Name:Kaushik Kotian Roll no.:29 Div:D20B Batch:B

AIDS-2 Experiment no.10

Aim : To implement ada boosting supervised learning algorithm.

Theory :

Introduction

AdaBoost (Adaptive Boosting) is a powerful ensemble learning algorithm used in supervised machine learning for classification tasks. It works by combining multiple weak learners, typically decision trees, to form a strong classifier. The key idea behind AdaBoost is to sequentially train weak classifiers, giving more importance (weight) to instances that were misclassified by the previous ones. This iterative process continues until the model reaches a specified number of weak classifiers or achieves a desirable level of accuracy. AdaBoost adjusts the weights of each classifier based on its performance, ensuring that future classifiers focus on harder-to-classify examples, making the final model more accurate and robust.

How AdaBoost Works

To understand how AdaBoost works, smash down its working mechanism right into a step-by-step process:

1. Weight Initialization

At the start, every schooling instance is assigned an identical weight. These weights determine the importance of every example in the getting-to-know method.

2. Model Training

A weak learner is skilled at the dataset, with the aim of minimizing class errors. A weak learner is usually an easy model, which includes a selection stump (a one-stage decision tree) or a small neural network.

3. Weighted Error Calculation

After the vulnerable learner is skilled, its miles are used to make predictions at the education dataset. The weighted mistakes are then calculated by means of summing up the weights of the misclassified times. This step emphasizes the importance of the samples which are tough to classify.

4. Model Weight Calculation

The weight of the susceptible learner is calculated primarily based on their Performance in classifying the training data. Models that perform properly are assigned higher weights, indicating that they're more reliable.

5. Update Instance Weights

The example weights are updated to offer more weight to the misclassified samples from the previous step. This adjustment focuses on the studying method at the times that the present-day model struggles with.

6. Repeat

Steps 2 through five are repeated for a predefined variety of iterations or till a distinctive overall performance threshold is met.

7. Final Model Creation

The very last sturdy model (also referred to as the ensemble) is created by means of combining the weighted outputs of all weak newcomers. Typically, the fashions with better weights have an extra influence on the final choice.

8. Classification

To make predictions on new records, AdaBoost uses the very last ensemble model. Each vulnerable learner contributes its prediction, weighted with the aid of its significance, and the blended result is used to categorize the enter.

Key Concepts in AdaBoost

To gain deeper information about AdaBoost, it's critical to be acquainted with some key principles associated with the algorithm:

1. Weak Learners

Weak novices are the individual fashions that make up the ensemble. These are generally fashions with accuracy barely higher than random hazards. In the context of AdaBoost, weak beginners are trained sequentially, with each new model focusing on the instances that preceding models determined difficult to classify.

2. Strong Classifier

The strong classifier, additionally known as the ensemble, is the final version created by combining the predictions of all weak first-year students. It has the collective know-how of all of the fashions and is capable of making correct predictions.

3. Weighted Voting

In AdaBoost, every susceptible learner contributes to the very last prediction with a weight-based totally on its Performance. This weighted vote-casting machine ensures that the greater correct fashions have a greater say in the final choice.

4. Error Rate

The error rate is the degree of ways a vulnerable learner plays on the schooling statistics. It is used to calculate the load assigned to each vulnerable learner. Models with lower error fees are given higher weights.

5. Iterations

The range of iterations or rounds in AdaBoost is a hyperparameter that determines what number of susceptible newbies are educated. Increasing the range of iterations may additionally result in a more complex ensemble; however, it can also increase the risk of overfitting.

Advantages of AdaBoost

AdaBoost gives numerous blessings that make it a popular choice in gadget mastering:

1. Improved Accuracy

AdaBoost can notably improve the accuracy of susceptible, inexperienced persons, even when the usage of easy fashions. By specializing in misclassified instances, it adapts to the tough areas of the records distribution.

2. Versatility

AdaBoost can be used with a number of base newbies, making it a flexible set of rules that may be carried out for unique forms of problems.

3. Feature Selection

It routinely selects the most informative features, lowering the need for giant function engineering.

4. Resistance to Overfitting

AdaBoost tends to be much less at risk of overfitting compared to a few different ensemble methods, thanks to its recognition of misclassified instances.

Limitations and Challenges

While AdaBoost is an effective algorithm, it is important to be aware of its barriers and challenges:

1. Sensitivity to Noisy Data

AdaBoost may be sensitive to noisy facts and outliers because it offers greater weight to misclassified times. Outliers can dominate the studying system and result in suboptimal consequences.

2. Computationally Intensive

Training AdaBoost may be computationally intensive, especially while using a massive wide variety of susceptible learners. This could make it much less appropriate for real-time applications.

3. Overfitting

Although AdaBoost is much less prone to overfitting than a few different algorithms, it may nonetheless overfit if the number of iterations is too excessive.

4. Model Selection

Selecting the proper vulnerable learner and tuning hyperparameters may be difficult, as the Performance of AdaBoost is noticeably dependent on these alternatives.

<u>Practical Applications</u>

AdaBoost has found applications in a huge range of domains, along with but not constrained to:

1. Face Detection

AdaBoost has been used in computer imagination and prescient for obligations like face detection, in which it allows the perception of faces in pics or motion pictures.

