment

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future**.**
After checking the performance, we decided to save the KNN model built with 45 features.

We will be building a web application that is integrated to the model we built.
We will create three HTML files namely

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

And we will save them in the templates folder.



This Flask app named app.js serves a machine learning model for diagnosing medical conditions based on symptoms entered. It loads a pre-trained model from a 'model.pkl' file, receives symptom inputs through a form, converts the input into a format compatible with the model, predicts the possible medical condition, and displays the diagnosis on a webpage using Flask's render_template. The '/' route directs to the home page, '/details' to additional information, and '/predict' receives the form data, processes it, and shows the predicted diagnosis on 'results.html'.

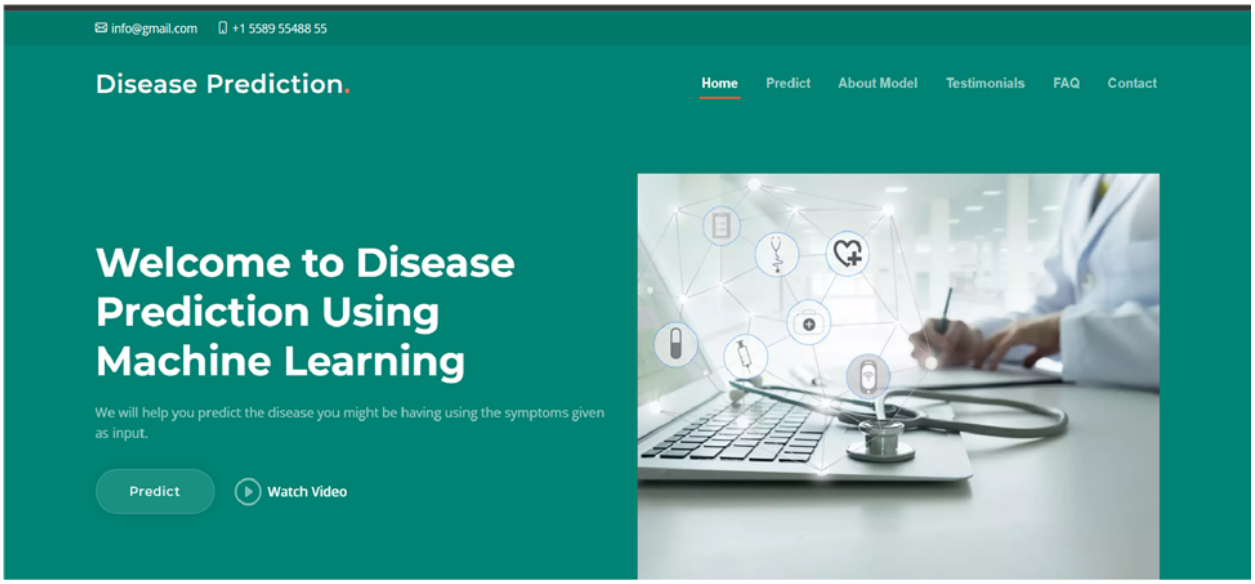
```python
from flask import Flask, render_template, request
import numpy as np
import pickle

model = pickle.load(open('C:/Users/ATHISH/Downloads/model.pkl','rb'))
app = Flask(__name__)

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

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

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

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


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

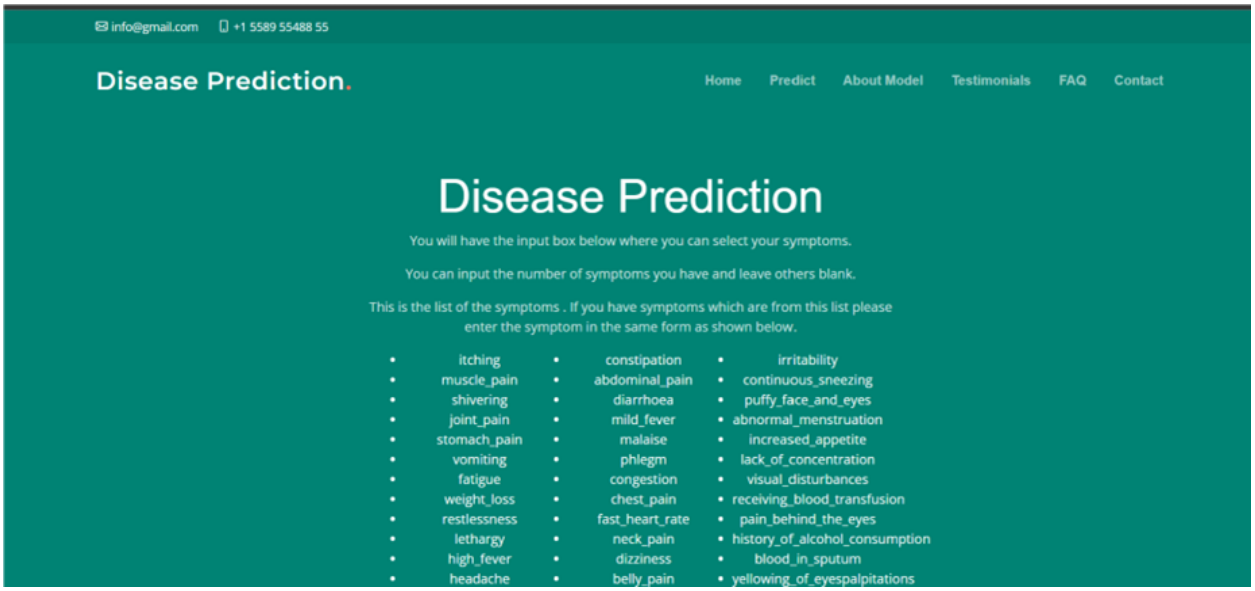
Output:

