CS6109 COMPILER DESIGN LAB

B.E CSE V Q - BATCH

LAB NO: 18 **DATE: 08/01/2022**

TEAM MEMBERS: **TEAM NO: 17**

S. No.	REG. NO.	NAME
1.	2019103506	Anika Arivarashan Sangeetha
2.	2019013530	Jeyasri Meenakshi K
3.	2019103599	Vishnupriya N

Project Title: Smart Doc- A Disease Prediction Model

Abstract, Deliverables – Input & Output, Problem Statement & Objective and Modules (I/O) (2)	
Literature Survey (2)	
Block Diagram (3)	
Dataset Description (3)	
Result Implementation (5)	
Total (15)	
Signature	

COMPILER DESIGN PROJECT

TEAM 17
-Anika A S (2019103506)
-Jeyasri Meenakshi K (2019103530)
-Vishnupriya N (2019103599)

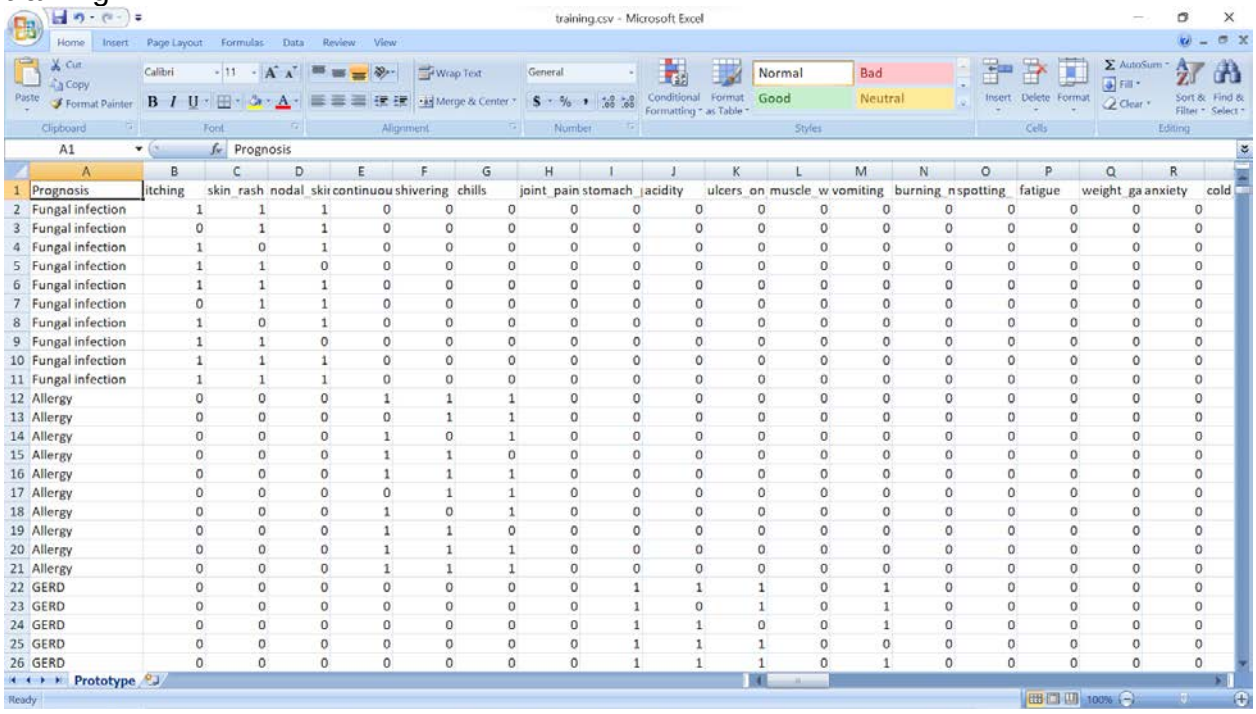
Smart Doc- A Disease Prediction Model

Abstract

Diseases and health related problems like malaria, dengue, Impetigo, Diabetes, Migraine, Jaundice, Chickenpox etc., cause significant effect on one's health and sometimes might also lead to death if ignored. The healthcare industry can make an effective decision making by "mining" the huge database they possess i.e. by extracting the hidden patterns and relationships in the database. Data mining algorithms like Decision Tree, Random Forest KNN and Naïve Bayes algorithms can give a remedy to this situation. Hence, we have developed an automated system that can discover and extract hidden knowledge associated with the diseases from a historical(diseases-symptoms) dataset according to the rule set by the respective algorithms.

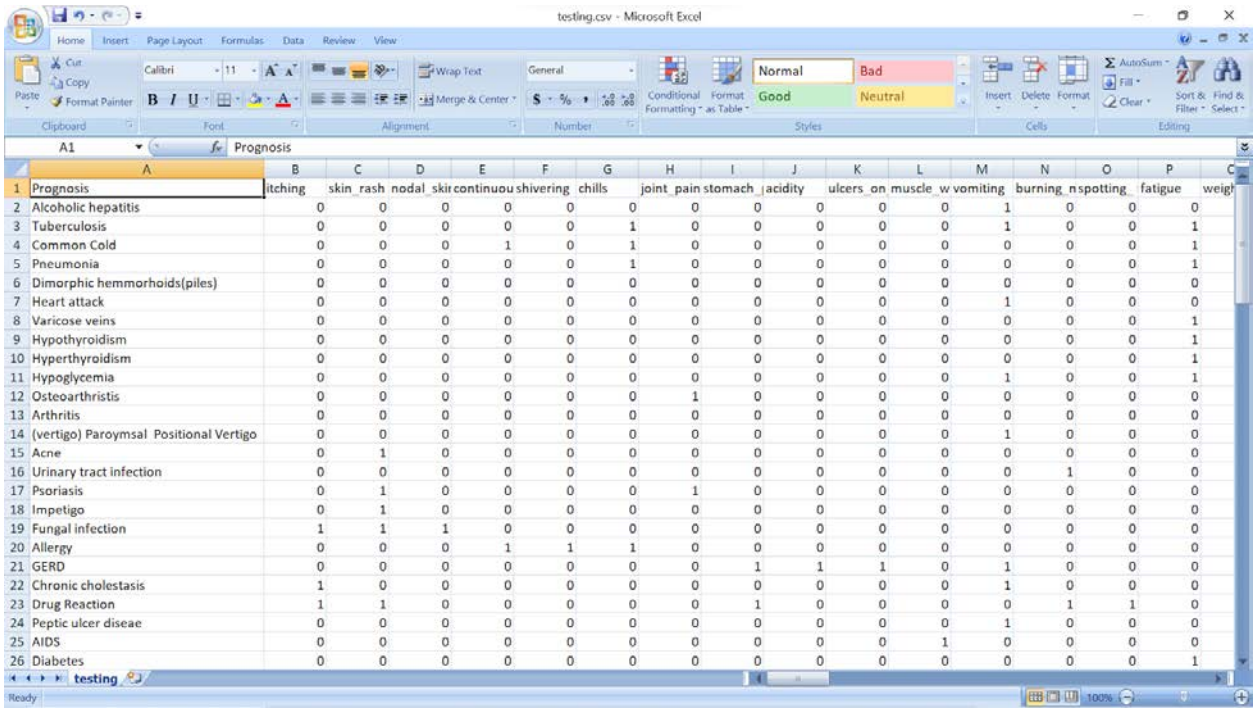
Deliverables

Input:
A dataset containing various prognosis and possible symptoms for each prognosis for training.



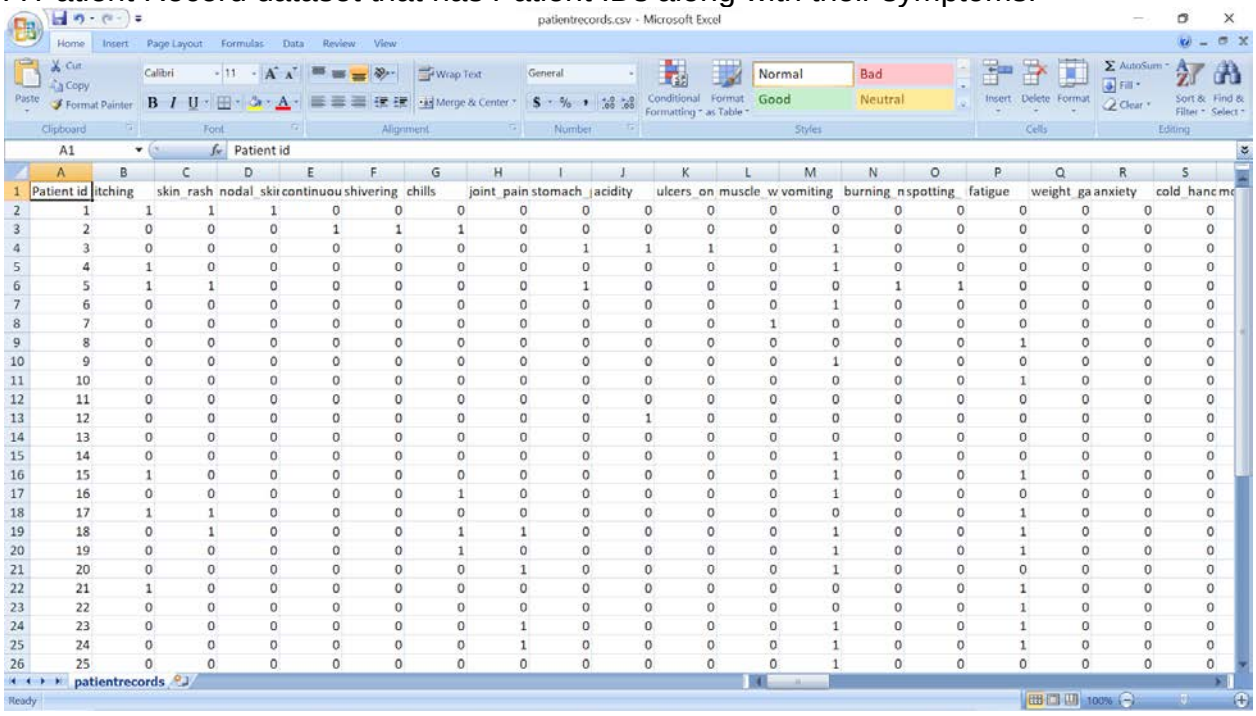
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	Prognosis	itching	skin_rash	nodal_skir	continuou	shivering	chills	joint_pain	stomach_p	acidity	ulcers_on	muscle_w	vomiting	burning_n	spotting_	fatigue	weight_ga	anxiety	cold
2	Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	Fungal infection	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	Fungal infection	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	Fungal infection	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	Fungal infection	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	Fungal infection	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	Fungal infection	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11	Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
13	Allergy	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	
14	Allergy	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
15	Allergy	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
16	Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
17	Allergy	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	
18	Allergy	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
19	Allergy	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
20	Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
21	Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
22	GERD	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	
23	GERD	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	
24	GERD	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	
25	GERD	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	
26	GERD	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	

A dataset containing various prognosis and possible symptoms for each prognosis for testing.



Prognosis	itching	skin_rash	nodal_skir	continuou	shivering	chills	joint_pain	stomach_	acidity	ulcers_on	muscle_w	vomiting	burning_n	spotting	fatigue	weigh
Alcoholic hepatitis	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Tuberculosis	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	
Common Cold	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	
Pneumonia	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	
Dimorphic hemmorhoids(piles)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Heart attack	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Varicose veins	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Hypothyroidism	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Hyperthyroidism	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Hypoglycemia	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	
Osteoarthristis	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
Arthritis	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
(vertigo) Paroymsal Positional Vertigo	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Acne	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
Urinary tract infection	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Psoriasis	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	
Impetigo	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
Fungal infection	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
Allergy	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	
GERD	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	
Chronic cholestasis	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Drug Reaction	1	1	0	0	0	0	0	1	0	0	0	0	1	1	0	
Peptic ulcer diseae	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
AIDS	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Diabetes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

A Patient Record dataset that has Patient IDs along with their symptoms.



Patient id	itching	skin_rash	nodal_skir	continuou	shivering	chills	joint_pain	stomach_	acidity	ulcers_on	muscle_w	vomiting	burning_n	spotting	fatigue	weight_ga	anxiety	cold_hanc	mo
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
5	1	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
15	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	
16	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	
17	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
18	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0	0	0	
19	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	
20	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
23	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	
24	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	
25	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	

Output:

A file containing a *final prediction* of the disease for each *Patient* .

1. Introduction

The accurate analysis of medical database benefits in early disease prediction, patient care and community services. The techniques of machine learning have been successfully employed in assorted applications including Disease prediction. The aim of developing a classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage.

A sample data of 4920 patients' records diagnosed with 41 diseases was selected for analysis. A dependent variable was composed of 41 diseases. 95 of 132 independent variables(symptoms) closely related to diseases were selected and optimized. This project demonstrates the disease prediction system developed using Machine learning algorithms such as Decision Tree classifier, Random forest classifier, KNearestNeighbour and Naïve Bayes classifier.

We aim to develop a prediction model that takes in the symptoms from the users and predicts the disease they are more likely to have.

2. Problem Statement

Challenges faced by many people are looking online for health information regarding diseases, diagnoses and different treatments. If a prediction system is made for doctors and medicine while using review mining, it will save a lot of time. Also, the problems faced by the users in understanding the heterogeneous medical vocabulary are laymen.

3. Objective

The aim of developing a classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage.

4. Literature Survey

1) Introduction

The accurate analysis of medical database benefits in early disease prediction, patient care and community services. The techniques of machine learning have been successfully employed in assorted applications including Disease prediction. The aim of developing a classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage.

2) Summary of the base paper

Based on the records of patients, considering their symptoms, the combination or permutations of which leads to 41 diseases. *Sneha Grampurohit* and *Chetan Sagarna*l have aimed to develop a prediction model that takes in the symptoms from the user and predicts the disease he is more likely to have. The Prediction that they developed considers 95 symptoms, amidst which the user can give the symptoms his processing as the input.

The four algorithms they have trained the system to predict the disease are

- Decision Tree Classifier
- Random forest Classifier
- KNearestNeighbour Classifier
- Naïve Bayes Classifier

Once the system is trained with the training set using the mentioned algorithms a rule set is formed and when the user the symptoms are given as an input to the model, those symptoms are processed according to the rule set developed, thus making classifications and predicting the most likely disease after a comparative study at the end of work.

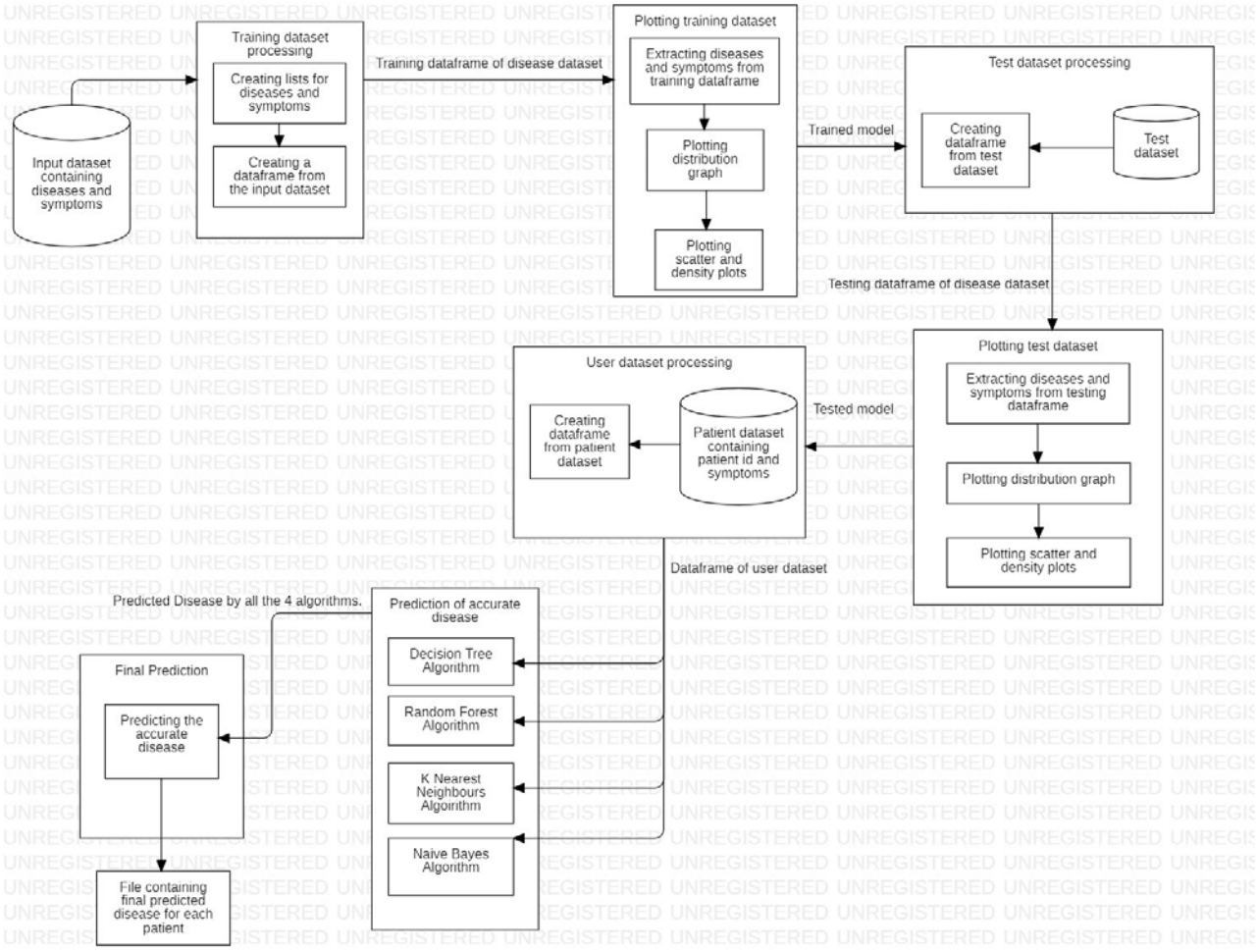
3) Conclusion

From the historical development of machine learning and its applications in the medical sector, it can be shown that systems and methodologies have emerged that have enabled sophisticated data analysis by simple and straightforward use of machine learning algorithms. This paper presents a comprehensive comparative study of four algorithms performance on a medical record each yielding an accuracy up to 95 percent. The performance is analyzed through confusion matrix and accuracy score.

4) References

1. Min Chen, Yixue Hao et.al "*Disease Prediction by Machine Learning over big data from Healthcare Communities*", IEEE[Access 2017]
2. Sayantan Saha, Argha Roy Chowdhuri et,al "*Web Based Disease Detection System*",IJERT, ISSN:22780181,Vol.2 Issue 4, April-2013

5. Block Diagram



6. Modules

Module 1 : Training dataset processing

A list is made to contain the different symptoms from the given input dataset and stored in l1. Then a list of diseases is listed in list disease. Create a vacant list L2, and append a number of zeroes equivalent to a number of diseases in list L1. Creating a dataframe df to read the training.csv file which stores the different diseases from the csv file.
Now, we replace the values in the imported file by pandas by the inbuilt function replace in pandas.

Input : Input dataset - training.csv
Output : List l1, Training dataset, dataframe df

Module 2 : Plotting and Visualizing of Training dataset

Plotting distribution graphs (histogram/bar graph) of column data for the symptoms for training dataset. Plotting Scatter and density plots for the symptoms for training dataset. Extracting diseases and symptoms from the training dataframe

Input : dataframe df
Output : Distribution graph, Scatter and Density plot of training dataset.

Module 3 : Test dataset processing

Creating dataframe from test dataset by reading the testing.csv file and replacing the imported values.

Input : Test dataset
Output : Testing dataframe of the test dataset

Module 4 : Plotting Test Dataset

Extracting disease and symptoms from testing dataframe. Plotting distribution graphs for the symptoms for testing dataset. Plotting Scatter and density plots for the symptoms for testing dataset.

Input : Testing dataframe
Output : Tested Model

Module 5 : User dataset processing

Creating a dataframe from a patient dataset.

Input : Patient dataset containing patient id and symptoms.
Output : Dataframe of user dataset

Module 6 : Prediction of accurate disease

1. **Decision Tree algorithm** - In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

   This algorithm works on 5 basic steps -

   Step 1: Begin the tree with the root node, says S, which contains the complete dataset.