2. Speech Recognition

In speech popularity, AdaBoost can be used to improve the accuracy of phoneme or word popularity structures.

3. Anomaly Detection

AdaBoost can be applied to anomaly detection problems in numerous fields, such as finance, healthcare, and cybersecurity.

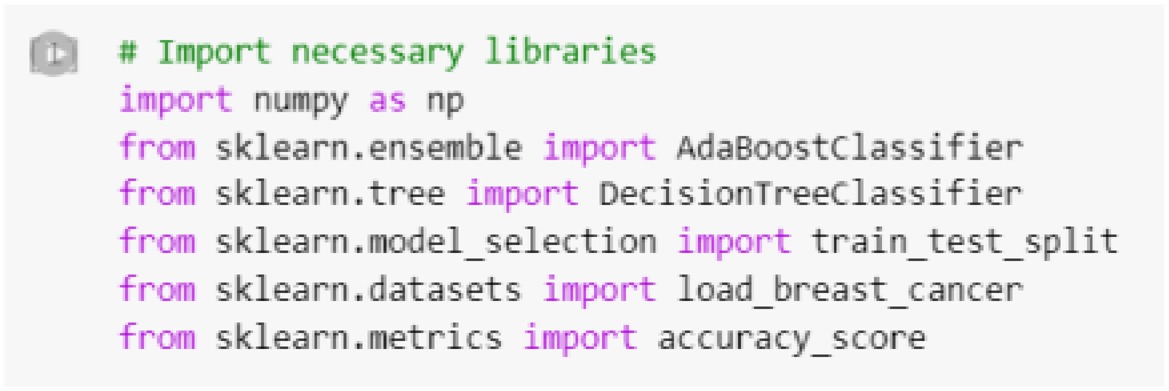
4. Natural Language Processing

In NLP, AdaBoost can decorate the overall Performance of sentiment analysis and textual content category fashions.

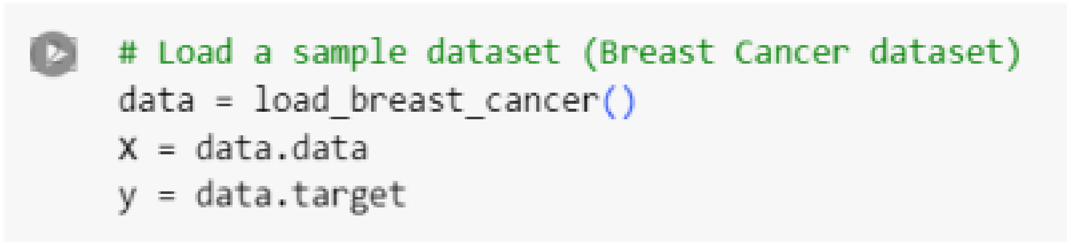
5. Biology and Bioinformatics

AdaBoost has been used for protein type, gene prediction, and different bioinformatics duties.

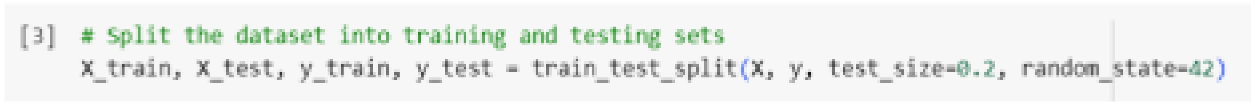
Code Implementation :



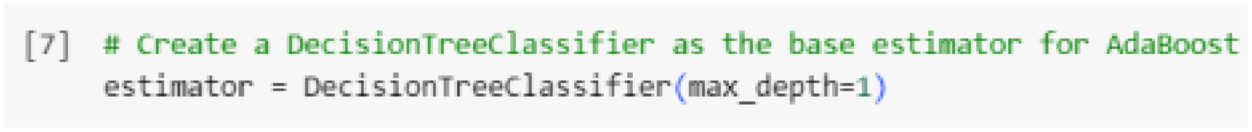
```
# Import necessary libraries
import numpy as np
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import accuracy_score
```



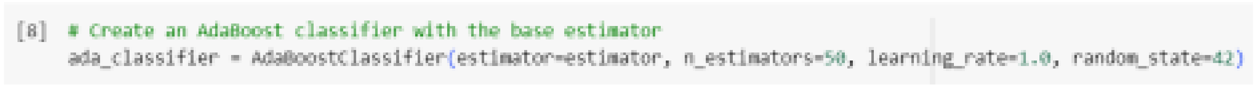
```
# Load a sample dataset (Breast Cancer dataset)
data = load_breast_cancer()
X = data.data
y = data.target
```



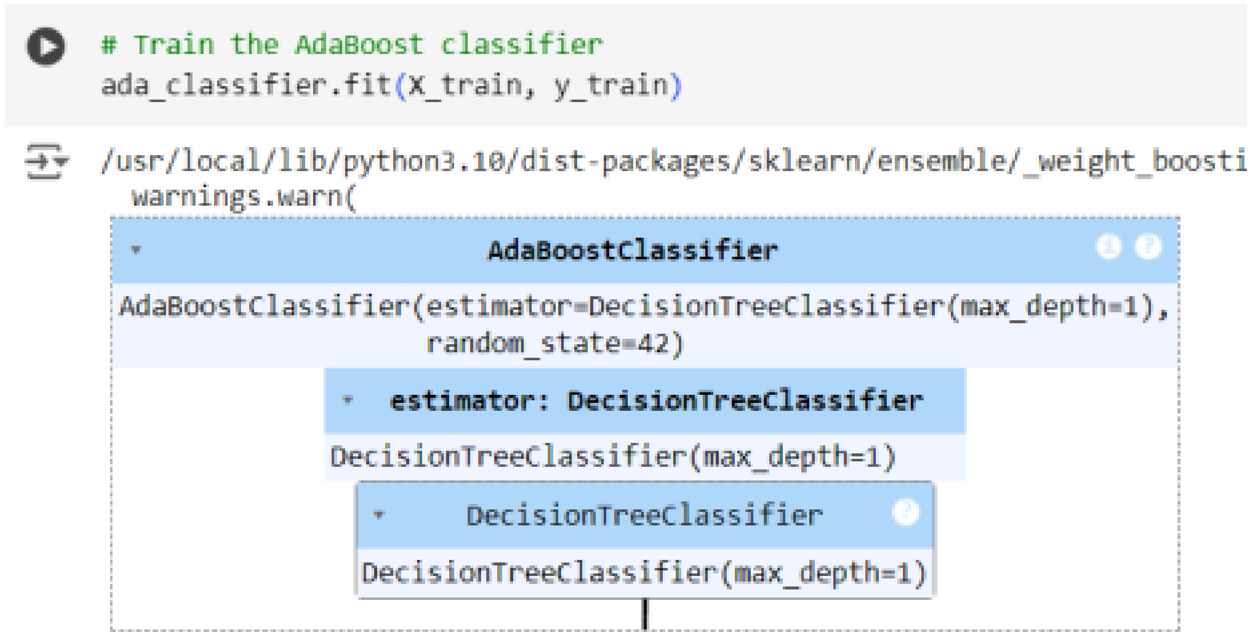
```
[3]  # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



```
[7]  # Create a DecisionTreeClassifier as the base estimator for AdaBoost
     estimator = DecisionTreeClassifier(max_depth=1)
```



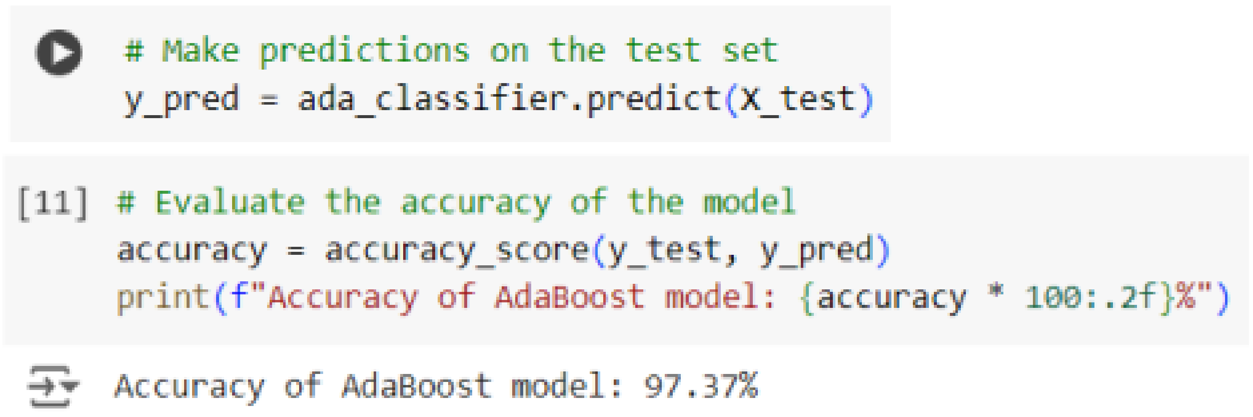
```
[8]  # Create an AdaBoost classifier with the base estimator
     ada_classifier = AdaBoostClassifier(estimator=estimator, n_estimators=50, learning_rate=1.0, random_state=42)
```



```
# Train the AdaBoost classifier
ada_classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosti
  warnings.warn(
```

```
AdaBoostClassifier
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1),
                   random_state=42)
  estimator: DecisionTreeClassifier
  DecisionTreeClassifier(max_depth=1)
    DecisionTreeClassifier
    DecisionTreeClassifier(max_depth=1)
```



```
# Make predictions on the test set
y_pred = ada_classifier.predict(X_test)
```

```
[11] # Evaluate the accuracy of the model
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy of AdaBoost model: {accuracy * 100:.2f}%")
```

```
Accuracy of AdaBoost model: 97.37%
```

Conclusion :

We have implemented the AdaBoost classifier, which improved accuracy by combining weak decision tree classifiers into a strong model. It effectively adapted to misclassified examples, enhancing overall performance. This demonstrates AdaBoost's capability to boost weak learners for better predictions.