```
[Running] python -u "c:\Users\ATHISH\Downloads\Disease-Prediction-ML-Flask-main\Disease-Prediction-ML-Flask-main\app.py"
C:\Users\ATHISH\AppData\Roaming\Python\Python38\site-packages\sklearn\base.py:348: InconsistentVersionWarning: Trying to unpickle estimator
KNeighborsClassifier from version 1.2.2 when using version 1.3.2. This might lead to breaking code or invalid results. Use at your own risk. For more info
please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * 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 the home page or in the header bar you will be redirected to the details.html page.

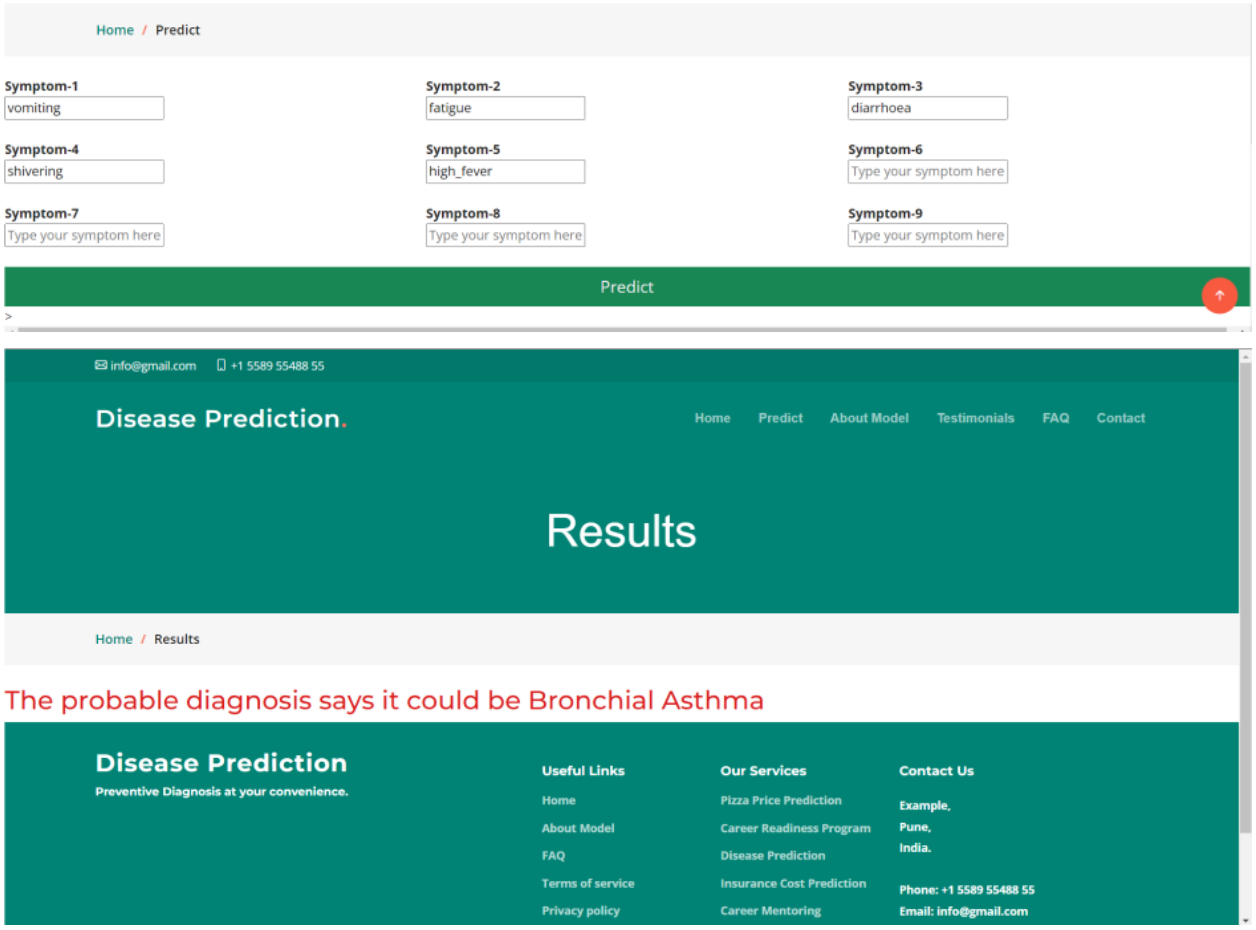



We are provided with 9 input fields where we can input our symptoms. We can fill all 9 boxes or can just fill 1. We are provided with a list of symptoms in the form of bullet points above.
We can input symptoms from above but they have to be in the same form as given in the list since they are the column names.

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

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




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