Breast cancer

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=UserWarning)




def breast_cancer():

    df = 
pd.read_csv('C:/Users/kshah/OneDrive/Desktop/test_major_project/disease_pred/b
r.csv')


 x_train, x_test, y_train, y_test = 
train_test_split(df.drop(columns=['diagnosis']), df['diagnosis'], 
test_size=0.2, random_state=42)
    clf = svm.SVC(kernel='linear')
    clf.fit(x_train, y_train)


    # Accuracy Score on training data
    x_train_pred = clf.predict(x_train)
    training_data_accuracy = accuracy_score(y_train, x_train_pred)
    print('Accuracy (Training Data) :', training_data_accuracy*100,'%')


    # Accuracy Score on test data
    x_test_pred = clf.predict(x_test)
    testing_data_accuracy = accuracy_score(y_test, x_test_pred)
    print('Accuracy (Testing Data) :', testing_data_accuracy*100,'%')

print("Please enter radius_mean Range(6.981 - 28.11): 
")
    sp.speak("Please enter radius_mean Range(6.981 - 28.11): ")
    radius_mean = cmd.takeCommand().lower()
    print(radius_mean)
    sp.speak(radius_mean)

    print("Please enter texture_mean (9.71 - 39.28): ")
    sp.speak("Please enter texture_mean (9.71 - 39.28): ")
    texture_mean = cmd.takeCommand().lower()
    print(texture_mean)
    sp.speak(texture_mean)
 # Splitting the data into testing and training set
```

```
    x_train, x_test, y_train, y_test =
train_test_split(df.drop(columns=['status']), df['status'], test_size=0.2,
random_state=42)

    # Data Standardization
    scaler = StandardScaler()
    a = scaler.fit(x_train)

    x_train = scaler.transform(x_train)
    x_test = scaler.transform(x_test)

    # Model Training (DecisionTreeClassifier)
    clf = DecisionTreeClassifier()
    clf.fit(x_train, y_train)



    print("Please enter perimeter_mean (43.79 - 188.5): ")
    sp.speak("Please enter perimeter_mean (43.79 - 188.5): ")
    perimeter_mean = cmd.takeCommand().lower()
    print(perimeter_mean)
    sp.speak(perimeter_mean)


    print("Please enter area_mean(143.5 - 2501): ")
    sp.speak("Please enter area_mean(143.5 - 2501): ")
    area_mean = cmd.takeCommand().lower()
    print(area_mean)
    sp.speak(area_mean)


    preds = clf.predict([[radius_mean, texture_mean, perimeter_mean,
area_mean]])
    f_pred = (' '.join(preds))

    if f_pred == 'B':
        f_pred = 'Benign'
    else:
        f_pred = 'Malignant'
    sp.speak(f_pred)
    return f_pred
```

Parkinson Disease

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=UserWarning)

def parkinson():




    df =
pd.read_csv('C:/Users/kshah/OneDrive/Desktop/test_major_project/disease_pred/p
arkinsons.csv')
    # print(df.info())
    # print(df.describe())
    df.isnull().sum()#checking for missing values


    #dropping column axis = 1; dropping row then axis = 0
    #Data Pre-Processing - Seperating Features and Target variables according
to their Correlation

    df.drop(["name",'spread1', 'MDVP:Flo(Hz)','MDVP:Fhi(Hz)','MDVP:Fo(Hz)'],
axis=1, inplace=True)
    columns = list(df.columns)
    for column in columns:
        if column == "status":
            continue

        filtered_columns = [column]
        for col in df.columns:
            if (column == col) | (column == "status"):
                continue
            cor_val = df[column].corr(df[col])
            if cor_val > 0.75:
                columns.remove(col)
                continue
            else:
                filtered_columns.append(col)
        df = df[filtered_columns]
    df.isnull().sum() #checking null value


    # converting Data in the form of hundred
    df.iloc[:,:8] = (df.iloc[:, :8]).mul(100).astype(int)
    # Accuracy Score on training data
    x_train_pred = clf.predict(x_train)
    training_data_accuracy = accuracy_score(y_train, x_train_pred)

    print('Accuracy (Training Data) :', training_data_accuracy*100,'%')


    # Accuracy Score on test data
    x_test_pred = clf.predict(x_test)
    testing_data_accuracy = accuracy_score(y_test, x_test_pred)


    print('Accuracy (Testing Data) :', testing_data_accuracy*100,'%')
 print("Enter your First nonlinear dynamical complexity measures (142 - 367)")
    sp.speak("Enter your First nonlinear dynamical complexity measures (142 - 367)")
    D2 = cmd.takeCommand().lower()
    print(D2)
    sp.speak(D2)

    print("Enter your second nonlinear dynamical complexity measures (25 - 68)")
    sp.speak("Enter your second nonlinear dynamical complexity measures (25 - 68)")
    RPDE = cmd.takeCommand().lower()
    print(RPDE)
    sp.speak(RPDE)


    print('Enter your third nonlinear measures of fundamental frequency variation (4 - 52)')
    sp.speak('Enter your third nonlinear measures of fundamental frequency variation (4 - 52)')
    PPE = cmd.takeCommand().lower()
    print(PPE)
    sp.speak(PPE)

    print("Enter your nonlinear fundamental frequency variation (0 - 45)")
    sp.speak("Enter your nonlinear fundamental frequency variation (0 - 45)")
    spread2 = cmd.takeCommand().lower()
    print(spread2)
    sp.speak(spread2)
    print("Enter your Signal fractal scaling exponent (57 - 82)")
    sp.speak("Enter your Signal fractal scaling exponent (57 - 82)")
    DFA = cmd.takeCommand().lower()
    print(DFA)
    sp.speak(DFA)

    print("Enter your ratio of noise to tonal components in the voice (844 -
3304)")
    sp.speak("Enter your ratio of noise to tonal components in the voice (844
- 3304)")
    HNR = cmd.takeCommand().lower()
    print(HNR)
    sp.speak(HNR)




    print("Enter your Several measures of variation in amplitude(0 - 5)")
    sp.speak("Enter your Several measures of variation in amplitude(0 - 5)")
    Shimar = input('Enter variation in amplitude: ')

    print("Enter your Several measures of variation in fundamental frequency
(0 - 3)")
    sp.speak("Enter your Several measures of variation in fundamental
frequency (0 - 3)")
    Jitter= input('Enter fundamental frequency: ')


    p_pred = clf.predict([[D2, RPDE, PPE, spread2, DFA, HNR,Shimar, Jitter]])

    p_pred = (','.join(str(x) for x in p_pred))
    predicted = ""

    if p_pred == 0:
        predicted = 'Not Affected'


    else:
        p_pred == 1
        predicted = 'Affected'


        return predicted
```