In [ ]:

```
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

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']

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']

l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
df=pd.read_csv("training.csv")
DF= pd.read_csv('training.csv', index_col='prognosis')

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()

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()

def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number])

    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]]
    columnNames = list(df)
    if len(columnNames) > 10:
        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()

X= df[l1]
y = df[["prognosis"]]
np.ravel(y)
print(X)
print(y)
tr=pd.read_csv("testing.csv")

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()

X_test= tr[l1]
y_test = tr[["prognosis"]]
np.ravel(y_test)
print(X_test)
print(y_test)

def scatterplt(disea):
    x = ((DF.loc[disea]).sum())
    x.drop(x[x==0].index,inplace=True)
    print(x.values)
    y = x.keys()
    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]
    y = [0,0,0,0,0]
    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()

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")

        import sqlite3
        conn = sqlite3.connect('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()

        scatterinp(Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get
        scatterplt(pred1.get())
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))

        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")

        import sqlite3
        conn = sqlite3.connect('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()

        scatterplt(pred2.get())
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")
        import sqlite3
        conn = sqlite3.connect('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()

        scatterplt(pred4.get())
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 S
        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,a
        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.g
        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")

        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symt
        c.execute("INSERT INTO NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtc
        conn.commit()
        c.close()
        conn.close()
        scatterplt(pred3.get())

root.configure()
root.title('Prognostic Disease Analytics')
root.resizable(0,0)
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()
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

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

    if qExit:
        root.destroy()
        exit()

w2 = Label(root, justify=LEFT, text="                      Prognostic Diseas
w2.config(font=("Helvetica",30,"bold italic"))
w2.grid(row=1, column=0, columnspan=2, padx=40)

NameLb = Label(root, text="Name of the Patient *", fg="Blue")
NameLb.config(font=("Helvetica",17,"bold italic"))
NameLb.grid(row=6, column=0, pady=15, sticky=W)

S1Lb = Label(root, text="Symptom 1 *", fg="Black")
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")
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")
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")
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")
S5Lb.config(font=("Times",15,"bold italic"))
S5Lb.grid(row=11, column=0, pady=10, sticky=W)

lrLb = Label(root, text="DecisionTree", fg="white", bg="red", 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="Red", bg="Orange", width =
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="green", 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="Red", bg="Sky Blue", widt
knnLb.config(font=("Times",15,"bold italic"))
knnLb.grid(row=21, column=0, pady=10, sticky=W)
OPTIONS = sorted(l1)

NameEn = Entry(root, textvariable=Name)
NameEn.grid(row=6, column=1)

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)

dst = Button(root, text="Prediction 1", command=DecisionTree,bg="red",fg="wh
dst.config(font=("Times",15,"bold italic"))
dst.grid(row=6, column=3,padx=10)

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

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

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

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

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

t1=Label(root,font=("Helvetica",15,"bold italic"),text="Decision Tree",heigh
         ,width=40,fg="red",textvariable=pred1,relief="sunken").grid(row=15,

t2=Label(root,font=("Helvetica",15,"bold italic"),text="Random Forest",heigh
         ,width=40,fg="red",textvariable=pred2,relief="sunken").grid(row=17,

t3=Label(root,font=("Helvetica",15,"bold italic"),text="Naive Bayes",height=
         ,width=40,fg="red",textvariable=pred3,relief="sunken").grid(row=19,

t4=Label(root,font=("Helvetica",15,"bold italic"),text="kNearest Neighbour",
         ,width=40,fg="red",textvariable=pred4,relief="sunken").grid(row=21,

root.mainloop()
```