

prediction

April 27, 2024

1 Disease Prediction based on Symtoms

```
[1]: #Importing Libraries
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from tkinter import *
import numpy as np
import pandas as pd
import os
```

```
[2]: #List of the symptoms is listed here in list l1.

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_
      'abnormal_menstruation','dischromic_
    ↳_patches','watering_from_eyes','increased_appetite','polyuria','family_history','mucoid_spu
    ↳
    ↳'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_cal
    ↳
    ↳'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling
    ↳
    ↳'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around
      'yellow_crust_ooze']

```

[3]: *#List of Diseases is listed in list disease.*

```

disease=['Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis',
        'Drug Reaction', 'Peptic ulcer disease', '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)', 'Heart attack', 'Varicose veins',
        'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
        'Osteoarthritis', 'Arthritis',
        '(vertigo) Parosymal Positional Vertigo', 'Acne',
        'Urinary tract infection', 'Psoriasis', 'Impetigo']

#disease = [df['prognosis'].unique()]
#print(disease)

```

[4]:

```

12=[]
for i in range(0,len(11)):
    12.append(0)
print(12)

```

```

[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, 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, 0, 0]

```

[5]: *#Reading the training .csv file*

```

df=pd.read_csv("training.csv")
DF= pd.read_csv('training.csv', index_col='prognosis')

```

```
#Replace the values in the imported file by pandas by the inbuilt function
↳replace in pandas.

df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic
↳cholestasis':3,'Drug Reaction':4,
    'Peptic ulcer disease':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,'Osteoarthritis':34,'Arthritis':35,
    '(vertigo) Paroymsal Positional Vertigo':36,'Acne':37,'Urinary tract
↳infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
#df.head()
DF.head()
```

```
[5]:          itching  skin_rash  nodal_skin_eruptions  \
prognosis
Fungal infection      1         1                   1
Fungal infection      0         1                   1
Fungal infection      1         0                   1
Fungal infection      1         1                   0
Fungal infection      1         1                   1

          continuous_sneezing  shivering  chills  joint_pain  \
prognosis
Fungal infection              0         0      0           0
Fungal infection              0         0      0           0
Fungal infection              0         0      0           0
Fungal infection              0         0      0           0
Fungal infection              0         0      0           0

          stomach_pain  acidity  ulcers_on_tongue  ...  \
prognosis
Fungal infection          0         0              0  ...
Fungal infection          0         0              0  ...
Fungal infection          0         0              0  ...
Fungal infection          0         0              0  ...
Fungal infection          0         0              0  ...

          pus_filled_pimples  blackheads  scurring  skin_peeling  \
```

prognosis				
Fungal infection	0	0	0	0
Fungal infection	0	0	0	0
Fungal infection	0	0	0	0
Fungal infection	0	0	0	0
Fungal infection	0	0	0	0

	silver_like_dusting	small_dents_in_nails	\
prognosis			
Fungal infection	0	0	
Fungal infection	0	0	
Fungal infection	0	0	
Fungal infection	0	0	
Fungal infection	0	0	

	inflammatory_nails	blister	red_sore_around_nose	\
prognosis				
Fungal infection	0	0	0	
Fungal infection	0	0	0	
Fungal infection	0	0	0	
Fungal infection	0	0	0	
Fungal infection	0	0	0	

	yellow_crust_ooze
prognosis	
Fungal infection	0
Fungal infection	0
Fungal infection	0
Fungal infection	0
Fungal infection	0

[5 rows x 132 columns]

```
[6]: # Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df1 if nunique[col] > 1 and nunique[col] < 50]] #
    ↪For displaying purposes, pick columns that have between 1 and 50 unique
    ↪values
    nRow, nCol = df1.shape
    columnNames = list(df1)
    nGraphRow = round((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 = df1.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()

```

```

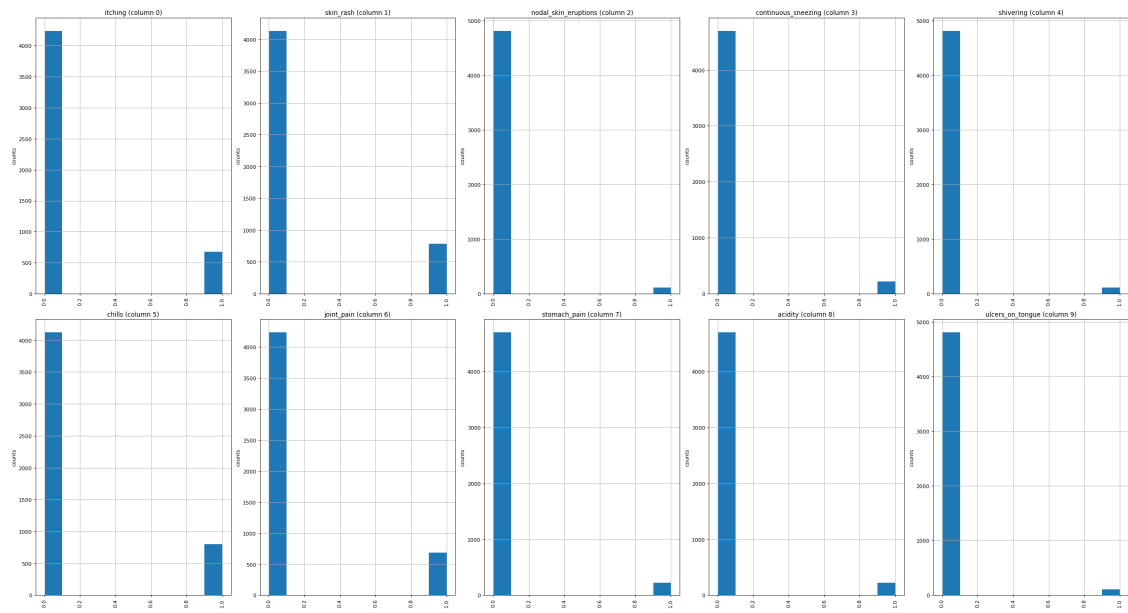
[7]: # Scatter and density plots
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna(axis='columns')
    df1 = df1[[col for col in df if 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()

