In [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
```

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

l1=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','ye
    'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_
    'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm',
    'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain
    'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','b
    'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesit
    'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittl
    'swollen_extremeties','excessive_hunger','extra_marital_contacts','dryin
    'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_n
    'movement_stiffness','spinning_movements','loss_of_balance','unsteadines
    'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_s
    'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_
    'depression','irritability','muscle_pain','altered_sensorium','red_spots
    'abnormal_menstruation','dischromic _patches','watering_from_eyes','incr
    'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_
    'receiving_unsterile_injections','coma','stomach_bleeding','distention_o
    'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','pro
    'palpitations','painful_walking','pus_filled_pimples','blackheads','scur
    'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blist
    'yellow_crust_ooze']
```

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

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)', 'Heart attack', 'Varicose veins',
       'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
       'Osteoarthristis', 'Arthritis',
       '(vertigo) Paroymsal  Positional Vertigo', 'Acne',
       'Urinary tract infection', 'Psoriasis', 'Impetigo']

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

In [4]:
```
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, 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]
```

In [5]:
```
#Reading the training .csv file
df=pd.read_csv("C:\\Users\\anith\\Downloads\\Testing.csv")
DF= pd.read_csv("C:\\Users\\anith\\Downloads\\Training.csv",index_col='progn
#Replace the values in the imported file by pandas by the inbuilt function r

df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic 
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bron
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken po
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alc
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract in
    'Impetigo':40}},inplace=True)
#df.head()
DF.head()
```

Out[5]:

prognosis	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_p
Fungal infection	1	1	1	0	0	0	
Fungal infection	0	1	1	0	0	0	
Fungal infection	1	0	1	0	0	0	
Fungal infection	1	1	0	0	0	0	
Fungal infection	1	1	1	0	0	0	

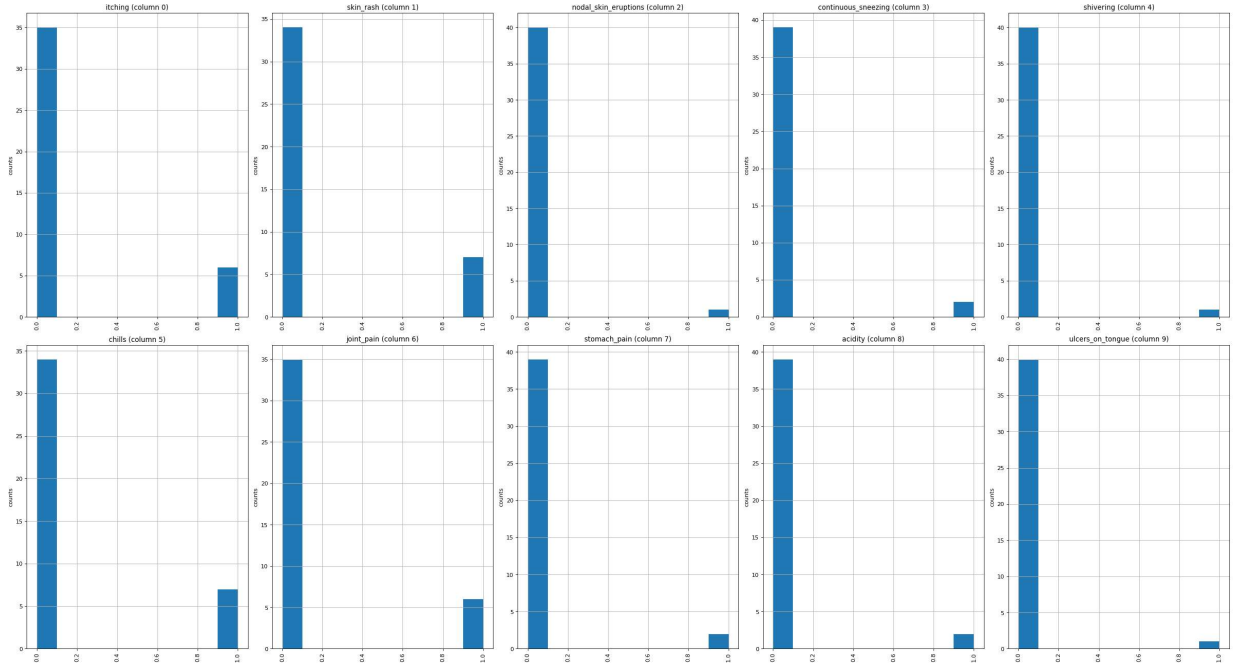
5 rows × 132 columns

In [6]:
```
# Distribution graphs (histogram/bar graph) of column data
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
    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()
```

