Problem Statement

Detecting Parkinson's Disease –Machine Learning Project

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; you can download it here. The dataset has 24 columns and 195 records and is only 39.7 KB or you can refer to the data file "parkinsons.data" provided.

Prerequisites

- Download the latest version of Anaconda for Python 3.8.
- basics of installing Jupyter and creating your first notebook
- Delve deeper and learn all the important terminology
- You'll need to install the following libraries with pip
- pip install numpy pandas sklearn xgboost
- 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

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

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.

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

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.

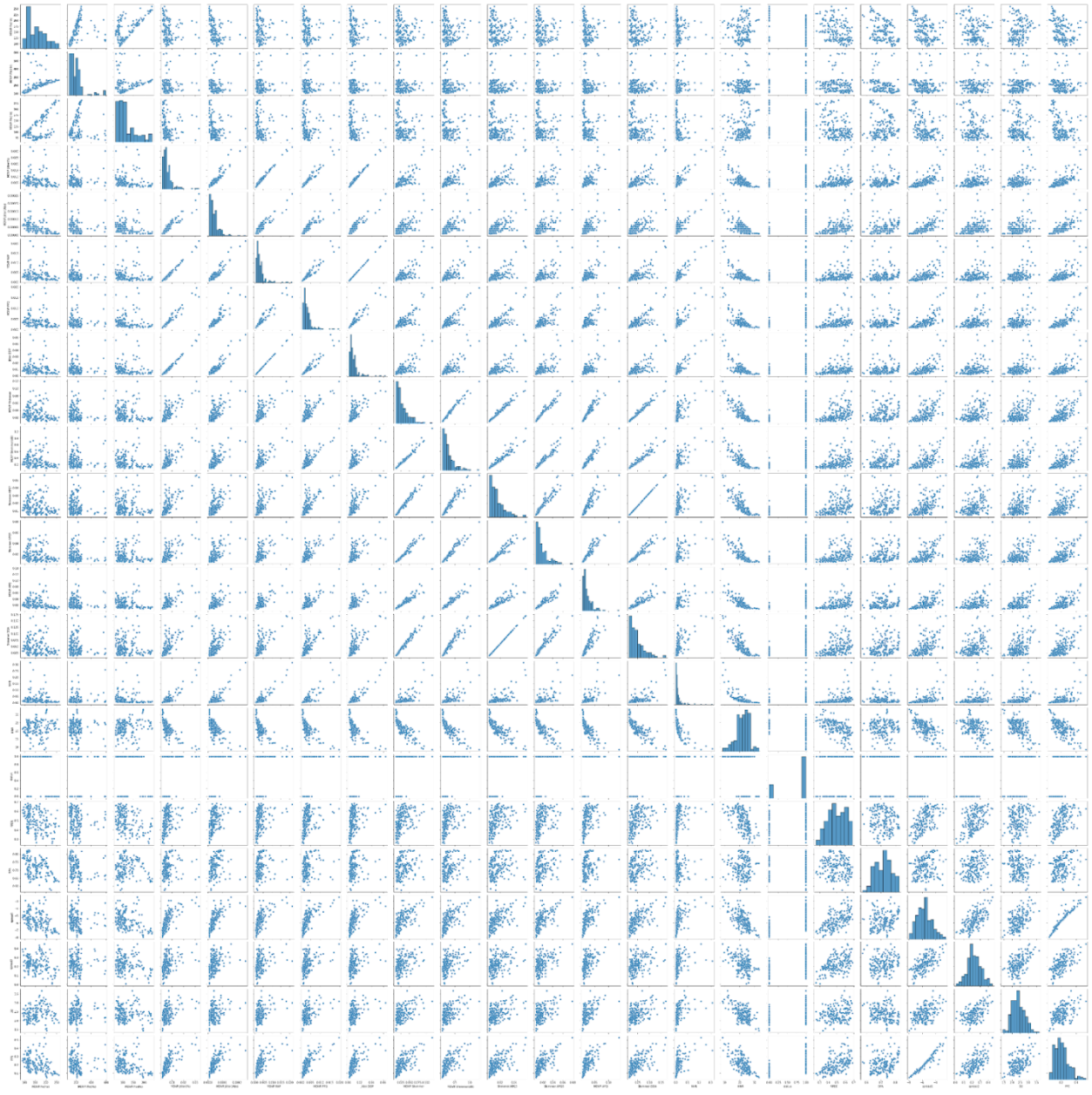
6. Now, split the dataset into training and testing sets keeping 20% of the data for testing.

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.

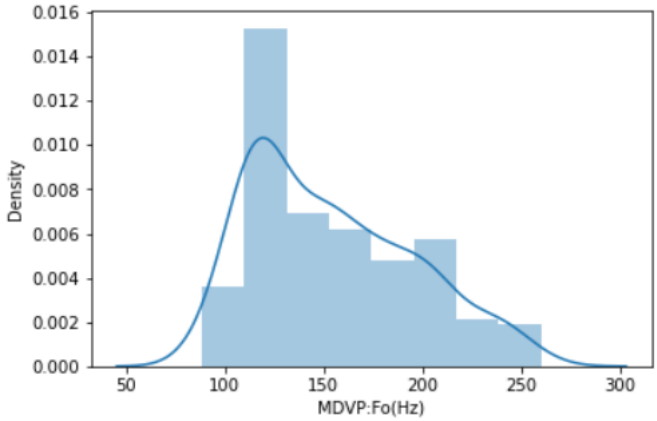
8. Finally, generate y_pred (predicted values for x_test) and calculate the accuracy for the model. Print it out.

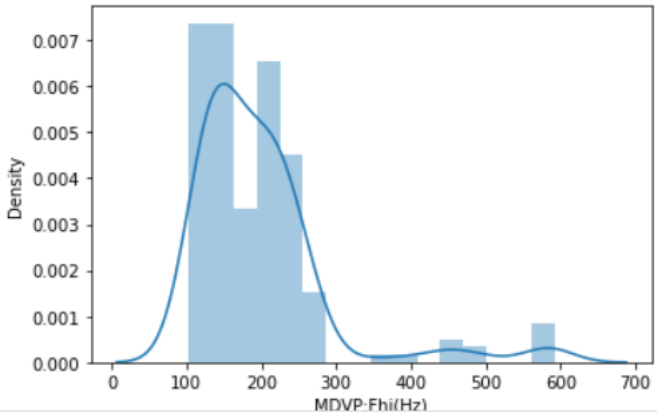
Model Evaluation:

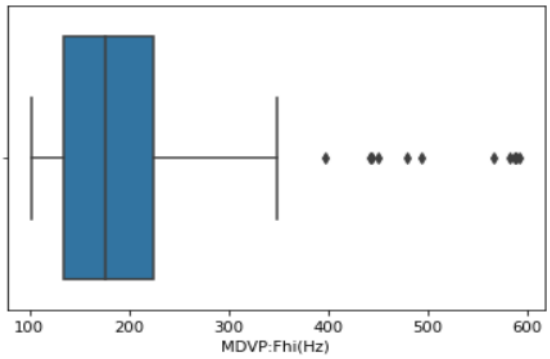
Sample Output:

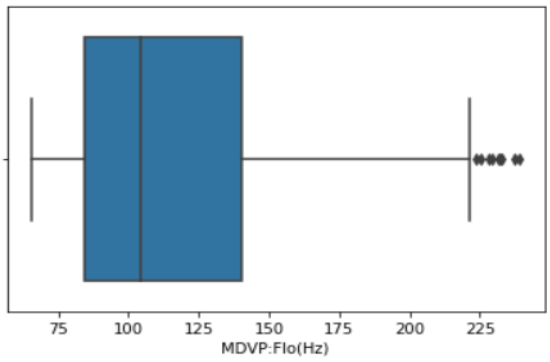


Sample Output:

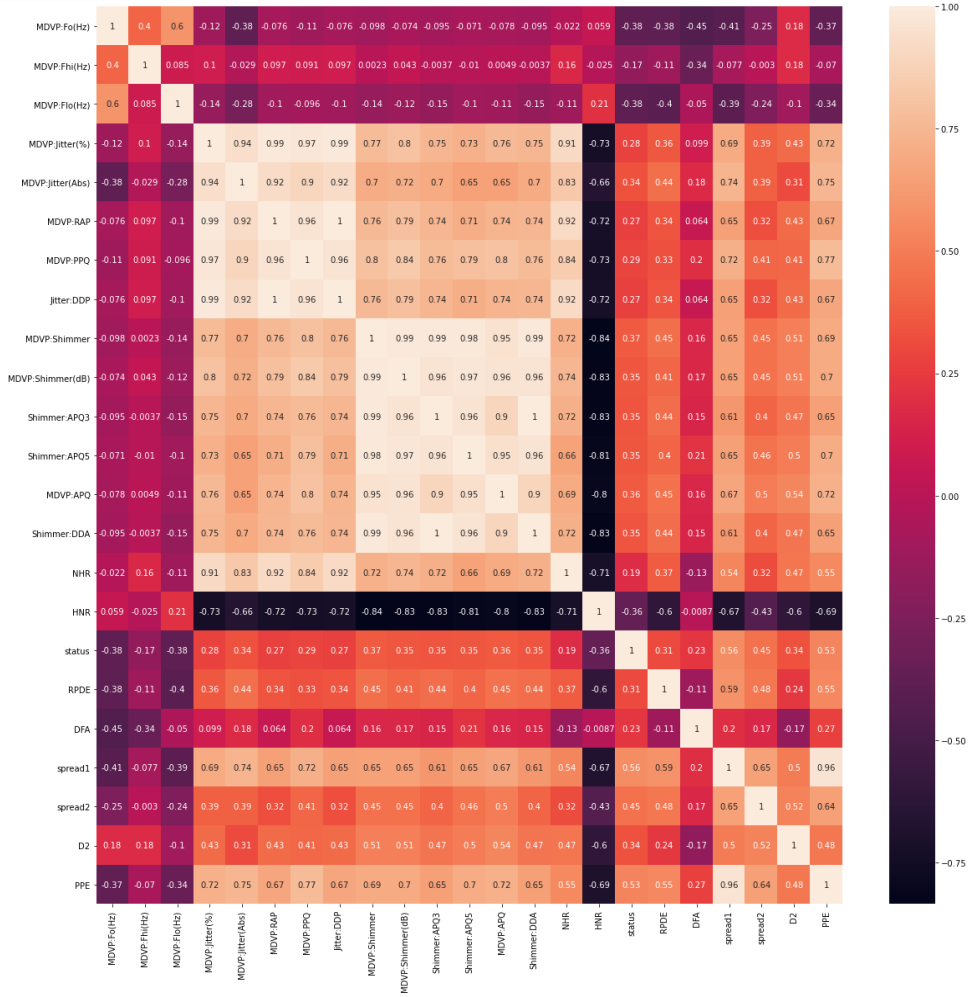




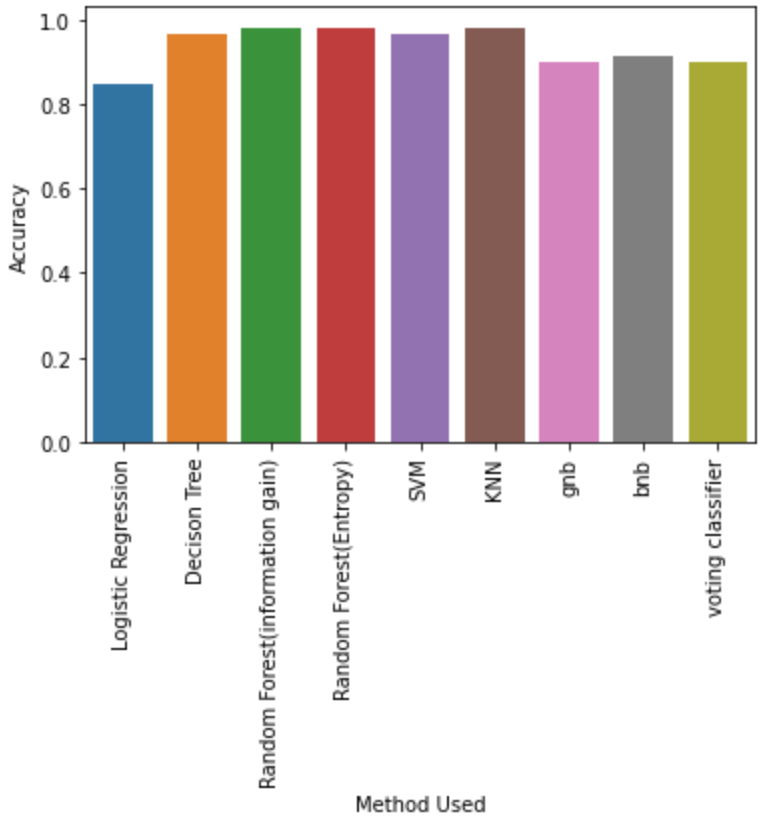




Sample Output:



Sample Output:




```
****************************************  LogisticRegression(C=0.4, max_iter=1000, solver='liblinear')  ****************************
              precision    recall  f1-score   support

           0       0.78      0.88      0.82        24
           1       0.91      0.83      0.87        35

    accuracy                           0.85        59
   macro avg       0.84      0.85      0.84        59
weighted avg       0.85      0.85      0.85        59

Confusion Matrix:
[[24  0]
 [ 0 35]]

****************************************  DecisionTreeClassifier(random_state=14)  ****************************************
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        24
           1       1.00      0.94      0.97        35

    accuracy                           0.97        59
   macro avg       0.96      0.97      0.97        59
weighted avg       0.97      0.97      0.97        59

Confusion Matrix:
[[24  0]
 [ 0 35]]
```

Sample output: