**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/br.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/parkinsons.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)

    # 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)

    # Model Training (DecisionTreeClassifier)

    clf = DecisionTreeClassifier()

    clf.fit(x\_train, y\_train)

 # Model Evaluation

    # Accuracy Score

    # 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 = cmd.takeCommand().lower()

    # print(Shimar)

    # sp.speak(Shimar)

    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 = cmd.takeCommand().lower()

    # print(Jitter)

    # sp.speak(Jitter)

    Jitter= input('Enter fundamental frequency: ')

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

    predicted = ""

    if p\_pred == 0:

        predicted = 'Not Affected'

    else:

        p\_pred == 1

        predicted = 'Affected'

        return predicted