In [7]:
```
# Scatter and density plots
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical colu
    # 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 w
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inve
        columnNames = columnNames[:10]
    df1 = df1[columnNames]
    ax = pd.plotting.scatter_matrix(df1, alpha=0.75, figsize=[plotSize, plot
    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), xyc
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

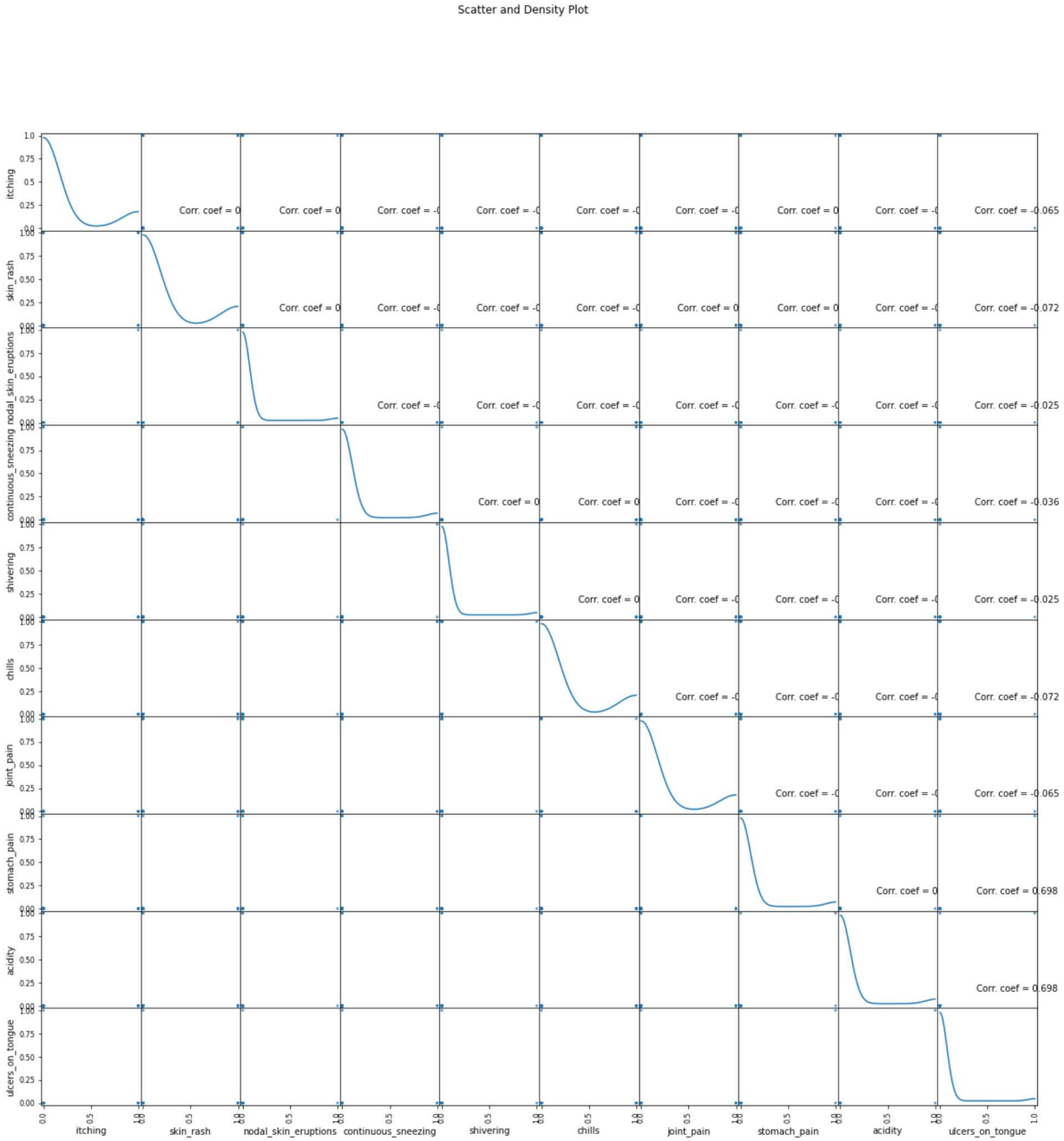
In [8]:
```
1 plotPerColumnDistribution(df, 10, 5)
```

```
C:\Users\anith\AppData\Local\Temp/ipykernel_25140/3685093358.py:10: MatplotlibD
eprecationWarning: Passing non-integers as three-element position specification
is deprecated since 3.3 and will be removed two minor releases later.
  plt.subplot(nGraphRow, nGraphPerRow, i + 1)
```



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

```
C:\Users\anith\AppData\Local\Temp/ipykernel_25140/1304029198.py:5: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.dropna will be keyw
ord-only
  df1 = df1.dropna('columns')
```



In [10]:
```
1 X= df[l1]
2 y = df[["prognosis"]]
3 np.ravel(y)
4 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               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
```

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

```
    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
```

In [12]:
```
#Reading the  testing.csv file
tr=pd.read_csv("C:\\Users\\anith\\Downloads\\Testing.csv")

#Using inbuilt function replace in pandas for replacing the values

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bron
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken po
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alc
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract in
    'Impetigo':40}},inplace=True)
tr.head()
```

Out[12]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	ston
0	1	1	1	0	0	0	0	
1	0	0	0	1	1	1	0	
2	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	
4	1	1	0	0	0	0	0	

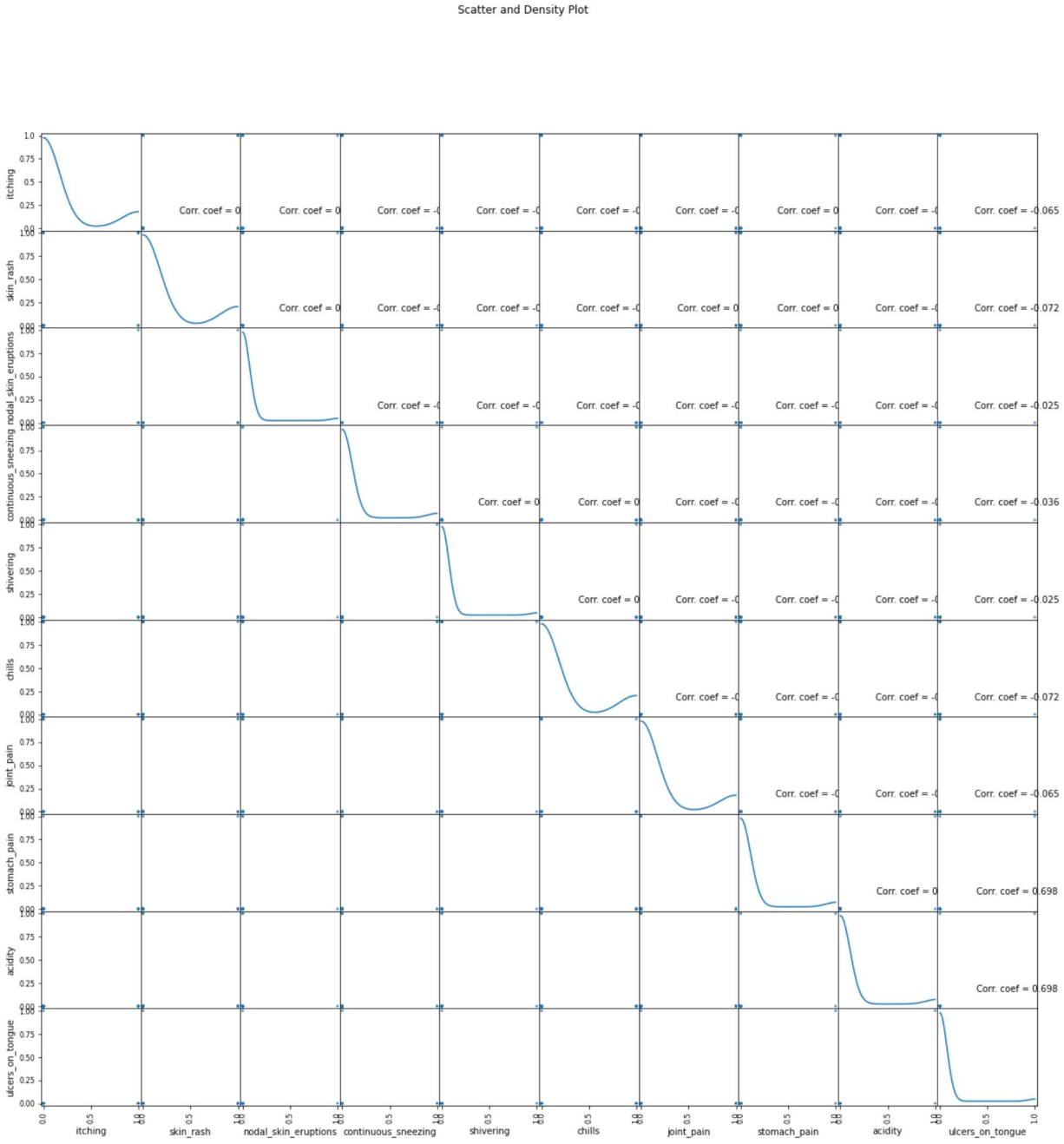
5 rows × 133 columns

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

```
C:\Users\anith\AppData\Local\Temp/ipykernel_25140/3685093358.py:10: MatplotlibDeprecationWarning: Passing non-integers as three-element position specification is deprecated since 3.3 and will be removed two minor releases later.
  plt.subplot(nGraphRow, nGraphPerRow, i + 1)
```

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

```
C:\Users\anith\AppData\Local\Temp/ipykernel_25140/1304029198.py:5: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.dropna will be keyw
ord-only
  df1 = df1.dropna('columns')
```



In [15]:
```
X_test= tr[l1]
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
```

In [16]:
```
1 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
```

In [17]:

```
#list1 = DF['prognosis'].unique()
def scatterplt(disea):
    x = ((DF.loc[disea]).sum())#total sum of symptom reported for given dise
    x.drop(x[x==0].index,inplace=True)#droping symptoms with values 0
    print(x.values)
    y = x.keys()#storing nameof 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()
```

In [18]:

```
#Decision Tree Algorithm
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 S
        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,a
        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.g

        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 creati
        import sqlite3
```

```
        conn = sqlite3.connect('C:\\Users\\anith\\Downloads\\database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name StringVar,Sy
        c.execute("INSERT INTO DecisionTree(Name,Symtom1,Symtom2,Symtom3,Sym
        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
        scatterplt(pred1.get())
```

In [19]:

```
#Random Forest Algorithm
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 S
        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,a
        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.g

        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 creat
        import sqlite3
        conn = sqlite3.connect('C:\\Users\\anith\\Downloads\\database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name StringVar,Sy
```

```
        c.execute("INSERT INTO RandomForest(Name,Symtom1,Symtom2,Symtom3,Sym
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms
        scatterplt(pred2.get())
```

In [20]:

```
#KNearestNeighbour Algorithm
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 S
        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,a
        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.g

        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 creat
        import sqlite3
        conn = sqlite3.connect('C:\\Users\\anith\\Downloads\\database.db')
        c = conn.cursor()
```

```
        c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name StringV
        c.execute("INSERT INTO KNearestNeighbour(Name,Symtom1,Symtom2,Symtom
        conn.commit()
        c.close()
        conn.close()
        #printing scatter plot of disease predicted vs its symptoms

        scatterplt(pred4.get())
```

In [21]:

```
d as NaiveBayes using sqlite3


Var,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease StringVar)")
,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4
```

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

Out[22]: ''

In [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()
```

In [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
```

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

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

In [26]:
```
#Headings for the GUI written at the top of GUI
w2 = Label(root, justify=LEFT, text="Disease Predictor using Machine Learnin
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2 = Label(root, justify=LEFT, text="Contributors:LBRCE STUDENTS", fg="Pink"
w2.config(font=("Times",30,"bold italic"))
w2.grid(row=2, column=0, columnspan=2, padx=100)
```

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

In [28]:
```
#Creating Labels for the symtoms
S1Lb = Label(root, text="Symptom 1 *", fg="Black", bg="Ivory")
S1Lb.config(font=("Times",15,"bold italic"))
S1Lb.grid(row=7, column=0, pady=10, sticky=W)

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

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

S4Lb = Label(root, text="Symptom 4", fg="Black", bg="Ivory")
S4Lb.config(font=("Times",15,"bold italic"))
S4Lb.grid(row=10, column=0, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5", fg="Black", bg="Ivory")
S5Lb.config(font=("Times",15,"bold italic"))
S5Lb.grid(row=11, column=0, pady=10, sticky=W)
```

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

destreeLb = Label(root, text="RandomForest", fg="white", bg="Sky Blue", widt
destreeLb.config(font=("Times",15,"bold italic"))
destreeLb.grid(row=17, column=0, pady=10, sticky=W)

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

knnLb = Label(root, text="kNearestNeighbour", fg="white", bg="red", width =
knnLb.config(font=("Times",15,"bold italic"))
knnLb.grid(row=21, column=0, pady=10, sticky=W)
OPTIONS = sorted(l1)
```

In [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)
```

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

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

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

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

rs = Button(root,text="Reset Inputs", command=Reset,bg="yellow",fg="purple",
rs.config(font=("Times",15,"bold italic"))
rs.grid(row=10,column=3,padx=10)

ex = Button(root,text="Exit System", command=Exit,bg="yellow",fg="purple",wi
ex.config(font=("Times",15,"bold italic"))
ex.grid(row=11,column=3,padx=10)
```

In [32]:
```
#Showing the output of different aldorithms
t1=Label(root,font=("Times",15,"bold italic"),text="Decision Tree",height=1,
         ,width=40,fg="red",textvariable=pred1,relief="sunken").grid(row=15,

t2=Label(root,font=("Times",15,"bold italic"),text="Random Forest",height=1,
         ,width=40,fg="white",textvariable=pred2,relief="sunken").grid(row=1

t3=Label(root,font=("Times",15,"bold italic"),text="Naive Bayes",height=1,bg
         ,width=40,fg="orange",textvariable=pred3,relief="sunken").grid(row=

t4=Label(root,font=("Times",15,"bold italic"),text="kNearest Neighbour",heig
         ,width=40,fg="yellow",textvariable=pred4,relief="sunken").grid(row=
```

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

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```