```

```

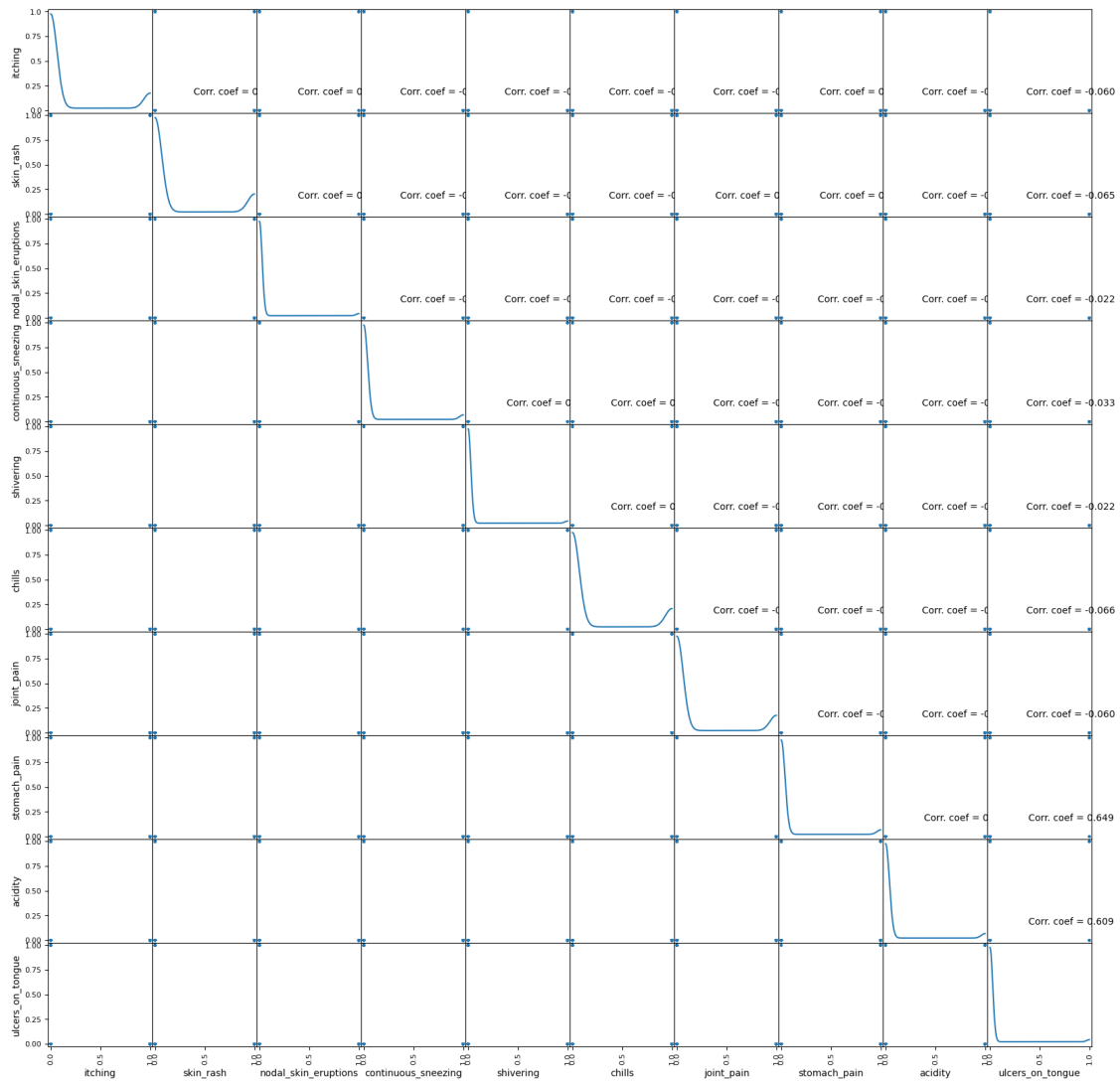
[8]: plotPerColumnDistribution(df, 10, 5)

```



```
[9]: plotScatterMatrix(df, 20, 10)
```

Scatter and Density Plot



```
[10]: X= df[l1]
      y = df[["prognosis"]]
      np.ravel(y)
      print(X)
```

	back_pain	constipation	abdominal_pain	diarrhoea	mild_fever	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	

4	0	0	0	0	0
...
4915	0	0	0	0	0
4916	0	0	0	0	0
4917	0	0	0	0	0
4918	0	0	0	0	0
4919	0	0	0	0	0

	yellow_urine	yellowing_of_eyes	acute_liver_failure	fluid_overload	\
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
...
4915	0	0	0	0	0
4916	0	0	0	0	0
4917	0	0	0	0	0
4918	0	0	0	0	0
4919	0	0	0	0	0

	swelling_of_stomach	...	pus_filled_pimples	blackheads	scurring	\
0	0	...	0	0	0	0
1	0	...	0	0	0	0
2	0	...	0	0	0	0
3	0	...	0	0	0	0
4	0	...	0	0	0	0
...
4915	0	...	0	0	0	0
4916	0	...	1	1	1	1
4917	0	...	0	0	0	0
4918	0	...	0	0	0	0
4919	0	...	0	0	0	0

	skin_peeling	silver_like_dusting	small_dents_in_nails	\
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	1	1	1	1
4919	0	0	0	0

inflammatory_nails blister 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	1	0	0	0
4919	0	1	1	1

[4920 rows x 95 columns]

```
[11]: print(y)
```

prognosis	
0	0
1	0
2	0
3	0
4	0
...	...
4915	36
4916	37
4917	38
4918	39
4919	40

[4920 rows x 1 columns]

```
[12]: #Reading the testing.csv file
tr=pd.read_csv("testing.csv")

#Using inbuilt function replace in pandas for replacing the values

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic_
↪cholestasis':3,'Drug Reaction':4,
    'Peptic ulcer disease':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,'Osteoarthritis':34,'Arthritis':35,
    '(vertigo) Parosmal Positional Vertigo':36,'Acne':37,'Urinary tract_
infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
tr.head()

```

```

[12]:  itching  skin_rash  nodal_skin_eruptions  continuous_sneezing  shivering  \
0         1         1                   1                   0         0
1         0         0                   0                   1         1
2         0         0                   0                   0         0
3         1         0                   0                   0         0
4         1         1                   0                   0         0

      chills  joint_pain  stomach_pain  acidity  ulcers_on_tongue  ...  \
0         0         0         0         0         0         0  ...
1         1         0         0         0         0         0  ...
2         0         0         1         1         1         1  ...
3         0         0         0         0         0         0  ...
4         0         0         1         0         0         0  ...

      blackheads  scurring  skin_peeling  silver_like_dusting  \
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

      small_dents_in_nails  inflammatory_nails  blister  red_sore_around_nose  \
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

      yellow_crust_ooze  prognosis
0         0         0
1         0         1
2         0         2
3         0         3
4         0         4

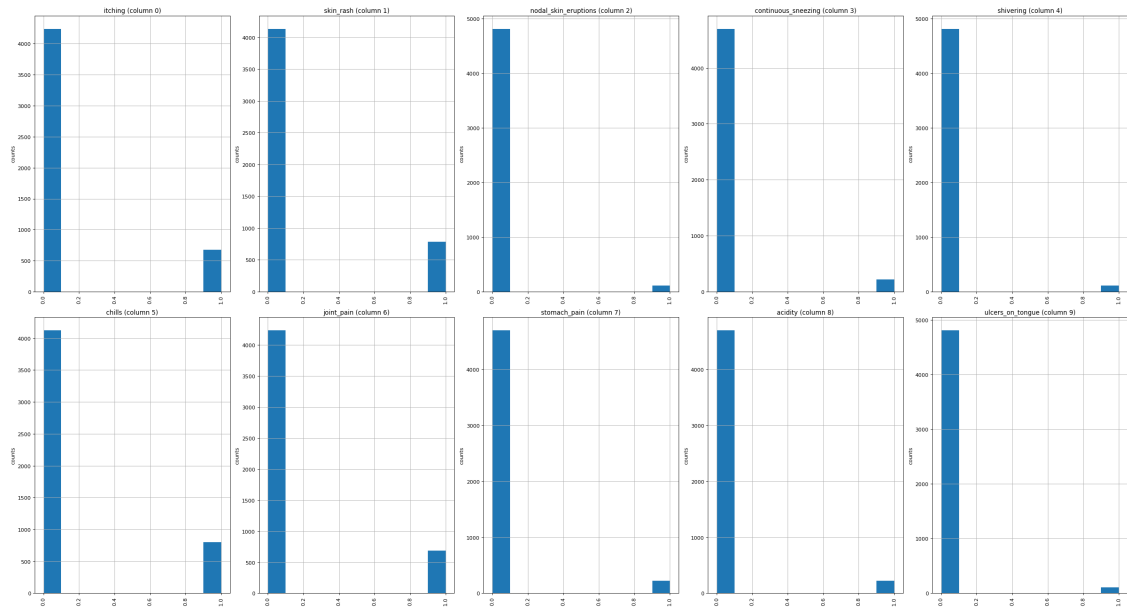
[5 rows x 133 columns]

```

```

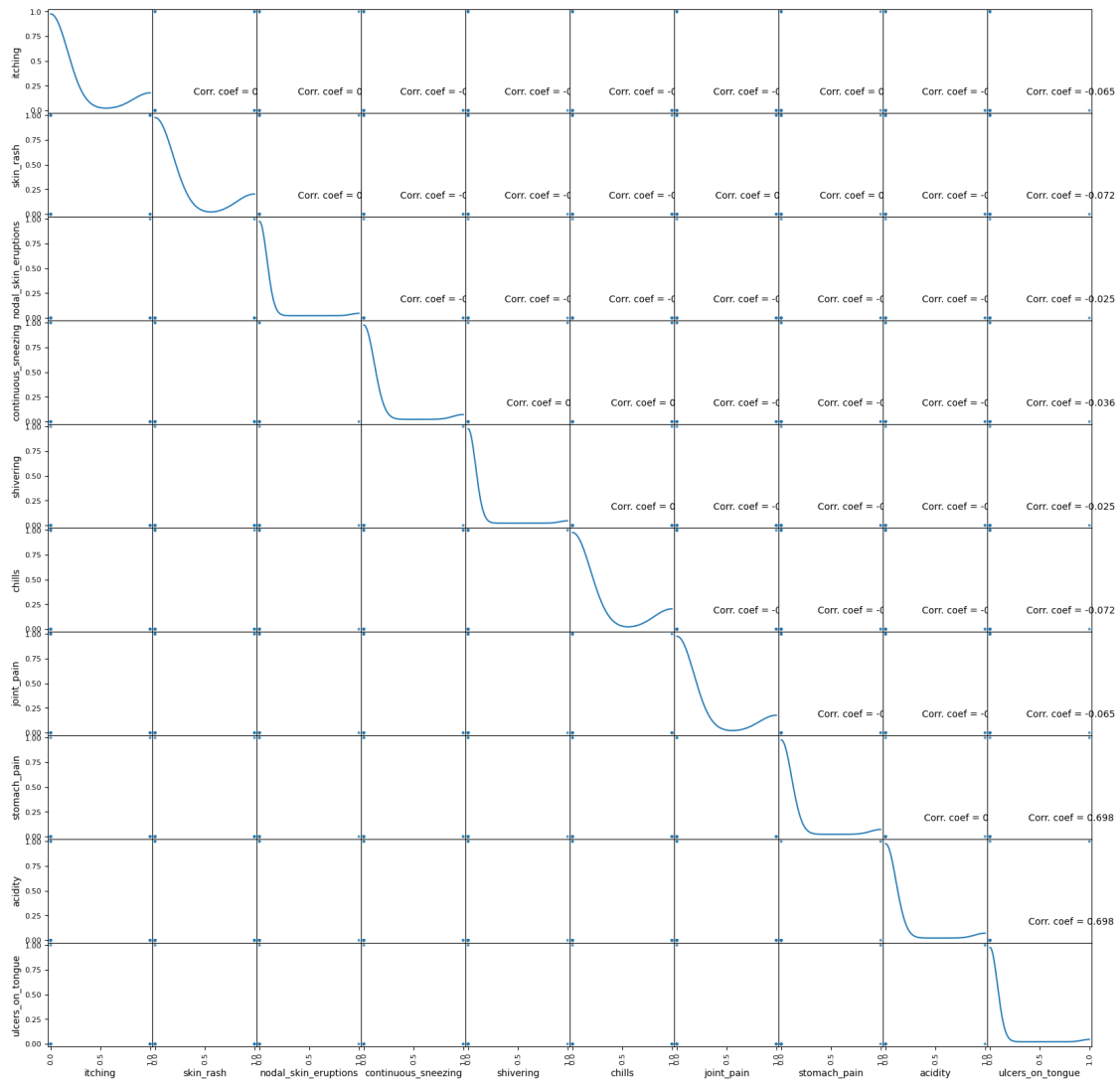
[13]: plotPerColumnDistribution(tr, 10, 5)

```



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

Scatter and Density Plot



```
[15]: X_test= tr[11]
      y_test = tr[["prognosis"]]
      np.ravel(y_test)
      print(X_test)
```

	back_pain	constipation	abdominal_pain	diarrhoea	mild_fever	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	1	0	0	

4	0	0	0	0	0
5	0	0	1	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	1	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	1	0	0	0	0
13	0	0	0	0	0
14	0	0	1	0	0
15	0	0	0	1	0
16	0	0	0	0	1
17	1	0	0	0	0
18	0	1	1	1	0
19	0	0	1	1	1
20	0	0	1	0	0
21	0	0	0	0	0
22	0	0	1	0	0
23	0	0	1	0	0
24	0	0	1	0	0
25	0	0	0	0	1
26	0	0	0	0	0
27	0	0	0	0	0
28	0	1	0	0	0
29	0	0	0	0	0
30	0	0	0	0	0
31	0	0	0	0	0
32	0	0	0	1	0
33	0	0	0	0	0
34	0	0	0	0	0
35	0	0	0	0	0
36	0	0	0	0	0
37	0	0	0	0	0
38	0	0	0	0	0
39	0	0	0	0	0
40	0	0	0	0	0

	yellow_urine	yellowing_of_eyes	acute_liver_failure	fluid_overload	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	1	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	
8	0	0	0	0	

9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	1	0	0
20	1	1	0	0
21	0	1	0	0
22	0	1	0	0
23	0	1	1	0
24	0	0	0	0
25	0	1	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
34	0	0	0	0
35	0	0	0	0
36	0	0	0	0
37	0	0	0	0
38	0	0	0	0
39	0	0	0	0
40	0	0	0	0

	swelling_of_stomach	...	pus_filled_pimples	blackheads	scurring	\
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	
5	0	...	0	0	0	
6	0	...	0	0	0	
7	0	...	0	0	0	
8	0	...	0	0	0	
9	0	...	0	0	0	
10	0	...	0	0	0	
11	0	...	0	0	0	
12	0	...	0	0	0	
13	0	...	0	0	0	

14	0 ...	0	0	0
15	0 ...	0	0	0
16	0 ...	0	0	0
17	0 ...	0	0	0
18	0 ...	0	0	0
19	0 ...	0	0	0
20	0 ...	0	0	0
21	0 ...	0	0	0
22	0 ...	0	0	0
23	0 ...	0	0	0
24	1 ...	0	0	0
25	0 ...	0	0	0
26	0 ...	0	0	0
27	0 ...	0	0	0
28	0 ...	0	0	0
29	0 ...	0	0	0
30	0 ...	0	0	0
31	0 ...	0	0	0
32	0 ...	0	0	0
33	0 ...	0	0	0
34	0 ...	0	0	0
35	0 ...	0	0	0
36	0 ...	0	0	0
37	0 ...	1	1	1
38	0 ...	0	0	0
39	0 ...	0	0	0
40	0 ...	0	0	0

	skin_peeling	silver_like_dusting	small_dents_in_nails	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	
15	0	0	0	
16	0	0	0	
17	0	0	0	
18	0	0	0	

19	0	0	0
20	0	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0
26	0	0	0
27	0	0	0
28	0	0	0
29	0	0	0
30	0	0	0
31	0	0	0
32	0	0	0
33	0	0	0
34	0	0	0
35	0	0	0
36	0	0	0
37	0	0	0
38	0	0	0
39	1	1	1
40	0	0	0

	inflammatory_nails	blister	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
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	0	0
22	0	0	0	0
23	0	0	0	0

24	0	0	0	0
25	0	0	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
34	0	0	0	0
35	0	0	0	0
36	0	0	0	0
37	0	0	0	0
38	0	0	0	0
39	1	0	0	0
40	0	1	1	1

[41 rows x 95 columns]

```
[16]: print(y_test)
```

	prognosis
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24

25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40

To build the precision of the model, we utilized three distinctive algorithms which are as per the following * Decision Tree algorithm * Random Forest algorithm * KNearestNeighbour algorithm * Naive Bayes algorithm

```
[17]: #list1 = DF['prognosis'].unique()
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())#total sum of symptom reported for given disease
    x.drop(x[x==0].index,inplace=True)#dropping symptoms with values 0
    print(x.values)
    y = x.keys()#storing name of symptoms in y
    print(len(x))
    print(len(y))
    plt.title(disea)
    plt.scatter(y,x.values)
    plt.show()

def scatterinp(sym1,sym2,sym3,sym4,sym5):
    x = [sym1,sym2,sym3,sym4,sym5]#storing input symptoms in y
    y = [0,0,0,0,0]#creating and giving values to the input symptoms
    if(sym1!='Select Here'):
        y[0]=1
    if(sym2!='Select Here'):
        y[1]=1
    if(sym3!='Select Here'):
        y[2]=1
    if(sym4!='Select Here'):
        y[3]=1
    if(sym5!='Select Here'):
        y[4]=1
    print(x)
    print(y)
```

```
plt.scatter(x,y)
plt.show()
```

2 Decision Tree Algorithm

```
[18]: root = Tk()
pred1=StringVar()
def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two_
↪Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn import tree

        clf3 = tree.DecisionTreeClassifier()
        clf3 = clf3.fit(X,y)

        from sklearn.metrics import_
↪classification_report,confusion_matrix,accuracy_score
        y_pred=clf3.predict(X_test)
        print("Decision Tree")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.
↪get(),Symptom5.get()]

        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        inputtest = [l2]
        predict = clf3.predict(inputtest)
```

```

predicted=predict[0]

h='no'
for a in range(0,len(disease)):
    if(predicted == a):
        h='yes'
        break

if (h=='yes'):
    pred1.set(" ")
    pred1.set(disease[a])
else:
    pred1.set(" ")
    pred1.set("Not Found")

#Creating the database if not exists named as database.db and creating
↪table if not exists named as DecisionTree using sqlite3
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name_
↪StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4_
↪TEXT,Symtom5 TEXT,Disease StringVar)")
c.execute("INSERT INTO_
↪DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?
↪?,?,?,?,(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.
↪get(),Symptom4.get(),Symptom5.get(),pred1.get()))
conn.commit()
c.close()
conn.close()

#printing scatter plot of input symptoms
#printing scatter plot of disease predicted vs its symptoms
scatterinp(Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.
↪get(),Symptom5.get())
scatterplt(pred1.get())

```

3 Random Forest Algorithm

```

[19]: pred2=StringVar()
def randomforest():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()

```

```

elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
    pred1.set(" ")
    sym=messagebox.askokcancel("System","Kindly Fill atleast first two_
↳Symptoms")
    if sym:
        root.mainloop()
else:
    from sklearn.ensemble import RandomForestClassifier
    clf4 = RandomForestClassifier(n_estimators=100)
    clf4 = clf4.fit(X,np.ravel(y))

    # calculating accuracy
    from sklearn.metrics import_
↳classification_report,confusion_matrix,accuracy_score
    y_pred=clf4.predict(X_test)
    print("Random Forest")
    print("Accuracy")
    print(accuracy_score(y_test, y_pred))
    print(accuracy_score(y_test, y_pred,normalize=False))
    print("Confusion matrix")
    conf_matrix=confusion_matrix(y_test,y_pred)
    print(conf_matrix)

    psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.
↳get(),Symptom5.get()]

    for k in range(0,len(l1)):
        for z in psymptoms:
            if(z==l1[k]):
                l2[k]=1

    inputtest = [l2]
    predict = clf4.predict(inputtest)
    predicted=predict[0]

    h='no'
    for a in range(0,len(disease)):
        if(predicted == a):
            h='yes'
            break
    if (h=='yes'):
        pred2.set(" ")
        pred2.set(disease[a])
    else:
        pred2.set(" ")
        pred2.set("Not Found")

```

```

        #Creating the database if not exists named as database.db and creating
↪table if not exists named as RandomForest using sqlite3
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name_
↪StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4_
↪TEXT,Symtom5 TEXT,Disease StringVar)")
c.execute("INSERT INTO_
↪RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?
↪,?,?,?,?), (NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.
↪get(),Symptom4.get(),Symptom5.get(),pred2.get()))
conn.commit()
c.close()
conn.close()
#printing scatter plot of disease predicted vs its symptoms
scatterplt(pred2.get())

```

4 KNearestNeighbour Algorithm

```

[20]: pred4=StringVar()
def KNN():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two_
↪Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
        knn=knn.fit(X,np.ravel(y))

        from sklearn.metrics import_
↪classification_report,confusion_matrix,accuracy_score
        y_pred=knn.predict(X_test)
        print("kNearest Neighbour")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")

```

```

conf_matrix=confusion_matrix(y_test,y_pred)
print(conf_matrix)

psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.
↪get(),Symptom5.get()]

for k in range(0,len(l1)):
    for z in psymptoms:
        if(z==l1[k]):
            l2[k]=1

inputtest = [l2]
predict = knn.predict(inputtest)
predicted=predict[0]

h='no'
for a in range(0,len(disease)):
    if(predicted == a):
        h='yes'
        break

if (h=='yes'):
    pred4.set(" ")
    pred4.set(disease[a])
else:
    pred4.set(" ")
    pred4.set("Not Found")

    #Creating the database if not exists named as database.db and creating
↪table if not exists named as KNearestNeighbour using sqlite3
import sqlite3
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name_
↪StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4_
↪TEXT,Symtom5 TEXT,Disease StringVar)")
c.execute("INSERT INTO_
↪KNearestNeighbour(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)_
↪VALUES(?,?,?,?,?,?,?)", (NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.
↪get(),Symptom4.get(),Symptom5.get(),pred4.get()))
conn.commit()
c.close()
conn.close()

#printing scatter plot of disease predicted vs its symptoms

scatterplt(pred4.get())

```

5 Naive Bayes Algorithm

```
[21]: pred3=StringVar()
def NaiveBayes():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two_
↪Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.naive_bayes import GaussianNB
        gnb = GaussianNB()
        gnb=gnb.fit(X,np.ravel(y))

        from sklearn.metrics import_
↪classification_report,confusion_matrix,accuracy_score
        y_pred=gnb.predict(X_test)
        print("Naive Bayes")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms = [Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.
↪get(),Symptom5.get()]
        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        inputtest = [l2]
        predict = gnb.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break
```



```

    if (h=='yes'):
        pred3.set(" ")
        pred3.set(disease[a])
    else:
        pred3.set(" ")
        pred3.set("Not Found")

    #Creating the database if not exists named as database.db and creating
    ↪table if not exists named as NaiveBayes using sqlite3
    import sqlite3
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symtom1_
    ↪StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5_
    ↪TEXT,Disease StringVar)")
    c.execute("INSERT INTO_
    ↪NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease) VALUES(?,?,?
    ↪?,?,?,?), (NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.
    ↪get(),Symptom4.get(),Symptom5.get(),pred3.get())")
    conn.commit()
    c.close()
    conn.close()

    #printing scatter plot of disease predicted vs its symptoms
    scatterplt(pred3.get())

```

6 Building Graphical User Interface

```

[22]: #Tk class is used to create a root window
root.configure(background='#001524')
root.title('Smart Disease Predictor System')
root.resizable(0,0)

```

```

[22]: ''

```

```

[23]: Symptom1 = StringVar()
Symptom1.set("Select Here")

Symptom2 = StringVar()
Symptom2.set("Select Here")

Symptom3 = StringVar()
Symptom3.set("Select Here")

Symptom4 = StringVar()
Symptom4.set("Select Here")

Symptom5 = StringVar()

```

```
Symptom5.set("Select Here")
Name = StringVar()
```

```
[24]: prev_win=None
def Reset():
    global prev_win

    Symptom1.set("Select Here")
    Symptom2.set("Select Here")
    Symptom3.set("Select Here")
    Symptom4.set("Select Here")
    Symptom5.set("Select Here")
    NameEn.delete(first=0,last=100)
    pred1.set(" ")
    pred2.set(" ")
    pred3.set(" ")
    pred4.set(" ")
    try:
        prev_win.destroy()
        prev_win=None
    except AttributeError:
        pass
```

```
[25]: from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")

    if qExit:
        root.destroy()
        exit()
```

```
[26]: #Headings for the GUI written at the top of GUI
w2 = Label(root, justify=LEFT, text="Disease Predictor using Machine Learning",
fg="#ffc300", bg="#001524")
w2.config(font=("Roboto",30,"bold italic"))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2.config(font=("Roboto",30,"bold italic"))
w2.grid(row=2, column=0, columnspan=2, padx=100)
```

```
[27]: #Label for the name
NameLb = Label(root, text="Name of the Patient *", fg="#ffc300", bg="#001524")
NameLb.config(font=("Roboto",15,"bold italic"))
NameLb.grid(row=6, column=0, pady=15, sticky=W)
```

```
[28]: #Creating Labels for the symptoms
S1Lb = Label(root, text="Symptom 1 *", fg="white", bg="#001524")
S1Lb.config(font=("Roboto",15,"bold italic"))
```

```

S1Lb.grid(row=7, column=0, pady=10, sticky=W)

S2Lb = Label(root, text="Symptom 2 *", fg="white", bg="#001524")
S2Lb.config(font=("Roboto",15,"bold italic"))
S2Lb.grid(row=8, column=0, pady=10, sticky=W)

S3Lb = Label(root, text="Symptom 3", fg="white",bg="#001524")
S3Lb.config(font=("Roboto",15,"bold italic"))
S3Lb.grid(row=9, column=0, pady=10, sticky=W)

S4Lb = Label(root, text="Symptom 4", fg="white", bg="#001524")
S4Lb.config(font=("Roboto",15,"bold italic"))
S4Lb.grid(row=10, column=0, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5", fg="white", bg="#001524")
S5Lb.config(font=("Roboto",15,"bold italic"))
S5Lb.grid(row=11, column=0, pady=10, sticky=W)

```

```

[29]: #Labels for the different algorithms
lrLb = Label(root, text="DecisionTree", width = 20)
lrLb.config(font=("Roboto",15,"bold italic"))
lrLb.grid(row=15, column=0, pady=10,sticky=W)

destreeLb = Label(root, text="RandomForest", width = 20)
destreeLb.config(font=("Roboto",15,"bold italic"))
destreeLb.grid(row=17, column=0, pady=10, sticky=W)

ranfLb = Label(root, text="NaiveBayes", width = 20)
ranfLb.config(font=("Roboto",15,"bold italic"))
ranfLb.grid(row=19, column=0, pady=10, sticky=W)

knnLb = Label(root, text="kNearestNeighbour", width = 20)
knnLb.config(font=("Roboto",15,"bold italic"))
knnLb.grid(row=21, column=0, pady=10, sticky=W)
OPTIONS = sorted(11)

```

```

[30]: #Taking name as input from user
NameEn = Entry(root, textvariable=Name)
NameEn.grid(row=6, column=1)

#Taking Symptoms as input from the dropdown from the user
S1 = OptionMenu(root, Symptom1,*OPTIONS)
S1.grid(row=7, column=1)

S2 = OptionMenu(root, Symptom2,*OPTIONS)
S2.grid(row=8, column=1)

```

```

S3 = OptionMenu(root, Symptom3,*OPTIONS)
S3.grid(row=9, column=1)

S4 = OptionMenu(root, Symptom4,*OPTIONS)
S4.grid(row=10, column=1)

S5 = OptionMenu(root, Symptom5,*OPTIONS)
S5.grid(row=11, column=1)

```

```

[31]: #Buttons for predicting the disease using different algorithms
dst = Button(root, text="Prediction 1", command=DecisionTree)
dst.config(font=("Roboto",15,"bold italic"))
dst.grid(row=6, column=3,padx=10)

rnf = Button(root, text="Prediction 2", command=randomforest)
rnf.config(font=("Roboto",15,"bold italic"))
rnf.grid(row=7, column=3,padx=10)

lr = Button(root, text="Prediction 3", command=NaiveBayes)
lr.config(font=("Roboto",15,"bold italic"))
lr.grid(row=8, column=3,padx=10)

kn = Button(root, text="Prediction 4", command=KNN)
kn.config(font=("Roboto",15,"bold italic"))
kn.grid(row=9, column=3,padx=10)

rs = Button(root,text="Reset Inputs", command=Reset,bg="#ffc300",fg="white")
rs.config(font=("Roboto",15,"bold italic"))
rs.grid(row=10,column=3,padx=10)

ex = Button(root,text="Exit System", command=Exit,bg="#ffc300",fg="white")
ex.config(font=("Roboto",15,"bold italic"))
ex.grid(row=11,column=3,padx=10)

```

```

[32]: #Showing the output of different algorithms
t1=Label(root,font=("Roboto",15,"bold italic"),text="Decision Tree",height=1
        ,width=40,fg="#ffc300",textvariable=pred1,relief="sunken").
    ↪grid(row=15, column=1, padx=10)

t2=Label(root,font=("Roboto",15,"bold italic"),text="Random Forest",height=1
        ,width=40,fg="#ffc300",textvariable=pred2,relief="sunken").
    ↪grid(row=17, column=1, padx=10)

t3=Label(root,font=("Roboto",15,"bold italic"),text="Naive Bayes",height=1
        ,width=40,fg="#ffc300",textvariable=pred3,relief="sunken").
    ↪grid(row=19, column=1, padx=10)

```

```
t4=Label(root,font=("Roboto",15,"bold italic"),text="kNearest_↵
↵Neighbour",height=1
        ,width=40,fg="#ffc300",textvariable=pred4,relief="sunken").
↵grid(row=21, column=1, padx=10)
```

```
[ ]: #calling this function because the application is ready to run
root.mainloop()
```

Decision Tree

Accuracy

0.9512195121951219

39

Confusion matrix

[[1 0 0 ... 0 0 0]

[0 1 0 ... 0 0 0]

[0 0 1 ... 0 0 0]

...

[0 0 0 ... 1 0 0]

[0 0 0 ... 0 1 0]

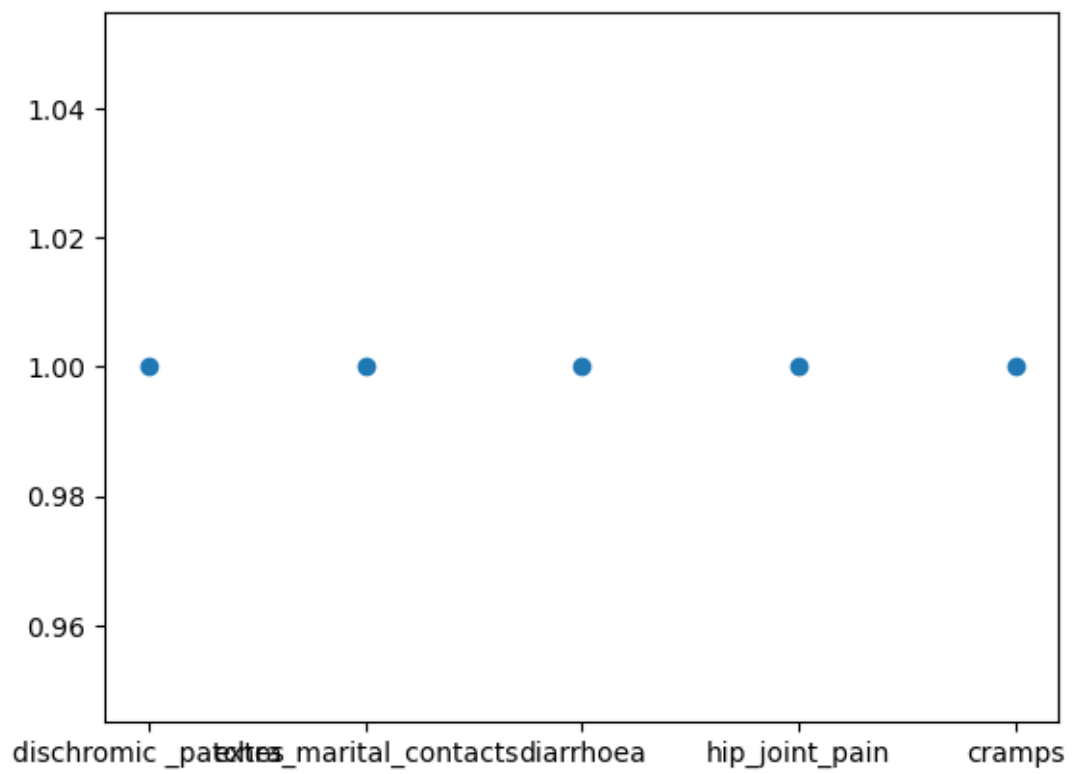
[0 0 0 ... 0 0 1]]

['dischromic _patches', 'extra_marital_contacts', 'diarrhoea', 'hip_joint_pain',
'cramps']

[1, 1, 1, 1, 1]

C:\Users\king1\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X
does not have valid feature names, but DecisionTreeClassifier was fitted with
feature names

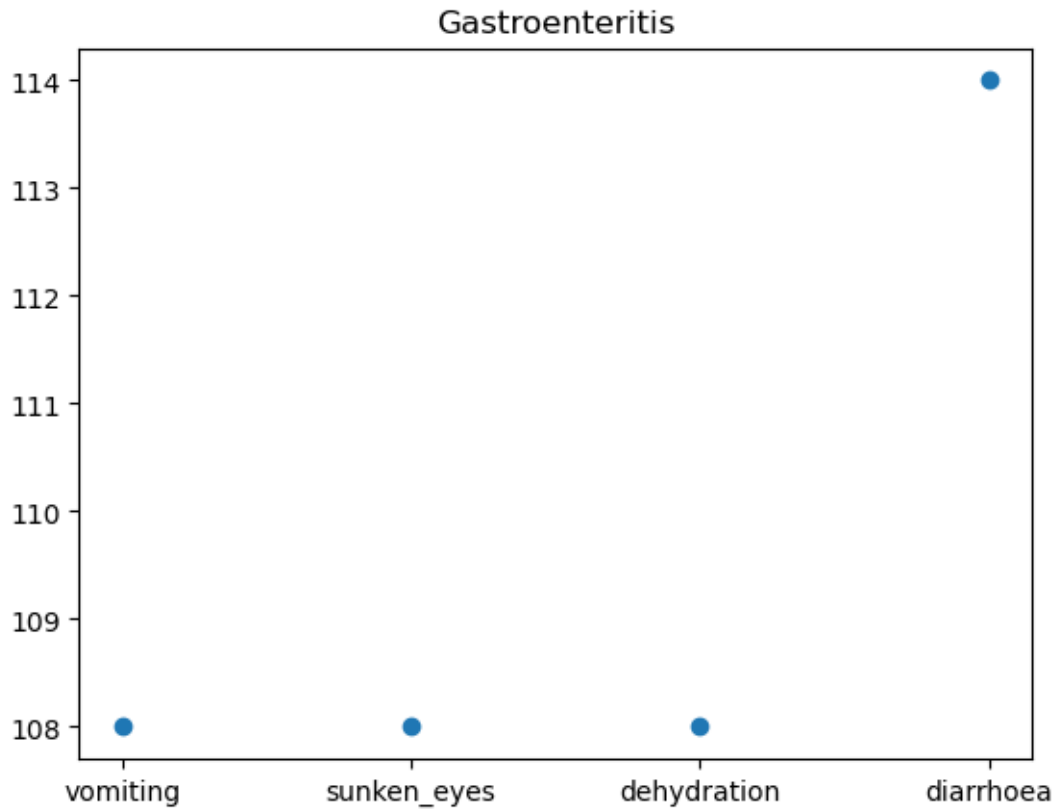
warnings.warn(



[108 108 108 114]

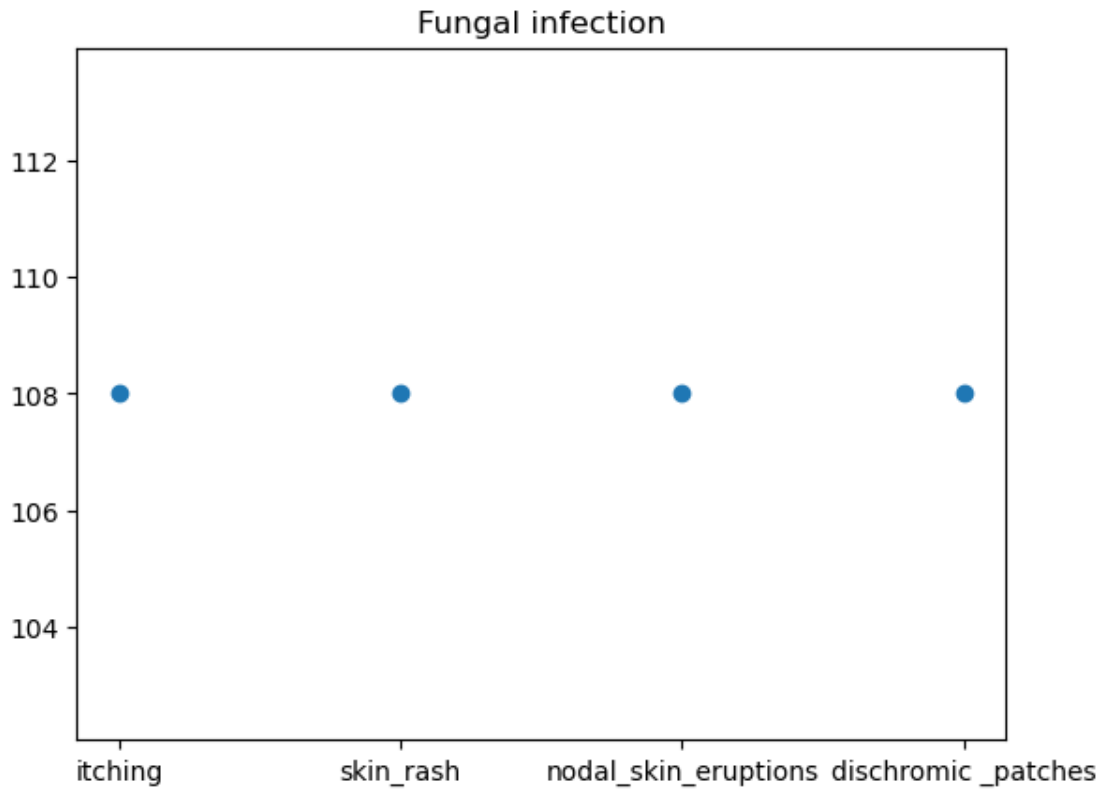
4

4



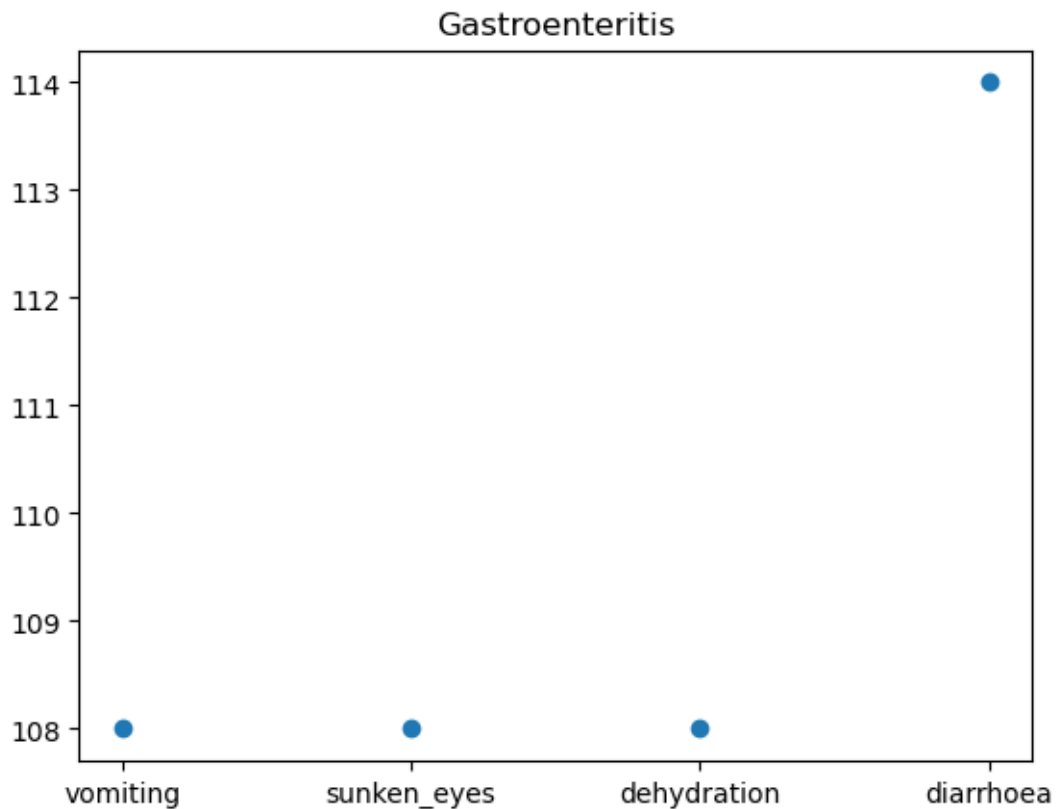
```
Random Forest
Accuracy
0.9512195121951219
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
[108 108 108 108]
4
4
```

```
C:\Users\king1\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
```



```
Naive Bayes
Accuracy
0.9512195121951219
39
Confusion matrix
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
[108 108 108 114]
4
4
```

```
C:\Users\king1\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X
does not have valid feature names, but GaussianNB was fitted with feature names
warnings.warn(
```

kNearest Neighbour

Accuracy

0.9512195121951219

39

Confusion matrix

[[1 0 0 ... 0 0 0]

[0 1 0 ... 0 0 0]

[0 0 0 ... 0 0 0]

...

[0 0 0 ... 1 0 0]

[0 0 0 ... 0 1 0]

[0 0 0 ... 0 0 1]]

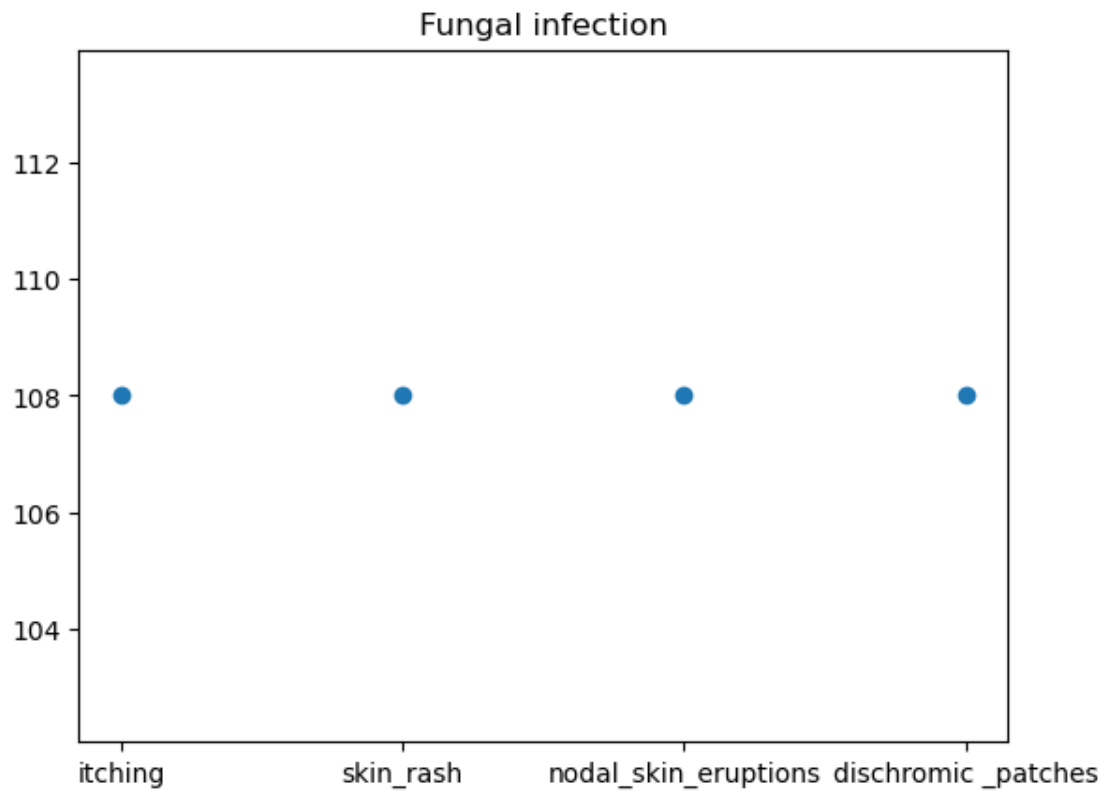
[108 108 108 108]

4

4

C:\Users\king1\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names

warnings.warn(



[]:

[]: