Disease Prediction Using Symptoms

A PROJECT BY

Parth Raut (33359)
Gaurav Sable (33360)
Dhananjay Salunke (33361)
Saurav Mohanty (33363)

ABSTRACT

Health is an important part of our lives. Health plays an important part in determining an individual's efficiency in day to day lives. When we encounter any disease it affects our routine and causes a break in our efficiency. In order to get back to our regular lifestyle we consult a doctor and follow their guidelines. Consulting Doctor isn’t always an easy task for some people as the hospitals aren't always reachable maybe due to location or maybe due to overburden on healthcare facilities. This project is about finding solutions such that people would be able to know what they might be suffering from in order to make a decision for further treatment . This problem when solved will prove to be beneficial in terms of both time and health. Accordingly we propose a disease predicting technique that consists of following key steps: First, the symptoms would be taken from the user. Then key questions based on the symptoms selected would be asked and based on the answers to these questions, a summary would be generated and would be presented to the user. Based on this report, users can decide their further steps.

Keywords : health, key-questions, summary, further-steps, report, disease-prediction.

INTRODUCTION

In today's modern e-health era, the internet has become a popular mode of acquiring knowledge regarding personal health. However, most healthcare consumers primarily use search engines to retrieve health information based upon their symptoms. Many times they end up trusting the top-ranked search results.This causes misunderstanding and confusion about their health status as they trust top-ranked search results even without filtering them.
This can result in social issues like suffering from cyberchondria which is a clinical phenomenon in which repeated Internet searches regarding medical information result in excessive concerns about physical health [1].
These search results can be very misleading and many times end up harming rather than helping the healthcare consumers. Primary disadvantages of using search engines to retrieve health information include:

A.RESULTS BASED ON SEARCH ENGINE'S RANKING ALGORITHM :

In general, the results which show up after searching symptoms on the internet are due to the search engine's ranking algorithm. This does not mean that the results which show up at top are most comprehensive, reliable and from accurate sources. They might be just up there due to clever Search Engine Optimization work.

B.INCREASE IN CASES OF CYBERCHONDRIA :

Cyberchondria refers to a clinical phenomenon in which repeated Internet searches regarding medical information result in excessive concerns about physical health. Cyberchondria is positively associated with symptoms of health anxiety, though it remains unclear as to whether cyberchondria poses a unique public burden.

C.DISREGARD OF PERSONAL HEALTH :

Searching for personal health on the search engine leads to disregard on personal health. Most of the healthcare consumers end up following the information retrieved from the search engines and this leads to negligence towards getting proper treatment / diagnosis.

OBJECTIVES

1. To predict disease that the user may have from the symptoms entered by the user.
2. To give the user the basic idea of which disease they have before going to the doctor.
3. To give information about the disease.
4. To give basic and important precautions for that disease.

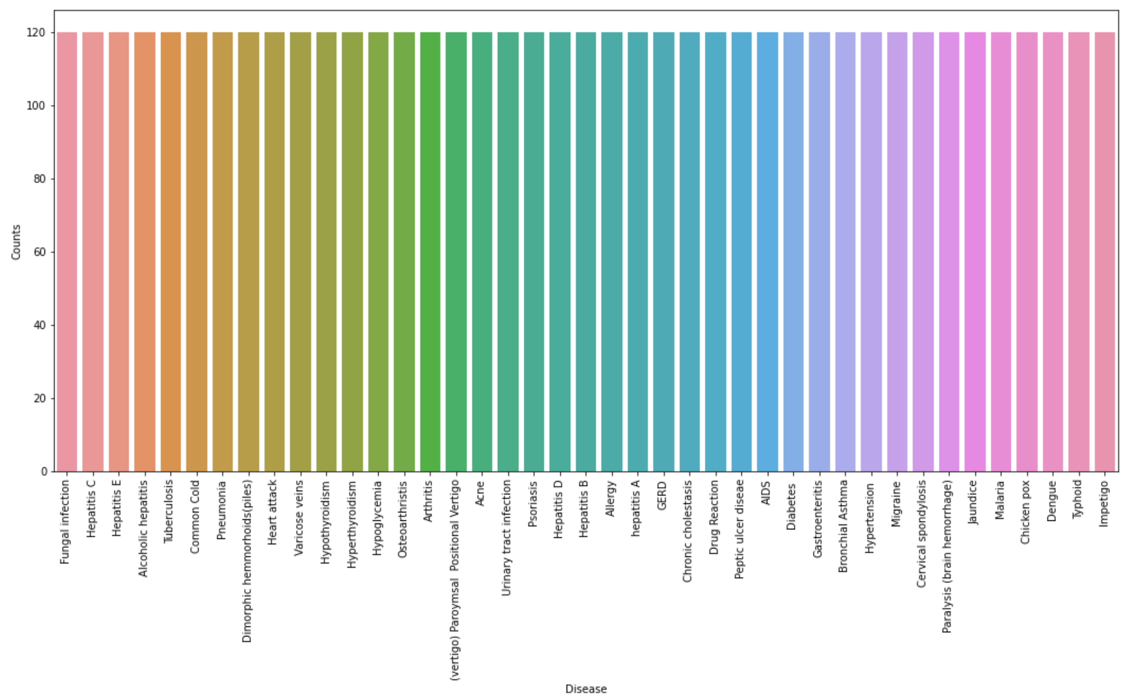
SCOPE

This project aims to help every healthcare consumer who wants to get help with diagnosing themselves based on their symptoms before consulting a doctor. The idea is to get symptoms from the user and generate a diagnosis by asking the user basic questions regarding the symptoms selected by them. The questions would judge the extent of severity of the symptoms and hence the conclusion will be drawn.

