**Detecting Parkinson’s Disease with XGBoost**

In this Python machine learning project, using the ***Python libraries*** scikit-learn, numpy, pandas, and xgboost, we will build a model using an XGBClassifier. We’ll load the data, get the features and labels, scale the features, then split the dataset, build an XGBClassifier, and then calculate the accuracy of our model.

**PREREQUISITES:**

1. You’ll need the UCI ML Parkinsons dataset for this. The dataset has 24 columns and 195 records

2. You will need to install Jupyter lab and install XGBOOST using:

pip install numpy pandas sklearn xgboost

## **Steps for Detecting Parkinson’s Disease with XGBoost**

1. Make necessary imports:

import numpy as np

import pandas as pd

import os, sys

from sklearn.preprocessing import MinMaxScaler

from xgboost import XGBClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

2. Now, let’s read the data into a DataFrame and get the first 5 records.

#Read the data

df=pd.read\_csv(' C:\\Users\\User\\Desktop\\parkinsons.data')

df.head()

3. Get the features and labels from the DataFrame (dataset). The features are all the columns except ‘status’, and the labels are those in the ‘status’ column.

# Get the features and labels

features=df.loc[:,df.columns!='status'].values[:,1:]

labels=df.loc[:,'status'].values

4. The ‘status’ column has values 0 and 1 as labels; let’s get the counts of these labels for both- 0 and 1.

# Get the count of each label (0 and 1) in labels

print(labels[labels==1].shape[0], labels[labels==0].shape[0])

5. Initialize a MinMaxScaler and scale the features to between -1 and 1 to normalize them. The MinMaxScaler transforms features by scaling them to a given range. The fit\_transform() method fits to the data and then transforms it. We don’t need to scale the labels.

# Scale the features to between -1 and 1

scaler=MinMaxScaler((-1,1))

x=scaler.fit\_transform(features)

y=labels

6. Now, split the dataset into training and testing sets keeping 20% of the data for testing.

# Split the dataset

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x, y, test\_size=0.2, random\_state=7)

7. Initialize an XGBClassifier and train the model. This classifies using extreme Gradient Boosting- using***gradient boosting algorithms***for modern data science problems. It falls under the category of Ensemble Learning in ML, where we train and predict using many models to produce one superior output.

# Train the model

model=XGBClassifier()

model.fit(x\_train,y\_train)

8. Finally, generate y\_pred (predicted values for x\_test) and calculate the accuracy for the model. Print it out.

#Calculate the accuracy

y\_pred=model.predict(x\_test)

print(accuracy\_score(y\_test, y\_pred)\*100)

