Detecting Parkinson's Disease

What is Parkinson's Disease?

Parkinson's disease is a progressive disorder of the central nervous system affecting movement and inducing tremors and stiffness. It has 5 stages to it and affects more than 1 million individuals every year in India. This is chronic and has no cure yet. It is a neurodegenerative disorder affecting dopamine-producing neurons in the brain.

What is XGBoost?

XGBoost is a new Machine Learning algorithm designed with speed and performance in mind. XGBoost stands for eXtreme Gradient Boosting and is based on decision trees. In this project, we will import the XGBClassifier from the xgboost library; this is an implementation of the scikit-learn API for XGBoost classification.

Detecting Parkinson's Disease with XGBoost – Objective

To build a model to accurately detect the presence of Parkinson's disease in an individual.

Detecting Parkinson's Disease with XGBoost – About the Python Machine Learning Project

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.

Dataset for Python Machine Learning Project

You'll need the UCI ML Parkinsons dataset for this. The dataset has 24 columns and 195 records and is only 39.7 KB.

Prerequisites

You'll need to install the following libraries with pip:

```
pip install numpy pandas sklearn xgboost
```

You'll also need to install Jupyter Lab, and then use the command prompt to run it:

```
C:\ >jupyter lab
```

This will open a new JupyterLab window in your browser. Here, you will create a new console and type in your code, then press Shift+Enter to execute one or more lines at a time.

Steps for Detecting Parkinson's Disease with XGBoost

Below are some steps required to practice Python Machine Learning Project –

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
```

Screenshot:

```
[1]: #DataFlair - 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.

```
#DataFlair - Read the data
df=pd.read_csv('D:\parkinsons.data')
df.head()
```

Output Screenshot:

```
[2]: #DataFlair - Read the data
     df=pd.read_csv('D:\\DataFlair\\parkinsons.data')
     df.head()
```

[2]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status	
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1	0
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1	0
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1	0
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1	0
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1	0

5 rows × 24 columns

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.

```
#DataFlair - Get the features and labels
features=df.loc[:,df.columns!='status'].values[:,1:]
labels=df.loc[:,'status'].values
```

Screenshot:

```
[3]: #DataFlair - 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.

```
#DataFlair - Get the count of each label (0 and 1) in labels
print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

Output Screenshot:

```
[4]: #DataFlair - Get the count of each label (0 and 1) in labels
     print(labels[labels==1].shape[0], labels[labels==0].shape[0])

     147 48
```

We have 147 ones and 48 zeros in the status column in our dataset.

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
```

Screenshot:

```
[5]: #DataFlair - 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.

```
#DataFlair - Split the dataset
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_state=7)
```

Screenshot:

```
[6]: #DataFlair - 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.

```
#DataFlair - Train the model
model=XGBClassifier()
model.fit(x_train,y_train)
```

Output Screenshot:

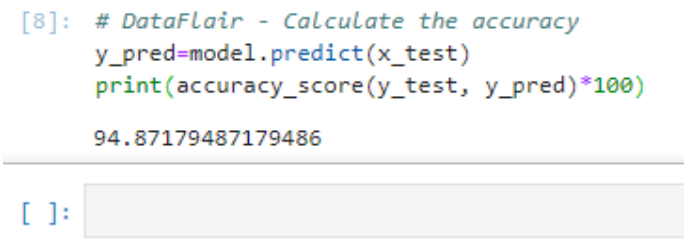
```
[7]: #DataFlair - Train the model
     model=XGBClassifier()
     model.fit(x_train,y_train)

[7]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                   colsample_bynode=1, colsample_bytree=1, gamma=0,
                   learning_rate=0.1, max_delta_step=0, max_depth=3,
                   min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                   nthread=None, objective='binary:logistic', random_state=0,
                   reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                   silent=None, subsample=1, verbosity=1)
```

8. Finally, generate y_pred (predicted values for x_test) and calculate the accuracy for the model. Print it out.

```
# DataFlair - Calculate the accuracy
y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
```

Output Screenshot:



Summary

In this Python machine learning project, we learned to detect the presence of Parkinson's Disease in individuals using various factors. We used an XGBClassifier for this and made use of the sklearn library to prepare the dataset. This gives us an accuracy of 94.87%, which is great considering the number of lines of code in this python project.