The main objective of the project is to take symptoms as input from the user and then feed the input to the machine learning model and then predict the disease based on the inputted symptoms. On the basis of prediction the severity of the symptom would be considered. Finally considering all symptoms a conclusion will be drawn and the user will be prompted about the disease that he/she is most likely to get affected.
The current scope of the project involves diagnosis and report generation. The future scope of the project is discussed later in this report.

Implementation

```
# Importing libraries
import numpy as np
import pandas as pd
from scipy.stats import mode
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neural_network import MLPClassifier

%matplotlib inline

# Reading the train.csv by removing the
# last column since it's an empty column
DATA_PATH = "Training.csv"
data = pd.read_csv(DATA_PATH).dropna(axis = 1)

# Checking whether the dataset is balanced or not
disease_counts = data["prognosis"].value_counts()
temp_df = pd.DataFrame({
      "Disease": disease_counts.index,
      "Counts": disease_counts.values
})

plt.figure(figsize = (18,8))
sns.barplot(x = "Disease", y = "Counts", data = temp_df)
plt.xticks(rotation=90)
plt.show()
```



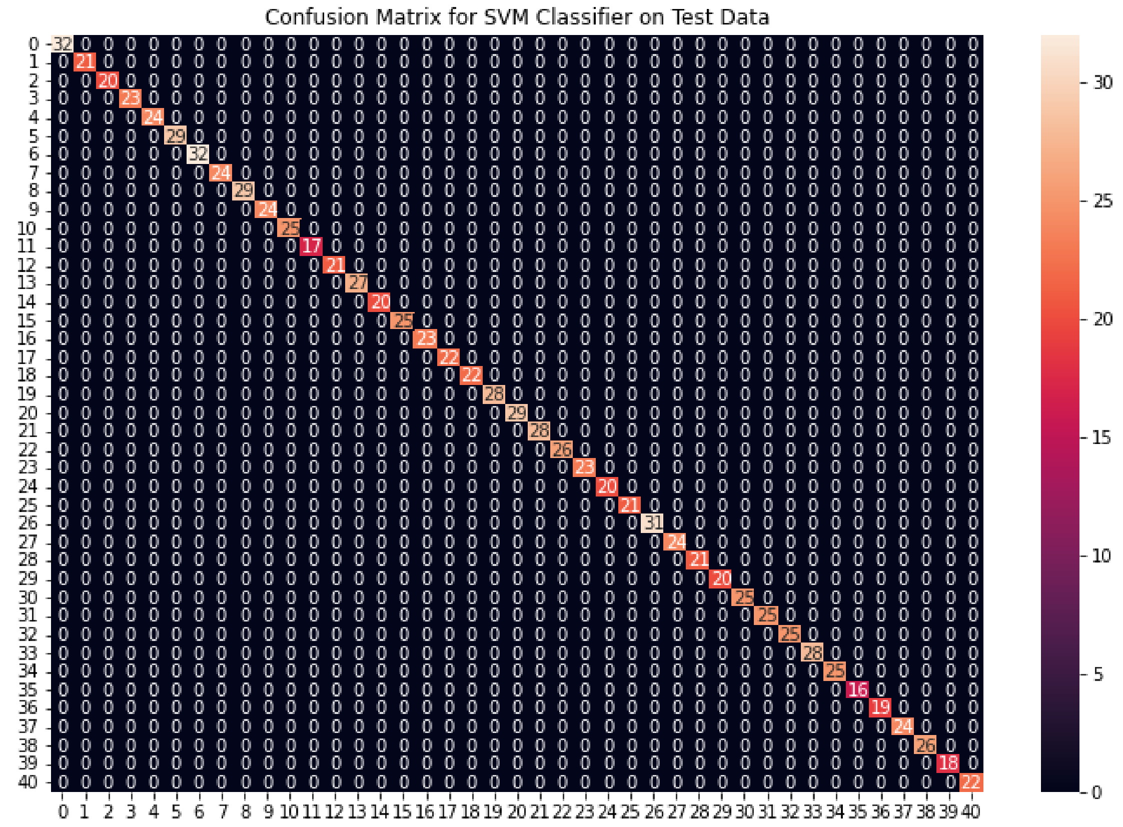
```
# Encoding the target value into numerical
# value using LabelEncoder
encoder = LabelEncoder()
data["prognosis"] = encoder.fit_transform(data["prognosis"])



X = data.iloc[:,:-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test =train_test_split(
X, y, test_size = 0.2, random_state = 24)

print(f"Train: {X_train.shape}, {y_train.shape}")
print(f"Test: {X_test.shape}, {y_test.shape}")

Train: (3936, 132), (3936,)
Test: (984, 132), (984,)

# Defining scoring metric for k-fold cross validation
def cv_scoring(estimator, X, y):
      return accuracy_score(y, estimator.predict(X))

# Initializing Models
models = {
      "SVC":SVC(),
      "Gaussian NB":GaussianNB(),
      "Random Forest":RandomForestClassifier(random_state=18)
}

# Producing cross validation score for the models
for model_name in models:
      model = models[model_name]
      scores = cross_val_score(model, X, y, cv = 10,
                                            n_jobs = -1,
                                            scoring = cv_scoring)
      print("=="*30)
      print(model_name)
      print(f"Scores: {scores}")
      print(f"Mean Score: {np.mean(scores)}")

============================================================
SVC
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Gaussian NB
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Random Forest
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0

print(X_test)

      itching  skin_rash  nodal_skin_eruptions  continuous_sneezing  \
3825        0          0                     0                    0
2368        0          0                     0                    0
178         0          1                     0                    0
4368        0          0                     0                    0
4078        0          0                     0                    0
...       ...        ...                   ...                  ...
2407        0          0                     0                    0
706         0          0                     0                    0
1822        0          0                     0                    0
2022        0          0                     0                    0
2154        0          0                     0                    0

      shivering  chills  joint_pain  stomach_pain  acidity  ulcers_on_tongue
\
3825          0       0           0             0        0                 0
2368          0       0           0             0        0                 0
178           0       1           1             0        0                 0
4368          0       0           1             0        0                 0
4078          0       0           1             0        0                 0
```

```
...        ...     ...        ...           ...      ...            ...
2407         0       0          0             0        0              0
706          0       0          0             0        0              0
1822         0       1          0             0        0              0
2022         0       0          0             0        0              0
2154         0       0          0             0        0              0

      ...  pus_filled_pimples  blackheads  scurring  skin_peeling  \
3825  ...                   0           0         0             0
2368  ...                   0           0         0             0
178   ...                   0           0         0             0
4368  ...                   0           0         0             0
4078  ...                   0           0         0             0
...   ...                 ...         ...       ...           ...
2407  ...                   0           0         0             0
706   ...                   0           0         0             0
1822  ...                   0           0         0             0
2022  ...                   0           0         0             0
2154  ...                   0           0         0             0

      silver_like_dusting  small_dents_in_nails  inflammatory_nails  blister
\
3825                    0                     0                   0        0
2368                    0                     0                   0        0
178                     0                     0                   0        0
4368                    0                     0                   0        0
4078                    0                     0                   0        0
...                   ...                   ...                 ...      ...
2407                    0                     0                   0        0
706                     0                     0                   0        0
1822                    0                     0                   0        0
2022                    0                     0                   0        0
2154                    0                     0                   0        0

      red_sore_around_nose  yellow_crust_ooze
3825                     0                  0
2368                     0                  0
178                      0                  0
4368                     0                  0
4078                     0                  0
...                    ...                ...
2407                     0                  0
706                      0                  0
1822                     0                  0
2022                     0                  0
2154                     0                  0

[984 rows x 132 columns]
```

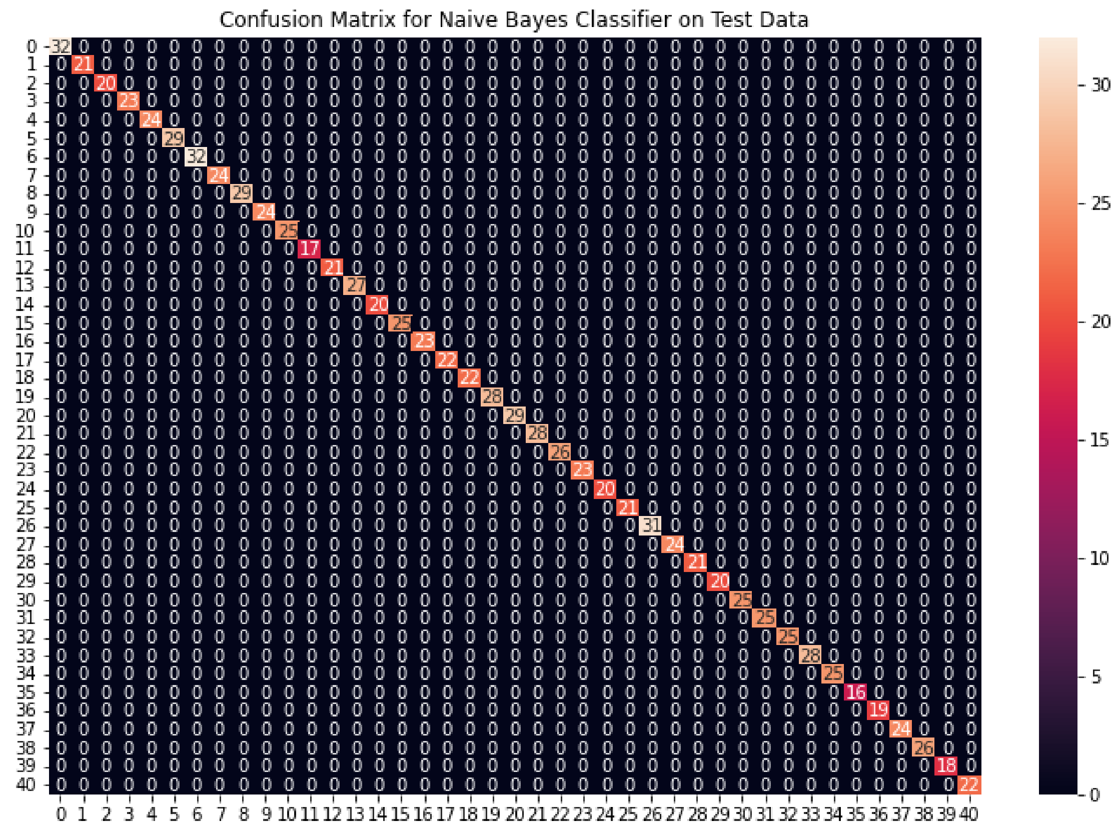
```
# Training and testing SVM Classifier
svm_model = SVC()
svm_model.fit(X_train, y_train)
preds = svm_model.predict(X_test)

print(f"Accuracy on train data by SVM Classifier\
: {accuracy_score(y_train, svm_model.predict(X_train))*100}")

print(f"Accuracy on test data by SVM Classifier\
: {accuracy_score(y_test, preds)*100}")
cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for SVM Classifier on Test Data")
plt.show()

# Training and testing Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
preds = nb_model.predict(X_test)
print(f"Accuracy on train data by Naive Bayes Classifier\
: {accuracy_score(y_train, nb_model.predict(X_train))*100}")

print(f"Accuracy on test data by Naive Bayes Classifier\
: {accuracy_score(y_test, preds)*100}")
cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Naive Bayes Classifier on Test Data")
plt.show()

# Training and testing Random Forest Classifier
rf_model = RandomForestClassifier(random_state=18)
rf_model.fit(X_train, y_train)
preds = rf_model.predict(X_test)
print(f"Accuracy on train data by Random Forest Classifier\
: {accuracy_score(y_train, rf_model.predict(X_train))*100}")

print(f"Accuracy on test data by Random Forest Classifier\
: {accuracy_score(y_test, preds)*100}")

mlp_clf = MLPClassifier(hidden_layer_sizes=(6,4),
                        max_iter = 300,activation = 'relu',
                        solver = 'adam')

mlp_clf.fit(X_train, y_train)
```

```
mlp_preds = mlp_clf.predict(X_test)
print(f"Accuracy on train data by MLP Classifier\
: {accuracy_score(y_train, mlp_clf.predict(X_train))*100}")

print(f"Accuracy on test data by MLP Classifier\
: {accuracy_score(y_test, mlp_preds)*100}")

cf_matrix = confusion_matrix(y_test, preds)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Random Forest Classifier on Test Data")
plt.show()
```

