A Major Project Report On

"Disease Prediction System"

Submitted in Partial Fulfilment of

the Requirements for the Award

Of

Degree of B.Tech.

To

Guru Gobind Singh Indraprastha University, Delhi

Under the guidance of Ms. Upasna Joshi

Submitted By:

Gaurav Birdi (01918002718)

Sachin Sharma (04418002718)

Md Omer (03018002718)

Prakhar Katiyar (40118002718)

Affiliated to GGSIP University, New Delhi
Approved by AICTE & Council of Architecture

COMPUTER SCIENCE DEPARTMENT, DELHI TECHNICAL CAMPUS,

KNOWLEDGE PARK-III, GREATER NOIDA

CANDIDATE'S DECLARATION

We therefore announce that the work presented in this report entitled "Disease Prediction System", to meet the need for the Bachelor of Technology degree in Computer Science and Engineering, submitted to Computer Science and Engineering Department, DTC in collaboration with Guru Gobind. Singh Indraprastha University, New Delhi, is a true record of our work undertaken under the direction of Ms. Upasna Joshi.

The work reported on this was not submitted by us for any other degree or diploma.

Date: 03/06/2022

Gaurav Birdi (01918002718)

Md. Omer (03018002718)

Sachin Sharma (04418002718)

Prakhar Katiyar (40118002718)

CERTIFICATE

This is to ensure that the Project work entitled “Disease Prevention Program” submitted by Gaurav Birdi, Md Omer, Prakhar Katiyar and Sachin Sharma in fulfillment of the Bachelor of Technology Degree in Computer Science and Engineering award requirements at DTC, Greater Noida. it is a real work being done under my direction and direction. To the best of our knowledge, the content contained in this work has not been submitted to any other University / Institution for any Bachelor's Degree.

Date: 03/06/2022

Ms. Upasna Joshi
(Asst. Prof., CSE)

ACKNOWLEDGEMENT

We express our sincere gratitude to Dr. Seema Verma (HOD, CSE) and Ms. Upasna Joshi (A.P, CSE), Delhi Technical Campus, for his valuable guidance and timely suggestions throughout my research career, otherwise this work. it would not have happened. We would also like to extend my deepest greetings to all the other members of the Department of Computer Science, who have given their great effort and guidance at the right times otherwise it would have been very difficult for me to complete this task. Lastly, we would also like to thank my friends for their advice and point out my mistakes.

CONTENTS

S.No.	Topic	Page No.
1.	Abstract	6
2.	Introduction	7
3.	Background	9
4.	Hardware Software Requirements	12
5.	Detailed Design	13
6.	Implementation	15
7.	Testing/Performance Matrices	41
8.	Conclusion And Future Work	48
9.	References	49
10.	Declaration by Student	51
11.	Verification by Faculty Project Guide	52

ABSTRACT

Disease Prediction helps patients to identify the risk of disease or health problems/disorder. In certain circumstances, people feel unsafe to visit hospitals or individual doctors. Further, sometime people just want to know the type of disease, they have been suffering with. Therefore, a need for a generic disease prediction system arises using which people can predict the disease on the basis of symptoms that they are facing. This would also create an awareness of diseases and available medical advice thus avoiding further health complications.

The proposed project aims at developing a web-based system that can predict diseases based on the symptoms reported by the user

In today's world almost everyone has an access to smartphones, it is going to play a crucial role in improving future of public health care system. This will further help patients in getting fast and appropriate medical opinion, improve patient care, decrease in resource consumption and further reduce health care costs.

Introduction

This Chapter contains an overview of the work done for our project, project objective and a brief intro to the two sections of our project i.e., General Disease Prediction, Covid Probability Prediction.

1.1 GENERAL DISEASE PREDICTION AND COVID PROBABILITY PREDICTION

In the present scenario, human beings are facing several diseases due to existing environmental conditions and lifestyle routines. Identification and prediction of diseases at an earlier stage are extremely important before it triggers a major health hazard. Disease and Covid Prediction using Machine learning is a Web based App which predicts the disease based on the symptoms provided by the user. If the patient is not much serious and just wants to know the type of disease, he/she may be suffering with, this Web based App will provide the prediction of the disease by just asking the symptoms. This system will help to predict disease on the basis of symptoms selected by the user. Early detection of disease may lead to more cures or longer survival. In today's world almost everyone has access to smartphones and hence can access this type of system, it is going to play a crucial role in improving future of public health care system. This will further help patients in getting fast and appropriate medical opinion, improve patient care, decrease in resource consumption and further reduce health care costs.

1.2 OVERVIEW OF PROJECT

With the fast advancement of technology and data, the healthcare sector is one of the most significant study topics in the contemporary era. There are several ways for treating various ailments all throughout the world. Machine Learning is a new method that aids in disease prediction.

The purpose of making this project is to predict the accurate disease of the patient using all their general information's and also the symptoms. On the presented dataset, machine learning methods such as Naive Bayes, Decision Tree, Random Forest, and KNN are used to predict the illness. In this project We have used two datasets one for Covid probability prediction which We have taken from Israel Govt Website and the other dataset for the General Disease Prediction taken from Kaggle. We have developed a website where the user can enter the symptoms of the disease and predict the disease based on the different machine learning algorithms.

1.3 PROJECT OBJECTIVE

The Objective of the project is to create a web app, where the user can perform two types of disease prediction i.e., General Disease Prediction and Covid Probability Prediction.

For the Disease Prediction we aim to work on two datasets, first dataset for the covid probability prediction part which has been taken from Israel Govt. Site and the second dataset for the General Disease Dataset which has been taken from Kaggle. Finally, we will Compare the accuracy of different machine learning models and take the best for the prediction.

Background

2.1 LITERATURE SURVEY (WITH PROPER REFERENCES CITED)

Harish Rajora, N. Punn, S. K. Sonbhadra, Sonali Agarwal. [1] In their paper implemented a web app. In their work, the prediction is done using four different machine learning algorithms (Random Forest, Naïve Bayes, KNN and Ensemble). The source of data is not mentioned. The accuracy achieved in this paper is between 84 and 93.5 for different algorithms.

Mohan Kumar K N, S.Sampath, Mohammed Imran [2] In their paper discuss the need for an affordable disease prediction system which can identify diseases in the early stage. It also lists the techniques currently being used to predict disease. Logistic regression, Support Vector Machine (SVM), Decision tree and Clustering techniques dominate with 27.5%, 25%, 22.5% and 20%. Logistic regression dominates the list because of the nature of medical data which is binary in most of the cases.

Md Ekramul Hossain, Arif Khan, Mohammad Ali Moni, Shahadat Uddin [3] In their paper discuss the different methods already used for specific disease prediction and their accuracies. The term ‘Electronic Health Data’ refers to the digitised health data which contains information of diseases, diagnostics and treatments of patients. All the papers with the keywords ‘Electronic Health data’ and ‘Disease prediction’ were compared and the listed in this comprehensive literature review.

Ashish Kumar, Priya Ghansela , Purnima Soni , Chirag Goswami , Parasmani Sharma [4] contains the prediction of diseases taken from several sources like hospitals, discharge slips of patients and from UCI repository. Then it applies supervised machine learning algorithms such as Decision tree, Random Forest, SVM (Support Vector Machine) and Naive-Bayes to train the model. The accuracies range from 54 in SVM to 95 in Random Forest. The exact source of the data is not given in this paper.

Min Chen, Yixue Hao , Kai Hwang , Lin Wang , Lu Wang [5] suggest the use of a new convolutional neural network based multimodal disease risk prediction (CNN-MDRP).

The data is collected from real-life hospital data from central China in 2013-2015. The prediction accuracy of the proposed algorithm reaches 94.8%.

Ch Aishwarya, K Suvarchala, B Aravind, G Shashank [6] basically focused on predicting a specific disease before its onset. Given that prevention is better than cure, it is very important to diagnose a specific disease and follow the necessary guidelines before it can increase. Therefore, we have come up with the idea of predicting diseases in advance. There is already a system for predicting the disease, but it only focuses on data sets available from local health care communities (structured data). The proposed program focuses on big data that is widely used today. It uses a multimedia CNN algorithm with a combination of random data and systematic data. This algorithmic model basically consists of three layers, namely the input layer, the hidden (duplicate layer) and the output layer.

Marouane Fethi Ferjani [7] test the proposed hypothesis that supervised ML algorithms can improve health care by the accurate and early detection of diseases. In this study, we investigate studies that utilize more than one supervised ML model for each disease recognition problem. This approach renders more comprehensiveness and precision because the evaluation of the performance of a single algorithm over various study settings induces bias which generates imprecise results. The analysis of ML models will be conducted on few diseases located at heart, kidney, breast, and brain. For the detection of the disease, numerous methodologies will be evaluated such as KNN, NB, DT, CNN, SVM, and LR.

Dhiraj Dahiwade, Prof Gajanan Patle, Prof Ektaa Meshraam [8] made a prediction of three diseases such as diabetes, mental illness and heart disease. Disease prognosis is done with systematic data. Predictors of heart disease, diabetes and brain disorders are made using a different machine learning algorithm such as naïve bayes, decision tree and KNN algorithm. The result of the Decision Tree algorithm is better than the Naïve bayes and the KNN algorithm. Also, they predict whether the patient is at high risk for cerebral infarction or low risk of cerebral infarction. To predict the risk of brain damage, they used CNN multimodel disease risk prediction in text data. Accurate comparisons occur between CNN's unimodel risk predictors based on CNN based multimodel disease risk algorithm. Disease prediction accuracy reaches 94.8% faster than CNN-based unimodal

risk-based algorithm.

Rinkal Keniya, Aman Khakharia, Vruddhi Shah, Vrushabh Gada, Ruchi Manjalkar, Tirth Thaker, Mahesh Warang, Ninad Mehendale [9] Proposed disease reporting system. A doctor may not always be available when needed. However, in the current context, one can necessarily use this predictive system at any time. Individual characteristics and age and gender can be assigned to the ML model for further processing. After initial data processing, the ML model uses the current input, trains and evaluates the algorithm that leads to the predicted disease.

Dong Jin Park, Min Woo Park, Homin Lee, Young-Jin Kim, Yeongsic Kim, Young Hoon Park [10] aims to build a new integration model by integrating the DNN (deep neural model) network) and two ML models for diagnosis. using laboratory test results. 86 attributes (laboratory tests) were selected from data sets based on statistical calculation, features related to clinical significance, and missing values. The prepared combination model obtained 81% F1 points and 92% predictive accuracy in the five most common diseases. In-depth studies and ML models have shown differences in predictable strength and disease classification patterns.

Hardware and Software requirements

This Chapter contains the Hardware Requirements and Software Requirements that are needed for the implementation of the project.

3.1 HARDWARE REQUIREMENTS

- PROCESSOR : Intel dual Core ,i3
- RAM : 8 GB
- HARD DISK : 80 GB
- INPUT DEVICE: Standard Keyboard, Standard Mouse
- OUTPUT DEVICE: Laptop/PC

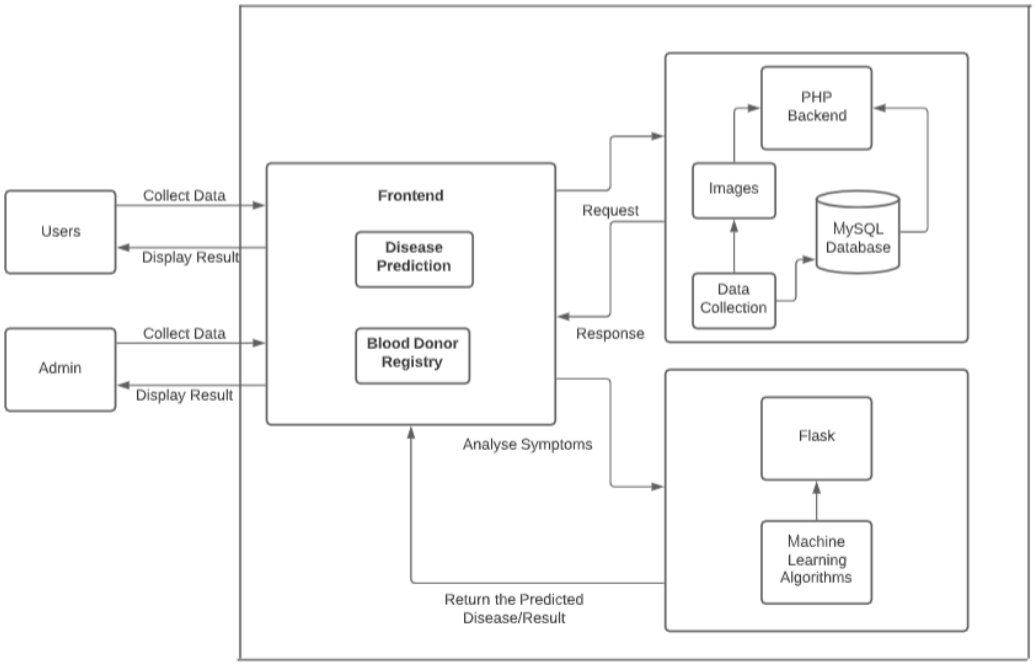
3.2 SOFTWARE REQUIREMENTS

- OPERATING SYSTEM : Windows 7/ XP/8
- FRONT END : Html,css,java script.
- LANGUAGES: Python
- OTHER SOFTWARE/TOOLS: Visual studio Code, Google Collab, Jupyter Notebook

Detailed Design

This Chapter contains the system architecture diagram along with other UML Diagrams pertinent to our project and features of our project.

4.1 SYSTEM ARCHITECTURE



4.2 use case diagram

4.3 MODULE DESCRIPTION

4.3.1 User Module:

In this module the user can predict the disease.

4.3.2 Covid-Prediction Module:

In this module the covid probability prediction takes place using Machine Learning

Algorithms.

4.3.3 General-Disease Prediction Module:

In this module prediction based on the general disease dataset takes place.

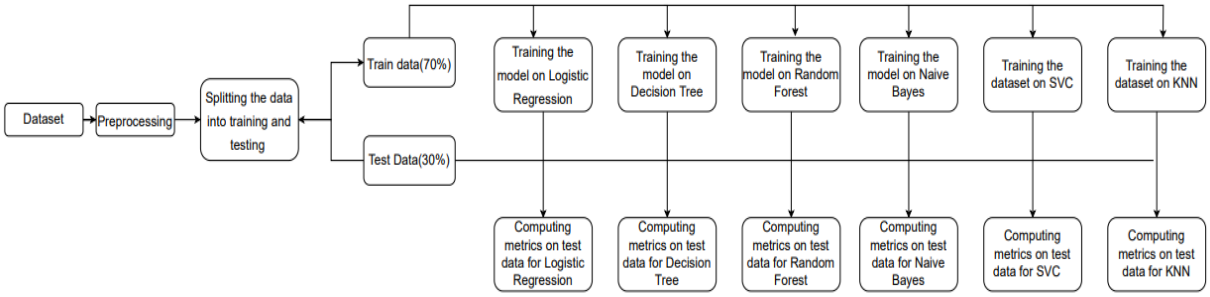
4.3.4 Model selection module:

In this module developer select different Machine Learning models and compare them and take the best fit model (highest accuracy level) for making the system.

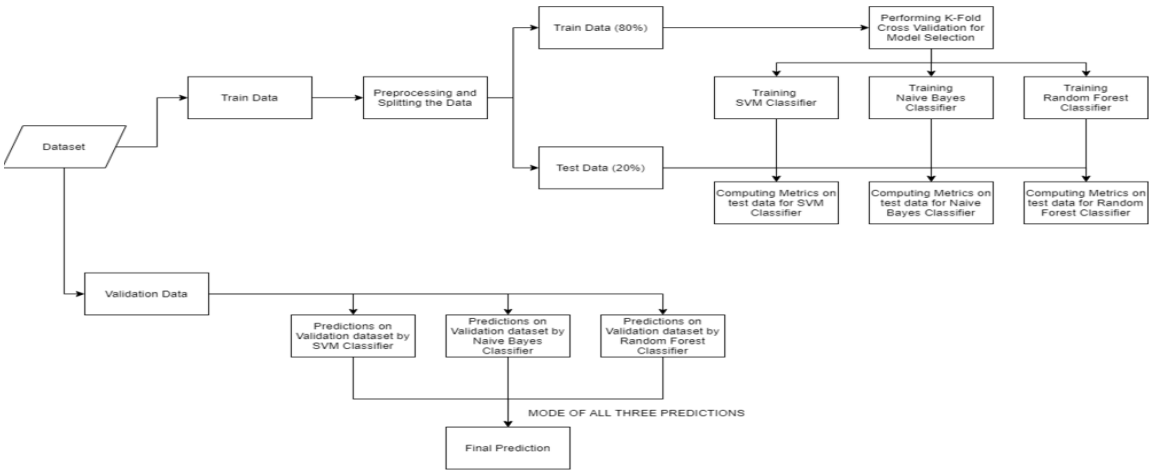
4.3.5 Flask Module:

This module is used to convert the machine learning algorithm-based prediction into a web app.

4.4 CORONAVIRUS PREDICTION FLOWCHART



4.5 GENERAL DISEASE PREDICTION FLOWCHART



Implementation

This Chapter includes the steps required for the implementation and the relevant snapshots.

5.1 DATA PREPROCESSING FOR GENERAL DISEASE DATASET

First Step is to import the required Libraries and import the dataset that dataset that we will be working on.

Importing the Libraries

```
In [ ]: import pandas as pd
        import numpy as np
```

Importing the Dataset

```
In [ ]: data = pd.read_csv('dataset.csv', header= None)
```

```
In [ ]: data.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	
0	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Symptom_9	Symptom_10	Symptor
1	Fungal infection	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	
2	Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	Fungal infection	itching	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	Fungal infection	itching	skin_rash	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

After importing the dataset, I start the data cleaning process.

Cleaning the Dataset

```
In [ ]: # Removing the first row
        data.drop(index = 0, axis = 0, inplace = True)
```

```
In [ ]: data.head()
```

Out[34]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	Fungal infection	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Fungal infection	itching	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Fungal infection	itching	skin_rash	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	Fungal infection	itching	skin_rash	nodal_skin_eruptions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
In [ ]: data.shape
```

Out[6]: (4920, 18)

```
In [ ]: # Creating the symptom dataset in which only symptoms are present
        symptom_df = data.iloc[:, 1:]
```

```
In [ ]: symptom_df.head()
```

In [ ]:
```
symptom_df.head()
```

Out[8]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	itching	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	itching	skin_rash	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	itching	skin_rash	nodal_skin_eruptions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
# Resetting the index
symptom_df.reset_index(inplace = True, drop = True)
```

In [ ]:
```
symptom_df.head()
```

Out[10]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	itching	nodal_skin_eruptions	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	itching	skin_rash	dischromic _patches	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	itching	skin_rash	nodal_skin_eruptions	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
symptom_df.shape
```

Out[11]:
```
(4920, 17)
```

Now We have created a symptom array containing the list of unique symptoms available in the used dataset.

In [ ]:
```
# Creating the list of all symptoms for each disease
symptom = []
for i in range(0,4920):
    symptom.append([str(symptom_df.values[i,j]) for j in range(0,17)])
```

In [ ]:
```
symptom[0]
```

Out[13]:
```
['itching',
 ' skin_rash',
 ' nodal_skin_eruptions',
 ' dischromic _patches',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan',
 'nan']
```

In [ ]:
```
# Getting the unique symptom names
column = pd.unique(symptom_df.iloc[:,:].values.ravel('K'))
```

In [ ]:
```
column
```

Out[15]:
```
array(['itching', ' skin_rash', ' continuous_sneezing', ' shivering',
       ' stomach_pain', ' acidity', ' vomiting', ' indigestion',
```

```
In [ ]: column
```

```
Out[15]: array(['itching', ' skin_rash', ' continuous_sneezing', ' shivering',
       ' stomach_pain', ' acidity', ' vomiting', ' indigestion',
       ' muscle_wasting', ' patches_in_throat', ' fatigue',
       ' weight_loss', ' sunken_eyes', ' cough', ' headache',
       ' chest_pain', ' back_pain', ' weakness_in_limbs', ' chills',
       ' joint_pain', ' yellowish_skin', ' constipation',
       ' pain_during_bowel_movements', ' breathlessness', ' cramps',
       ' weight_gain', ' mood_swings', ' neck_pain', ' muscle_weakness',
       ' stiff_neck', ' pus_filled_pimples', ' burning_micturition',
       ' bladder_discomfort', ' high_fever', ' nodal_skin_eruptions',
       ' ulcers_on_tongue', ' loss_of_appetite', ' restlessness',
       ' dehydration', ' dizziness', ' weakness_of_one_body_side',
       ' lethargy', ' nausea', ' abdominal_pain', ' pain_in_anal_region',
       ' sweating', ' bruising', ' cold_hands_and_feets', ' anxiety',
       ' knee_pain', ' swelling_joints', ' blackheads',
       ' foul_smell_of urine', ' skin_peeling', ' blister',
       ' dischromic _patches', ' watering_from_eyes',
       ' extra_marital_contacts', ' diarrhoea', ' loss_of_balance',
       ' blurred_and_distorted_vision', ' altered_sensorium',
       ' dark_urine', ' swelling_of_stomach', ' bloody_stool', ' obesity',
       ' hip_joint_pain', ' movement_stiffness', ' spinning_movements',
       ' scurring', ' continuous_feel_of_urine', ' silver_like_dusting',
       ' red_sore_around_nose', nan, ' spotting_ urination',
       ' passage_of_gases', ' irregular_sugar_level', ' family_history',
       ' lack_of_concentration', ' excessive_hunger',
       ' yellowing_of_eyes', ' distention_of_abdomen',
       ' irritation_in_anus', ' swollen_legs', ' painful_walking',
       ' small_dents_in_nails', ' yellow_crust_ooze', ' internal_itching',
       ' mucoid_sputum', ' history_of_alcohol_consumption',
       ' swollen_blood_vessels', ' unsteadiness', ' inflammatory_nails',
       ' depression', ' fluid_overload', ' swelled_lymph_nodes',
       ' malaise', ' prominent_veins_on_calf', ' puffy_face_and_eyes',
       ' fast_heart_rate', ' irritability', ' muscle_pain', ' mild_fever',
       ' yellow_urine', ' phlegm', ' enlarged_thyroid',
       ' increased_appetite', ' visual_disturbances', ' brittle_nails',
       ' drying_and_tingling_lips', ' polyuria', ' pain_behind_the_eyes',
       ' toxic_look_(typhos)', ' throat_irritation',
       ' swollen_extremeties', ' slurred_speech', ' red_spots_over_body',
       ' belly_pain', ' receiving_blood_transfusion',
       ' acute_liver_failure', ' redness_of_eyes', ' rusty_sputum',
       ' abnormal_menstruation', ' receiving_unsterile_injections',
```

Here we can see that there are 132 symptoms available in the dataset.

Now we start working towards creating the final dataset on which we will be using the Machine Learning algorithms for prediction.

```
In [ ]: column.shape
```

```
Out[16]: (132,)
```

```
In [ ]: # Creating a new dataframe having symptom names as columns
new_df = pd.DataFrame(columns = column)
new_df.head()
```

```
Out[17]:
```

itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	patches_in_throat	...	abnormal_menstruatio

0 rows × 132 columns

```
In [ ]: # Creating the encoded list of arrays for each symptom
new_list = np.zeros((4920,132), dtype = int)

for row in range(0,4920):
    for col in range(0,132):
        for sym in range(0,17):
            if column[col] == symptom[row][sym]:
                new_list[row][col] = 1
```

```
In [ ]: new_list.shape
```

```
Out[19]: (4920, 132)
```

```
In [ ]: new_list
```

```
Out[20]: array([[1, 1, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]])
```

Now We have converted the dataset into binary form.

We have also created a disease array which we would concatenate with the converted dataset.

```
In [ ]: # Encoding this array to the new_df dataframe of symptoms
        for i in range(len(new_list)):
            new_df.loc[i] = new_list[i]

In [ ]: new_df.head()
```

Out[22]:

	itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	patches_in_throat	...	abnormal_menstruati
0	1	1	0	0	0	0	0	0	0	0	...	
1	0	1	0	0	0	0	0	0	0	0	...	
2	1	0	0	0	0	0	0	0	0	0	...	
3	1	1	0	0	0	0	0	0	0	0	...	
4	1	1	0	0	0	0	0	0	0	0	...	

5 rows × 132 columns

```
In [ ]: # We need to add disease name to it
        disease_df = pd.read_csv('dataset.csv')

In [ ]: disease = disease_df['Disease']

In [ ]: disease.head()
```

Out[25]:

```
0    Fungal infection
1    Fungal infection
2    Fungal infection
3    Fungal infection
4    Fungal infection
Name: Disease, dtype: object
```

```
In [ ]: disease.shape
```

Out[26]: (4920,)

```
In [ ]: new_df.shape
```

Finally, we concatenate the disease array with the converted dataset.

Out[26]: (4920,)

```
In [ ]: new_df.shape
```

Out[27]: (4920, 132)

```
In [ ]: # Concatenating the Disease column with the previous dataframe
        disease_concat_df = pd.concat([disease,new_df], axis=1)
        disease_concat_df.head()
```

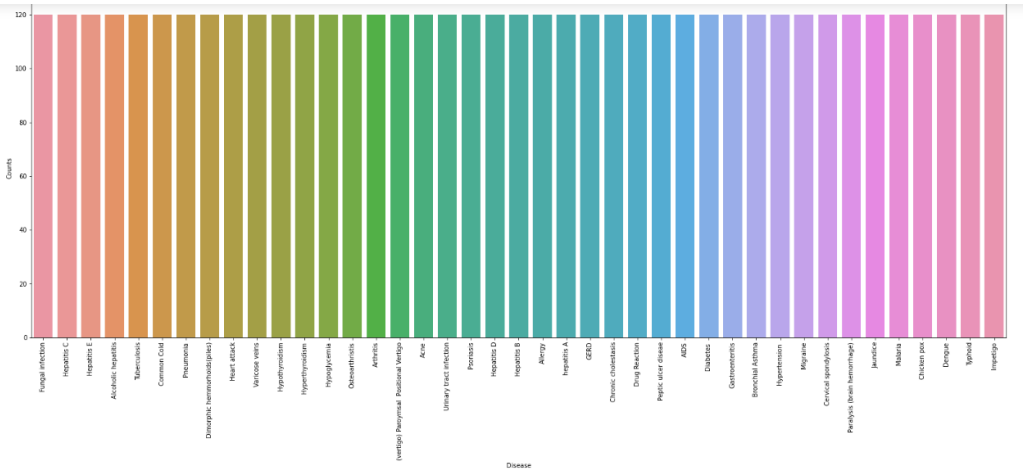
Out[28]:

	Disease	itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	...	abnormal_menstruation	recei
0	Fungal infection	1	1	0	0	0	0	0	0	0	...	0	
1	Fungal infection	0	1	0	0	0	0	0	0	0	...	0	
2	Fungal infection	1	0	0	0	0	0	0	0	0	...	0	
3	Fungal infection	1	1	0	0	0	0	0	0	0	...	0	
4	Fungal infection	1	1	0	0	0	0	0	0	0	...	0	

5 rows × 133 columns

```
In [ ]: import matplotlib.pyplot as plt
        import seaborn as sns

In [ ]: # Reading the train.csv by removing the
        # last column since it's an empty column
        data = disease_concat_df

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

        plt.figure(figsize = (28,10))
        sns.barplot(x = "Disease", y = "Counts", data = temp_df)
        plt.xticks(rotation=90)
```

Final Dataset after Cleaning.



In [ ]:
```
# Saving this new dataframe
disease_concat_df.to_csv("final_healthdata.csv", index=False)
```

In [ ]:
```
disease_concat_df
```

Out[30]:

	Disease	itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	...	abnormal_menstruation
0	Fungal infection	1	1	0	0	0	0	0	0	0	...	0

In [ ]:
```
# Saving this new dataframe
disease_concat_df.to_csv("final_healthdata.csv", index=False)
```

In [ ]:
```
disease_concat_df
```

Out[30]:

	Disease	itching	skin_rash	continuous_sneezing	shivering	stomach_pain	acidity	vomiting	indigestion	muscle_wasting	...	abnormal_menstruation
0	Fungal infection	1	1	0	0	0	0	0	0	0	...	0
1	Fungal infection	0	1	0	0	0	0	0	0	0	...	0
2	Fungal infection	1	0	0	0	0	0	0	0	0	...	0
3	Fungal infection	1	1	0	0	0	0	0	0	0	...	0
4	Fungal infection	1	1	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4915	(vertigo) Paroymsal Positional Vertigo	0	0	0	0	0	0	1	0	0	...	0
4916	Acne	0	1	0	0	0	0	0	0	0	...	0
4917	Urinary tract infection	0	0	0	0	0	0	0	0	0	...	0
4918	Psoriasis	0	1	0	0	0	0	0	0	0	...	0
4919	Impetigo	0	1	0	0	0	0	0	0	0	...	0

5.2 MODEL TRAINING ON GENERAL DISEASE DATASET

First Step is to import the dataset along with the required Libraries.

In [ ]:
```
import pandas as pd
```

Reading the Data,Train Test Split

In [ ]:
```
test = pd.read_csv('/content/sample_data/testing.csv')
train = pd.read_csv('/content/sample_data/training.csv')
```

In [ ]:
```
train.dropna(inplace=True,axis=1)
```

In [ ]:
```
train.head()
```

Out[5]:

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

5 rows × 133 columns

In [ ]:
```
X_train = train.drop(['prognosis'], axis = 1)
y_train = train['prognosis']
X_test = test.drop(['prognosis'], axis = 1)
```

First Algorithm Used for Prediction is Random Forest.

In [ ]:
```
X_train = train.drop(['prognosis'], axis = 1)
y_train = train['prognosis']
X_test = test.drop(['prognosis'], axis = 1)
y_test = test['prognosis']
```

RandomForest Classifier

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
import numpy as np
```

In [ ]:
```
error= []
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
```

Out[8]:
```
RandomForestClassifier()
```

In [ ]:
```
pred_i = rfc.predict(X_test)
error.append(np.mean(pred_i != y_test))
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

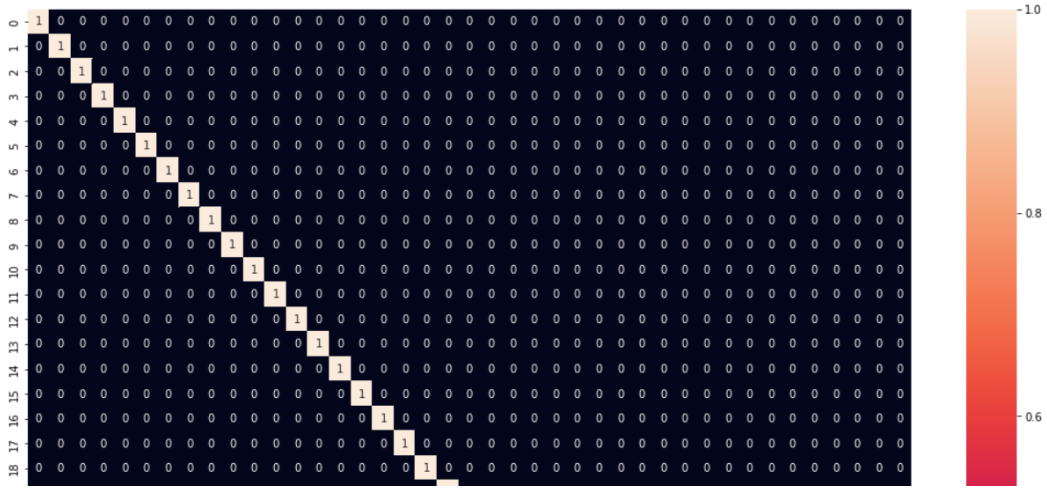
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
                                        precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                AIDS           1.00      1.00      1.00         1
                                Acne           1.00      1.00      1.00         1
                Alcoholic hepatitis            1.00      1.00      1.00         1
                             Allergy           0.50      1.00      0.67         1
                           Arthritis           1.00      1.00      1.00         1
                  Bronchial Asthma             1.00      1.00      1.00         1
              Cervical spondylosis             1.00      1.00      1.00         1
                        Chicken pox            1.00      1.00      1.00         1
              Chronic cholestasis              0.50      1.00      0.67         1
```

Random Forest Algorithm Accuracy: 95%

```
In [ ]: CM = confusion_matrix(y_test,pred_i)

        import seaborn as sns
        import matplotlib.pyplot as plt

        # labels = ['0', '1']

        fig, ax = plt.subplots(figsize=(18, 17))
        sns.heatmap(CM, annot=True, fmt=".0f")
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0bb5c982d0>
```



Second Algorithm Used for Prediction is Logistic Regression.

Logistic Regression

```
In [ ]: from sklearn.linear_model import LogisticRegression
        import numpy as np
```

```
In [ ]: error= []
        # Will take some time
        lr = LogisticRegression()
        lr.fit(X_train,y_train)
```

```
Out[14]: LogisticRegression()
```

```
In [ ]: pred_i = lr.predict(X_test)
        error.append(np.mean(pred_i != y_test))
```

```
In [ ]: from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

```
In [ ]: print(classification_report(y_test,pred_i))
```

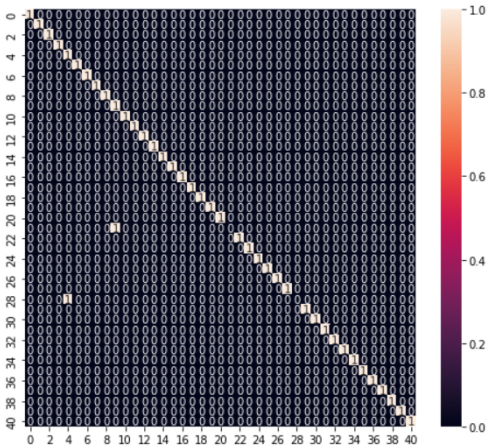
```
                                        precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                    AIDS       1.00      1.00      1.00         1
                                    Acne       1.00      1.00      1.00         1
                    Alcoholic hepatitis       1.00      1.00      1.00         1
                                 Allergy       0.50      1.00      0.67         1
                               Arthritis       1.00      1.00      1.00         1
                       Bronchial Asthma       1.00      1.00      1.00         1
                   Cervical spondylosis       1.00      1.00      1.00         1
                            Chicken pox       1.00      1.00      1.00         1
                   Chronic cholestasis       0.50      1.00      0.67         1
                            Common Cold       1.00      1.00      1.00         1
                                 Dengue       1.00      1.00      1.00         1
                               Diabetes       1.00      1.00      1.00         1
            Dimorphic hemmorhoids(piles)       1.00      1.00      1.00         1
                                Drug Reaction      1.00      1.00      1.00         1
```

Bronchial Asthma	1.00	1.00	1.00	1
Cervical spondylosis	1.00	1.00	1.00	1
Chicken pox	1.00	1.00	1.00	1
Chronic cholestasis	0.50	1.00	0.67	1
Common Cold	1.00	1.00	1.00	1
Dengue	1.00	1.00	1.00	1
Diabetes	1.00	1.00	1.00	1
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	1
Drug Reaction	1.00	1.00	1.00	1
Fungal infection	1.00	1.00	1.00	1
GERD	1.00	1.00	1.00	1
Gastroenteritis	1.00	1.00	1.00	1
Heart attack	1.00	1.00	1.00	1
Hepatitis B	1.00	1.00	1.00	1
Hepatitis C	1.00	1.00	1.00	1
Hepatitis D	0.00	0.00	0.00	1
Hepatitis E	1.00	1.00	1.00	1
Hypertension	1.00	1.00	1.00	1
Hyperthyroidism	1.00	1.00	1.00	1
Hypoglycemia	1.00	1.00	1.00	1
Hypothyroidism	1.00	1.00	1.00	1
Impetigo	1.00	1.00	1.00	1
Jaundice	0.00	0.00	0.00	1
Malaria	1.00	1.00	1.00	1
Migraine	1.00	1.00	1.00	1
Osteoarthristis	1.00	1.00	1.00	1
Paralysis (brain hemorrhage)	1.00	1.00	1.00	1
Peptic ulcer diseae	1.00	1.00	1.00	1
Pneumonia	1.00	1.00	1.00	1
Psoriasis	1.00	1.00	1.00	1
Tuberculosis	1.00	1.00	1.00	1
Typhoid	1.00	1.00	1.00	1
Urinary tract infection	1.00	1.00	1.00	1
Varicose veins	1.00	1.00	1.00	1
hepatitis A	1.00	1.00	1.00	1
accuracy			0.95	41
macro avg	0.93	0.95	0.93	41

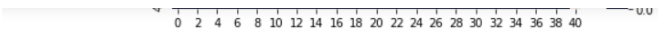
Logistic Regression Algorithm Accuracy: 95%

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0bb1c2e8d0>
```



Third Algorithm used for prediction is Decision Tree.



Decision Tree

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
import numpy as np
```

In [ ]:
```
error= []
# Will take some time
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
```

Out[20]:
```
DecisionTreeClassifier()
```

In [ ]:
```
pred_i = dt.predict(X_test)
error.append(np.mean(pred_i != y_test))
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

In [ ]:
```
print(classification_report(y_test,pred_i))
```

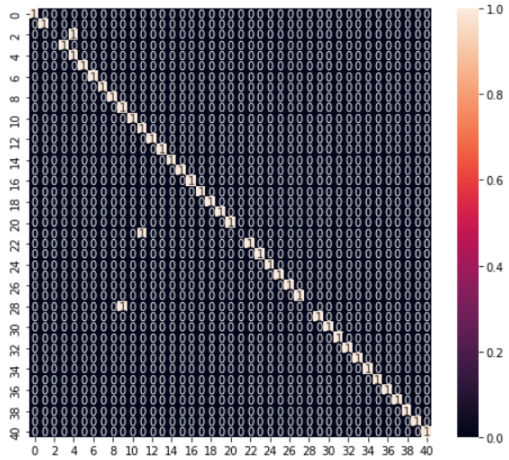
```
                                    precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                AIDS       1.00      1.00      1.00         1
                                Acne       0.00      0.00      0.00         1
                Alcoholic hepatitis       1.00      1.00      1.00         1
                             Allergy       0.50      1.00      0.67         1
                           Arthritis       1.00      1.00      1.00         1
                   Bronchial Asthma       1.00      1.00      1.00         1
                Cervical spondylosis       1.00      1.00      1.00         1
                           Chicken pox       1.00      1.00      1.00         1
                Chronic cholestasis       0.50      1.00      0.67         1
                           Common Cold       1.00      1.00      1.00         1
```

Decision Tree Algorithm Accuracy: 93%

```
                           Chicken pox       1.00      1.00      1.00         1
                Chronic cholestasis       0.50      1.00      0.67         1
                           Common Cold       1.00      1.00      1.00         1
                                Dengue       0.50      1.00      0.67         1
                              Diabetes       1.00      1.00      1.00         1
Dimorphic hemmorhoids(piles)       1.00      1.00      1.00         1
                          Drug Reaction       1.00      1.00      1.00         1
                    Fungal infection       1.00      1.00      1.00         1
                                GERD       1.00      1.00      1.00         1
                    Gastroenteritis       1.00      1.00      1.00         1
                       Heart attack       1.00      1.00      1.00         1
                        Hepatitis B       1.00      1.00      1.00         1
                        Hepatitis C       1.00      1.00      1.00         1
                        Hepatitis D       0.00      0.00      0.00         1
                        Hepatitis E       1.00      1.00      1.00         1
                       Hypertension       1.00      1.00      1.00         1
                    Hyperthyroidism       1.00      1.00      1.00         1
                       Hypoglycemia       1.00      1.00      1.00         1
                     Hypothyroidism       1.00      1.00      1.00         1
                            Impetigo       1.00      1.00      1.00         1
                            Jaundice       0.00      0.00      0.00         1
                             Malaria       1.00      1.00      1.00         1
                            Migraine       1.00      1.00      1.00         1
                      Osteoarthristis       1.00      1.00      1.00         1
Paralysis (brain hemorrhage)       1.00      1.00      1.00         1
             Peptic ulcer diseae       1.00      1.00      1.00         1
                       Pneumonia       1.00      1.00      1.00         1
                       Psoriasis       1.00      1.00      1.00         1
                    Tuberculosis       1.00      1.00      1.00         1
                         Typhoid       1.00      1.00      1.00         1
     Urinary tract infection       1.00      1.00      1.00         1
                  Varicose veins       1.00      1.00      1.00         1
                    hepatitis A       1.00      1.00      1.00         1

                        accuracy                           0.93        41
                       macro avg       0.89      0.93      0.90        41
                    weighted avg       0.89      0.93      0.90        41
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0bb0a3ccd0>



Fourth Algorithm used for prediction is Support Vector Classification.

Support Vector Classification

In [ ]:
```
from sklearn.svm import SVC
import numpy as np
```

In [ ]:
```
error= []
# Will take some time
svc = SVC()
svc.fit(X_train,y_train)
```

Out[26]: SVC()

In [ ]:
```
pred_i = svc.predict(X_test)
error.append(np.mean(pred_i != y_test))
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

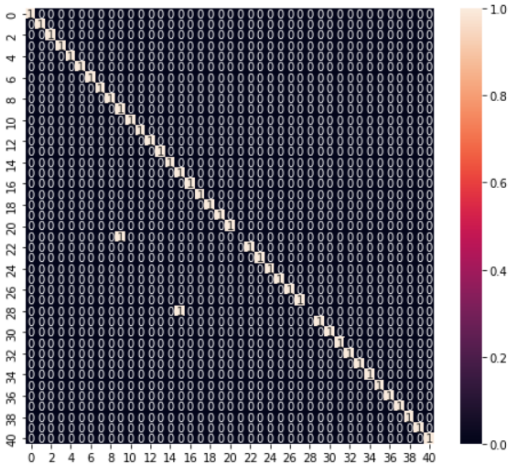
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
                                        precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                    AIDS       1.00      1.00      1.00         1
                                    Acne       1.00      1.00      1.00         1
                    Alcoholic hepatitis       1.00      1.00      1.00         1
                                Allergy       1.00      1.00      1.00         1
                              Arthritis       1.00      1.00      1.00         1
                       Bronchial Asthma       1.00      1.00      1.00         1
                   Cervical spondylosis       1.00      1.00      1.00         1
                            Chicken pox       1.00      1.00      1.00         1
                   Chronic cholestasis       0.50      1.00      0.67         1
                            Common Cold       1.00      1.00      1.00         1
                                 Dengue       1.00      1.00      1.00         1
                               Diabetes       1.00      1.00      1.00         1
```

Support Vector Classification Algorithm Accuracy: 95%

```
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0bb09570d0>



Fifth Algorithm used for prediction is Gaussian Naïve Bayes.

Gaussian Naive Bayes

```
In [ ]: from sklearn.naive_bayes import GaussianNB
        import numpy as np

In [ ]: error= []
        # Will take some time
        nb = GaussianNB()
        nb.fit(X_train,y_train)
```

Out[32]: GaussianNB()

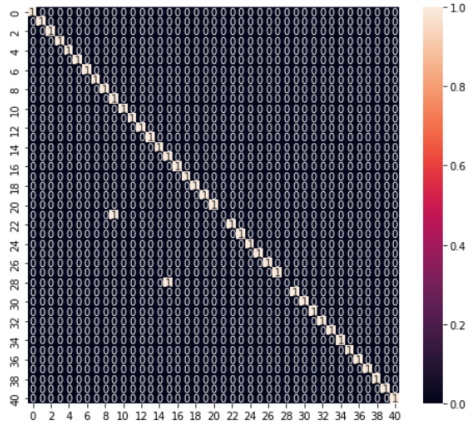
```
In [ ]: pred_i = nb.predict(X_test)
        error.append(np.mean(pred_i != y_test))

In [ ]: from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score

In [ ]: print(classification_report(y_test,pred_i))
```

	precision	recall	f1-score	support
(vertigo) Paroymsal	1.00	1.00	1.00	1
Positional Vertigo	1.00	1.00	1.00	1
AIDS	1.00	1.00	1.00	1
Acne	1.00	1.00	1.00	1
Alcoholic hepatitis	1.00	1.00	1.00	1
Allergy	1.00	1.00	1.00	1
Arthritis	1.00	1.00	1.00	1
Bronchial Asthma	1.00	1.00	1.00	1
Cervical spondylosis	1.00	1.00	1.00	1
Chicken pox	1.00	1.00	1.00	1
Chronic cholestasis	0.50	1.00	0.67	1
Common Cold	1.00	1.00	1.00	1
Dengue	1.00	1.00	1.00	1
Diabetes	1.00	1.00	1.00	1
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	1
Drug Reaction	1.00	1.00	1.00	1

Gaussian Naïve Bayes Algorithm Accuracy: 95%

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0baee18e10>



5.3 DATA PREPROCESSING FOR CORONAVIRUS DATASET

First step is to import the Coronavirus dataset along with the required Libraries.

Reading the Data

In [ ]:
```
import pandas as pd
df = pd.read_csv('/content/sample_data/corona_tested_individuals_ver_0083.english.csv')
```

Data Cleaning

In [ ]:
```
df.head(5)
```

Out[2]:

	test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication
0	2020-11-12	0	0	0	0	0	negative	No	male	Other
1	2020-11-12	0	1	0	0	0	negative	No	male	Other
2	2020-11-12	0	0	0	0	0	negative	Yes	female	Other
3	2020-11-12	0	0	0	0	0	negative	No	male	Other
4	2020-11-12	0	1	0	0	0	negative	No	male	Contact with confirmed

In [ ]:
```
df.describe
```

Out[3]:
```
<bound method NDFrame.describe of          test_date  cough  fever  sore_throat  shortness_of_breath  \
0        2020-11-12      0      0            0                    0
1        2020-11-12      0      1            0                    0
2        2020-11-12      0      0            0                    0
3        2020-11-12      0      0            0                    0
4        2020-11-12      0      1            0                    0
...             ...    ...    ...          ...                  ...
2742591  2020-03-11      0      0            0                    0
2742592  2020-03-11      0      0            0                    0
```

Analysing the Dataset information.

```
         head_ache corona_result age_60_and_above  gender  \
0                0      negative                No    male
1                0      negative                No    male
2                0      negative               Yes  female
3                0      negative                No    male
4                0      negative                No    male
...            ...           ...               ...     ...
2742591          0      negative               NaN  female
2742592          0      negative               NaN  female
2742593          0         other               NaN    male
2742594          0      negative               NaN  female
2742595          0      negative               NaN    male

                test_indication
0                         Other
1                         Other
2                         Other
3                         Other
4         Contact with confirmed
...                          ...
2742591                   Other
2742592                   Other
2742593                   Other
2742594                   Other
2742595                   Other

[2742596 rows x 10 columns]>
```

In [ ]:
```
df.info()
df1 =pd.DataFrame()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2742596 entries, 0 to 2742595
Data columns (total 10 columns):
 #   Column              Dtype
---  ------              -----
 0   test_date           object
```

Removing the None Values from the dataset.

```
df1 =pd.DataFrame()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2742596 entries, 0 to 2742595
Data columns (total 10 columns):
 #   Column               Dtype
---  ------               -----
 0   test_date            object
 1   cough                int64
 2   fever                int64
 3   sore_throat          int64
 4   shortness_of_breath  int64
 5   head_ache            int64
 6   corona_result        object
 7   age_60_and_above     object
 8   gender               object
 9   test_indication      object
dtypes: int64(5), object(5)
memory usage: 209.2+ MB
```

In [ ]:
```
df['cough'].value_counts()
```

Out[5]:
```
0    2631258
1     111338
Name: cough, dtype: int64
```

In [ ]:
```
df = df[df.cough != "None"]
df = df[df.fever != "None"]
```

In [ ]:
```
df = df.astype({'cough': int,'fever': int,'sore_throat': int,'head_ache': int, 'shortness_of_breath': int})
```

In [ ]:
```
df.isnull().sum()
```

Out[8]:
```
test_date                0
cough                    0
```

```
In [ ]: df = df.astype({'cough': int,'fever': int,'sore_throat': int,'head_ache': int, 'shortness_of_breath': int})
```

```
In [ ]: df.isnull().sum()
```

```
Out[8]: test_date                  0
        cough                      0
        fever                      0
        sore_throat                0
        shortness_of_breath        0
        head_ache                  0
        corona_result              0
        age_60_and_above      547644
        gender                 92886
        test_indication            0
        dtype: int64
```

```
In [ ]: df['test_indication'].value_counts()
```

```
Out[9]: Other                    2547559
        Contact with confirmed    170742
        Abroad                     24295
        Name: test_indication, dtype: int64
```

```
In [ ]: df['corona_result'].value_counts()
```

```
Out[10]: negative    2480403
         positive     220975
         other         41218
         Name: corona_result, dtype: int64
```

```
In [ ]: df['age_60_and_above'].value_counts()
```

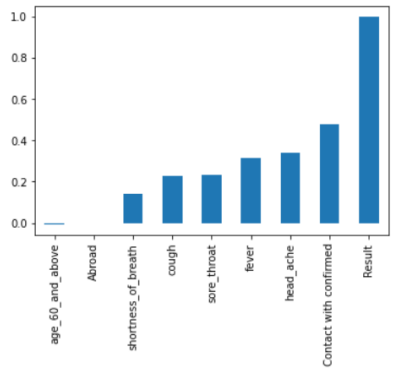
```
Out[11]: No     1908553
         Yes     286399
         Name: age_60_and_above, dtype: int64
```

Getting a correlation graph of all the features that will be used for prediction.

```
In [ ]: df['Result'] = df['corona_result'].map({'negative':0, 'positive':1})
        df2 =pd.get_dummies(df['test_indication'])
        df = pd.concat([df, df2],axis=1)
```

```
In [ ]: df.corr()["Result"][:-1].sort_values().plot(kind='bar')
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa6c554350>
```




```
In [ ]: df.corr()['Result'].sort_values()
```

```
Out[15]: Other                   -0.447394
         age_60_and_above        -0.008201
         Abroad                   0.000557
         shortness_of_breath      0.140696
         cough                    0.229705
```

Final Dataset after cleaning that will used for prediction.

```
head_ache                0.339869
Contact with confirmed   0.476043
Result                   1.000000
Name: Result, dtype: float64
```

```
In [ ]: df.dropna(inplace=True)
        df.columns
```

```
Out[16]: Index(['test_date', 'cough', 'fever', 'sore_throat', 'shortness_of_breath',
                'head_ache', 'corona_result', 'age_60_and_above', 'gender',
                'test_indication', 'Result', 'Abroad', 'Contact with confirmed',
                'Other'],
               dtype='object')
```

```
In [ ]: df.to_csv('/content/sample_data/Corona_PreProcessed2.csv')
```

```
In [ ]: df.head()
```

Out[18]:

	test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication	Result	Abroad	Contact with confirmed
0	2020-11-12	0	0	0	0	0	negative	0.0	male	Other	0.0	0	0
1	2020-11-12	0	1	0	0	0	negative	0.0	male	Other	0.0	0	0
2	2020-11-12	0	0	0	0	0	negative	1.0	female	Other	0.0	0	0
3	2020-11-12	0	0	0	0	0	negative	0.0	male	Other	0.0	0	0
4	2020-11-12	0	1	0	0	0	negative	0.0	male	Contact with confirmed	0.0	0	1

5.4 MODEL TRAINING ON CORONAVIRUS DATASET

First step is to import the dataset and the required libraries used for prediction.

```
In [ ]: import pandas as pd
```

Reading the Data

```
In [ ]: df = pd.read_csv(r'C:\Users\abhis\Documents\Covid Pred ver_0083\PreprocessedData.csv', index_col = 0)
```

```
In [ ]: df.head()
```

Out[3]:

	test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication	Result	Abroad	Contact with confirmed
0	2020-11-12	0	0	0	0	0	negative	0.0	male	Other	0.0	0	0
1	2020-11-12	0	1	0	0	0	negative	0.0	male	Other	0.0	0	0
2	2020-11-12	0	0	0	0	0	negative	1.0	female	Other	0.0	0	0
3	2020-11-12	0	0	0	0	0	negative	0.0	male	Other	0.0	0	0
4	2020-11-12	0	1	0	0	0	negative	0.0	male	Contact with confirmed	0.0	0	1

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

```
Out[4]: test_date              0
        cough                  0
        fever                  0
        sore_throat            0
        shortness_of_breath    0
        head_ache              0
```

Splitting the data into training and testing with a test size of 30%.

```
In [ ]: df[df['fever'].isna()]
```

Out[5]:

test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication	Result	Abroad	Contact with confirmed

```
In [ ]: df.shape
```

Out[6]: (2151898, 14)

```
In [ ]: df.dropna(inplace=True)
```

Train Test Split

```
In [ ]: from sklearn.model_selection import train_test_split
        X=df.drop(['Result','corona_result','gender','test_indication','Other','test_date'],axis=1).values
        y=df["Result"].values
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
In [ ]: df3=df
```

```
In [ ]: df3.drop(['Result','corona_result','gender','test_indication','Other','test_date'],axis=1)
```

Out[10]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	age_60_and_above	Abroad	Contact with confirmed
0	0	0	0	0	0	0.0	0	0
1	0	1	0	0	0	0.0	0	0
2	0	0	0	0	0	1.0	0	0
3	0	0	0	0	0	0.0	0	0
4	0	1	0	0	0	0.0	0	1

First Algorithm Used for Prediction is Random Forest.

Random Forest Algorithm Accuracy: 93%

2731643	1	0	0	0	0	0.0	0	1
2731644	0	0	0	0	0	0.0	0	0
2731647	0	0	0	0	0	0.0	0	0

2151898 rows × 8 columns

Randon Forest Classifier

```
In [ ]: from sklearn.ensemble import RandomForestClassifier
        import numpy as np
```

```
In [ ]: rfc = RandomForestClassifier()
        rfc.fit(X_train,y_train)
```

Out[12]: RandomForestClassifier()

```
In [ ]: pred_i = rfc.predict(X_test)
        error = np.mean(pred_i != y_test)
```

```
In [ ]: from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

```
In [ ]: print(classification_report(y_test,pred_i))
```

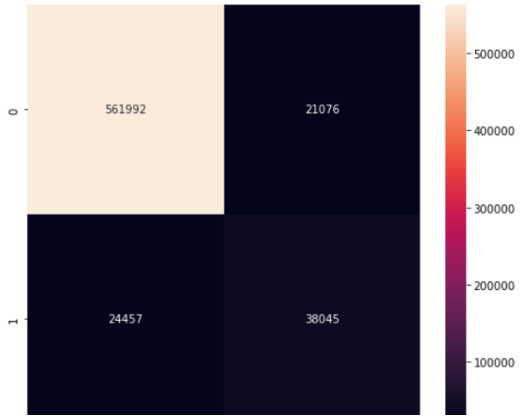
```
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96    583068
         1.0       0.64      0.61      0.63     62502

    accuracy                           0.93    645570
   macro avg       0.80      0.79      0.79    645570
weighted avg       0.93      0.93      0.93    645570
```

Confusion Matrix for Random Forest Algorithm.

```
weighted avg       0.93      0.93      0.93    645570
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[16]: <AxesSubplot:>



Second Algorithm Used for Prediction is Logistic Regression.

Logistic Regression Algorithm Accuracy: 92%

Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
import numpy as np
```

In [ ]:
```
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

Out[18]: LogisticRegression()

In [ ]:
```
pred_i = lr.predict(X_test)
error = np.mean(pred_i != y_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

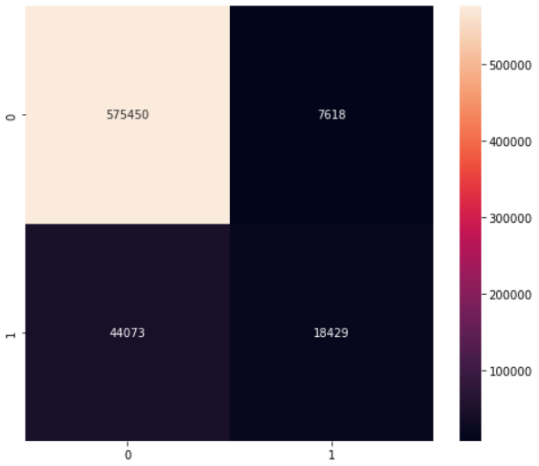
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96    583068
         1.0       0.71      0.29      0.42     62502

    accuracy                           0.92    645570
   macro avg       0.82      0.64      0.69    645570
weighted avg       0.91      0.92      0.90    645570
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
```

```
In [ ]: CM = confusion_matrix(y_test,pred_i)

        import seaborn as sns
        import matplotlib.pyplot as plt

        # labels = ['0', '1']

        fig, ax = plt.subplots(figsize=(8, 7))
        sns.heatmap(CM, annot=True, fmt=".0f")
```

```
Out[22]: <AxesSubplot:>
```



Third Algorithm Used for Prediction is Decision Tree.

Decision Tree Algorithm Accuracy: 93%

Decision Tree

```
In [ ]: from sklearn.tree import DecisionTreeClassifier
        import numpy as np
```

```
In [ ]: # Will take some time
        dt = DecisionTreeClassifier()
        dt.fit(X_train,y_train)
```

```
Out[24]: DecisionTreeClassifier()
```

```
In [ ]: pred_i = dt.predict(X_test)
        error= np.mean(pred_i != y_test)
```

```
In [ ]: from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

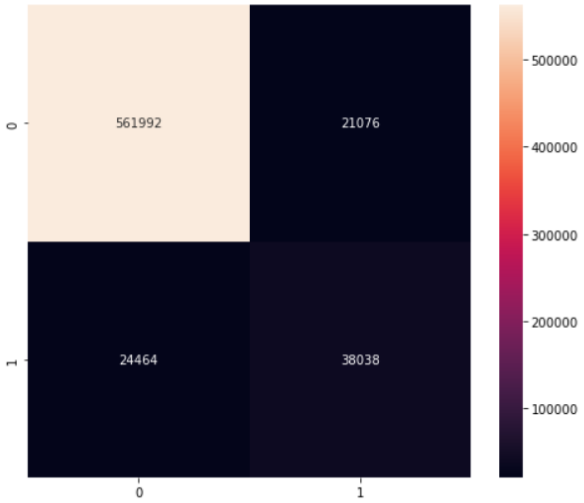
```
In [ ]: print(classification_report(y_test,pred_i))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96    583068
         1.0       0.64      0.61      0.63     62502

    accuracy                           0.93    645570
   macro avg       0.80      0.79      0.79    645570
weighted avg       0.93      0.93      0.93    645570
```

```
In [ ]: CM = confusion_matrix(y_test,pred_i)

        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[28]: <AxesSubplot:>



Fourth Algorithm Used for Prediction is Support Vector Classification.

Support Vector Classification Algorithm Accuracy: 93%

Support Vector Classification

In [ ]:
```
from sklearn.svm import SVC
import numpy as np
```

In [ ]:
```
# Will take some time
svc = SVC()
svc.fit(X_train,y_train)
```

Out[30]: SVC()

In [ ]:
```
pred_i = svc.predict(X_test)
error = np.mean(pred_i != y_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

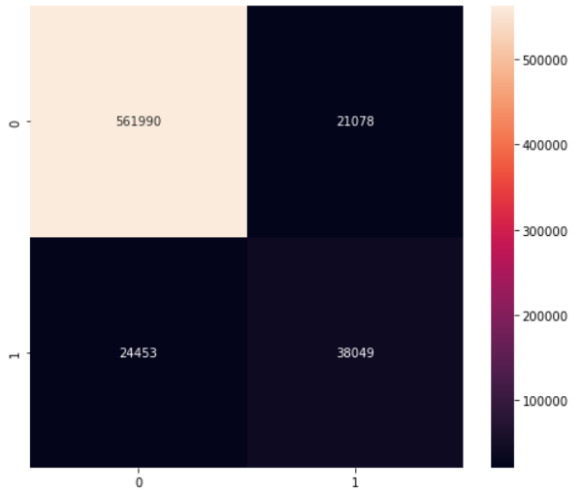
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.96      0.96    583068
         1.0       0.64      0.61      0.63     62502

    accuracy                           0.93    645570
   macro avg       0.80      0.79      0.79    645570
weighted avg       0.93      0.93      0.93    645570
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[34]: <AxesSubplot:>



Fifth Algorithm Used for Prediction is Gaussian Naïve Bayes.

Gaussian Naïve Bayes Algorithm Accuracy: 92%

Gaussian Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
import numpy as np
```

In [ ]:
```
# Will take some time
nb = GaussianNB()
nb.fit(X_train,y_train)
```

Out[79]: GaussianNB()

In [ ]:
```
pred_i = nb.predict(X_test)
error = np.mean(pred_i != y_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

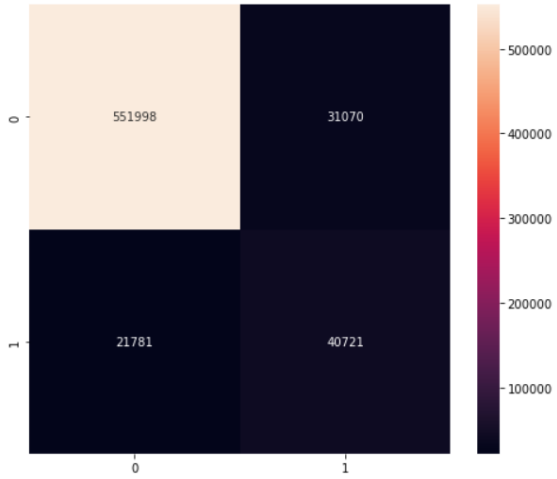
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.95      0.95    583068
         1.0       0.57      0.65      0.61     62502

    accuracy                           0.92    645570
   macro avg       0.76      0.80      0.78    645570
weighted avg       0.92      0.92      0.92    645570
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt

# labels = ['0', '1']

fig, ax = plt.subplots(figsize=(8, 7))
sns.heatmap(CM, annot=True, fmt=".0f")
```

Out[83]: <AxesSubplot:>



Sixth Algorithm Used for Prediction is K-Nearest Neighbour.

K-Nearest Neighbour Algorithm Accuracy: 93%

KNN

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
```

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

Out[42]: KNeighborsClassifier(n_neighbors=7)

In [ ]:
```
pred_i = knn.predict(X_test)
error = np.mean(pred_i != y_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix,classification_report,f1_score,accuracy_score
```

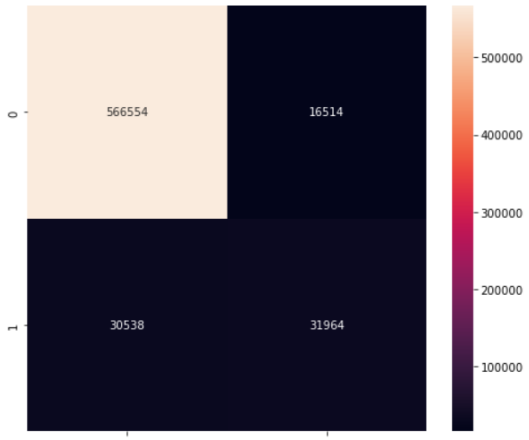
In [ ]:
```
print(classification_report(y_test,pred_i))
```

```
              precision    recall  f1-score   support

         0.0       0.95      0.97      0.96    583068
         1.0       0.66      0.51      0.58     62502

    accuracy                           0.93    645570
   macro avg       0.80      0.74      0.77    645570
weighted avg       0.92      0.93      0.92    645570
```

In [ ]:
```
CM = confusion_matrix(y_test,pred_i)

import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [ ]: CM = confusion_matrix(y_test,pred_i)

        import seaborn as sns
        import matplotlib.pyplot as plt

        # labels = ['0', '1']

        fig, ax = plt.subplots(figsize=(8, 7))
        sns.heatmap(CM, annot=True, fmt=".0f")
```

```
Out[106]: <AxesSubplot:>
```



5.5 DATASET DETAILS

We've worked on 2 Datasets in our project.

1) The first provided dataset for the "Disease Prediction System" was collected from Kaggle.

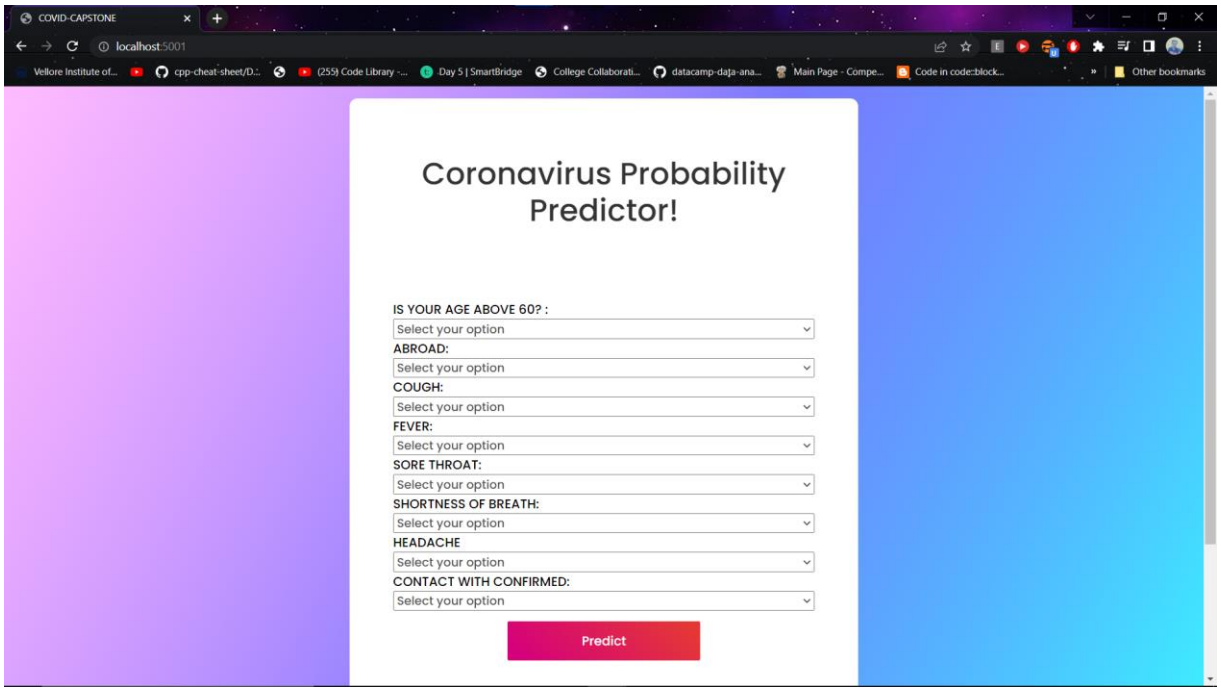
https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset?select=dataset.csv

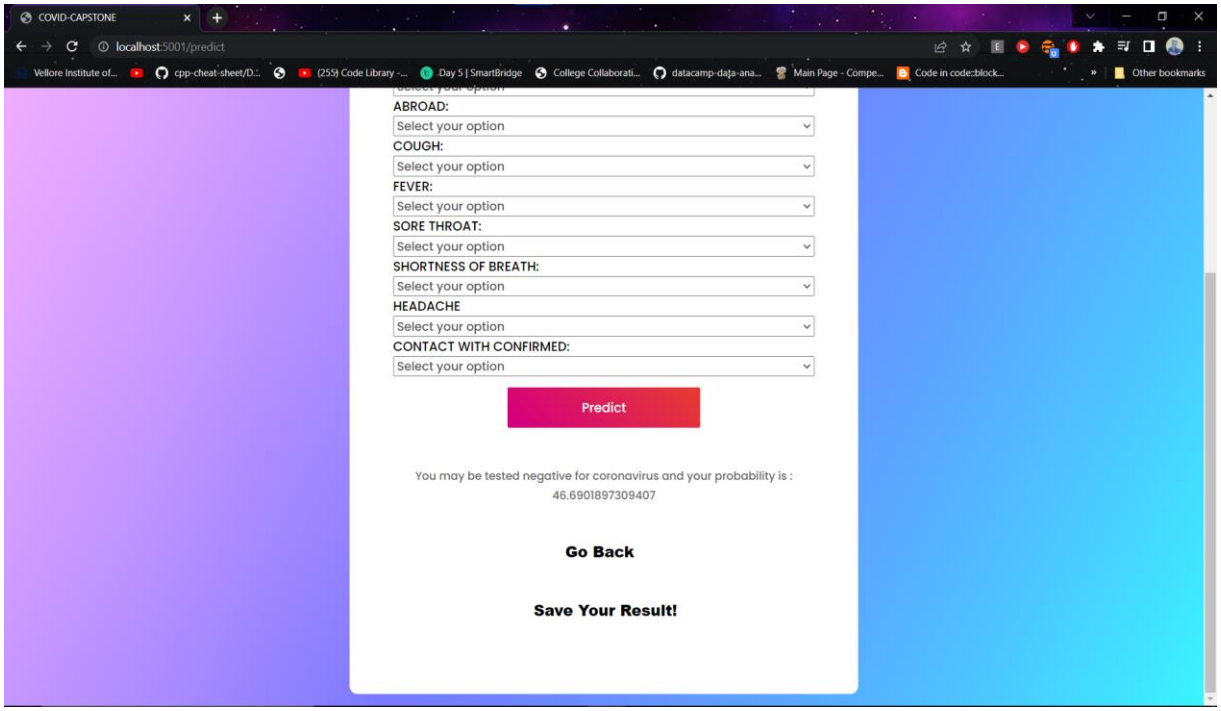
2) The second provided dataset for the "Covid-19 Prediction" contains covid test information such as the date, gender, whether the person has had cough, fever, sore throat, shortness of breath, headache, age, whether they were abroad or had contact with someone with covid and the test result. The dataset is obtained by a government website.

https://data.gov.il/dataset/covid-19/resource/d337959a-020a-4ed3-84f7-fca182292308

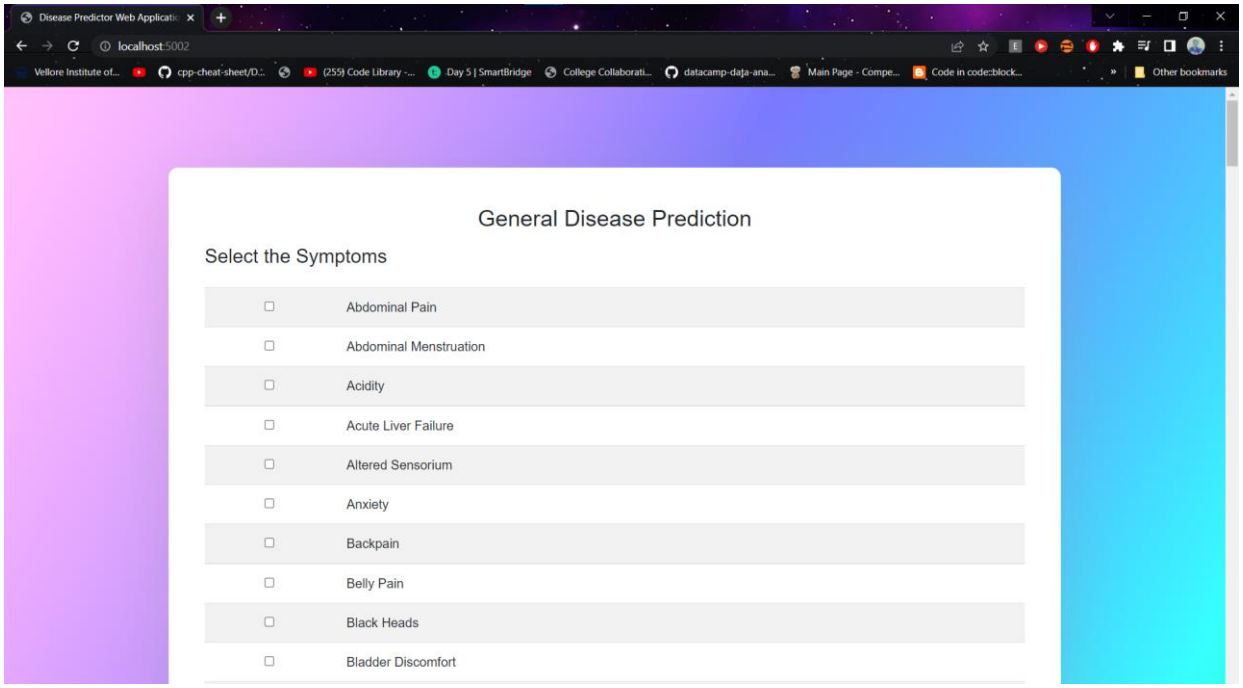
5.6 WEBSITE IMAGES

CORONAVIRUS PROBABLITY PREDICTION PAGE (USING FLASK)





GENEAL DISEASE PREDICTION PAGE (USING FLASK)



5.7 SAMPLE CODE

```
from flask import Flask, render_template, request
from joblib import load
import pandas as pd
import numpy as np
from scipy.stats import mode

app = Flask(__name__)

# model = load('./Saved Model/naive_bayes_model.joblib')
model1=load( './Saved Model/decision_tree.joblib')
model2=load('./Saved Model/random_forest_model.joblib')

symptoms_dict = {'itching': 0,
                 ' skin_rash': 1,
                 ' continuous_sneezing': 2,
```

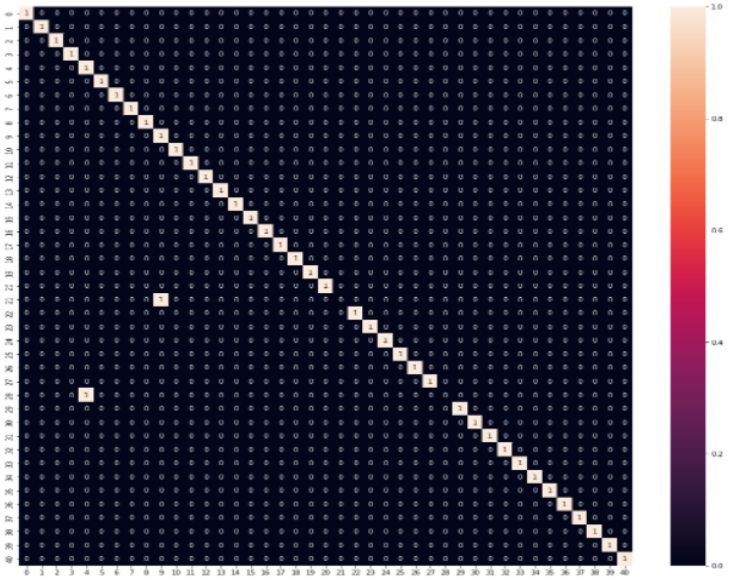
```
                ' shivering': 3,
                ' stomach_pain': 4,
                ' acidity': 5,
                ' vomiting': 6,
                ' indigestion': 7,
                ' muscle_wasting': 8,
                ' patches_in_throat': 9,
                ' fatigue': 10,
                ' weight_loss': 11,
                ' sunken_eyes': 12,
                ' cough': 13,
                ' headache': 14,
                ' restlessness': 37,
                ' dehydration': 38,
                ' dizziness': 39,
                ' weakness_of_one_body_side': 40,
                ' lethargy': 41,
                ' nausea': 42,
                ' abdominal_pain': 43,
                ' pain_in_anal_region': 44,
                ' sweating': 45,
                ' bruising': 46,
                ' cold_hands_and_feets': 47,
                ' anxiety': 48,
                ' knee_pain': 49,
                ' swelling_joints': 50,
                ' blackheads': 51,
                ' foul_smell_of urine': 52,
                ' skin_peeling': 53,
                ' blister': 54,
                ' dischromic _patches': 55,
                ' watering_from_eyes': 56,
                    '  extra_marital_contacts':  57,                    '
visual_disturbances': 106,
                ' brittle_nails': 107,
                ' drying_and_tingling_lips': 108,
                ' polyuria': 109,
                ' pain_behind_the_eyes': 110,
                ' toxic_look_(typhos)': 111,
                ' throat_irritation': 112,
                ' swollen_extremeties': 113,
                ' slurred_speech': 114,
                ' red_spots_over_body': 115,
                ' belly_pain': 116,
                ' receiving_blood_transfusion': 117,
                ' acute_liver_failure': 118,
                ' redness_of_eyes': 119,
                ' rusty_sputum': 120,
```

```
                ' abnormal_menstruation': 121,
                ' receiving_unsterile_injections': 122,
                ' coma': 123,
                ' sinus_pressure': 124,
                ' palpitations': 125,
                ' stomach_bleeding': 126,
                ' runny_nose': 127,
                ' congestion': 128,
                ' blood_in_sputum': 129,
                ' loss_of_smell': 130
                }
symptoms_dict    =    pd.DataFrame(list(symptoms_dict.items()),    columns=
['Symptoms', 'Count'])


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



@app.route("/", methods=['GET','POST'])
def predict():
    input_vector = np.zeros(len(symptoms_dict))


    if request.method == 'POST':
        symptoms = request.form.getlist('symptoms_checkbox')
        for i in range(0, len(symptoms)):
            symptoms[i] = int(symptoms[i])
        for symptom in symptoms:
            input_vector[symptom] = 1
            # symp = []
            # symp.append(symptoms_dict.iloc[symptom, 1])
            # input_vector[symp] = 1
            # pred = model.predict([input_vector])[0]
        pred1 = model1.predict([input_vector])[0]
        pred2 = model2.predict([input_vector])[0]
        finalprediction=mode([pred1,pred2])[0][0]

                      return     render_template('show.html',     disease=
finalprediction,d2=pred1,d3=pred2)
    return render_template('index.html')


if __name__ == "__main__":
    app.run(host="localhost",port=5003,debug=True)
```

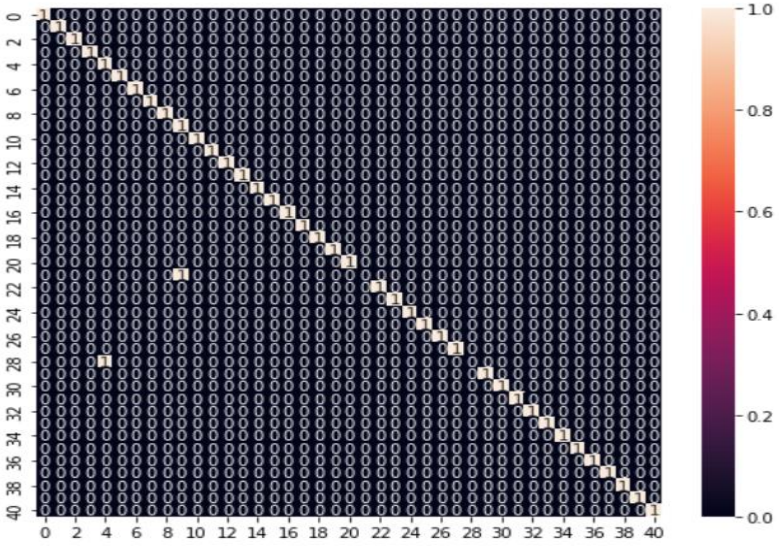
Testing / Performance metrics

6.1 CONFUSION MATRIX FOR DISEASE PREDICTION DATASET

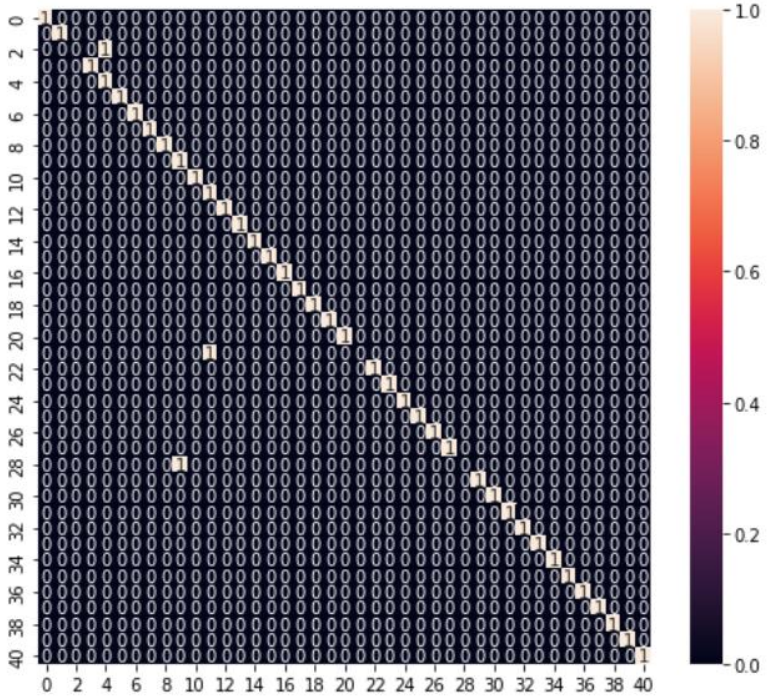
- RANDOM FOREST



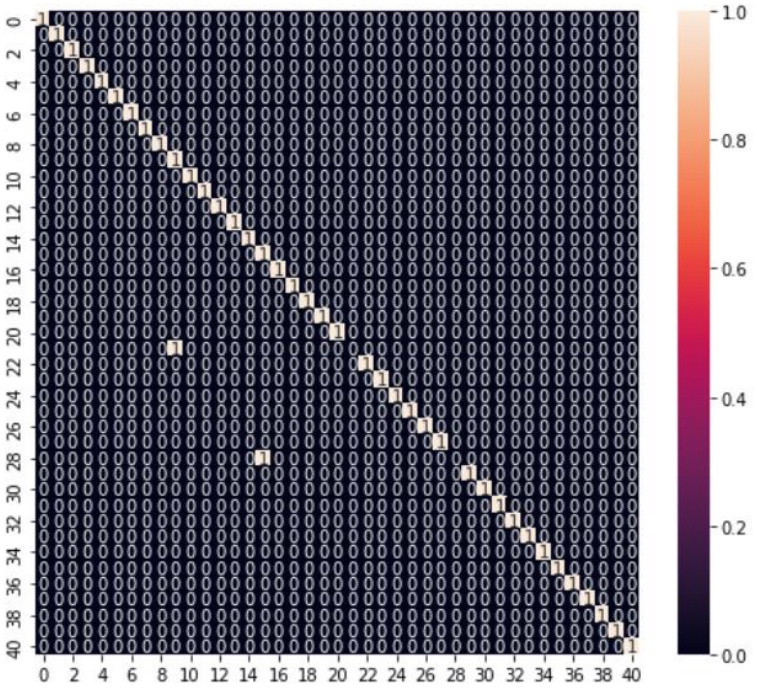
- LOGISTIC REGRESSION



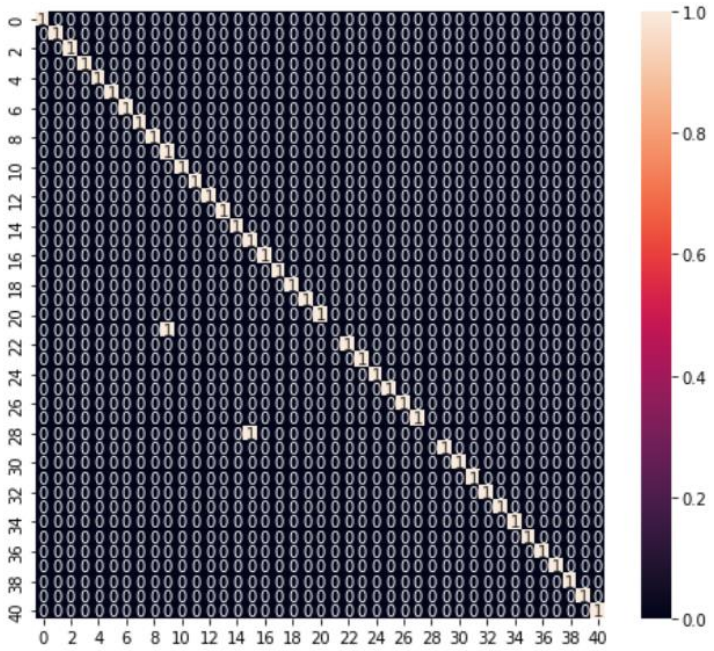
- DECISION TREE



- SUPPORT VECTOR CLASSIFICATION

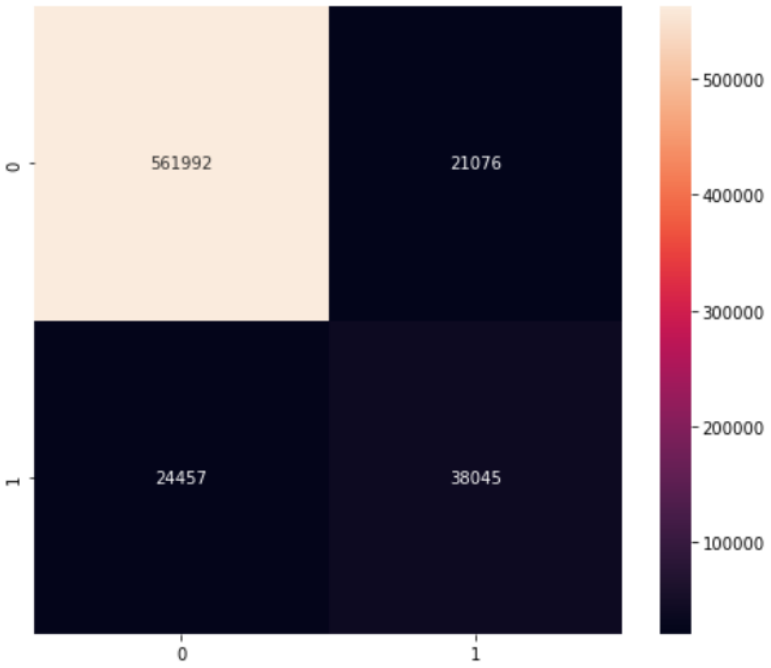


• GAUSSIAN NAÏVE BAYES

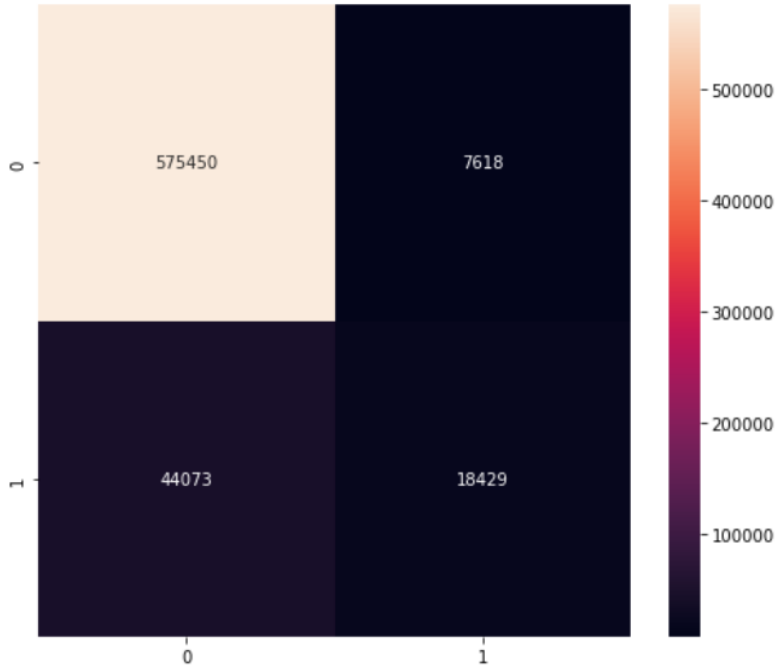


6.2 CONFUSION MATRIX FOR CORONAVIRUS DATASET

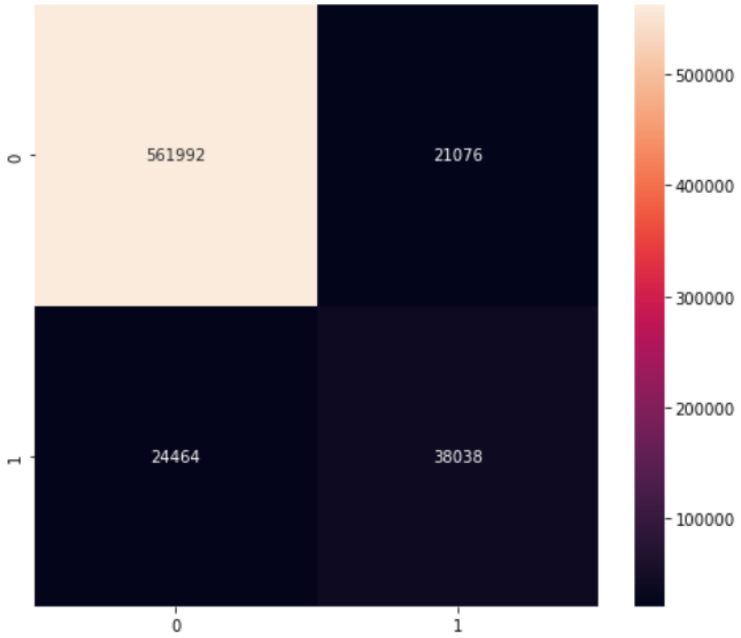
• RANDOM FOREST



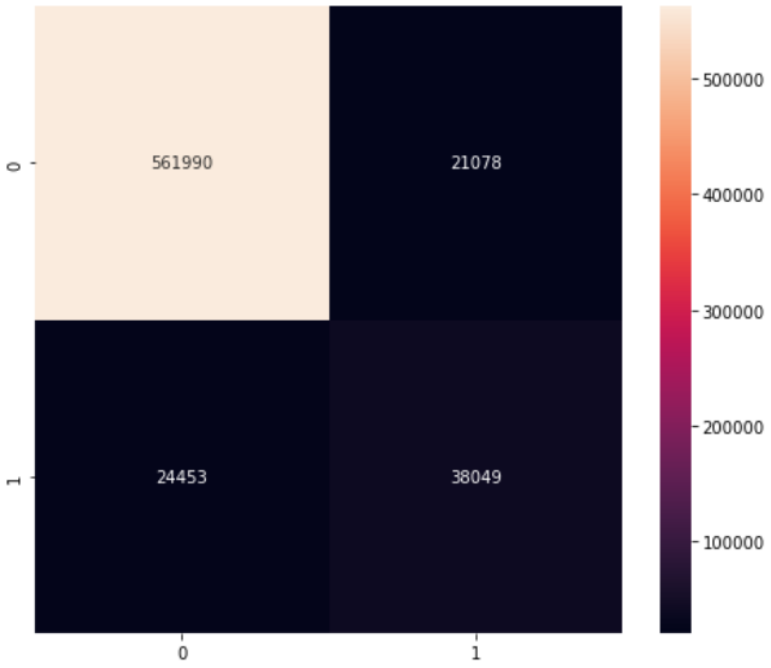
- LOGISTIC REGRESSION



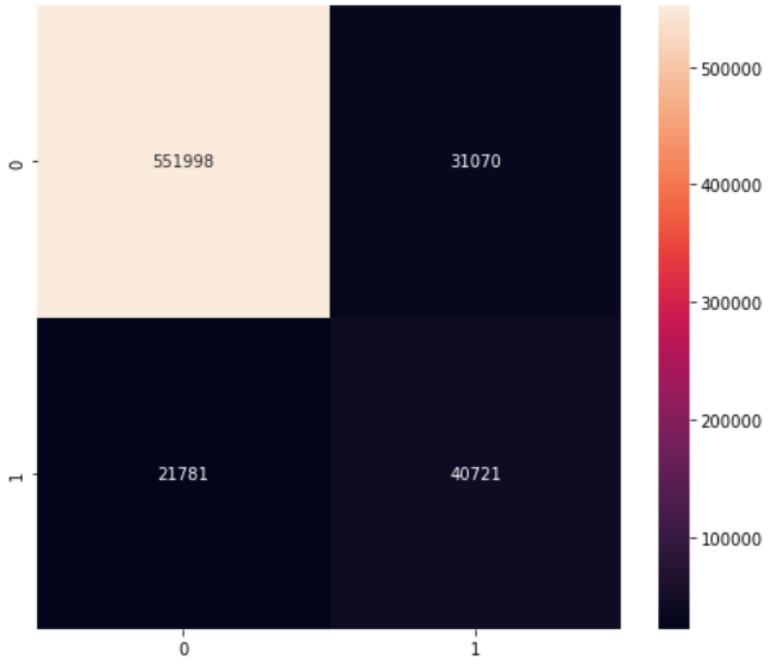
- DECISION TREE



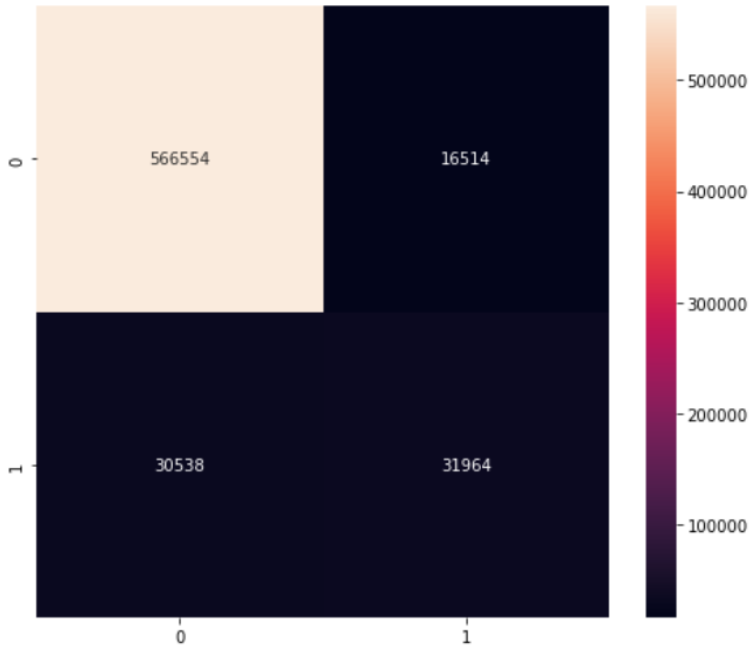
- SUPPORT VECTOR CLASSIFICATION



- GAUSSIAN NAÏVE BAYES



• K-NEAREST NEIGHBOR



6.3 COMPARISION BETWEEN MODELS

CORONAVIRUS DATASET

Algorithm	Accuracy	Precision		Recall		F1-score	
		0	1	0	1	0	1
Logistic Regression	92%	0.93	0.71	0.99	0.30	0.96	0.42
Decision Tree	93%	0.95	0.66	0.97	0.57	0.96	0.61
Random Forest	93%	0.95	0.66	0.97	0.57	0.96	0.61
Gaussian Naïve Bayes	92%	0.96	0.57	0.95	0.65	0.95	0.61
Support Vector Classification	93%	0.96	0.64	0.96	0.61	0.96	0.63
KNN	92%	0.96	0.57	0.95	0.65	0.95	0.61

GENERAL DISEASE DATASET

Algorithm	Accuracy
Logistic Regression	95%
Decision Tree	93%
Random Forest	95%
Gaussian Naïve Bayes	95%
Support Vector Classification	95%

Conclusion & Future Work

So finally, wse conclude by saying that this project will help in improving future of the public health care system and help patients in getting fast and appropriate medical opinion as it provides prediction for General Disease and Covid using symptoms entered by the user at the comfort of home. The system uses Logistic Regression and multiple machine learning algorithms for General Disease Prediction. Using Algorithms, the maximum accuracy achieved is 95%. Timely prediction and diagnosis of an ailment can evade a general disease turning into a fatal disease.

FUTURE WORK

The Future Vision of the Project encompasses collecting information and data of latest diseases. we also plan to include as part of the project recommendation of the medical experts based on the predicted disease and develop a mobile application for the current based web application.

REFERENCES

1) Rajora, H., Singh Punn, N., Sonbhadra, S. K., & Agarwal, S. (2021). Web based disease prediction and recommender system. *arXiv e-prints*, arXiv-2106.

2) Mohan Kumar, K. N., Sampath, S., & Imran, M. (2019). An overview on disease prediction for preventive care of health deterioration. *IJEAT*, *8*(5S), 255-261.

3) Hossain, M. E., Khan, A., Moni, M. A., & Uddin, S. (2019). Use of electronic health data for disease prediction: A comprehensive literature review. *IEEE/ACM transactions on computational biology and bioinformatics*, *18*(2), 745-758.

4) Ashish Kumar, Priya Ghansela, Purnima Soni , Chirag Goswami , Parasmani Sharma. (2020). Prediction of diseases using supervised learning. *International journal of creative research thoughts*.

5) Chen, M., Hao, Y., Hwang, K., Wang, L., & Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. *Ieee Access*, *5*, 8869-8879.

6) Aishwarya, C., Suvarchala, K., Aravind, B., Shashank, G., Anand, M., & Krishna Rao, N. V. (2020). Prediction of disease using machine learning and deep learning. In *Energy Systems, Drives and Automations* (pp. 69-79). Springer, Singapore.

7) Ferjani, M. F. Disease Prediction Using Machine Learning.

8) Dahiwade, D., Patle, G., & Meshram, E. (2019, March). Designing disease prediction model using machine learning approach. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1211-1215). IEEE.

9) Keniya, R., Khakharia, A., Shah, V., Gada, V., Manjalkar, R., Thaker, T., ... & Mehendale, N. (2020). Disease prediction from various symptoms using machine learning. *Available at SSRN 3661426*.

10) Park, D. J., Park, M. W., Lee, H., Kim, Y. J., Kim, Y., & Park, Y. H. (2021). Development of machine learning model for diagnostic disease prediction based on laboratory tests. *Scientific reports*, *11*(1), 1-11.

11) Thakral, B., Saluja, K., Bajpai, M., Sharma, R. R., & Marwaha, N. (2005). Importance of weak ABO subgroups. *Laboratory Medicine*, *36*(1), 32-34.

12) Kaur, R., & Jain, A. (2012). Rare blood donor program in the country: Right time to start. *Asian Journal of Transfusion Science*, *6*(1), 1.

Declaration by Student

We certify that we have properly verified all the items in the checklist and ensure that the report is inproper format as specified in the course handout.

Name: Gaurav Birdi, Md. Omer, Sachin Sharma, Prakhar Katiyar
Place: Delhi Technical Campus
Date: 07/04/2022
Signature of Project Guide:

Verification by Faculty Project Guide

We have duly verified all the items in the checklist and ensured that the report is in proper format.

Name: Ms. Upasna Joshi
Place: Delhi Technical Campus
Date: 07/04/2022
Signature of Project Guide: