Author: CHRISTOFOR JOHN S

Title: HOUSE PRICE PREDICTOR

Phase 2: Innovation

To Boost our model With using Advanced regression techniques.

Phase 2: Innovation

Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for improved prediction accuracy.

BOOSTING

Boosting is a method used in machine learning to reduce errors in predictive data analysis. Data scientists train machine learning software, called machine learning models, on labeled data to make guesses about unlabeled data.

Types of boosting:

- Boosting methods are focused on iteratively combining weak learners to build a strong learner that can predict more accurate outcomes. As a reminder, a weak learner classifies data slightly better than random guessing.
- Boosting algorithms can differ in how they create and aggregate weak learners during the sequential process.

Three popular types of boosting methods include:

1. **Adaptive boosting or AdaBoost**
2. **Gradient boosting**
3. **Extreme gradient boosting or XGBoost**

- **Adaptive boosting or AdaBoost:**
  Yoav Freund and Robert Schapire are credited with the creation of the AdaBoost algorithm. This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues optimize in a sequential fashion until it yields the strongest predictor.
- **Gradient boosting:**
  Building on the work of Leo Breiman, Jerome H. Friedman developed gradient boosting, which works by sequentially adding predictors to an ensemble with each one correcting for the errors of its predecessor. However, instead of changing weights of data points like AdaBoost, the gradient boosting trains on the residual errors of the previous predictor. The name, gradient boosting, is used since it combines the gradient descent algorithm and boosting method.
- **Extreme gradient boosting or XGBoost:**
  XGBoost is an implementation of gradient boosting that's designed for computational speed and scale. XGBoost leverages multiple cores on the CPU, allowing for learning to occur in parallel during training.

Benefits and challenges of boosting:

There are a number of key advantages and challenges that the boosting method presents when used for classification or regression problems.

The key benefits of boosting include:

- **Ease of Implementation:** Boosting can be used with several hyper-parameter tuning options to improve fitting. No data preprocessing is required, and boosting algorithms like have built-in routines to handle missing data. In Python, the scikit-learn library of ensemble methods (also known as sklearn.ensemble) makes it easy to implement the popular boosting methods, including AdaBoost, XGBoost, etc.
- **Reduction of bias:** Boosting algorithms combine multiple weak learners in a sequential method, iteratively improving upon observations. This approach can help to reduce high bias, commonly seen in shallow decision trees and logistic regression models.
- **Computational Efficiency:** Since boosting algorithms only select features that increase its predictive power during training, it can help to reduce dimensionality as well as increase computational efficiency.

The key challenges of boosting include:

- **Overfitting:** There's some dispute in the around whether or not boosting can help reduce overfitting or exacerbate it. We include it under challenges because in the instances that it does occur, predictions cannot be generalized to new datasets.
- **Intense computation:** Sequential training in boosting is hard to scale up. Since each estimator is built on its predecessors, boosting models can be computationally expensive, although XGBoost seeks to address scalability issues seen in other types of boosting methods. Boosting algorithms can be slower to train when compared to bagging as a large number of parameters can also influence the behavior of the model.

Applications of boosting

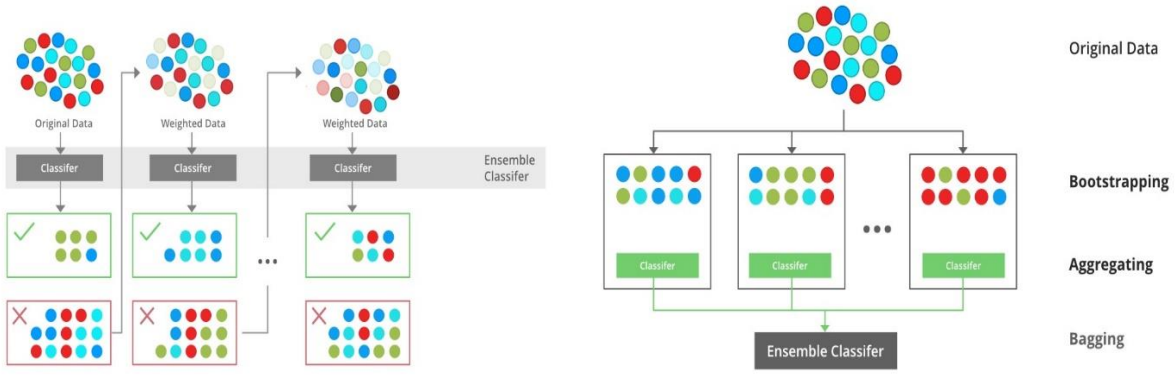
Boosting algorithms are well suited for artificial intelligence projects across a broad range of industries, including:

- **Healthcare**
- **IT**
- **Finance**

XGBoost:

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve

many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.



Requirements:

To implement the XGboost we need to satisfy the pre requirement as follows,

- Python 3.11
- XGBoost Module

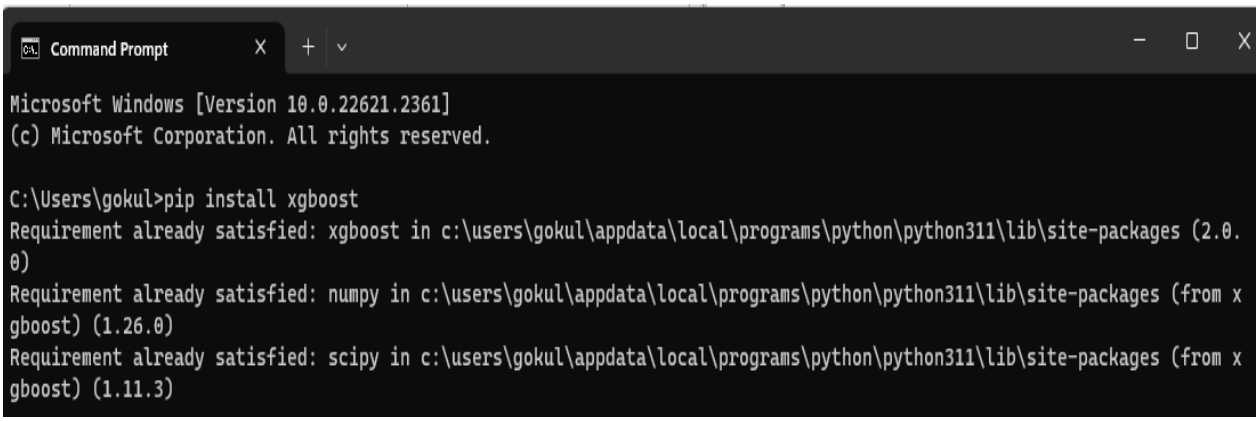
Installation Guide:

Install Python 3.11 .1 or more from online .

Then install working environment IDE like Anconda (Jupyter notebook), VS code or colab from online.

Install XGboost using the command in command prompt

>pip install xgboost



Methods in Xgboost for regression:

XGBRegressor()

Before using it we need to import it in python file as follows

from xgboost import XGBRegressor

Online Sample Code:

```
from xgboost import XGBClassifier
# read data
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data['data'], data['target'], test_size=.2)
# create model instance
Bst=XGBClassifier(n_estimators=2,max_depth=2,learning_rate=1,objective='binary:logistic')
# fit model
bst.fit(X_train, y_train)
# make predictions
preds = bst.predict(X_test)
```

Our code used in our model:

```
    # Load and preprocess the data
    X_train, X_test, y_train, y_test = load_and_preprocess_data('USA_Housing.csv')

    # Create an XGBoost Regressor model
    model = XGBRegressor(n_estimators=100, learning_rate=0.1, random_state=0)

    # Fit the model to your training data
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)
```

“USA_Housing.csv” is provided by your Team for our project House price prediction.

Source: Kaggle. (Dataset Link: https://www.kaggle.com/datasets/vedavyasv/usa-housing) Provided by your organization.

XGBRegressor() is Method to pass our trained data as the parameter. And this method help us to boost that is improve the performance of our model.

Advantages:

1. **High accuracy**
2. **Speed**
3. **Flexibility**

Disadvantages:

1. **Complexity**
2. **Overfitting**
3. **Memory usage**