Disease Prediction Using Machine Learning

Introduction

Disease prediction has become the most researched topic in the field of Machine Learning considering the health of a person is of utmost importance. The disease should be diagnosed accurately on the basis of symptoms shown by the patient as soon as possible, so that correct measures can be taken to cure it. There is a significant increase in the amount of medical data in healthcare industry from past few years, so data mining and machine learning have become necessary for finding the hidden patterns in data for prediction. Hence, this project has got many useful applications in real world.

Machine Learning Problem Formulation

Here, we are using Machine Learning algorithms like Decision Trees, Random Forest and Naive Bayes for the purpose of predicting the disease from given symptoms. We take symptoms of the user as input and then predict the disease accordingly and then compare the accuracy generated from it.

Data Set Source

The data set can be downloaded through the following link:

https://www.kaggle.com/kaushil268/disease-prediction-using-machine-learning/download

This dataset consists of 132 symptoms and 42 diseases can be predicted out of it.

Prerequisites

Software and important libraries needed are Python 3, Pycharm , numpy, pandas, tkinter (for GUI Interface) and scikit-learn.

Working Code:

Step 1: Importing the necessary pre installed libraries.

```
from tkinter import *
import numpy as np
import pandas as pd
```

Step 2: We make two arrays l1 and l2 consisting of the symptoms and diseases.

```
l1=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','
yellow_urine',
'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_st
omach',
'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','t
hroat_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','blo
ody_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_nec
k','swelling_joints',
'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness'
,
'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_sme
ll_of_urine',
'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_lo
ok_(typhos)',
'depression','irritability','muscle_pain','altered_sensorium','red_spots_o
ver_body','belly_pain',
'abnormal_menstruation','dischromic
_patches','watering_from_eyes','increased_appetite','polyuria','family_his
tory','mucoid_sputum',
'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_bl
ood_transfusion',
'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_
abdomen',
```

```
'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','promi
nent_veins_on_calf',
'palpitations','painful_walking','pus_filled_pimples','blackheads','scurri
ng','skin_peeling',
'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister
','red_sore_around_nose',
'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)',
'Heartattack','Varicoseveins','Hypothyroidism','Hyperthyroidism','Hypoglyc
emia','Osteoarthristis',
'Arthritis','(vertigo) Paroymsal  Positional Vertigo','Acne','Urinary
tract infection','Psoriasis',
'Impetigo']

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

Step 3: The training data is read which is used by the model to learn and train it. Here, the output value is specified for the given input values.

```
df=pd.read_csv(r"..\Training.csv")
df.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)
# print(df.head())
X= df[l1]
y = df[["prognosis"]]
np.ravel(y)
# print(y)
```

Step 4: Now, we read the Test dataset which is independent of the Training data and is used to evaluate the performance of the model.

```
tr=pd.read_csv(r"..\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)

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

Step 5: The following function uses Decision Tree for classification which uses a model that predicts the value of a required variable by learning decision rules obtained from the data features.

For detailed explanation, you can go to: https://www.geeksforgeeks.org/decision-tree/

```
def DecisionTree():

    from sklearn import tree
      clf3 = tree.DecisionTreeClassifier()    # empty model of the decision
tree
    clf3 = clf3.fit(X,y)

    # calculating accuracy
    from sklearn.metrics import accuracy_score
    y_pred=clf3.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    print(accuracy_score(y_test, y_pred,normalize=False))

    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'):
        t1.delete("1.0", END)
        t1.insert(END, disease[a])
    else:
        t1.delete("1.0", END)
        t1.insert(END, "Not Found")
```

Step 6: The below function uses Random Forest Model for classification. It is based on supervised learning and uses a combination of multiple decision trees on different subsets and then obtains the mean of them to improve the overall accuracy of result.

For detailed explanation, you can go to:
https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

```
def randomforest():
    from sklearn.ensemble import RandomForestClassifier
    clf4 = RandomForestClassifier()
    clf4 = clf4.fit(X,np.ravel(y))

    # calculating accuracy
    from sklearn.metrics import accuracy_score
    y_pred=clf4.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    print(accuracy_score(y_test, y_pred,normalize=False))

    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'):
        t2.delete("1.0", END)
        t2.insert(END, disease[a])
    else:
        t2.delete("1.0", END)
        t2.insert(END, "Not Found")
```

Step 7: The following function implements the Naive Bayes Model which uses Bayes theorem for the purpose of classification. It assumes that features present in a class are independent of each other. For detailed explanation you can go to: https://www.geeksforgeeks.org/naive-bayes-classifiers/

```
def NaiveBayes():
    from sklearn.naive_bayes import GaussianNB
    gnb = GaussianNB()
    gnb=gnb.fit(X,np.ravel(y))

    # calculating accuracy
    from sklearn.metrics import accuracy_score
    y_pred=gnb.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    print(accuracy_score(y_test, y_pred,normalize=False))

    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'):
        t3.delete("1.0", END)
        t3.insert(END, disease[a])
    else:
        t3.delete("1.0", END)
        t3.insert(END, "Not Found")
```

Step 8: Now we create a GUI Interface for taking the name of the user and 5 symptoms shown in the patient as input. The three models are used for predicting the disease according to the given symptoms. Three buttons are created, clicking on which the respective result is shown.

For detailed explanation you can go to: https://www.geeksforgeeks.org/python-gui-tkinter/

```
root = Tk()
root.configure(background='purple1')

# entry variables
Symptom1 = StringVar()
Symptom1.set(None)
Symptom2 = StringVar()
Symptom2.set(None)
Symptom3 = StringVar()
Symptom3.set(None)
Symptom4 = StringVar()
Symptom4.set(None)
Symptom5 = StringVar()
Symptom5.set(None)
Name = StringVar()

# Heading
w2 = Label(root, justify=LEFT, text="Disease Prediction based on Symptoms
provided", fg="black", bg="purple1")
w2.config(font=("Elephant", 30))
w2.grid(row=1, column=0, columnspan=2, padx=100)
w2 = Label(root, justify=LEFT, text="Machine Learning Project",
fg="black", bg="purple1")
w2.config(font=("Aharoni", 30))
w2.grid(row=2, column=0, columnspan=2, padx=100)

# labels for taking input from the user
NameLb = Label(root, text="Name of the Patient", fg="yellow", bg="black")
NameLb.grid(row=6, column=1, pady=15, sticky=W)

S1Lb = Label(root, text="Symptom 1",  fg="yellow", bg="black")
S1Lb.grid(row=7, column=1, pady=10, sticky=W)

S2Lb = Label(root, text="Symptom 2",  fg="yellow", bg="black")
S2Lb.grid(row=8, column=1, pady=10, sticky=W)

S3Lb = Label(root, text="Symptom 3",  fg="yellow", bg="black")
```

```
S3Lb.grid(row=9, column=1, pady=10, sticky=W)

S4Lb = Label(root, text="Symptom 4",    fg="yellow", bg="black")
S4Lb.grid(row=10, column=1, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5",    fg="yellow", bg="black")
S5Lb.grid(row=11, column=1, pady=10, sticky=W)

# taking input from the user
OPTIONS = sorted(l1)

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

S1En = OptionMenu(root, Symptom1,*OPTIONS)
S1En.grid(row=7, column=1)

S2En = OptionMenu(root, Symptom2,*OPTIONS)
S2En.grid(row=8, column=1)

S3En = OptionMenu(root, Symptom3,*OPTIONS)
S3En.grid(row=9, column=1)

S4En = OptionMenu(root, Symptom4,*OPTIONS)
S4En.grid(row=10, column=1)

S5En = OptionMenu(root, Symptom5,*OPTIONS)
S5En.grid(row=11, column=1)


dst = Button(root, text="DecisionTree",
command=DecisionTree,bg="black",fg="yellow")
dst.grid(row=15, column=1, pady=10, sticky=W)

rnf = Button(root, text="Randomforest",
command=randomforest,bg="black",fg="yellow")
rnf.grid(row=17, column=1, pady=10, sticky=W)

lr = Button(root, text="NaiveBayes",
command=NaiveBayes,bg="black",fg="yellow")
lr.grid(row=19, column=1, pady=10, sticky=W)

#displaying output to the user
t1 = Text(root, height=1, width=40,bg="white",fg="black")
t1.grid(row=15, column=1, padx=10)

t2 = Text(root, height=1, width=40,bg="white",fg="black")
```

```
t2.grid(row=17, column=1 , padx=10)

t3 = Text(root, height=1, width=40,bg="white",fg="black")
t3.grid(row=19, column=1 , padx=10)

root.mainloop()
```