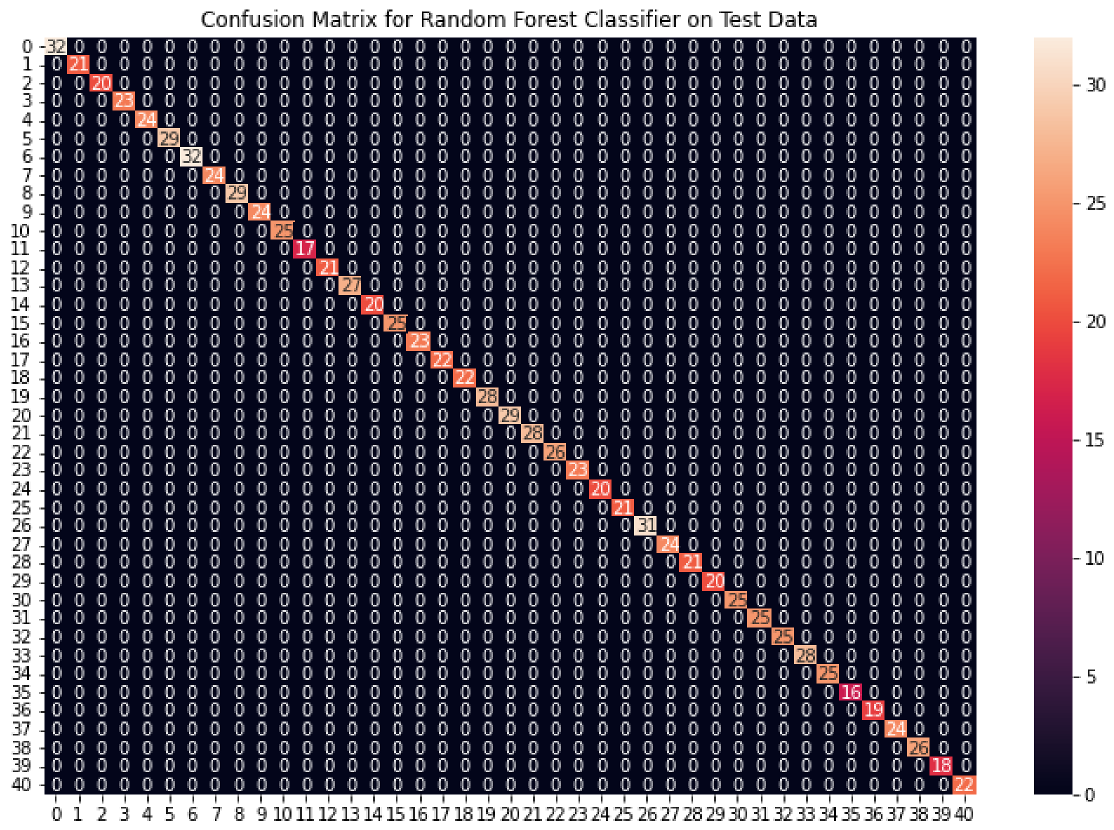
```
Accuracy on train data by SVM Classifier: 100.0
Accuracy on test data by SVM Classifier: 100.0
```



```
Accuracy on train data by Naive Bayes Classifier: 100.0
Accuracy on test data by Naive Bayes Classifier: 100.0
```

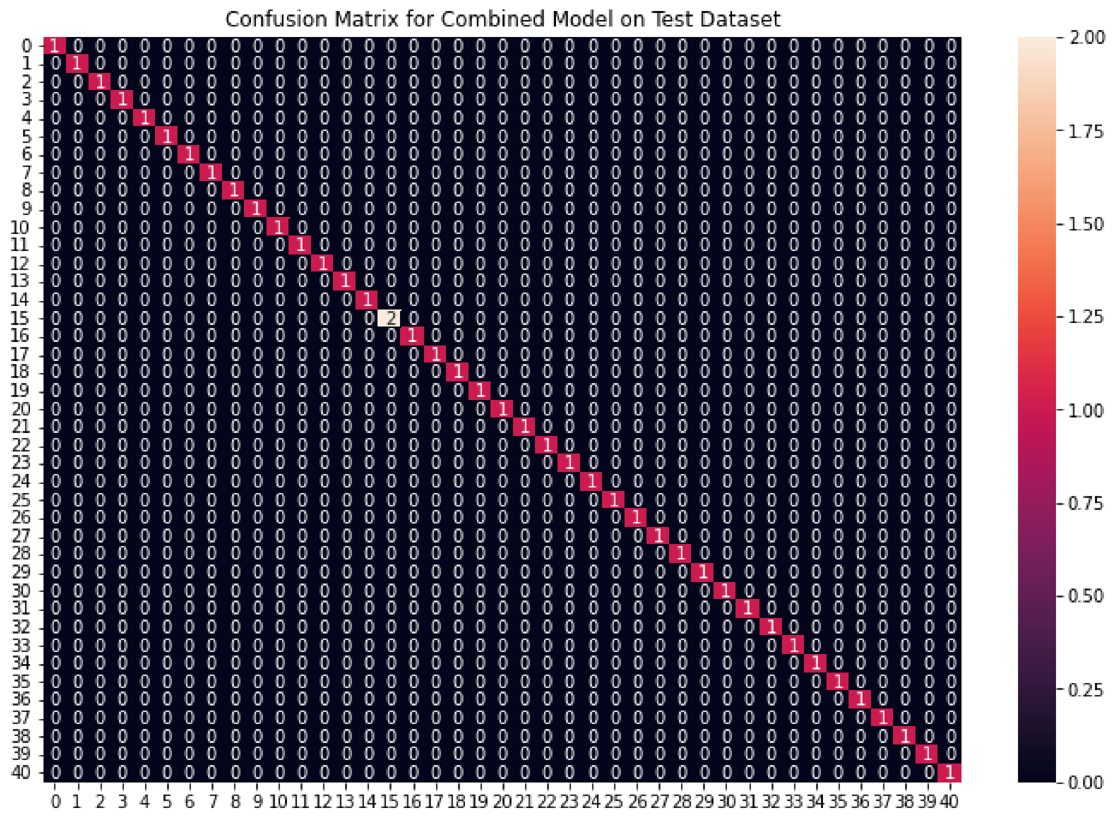


```
Accuracy on train data by Random Forest Classifier: 100.0
Accuracy on test data by Random Forest Classifier: 100.0
Accuracy on train data by MLP Classifier: 100.0
Accuracy on test data by MLP Classifier: 100.0
```



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

# Training the models on whole data
final_svm_model = SVC()
final_nb_model = GaussianNB()
final_rf_model = RandomForestClassifier(random_state=18)
final_mlp_model = MLPClassifier(hidden_layer_sizes=(6,4),
                        max_iter = 300,activation = 'relu',
                        solver = 'adam')

final_svm_model.fit(X, y)
final_nb_model.fit(X, y)
final_rf_model.fit(X, y)
final_mlp_model.fit(X, y)

# Reading the test data
test_data = pd.read_csv("Testing.csv").dropna(axis=1)

test_X = test_data.iloc[:, :-1]
test_Y = encoder.transform(test_data.iloc[:, -1])

# Making prediction by take mode of predictions
# made by all the classifiers
svm_preds = final_svm_model.predict(test_X)
nb_preds = final_nb_model.predict(test_X)
rf_preds = final_rf_model.predict(test_X)
mlp_preds = final_mlp_model.predict(test_X)

final_preds = [mode([i,j,k,l])[0][0] for i,j,
                  k,l in zip(svm_preds, nb_preds, rf_preds,mlp_preds)]

print(f"Accuracy on Test dataset by the combined model\
: {accuracy_score(test_Y, final_preds)*100}")

cf_matrix = confusion_matrix(test_Y, final_preds)
plt.figure(figsize=(12,8))

sns.heatmap(cf_matrix, annot = True)
plt.title("Confusion Matrix for Combined Model on Test Dataset")
plt.show()
```

```
Accuracy on Test dataset by the combined model: 100.0
```



```
symptoms = X.columns.values
```

```
# Creating a symptom index dictionary to encode the
# input symptoms into numerical form
symptom_index = {}
for index, value in enumerate(symptoms):
      symptom = " ".join([i.capitalize() for i in value.split("_")])
      symptom_index[symptom] = index

data_dict = {
      "symptom_index":symptom_index,
      "predictions_classes":encoder.classes_
}

# Defining the Function
# Input: string containing symptoms separated by commmas
# Output: Generated predictions by models
def predictDisease(symptoms):
      symptoms = symptoms.split(",")

      # creating input data for the models
      input_data = [0] * len(data_dict["symptom_index"])
      for symptom in symptoms:
            index = data_dict["symptom_index"][symptom]
            input_data[index] = 1

      # reshaping the input data and converting it
      # into suitable format for model predictions
      input_data = np.array(input_data).reshape(1,-1)

      # generating individual outputs
      rf_prediction = data_dict["predictions_classes"][final_rf_model.predict
(input_data)[0]]
      nb_prediction = data_dict["predictions_classes"][final_nb_model.predict
(input_data)[0]]
      svm_prediction = data_dict["predictions_classes"][final_svm_model.predi
ct(input_data)[0]]
      mlp_prediction = data_dict["predictions_classes"][final_mlp_model.predi
ct(input_data)[0]]

      # making final prediction by taking mode of all predictions
      final_prediction = mode([rf_prediction, nb_prediction, svm_prediction])
[0][0]
      predictions = {
            "rf_model_prediction": rf_prediction,
            "naive_bayes_prediction": svm_prediction,
            "svm_model_prediction": nb_prediction,
            "mlp_model_prediction": mlp_prediction,
            "final_prediction":final_prediction
      }
```

```
    return predictions

# Testing the function
print(predictDisease("Shivering,Cough"))
```

```
{'rf_model_prediction': 'Allergy', 'naive_bayes_prediction': 'Allergy', 'svm_
model_prediction': 'Allergy', 'mlp_model_prediction': 'Hypoglycemia', 'final_
prediction': 'Allergy'}

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X do
es not have valid feature names, but RandomForestClassifier was fitted with f
eature names
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X do
es not have valid feature names, but GaussianNB was fitted with feature names
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X do
es not have valid feature names, but SVC was fitted with feature names
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X do
es not have valid feature names, but MLPClassifier was fitted with feature na
mes
  "X does not have valid feature names, but"
```

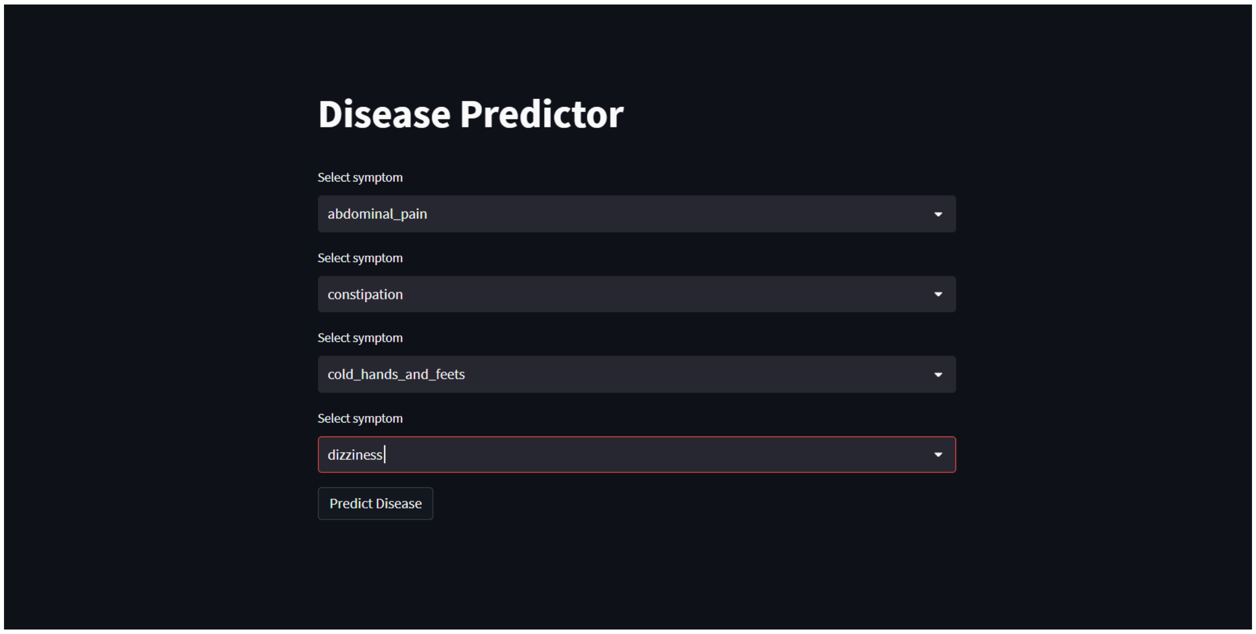
```
print(encoder.classes_)
```

```
['(vertigo) Paroymsal  Positional Vertigo' 'AIDS' 'Acne'
 'Alcoholic hepatitis' 'Allergy' 'Arthritis' 'Bronchial Asthma'
 'Cervical spondylosis' 'Chicken pox' 'Chronic cholestasis' 'Common Cold'
 'Dengue' 'Diabetes ' 'Dimorphic hemmorhoids(piles)' 'Drug Reaction'
 'Fungal infection' 'GERD' 'Gastroenteritis' 'Heart attack' 'Hepatitis B'
 'Hepatitis C' 'Hepatitis D' 'Hepatitis E' 'Hypertension '
 'Hyperthyroidism' 'Hypoglycemia' 'Hypothyroidism' 'Impetigo' 'Jaundice'
 'Malaria' 'Migraine' 'Osteoarthristis' 'Paralysis (brain hemorrhage)'
 'Peptic ulcer diseae' 'Pneumonia' 'Psoriasis' 'Tuberculosis' 'Typhoid'
 'Urinary tract infection' 'Varicose veins' 'hepatitis A']
```

```
symptoms = X_train.columns
print(symptoms)
```

```
Index(['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing',
       'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity',
       'ulcers_on_tongue',
       ...
       'pus_filled_pimples', 'blackheads', 'scurring', 'skin_peeling',
       'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails',
       'blister', 'red_sore_around_nose', 'yellow_crust_ooze'],
      dtype='object', length=132)
```

APP SCREENSHOTS:



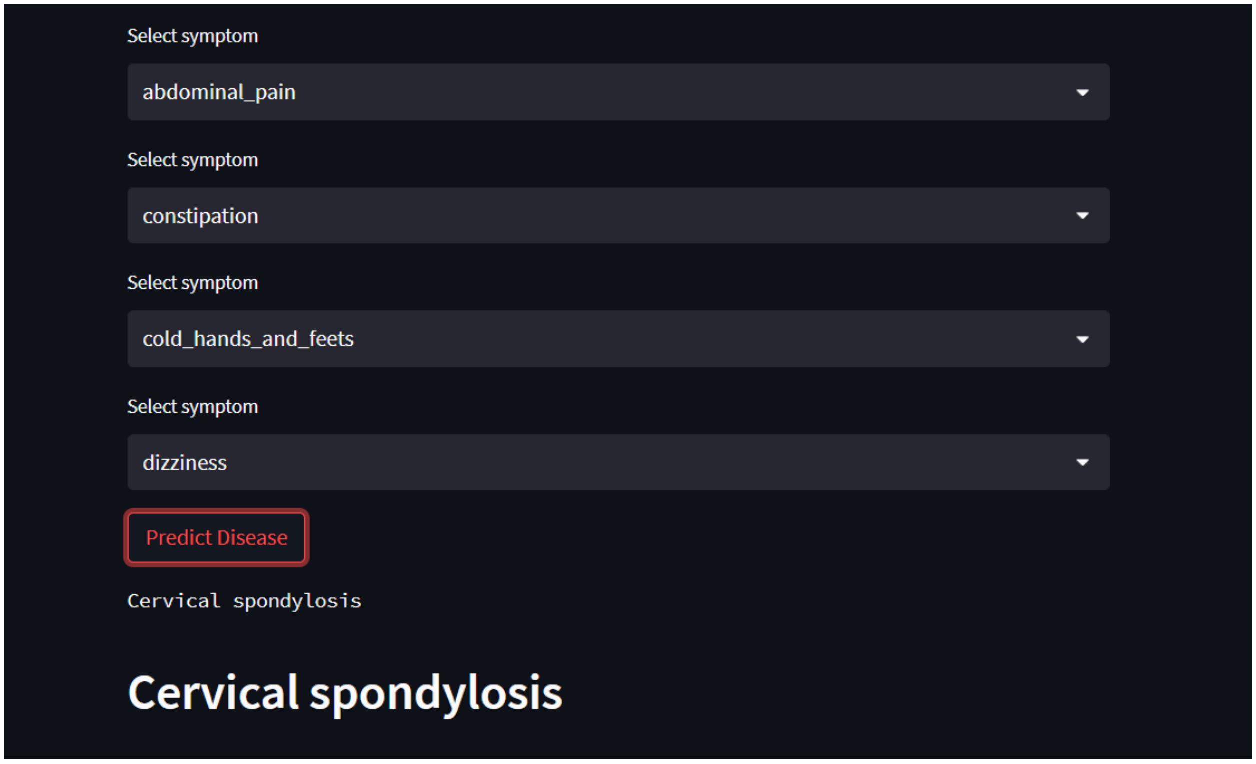
Disease Predictor

Select symptom
abdominal_pain

Select symptom
constipation

Select symptom
cold_hands_and_feets

Select symptom
dizziness



Select symptom
abdominal_pain

Select symptom
constipation

Select symptom
cold_hands_and_feets

Select symptom
dizziness

```
Cervical spondylosis
```

Cervical spondylosis

Technologies Used
FRONT-END
HTML
CSS
Streamlit
Python 3.8

1. Hardware Requirements :
    - Operating System-> Windows 8 or Later
    - Processor->Intel Pentium 4 or later
    - Memory ->2 GB minimum
    - Display -> Monitor or Screen

2. Software Requirements :
    - Editor- Vs Code or Colab
    - Web-Browser - Chrome or Microsoft Edge

FUTURE SCOPE

By developing an internet based disease detecting technique ,which is detecting the diseases on the basis of the symptoms that the user will specify , we are taking a step towards self health analyzing methods that will help people in reducing the unnecessary medical activities. By analyzing current major diseases, the problems patients are facing while dealing with those diseases and the increasing crowds in hospitals we are trying to develop a new web based method that will help people in analyzing their health on their own.
As in daily life we see that people have to reach hospitals even if they have just some minor problem that can be resolved at home, because of not having enough knowledge about the disease or being unable to predict the disease. We are providing a practical solution to these problems by developing a system that will help people in analyzing their own health and predicting the disease by taking symptoms under consideration. This system will majorly help those elder patients who have to visit the hospital daily for minor checkups. To fit this system in real clinical practices we will continuously take reviews and feedback from users while testing the system and will try to reach the maximum perfection.
In addition to the current features that the system is providing currently we will try to add new features and will try to make further technical improvements in future to make the system more smarter and efficient :

1. We will try to increase the precision in detecting the diseases by collecting more and more information and will also try to use ML techniques in future to make the system more efficient. We are providing a review and feedback system for the users so that the users can suggest about the needed improvements and developments. We will try to make the system user friendly based on the reviews and feedback we get. In future, with better and efficient ML algorithms , it is possible to achieve better processing of systems and better prediction of diseases.
2. In addition to current diseases we will try to expand the system by adding new disease predicting methods that will help in predicting most of the diseases. We will work towards developing the UI so that users can feel it more friendly and can feel free to input the symptoms on the platform without any hesitation.

3. Currently we have just developed the system to predict the diseases but in future we will also try to add some features so that along with predicting the diseases the system will also provide some primary precautionary measures to prevent or reduce the disease from becoming severe. For this we will try to add some experts from the medical field to our project and according to their suggestions we will further modify our system.

After all such modifications the system will become very useful in health checkups. In Today's era almost everyone has some type of health issue and some of them like diabetes, sugar, hypertension, and blood pressure are very common. For such a disease a patient regularly needs to visit a doctor just to make a checkup everyday. Our system will be very useful for such people. And in future it will be very useful in reducing the crowds from hospitals. Some diseases like Fever, cough can be cured at home if predicted before becoming a severe problem and if proper precautions are taken but people still have to visit hospitals for such common diseases. In future, our system will provide an option to such people of firstly taking a primary checkup and then to decide whether it is really needed to visit a doctor or not.
So our system has much more Future Scope in reducing the crowds in hospitals and indirectly reducing the infection of diseases that are spreading due to crowds in hospitals. It will also be very useful in the view of elder peoples and in many such areas.

REFERENCES

[1] Mathes BM, Norr AM, Allan NP, Albanese BJ, Schmidt NB. Cyberchondria: Overlap with health anxiety and unique relations with impairment, quality of life, and service utilization. Psychiatry Res. 2018 Mar;261:204-211. doi: 10.1016/j.psychres.2018.01.002. Epub 2018 Jan 3. PMID: 29324396.

[2] Intelligent Health Diagnosis Technique Exploiting Automatic Ontology Generation and Web-Based Personal Health Record Services.
GUN-WOO KIM AND DONG-HO LEE Department of Computer Science and Engineering, Haiyang University, Seoul 04763, South Korea

[3] Disease Prediction Based on Symptoms Using Machine Learning
Y. Deepthi, K. Pavan Kalyan, Mukul Vyas, K. Radhika, D. Kishore Babu & N. V. Krishna Rao