   Step 2 : Find the best attribute in the dataset using Attribute Selection Measure (ASM).

   Step 3 : Divide the S into subsets that contain possible values for the best attributes.

   Step 4 : Generate the decision tree node, which contains the best attribute.

   Step 5 : Recursively make new decision trees using the subsets of the dataset created in step 3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

   Input : Dataframe of user dataset.
   Output : Predicted Disease following decision tree algorithm.

2. **Random Forest algorithm** - This algorithm works in two-phase: first is to create the random forest by combining the N decision tree, and second is to make predictions for each tree created in the first phase.
   This algorithm works on 4 basic steps –

   Step 1 : It chooses random data samples from the dataset.

   Step 2 : It constructs decision trees for every sample dataset chosen.

   Step 3 : At this step every predicted result will be compiled and voted on.

   Step 4 : At last, the most voted prediction will be selected and presented as a result of classification.

   Input : Dataframe of user dataset.
   Output : Predicted Disease following random forest algorithm.

3. **K Nearest Neighbour algorithm** - It finds extensive use in pattern finding and data mining. It works by finding a pattern in data which links data to results and it improves upon the pattern recognition with every iteration.
This algorithm works on 5 basic steps –
Step 1 : It handles the data from the dataset
Step 2 : Calculates the distance
Step 3 : Find k nearest point
Step 4 : Predict the class
Step 5 : Check the accuracy

Input : Dataframe of user dataset.
Output : Predicted Disease following K Nearest Neighbour algorithm.

4. **Naïve Bayes algorithm** - They share a common principle that every pair of predictions is independent of each other. It also makes an assumption that features make an independent and equal contribution to the prediction.
This algorithm works on 5 basic steps –

Step 1 : Separate By Class.

Step 2 : Summarize Dataset.

Step 3 : Summarize Data By Class.

Step 4 : Gaussian Probability Density Function.

Step 5 : Class Probabilities

Input : Dataframe of user dataset.
Output : Predicted Disease following Naive Bayes algorithm.

Module 7 : Final prediction

- Final prediction of accurate disease is found after comparing the disease predicted by each of the four algorithms for each patient.
  Input : Predicted Disease by all the 4 algorithms.
  Output : File containing final predicted disease for each patient.

7. Dataset Description

The dataset we have considered is a sample data of 4920 patients' records consisting of 132 symptoms, the combination or permutations of which leads to 41 diseases is selected for analysis. A dependent variable was composed of 41 diseases. 95 of 132 independent variables(symptoms) closely related to diseases were selected and optimized.
The training dataset consists of mainly two columns "Disease" and "Symptoms" but this dataset is pre-processed so it helps in easily classifying the data. This dataset is used to train our model. Whereas the testing dataset will be used to test our developed system so that we can know the accuracy of our model.

8. Results Implementation

Under Module 1

```
[1]  from mpl_toolkits.mplot3d import Axes3D
     from sklearn.preprocessing import StandardScaler
     from sklearn.feature_extraction.text import CountVectorizer
     import matplotlib.pyplot as plt
     from tkinter import *
     import numpy as np
     import pandas as pd
     import collections
     import os
```

These are the imported libraries that are utilized to use various tools that are available in that specific libraries.

```
[2]  l1=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
         'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
         'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',
         'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_in_limbs',
         'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','bloody_stool',
         'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesity','swollen_legs',
         'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittle_nails',
         'swollen_extremeties','excessive_hunger','extra_marital_contacts','drying_and_tingling_lips',
         'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_neck','swelling_joints',
         'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness',
         'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_smell_of urine',
         'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_look_(typhos)',
         'depression','irritability','muscle_pain','altered_sensorium','red_spots_over_body','belly_pain',
         'abnormal_menstruation','dischromic _patches','watering_from_eyes','increased_appetite','polyuria','family_history','mucoid_sputum',
         'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
         'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',
         'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_calf',
         'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',
         'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose',
         'yellow_crust_ooze']
```

L1 is the list made for various Symptoms which are generally shown in people for various Diseases.

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

Disease is the list made for different Diseases which are for the most part appeared in different individuals.

```python
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

First L2 is the vacant list. At that point, equivalent to a number of diseases in list L1, L2 is appended to a number of zeroes.

```python
d=pd.read_csv("/root/training.csv")
d.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
df = d.select_dtypes(include =[np.number])
df.head()
```

There is a CSV document containing diseases and symptoms, named training.csv, which is utilized to prepare the model. Read_csv() function is utilized to store the information in the dataframe, named df. Utilizing the replace() function, the prognosis column for the different diseases, it is replaced by the numbers from 0 to n-1, where n is the number of different diseases present in the .csv record. d.select_dtypes function is used to select the numerical rows and head() function is utilized to print the initial five rows of the preparation dataframe.

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	muscle_wasting
0	1	1	1	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

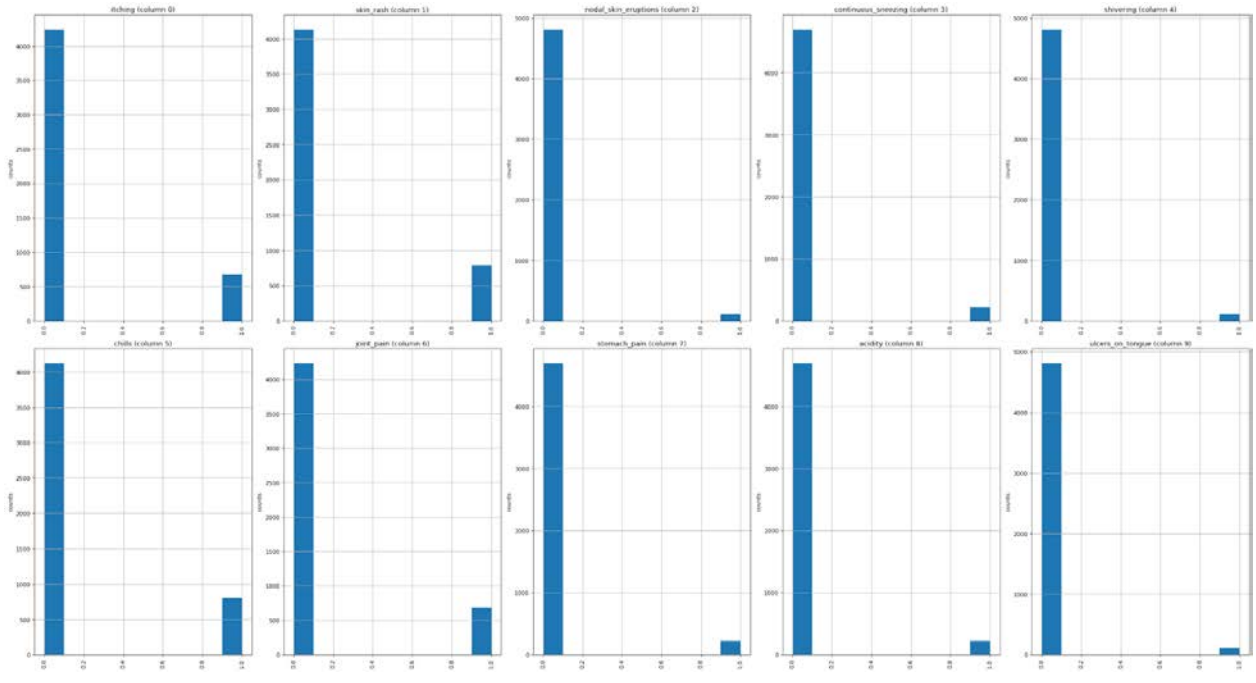
5 rows × 132 columns

This is the output produced which contains the initial five rows of the dataframe df.

Under Module 2

```
def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]]
    nRow, nCol = df1.shape
    columnNames = list(df1)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i+1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
plotPerColumnDistribution(df,10,5)
```

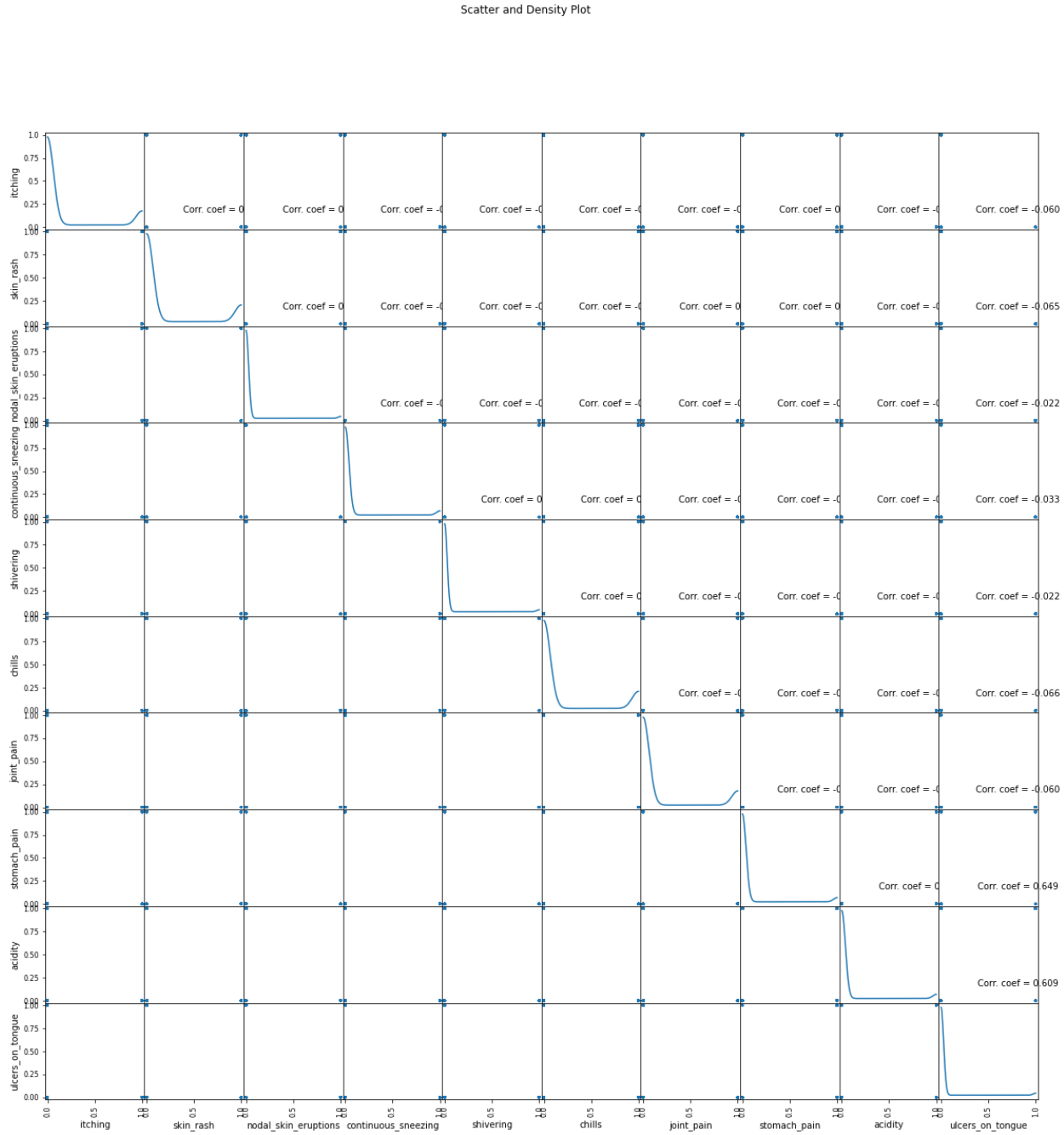
The code is used to display the distribution graph of the columns of the training.csv file.



Output for the distribution graph of the columns of the training.csv file.

```
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df1 = df1[columnNames]
    ax = pd.plotting.scatter_matrix(df1, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df1.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
plotScatterMatrix(df, 20, 10)
```

The code above is used to plot the ScatterMatrix and Density Plot of the training.csv file.



Output for the ScatterMatrix plot of the columns of the training.csv file.

```
[49] X= d[11]
     y = d[["Prognosis"]]
     np.ravel(y)
     print(X)
```

```
      back_pain  constipation  ...  red_sore_around_nose  yellow_crust_ooze
0             0             0  ...                     0                  0
1             0             0  ...                     0                  0
2             0             0  ...                     0                  0
3             0             0  ...                     0                  0
4             0             0  ...                     0                  0
...         ...           ...  ...                   ...                ...
4915          0             0  ...                     0                  0
4916          0             0  ...                     0                  0
4917          0             0  ...                     0                  0
4918          0             0  ...                     0                  0
4919          0             0  ...                     1                  1

[4920 rows x 95 columns]
```

Putting the Symptoms in X and prognosis/diseases in y for training the model.
Output for the print(X) in which different symptoms have the values ‘0’ or ‘1’ according to their presence in the particular diseases

```
print(y)
```

```
                                    Prognosis
0                           Fungal infection
1                           Fungal infection
2                           Fungal infection
3                           Fungal infection
4                           Fungal infection
...                                      ...
4915  (vertigo) Paroymsal  Positional Vertigo
4916                                    Acne
4917                 Urinary tract infection
4918                                Psoriasis
4919                                 Impetigo

[4920 rows x 1 columns]
```

Output for the print(y) in which different diseases have values according to their symptoms.

Under Module 3

```
[ ] tr=pd.read_csv("/root/testing.csv")
    tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
        'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,
        'Migraine':11,'Cervical spondylosis':12,
        'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
        'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,
        'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,
        'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
        '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,
        'Impetigo':40}},inplace=True)
    tr.head()
```

There is a CSV document containing diseases and symptoms, named testing.csv, which is utilized to test the model prepared. Read_csv() function is utilized to store the information in the dataframe, named *tr*. Utilizing the replace() function, the prognosis column for the different diseases, it is replaced by the numbers from 0 to n-1, where n is the number of different diseases present in the .csv record and head() function is utilized to print the initial five rows of the preparation dataframe.

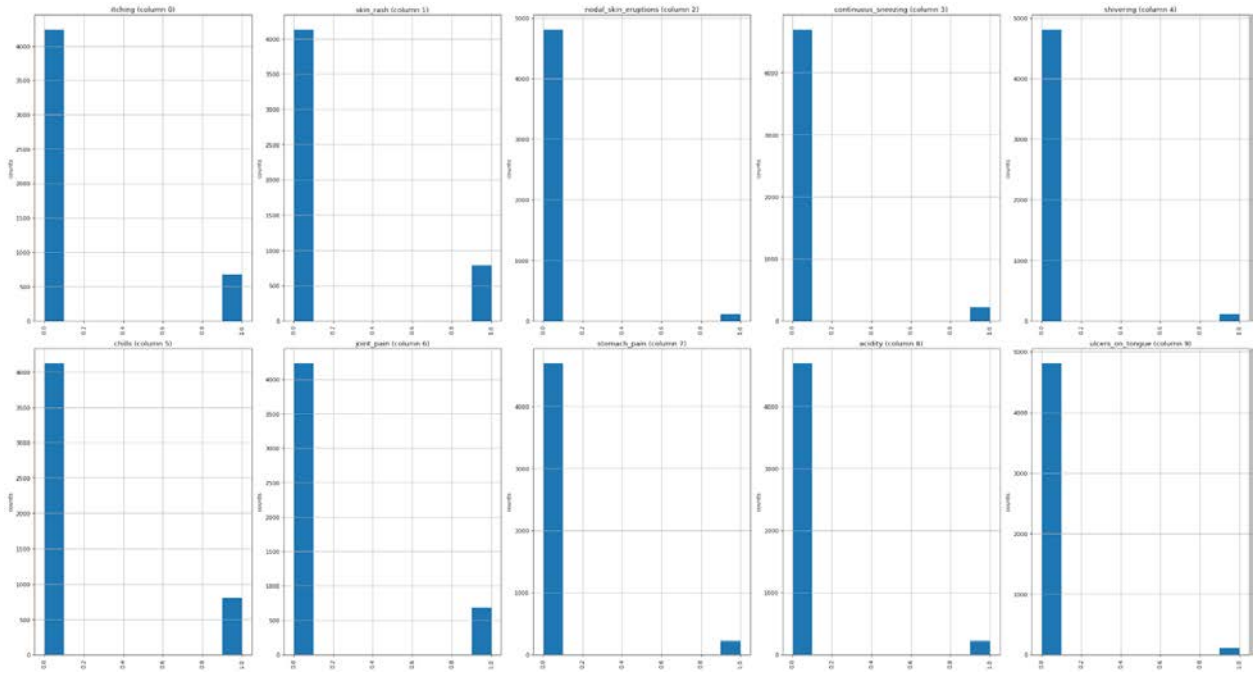
	Prognosis	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tong
0	Alcoholic hepatitis	0	0	0	0	0	0	0	0	0	
1	Tuberculosis	0	0	0	0	0	1	0	0	0	
2	Common Cold	0	0	0	1	0	1	0	0	0	
3	Pneumonia	0	0	0	0	0	1	0	0	0	
4	Dimorphic hemmorhoids(piles)	0	0	0	0	0	0	0	0	0	

5 rows × 133 columns

The initial five rows of the prepared dataframe *tr* is printed.

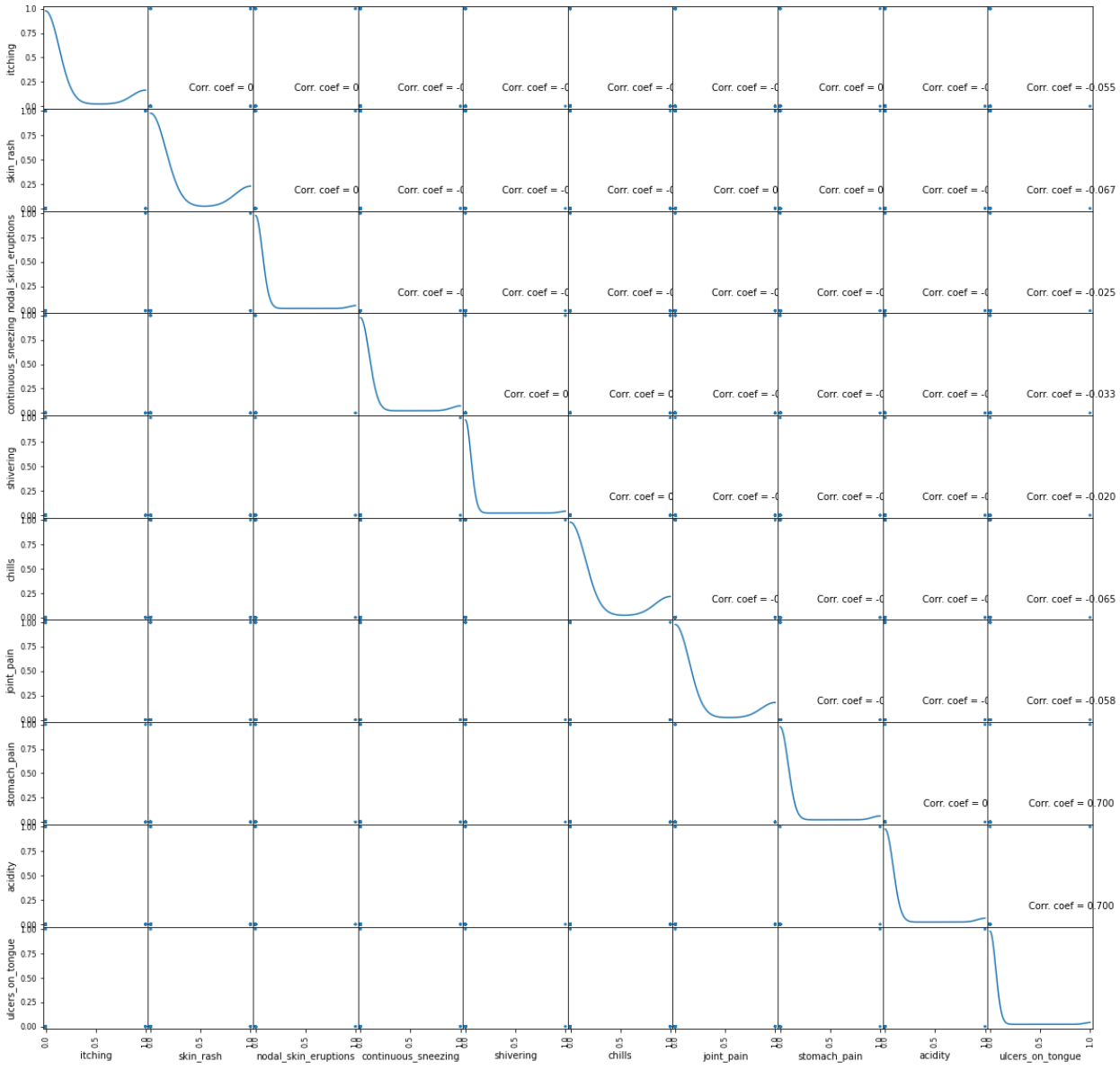
Under Module 4

```
plotPerColumnDistribution(tr, 10, 5)
```



Output for the distribution graph of the columns of the testing.csv file.

```
[ ]  plotScatterMatrix(tr, 20, 10)
```



Output for the ScatterMatrix plot of the columns of the testing.csv file.

```
[ ]  X_test= tr[11]
     y_test = tr[["Prognosis"]]
     np.ravel(y_test)
     print(X_test)
```

```
    back_pain  constipation  ...  red_sore_around_nose  yellow_crust_ooze
0           0             0  ...                     0                  0
1           0             0  ...                     0                  0
2           0             0  ...                     0                  0
3           0             0  ...                     0                  0
4           0             1  ...                     0                  0
..        ...           ...  ...                   ...                ...
95          0             0  ...                     0                  0
96          0             0  ...                     0                  0
97          0             0  ...                     0                  0
98          0             0  ...                     1                  1
99          0             0  ...                     0                  0

[100 rows x 95 columns]
```

Putting the Symptoms in X_test and prognosis in y_test for training the model.
Output for the print(X_test) in which different symptoms have the values ‘0’ or ‘1’ according to their presence in the particular diseases

```
[ ]  print(y_test)
```

```
                        Prognosis
0              Alcoholic hepatitis
1                     Tuberculosis
2                      Common Cold
3                        Pneumonia
4     Dimorphic hemmorhoids(piles)
..                             ...
95                            Acne
96         Urinary tract infection
97                       Psoriasis
98                        Impetigo
99                Fungal infection

[100 rows x 1 columns]
```

Output for the print(y_test) in which different diseases have values according to their symptoms.

Under Module 5

```
[ ]  ud=pd.read_csv("/root/patientrecords.csv")
     ud.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
         'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,'Hypertension ':10,
         'Migraine':11,'Cervical spondylosis':12,
         'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
         'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,'Tuberculosis':25,
         'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,
         'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
         '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract infection':38,'Psoriasis':39,
         'Impetigo':40}},inplace=True)
     ud.head()
```

Here a CSV document, named patientrecords.csv, is read to store the information in the dataframe, named *ud*. Utilizing the replace() function, the prognosis column for the different diseases, it is replaced by the numbers from 0 to n-1, where n is the number of different diseases present in the .csv record and head() function is utilized to print the initial five rows of the preparation dataframe.

	Patient id	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	musc
0	1	1	1	1	0	0	0	0	0	0	0	
1	2	0	0	0	1	1	1	0	0	0	0	
2	3	0	0	0	0	0	0	0	1	1	1	
3	4	1	0	0	0	0	0	0	0	0	0	
4	5	1	1	0	0	0	0	0	1	0	0	

5 rows × 133 columns

The initial five rows of the prepared dataframe *ud* is printed.

Under Module 6

To build the precision of the model, we utilized four distinctive algorithms which are as per the following

- Decision Tree algorithm
- Random Forest algorithm
- KNearestNeighbour algorithm
- Naive Bayes algorithm

Decision Tree Algorithm :

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

We import tree (decision tree classifier) along with certain selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

```
[16] def DecisionTree():
         clf1 = tree.DecisionTreeClassifier()
         clf1 = clf1.fit(X.values,y)
         y_pred=clf1.predict(X_test.values)
         print("Decision Tree")
         print("Accuracy")
         print(accuracy_score(y_test, y_pred))
         print(accuracy_score(y_test, y_pred,normalize=False))

         temp = ud.copy()
         dat = ['Patient_id','Decision_Tree_Prediction']
         with open('/root/output.csv', 'w', newline='') as write_obj:
             csv_writer = csv.writer(write_obj)
             csv_writer.writerow(dat)
         for it in temp.values.tolist():
             predict = clf1.predict([it[1:]])
             predicted=predict[0]
             h='no'
             for a in range(0,len(disease)):
                 if(predicted == a):
                     h='yes'
                     break
             if (h=='yes'):
               dat = [it[0],disease[a]]
               with open('/root/output.csv', 'a', newline='') as write_obj:
                 csv_writer = csv.writer(write_obj)
                 csv_writer.writerow(dat)
             else:
               dat = [it[0],'Not Found']
               with open('/root/output.csv', 'a', newline='') as write_obj:
                 csv_writer = csv.writer(write_obj)
                 csv_writer.writerow(dat)
     DecisionTree()
     #Accuracy-File Change
```

```
Decision Tree
Accuracy
1.0
100
```

DecisionTreeClassifier - Creates a decision tree classifier object,
clf.fit(X_train,y_train) – Trains the decision Tree Classifier,
y_pred – predicts the response for test dataset.
Accuracy can be computed by comparing actual test set values and predicted values.

Then we open a new csv file (output.csv) and write down the results for each patient id with their corresponding Decision Tree Prediction.

Random Forest Algorithm:

```
[17] from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
     import csv
```

We import RandomForest Classifier along with certain selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

```
[18] def randomforest():
            clf2 = RandomForestClassifier(n_estimators=100)
            clf2 = clf2.fit(X.values,np.ravel(y))
            # calculating accuracy
            y_pred=clf2.predict(X_test.values)
            print("Random Forest")
            print("Accuracy")
            print(accuracy_score(y_test, y_pred))
            print(accuracy_score(y_test, y_pred,normalize=False))

            new_row_list = []
            row_list = []
            temp = ud.copy()
            with open('/root/output.csv', mode = 'r') as read_obj:
                csv_reader = csv.reader(read_obj)
                for row in csv_reader:
                  row_list.append(row)
            i=0
            new_row_list.append(row_list[i]+['Random_Forest_Classifier'])
            i=i+1
            for it in temp.values.tolist():
                predict = clf2.predict([it[1:]])
                predicted=predict[0]
                h='no'
                for a in range(0,len(disease)):
                    if(predicted == a):
                        h='yes'
                        break
                if (h=='yes'):
                  new_row_list.append(row_list[i]+[disease[a]])
                else:
                  new_row_list.append(row_list[i]+['Not Found'])
                i=i+1
            with open('/root/output.csv', 'w', newline='') as write_obj:
               csv_writer = csv.writer(write_obj)
               csv_writer.writerows(new_row_list)
     randomforest()
```

```
Random Forest
Accuracy
1.0
100
```

RandomForestClassifier(n_estimators = 100) – Creates a random forest classifier object,
clf2.fit(X_train, y_train) – This function is used to train the model using the training sets as parameters
y_pred = clf.predict(X_test) - predicts the response for test dataset.
Accuracy can be computed by comparing actual test set values and predicted values.

Then we open the created csv file (output.csv) and append the results for each patient id with their corresponding Random Forest Tree Prediction.

KNearestNeighbour Algorithm :

```
[19] from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
     import csv
```

We import KNeighbourClassifier along with certain selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

```
[20] def KNN():
            clf3=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
            clf3=clf3.fit(X.values,np.ravel(y))
            y_pred=clf3.predict(X_test.values)
            print("KNN")
            print("Accuracy")
            print(accuracy_score(y_test, y_pred))
            print(accuracy_score(y_test, y_pred,normalize=False))

            new_row_list = []
            row_list = []
            temp = ud.copy()
            with open('/root/output.csv', mode = 'r') as read_obj:
                csv_reader = csv.reader(read_obj)
                for row in csv_reader:
                  row_list.append(row)
            i=0
            new_row_list.append(row_list[i]+['KNN_Classifier'])
            i=i+1
            for it in temp.values.tolist():
                predict = clf3.predict([it[1:]])
                predicted=predict[0]
                h='no'
                for a in range(0,len(disease)):
                    if(predicted == a):
                        h='yes'
                        break
                if (h=='yes'):
                  new_row_list.append(row_list[i]+[disease[a]])
                else:
                  new_row_list.append(row_list[i]+['Not Found'])
                i=i+1
            with open('/root/output.csv', 'w', newline='') as write_obj:
               csv_writer = csv.writer(write_obj)
               csv_writer.writerows(new_row_list)
     KNN()
```

```
KNN
Accuracy
1.0
100
```

Create KNN classifier object by passing argument number of neighbors in KNeighborsClassifier() function,
clf3.fit(features,label) – This function is used to train the model using the training sets as parameters
y_pred = clf.predict(X_test.values) - predicts the response for test dataset.
Accuracy can be computed by comparing actual test set values and predicted values.

Then we open the created csv file (output.csv) and append the results for each patient id with their corresponding KNN classifier algorithm.

Naïve Bayes Algorithm :

```
[21] from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
     import csv
```

We import GaussianNB (naïve bayes classifier) along with certain selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

```
[22] def NaiveBayes():
            clf4 = GaussianNB()
            clf4=clf4.fit(X.values,np.ravel(y))
            y_pred=clf4.predict(X_test.values)
            print("Naive Bayes")
            print("Accuracy")
            print(accuracy_score(y_test, y_pred))
            print(accuracy_score(y_test, y_pred,normalize=False))

            new_row_list = []
            row_list = []
            temp = ud.copy()
            with open('/root/output.csv', mode = 'r') as read_obj:
                csv_reader = csv.reader(read_obj)
                for row in csv_reader:
                  row_list.append(row)
            i=0
            new_row_list.append(row_list[i]+['NaiveBayes_Classifier'])
            i=i+1
            for it in temp.values.tolist():
                predict = clf4.predict([it[1:]])
                predicted=predict[0]
                h='no'
                for a in range(0,len(disease)):
                    if(predicted == a):
                        h='yes'
                        break
                if (h=='yes'):
                  new_row_list.append(row_list[i]+[disease[a]])
                else:
                  new_row_list.append(row_list[i]+['Not Found'])
                i=i+1
            with open('/root/output.csv', 'w', newline='') as write_obj:
               csv_writer = csv.writer(write_obj)
               csv_writer.writerows(new_row_list)
     NaiveBayes()
```

```
Naive Bayes
Accuracy
1.0
100
```

GaussianNB() function – Creates a gaussian classifier object,
clf4.fit(features,label) – This function is used to train the model using the training sets as parameters,
y_pred = clf4.predict(X_test.values) - predicts the response for test dataset.
Accuracy can be computed by comparing actual test set values and predicted values.

Then we open the created csv file (output.csv) and append the results for each patient id with their corresponding NaiveBayes classifier algorithm.

Under Module 7

```
[44] def most_frequent(List):
         dict = {}
         count, itm = 1, ''
         for item in reversed(List):
             dict[item] = dict.get(item, 0) + 1
             if dict[item] >= count :
                 count, itm = dict[item], item
         return(itm)
     def FinalDecision():
             new_row_list = []
             row_list = []
             with open('/root/output.csv', mode = 'r') as read_obj:
                 csv_reader = csv.reader(read_obj)
                 for row in csv_reader:
                   row_list.append(row)
             new_row_list.append(row_list[0]+['Final_Decision'])
             i=1
             for it in row_list:
                 if(it[0]!="Patient_id"):
                     new_row_list.append(row_list[i]+[most_frequent(it)])
                     i=i+1
             with open('/root/output.csv', 'w', newline='') as write_obj:
                csv_writer = csv.writer(write_obj)
                csv_writer.writerows(new_row_list)
     FinalDecision()
```

We are using python dictionary to save element as a key and its frequency as the value, and thus find the most frequent element.
Then we define the function FinalDecision() ,create two lists ,open output.csv file to read using csv_reader and append our final decision after comparing the results obtained by the four distinct algorithms.

output.csv

1 to 50 of 50 entries Filter

Patient_id	Decision_Tree_Prediction	Random_Forest_Classifier	KNN_Classifier	NaiveBayes_Classifier	Final_Decision
1	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
2	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
3	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
4	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
5	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
6	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
7	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
8	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
9	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
10	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection
11	Allergy	Allergy	Allergy	Allergy	Allergy
12	Allergy	Allergy	Allergy	Allergy	Allergy
13	Allergy	Allergy	Allergy	Allergy	Allergy
14	Allergy	Allergy	Allergy	Allergy	Allergy
15	Allergy	Allergy	Allergy	Allergy	Allergy
16	Allergy	Allergy	Allergy	Allergy	Allergy
17	Allergy	Allergy	Allergy	Allergy	Allergy
18	Allergy	Allergy	Allergy	Allergy	Allergy
19	Allergy	Allergy	Allergy	Allergy	Allergy
20	Allergy	Allergy	Allergy	Allergy	Allergy
21	GERD	GERD	GERD	GERD	GERD
22	GERD	GERD	GERD	GERD	GERD
23	GERD	GERD	GERD	GERD	GERD
24	GERD	GERD	GERD	GERD	GERD
25	GERD	GERD	GERD	GERD	GERD
26	GERD	GERD	GERD	GERD	GERD
27	GERD	GERD	GERD	GERD	GERD
28	GERD	GERD	GERD	GERD	GERD
29	GERD	GERD	GERD	GERD	GERD
30	GERD	GERD	GERD	GERD	GERD
31	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis
32	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis
33	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis
34	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis
35	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis	Chronic cholestasis

The final output obtained is found in output.csv.