Boosting with Decision Tree  
**Ex No**: 7  
**Date**: 14/08/2024  
**Name**: Ruban S - 71762233039

**Aim**:  
To build a Boosting ensemble model using a Decision Tree to predict employee attrition based on various features in the provided dataset.

**What is Boosting?**  
Boosting is an ensemble machine learning technique that sequentially builds models,

each attempting to correct the errors of the previous one. Unlike Bagging, which creates

independent models, Boosting focuses on models that complement each other by

giving more weight to previously misclassified data points. The combined model tends

to be more accurate and robust.

**Procedure**:

1. **Import Necessary Libraries and Load the Dataset**:
   * Import libraries such as pandas, numpy, scikit-learn, matplotlib, and seaborn.
   * Load the employee attrition dataset into a DataFrame.
2. **Encode Categorical Data**:
   * Use LabelEncoder or mapping to transform categorical columns (such as Gender, Department, Job\_Title) into numerical values.
3. **Prepare the Data**:
   * Select features like Age, Gender\_encode, Department\_encode, Job\_Title\_encode, Years\_at\_Company, Satisfaction\_Level, Average\_Monthly\_Hours, Promotion\_Last\_5Years, and Salary, and define the target variable (Attrition).
   * Split the data into training and testing sets using train\_test\_split.
4. **Define the Base Model**:
   * Use the DecisionTreeClassifier as the base estimator for the boosting algorithm.
5. **Implement Boosting**:
   * Use AdaBoostClassifier or GradientBoostingClassifier with the DecisionTreeClassifier as the base estimator.
   * Define the number of boosting rounds (n\_estimators) and learning rate.
   * Train the boosting model on the training data.
6. **Evaluate the Model**:
   * Make predictions on the test data.
   * Evaluate the model using metrics such as accuracy score, classification report, and confusion matrix.
7. **Plot Feature Distributions**:
   * Use seaborn to plot the distributions of features such as Age, Gender\_encode, Department\_encode, Job\_Title\_encode, and Satisfaction\_Level.
8. **Plot Confusion Matrix**:
   * Plot the confusion matrix for the Boosting model to visualize the classification performance.

**Choosing Features in Boosting**:  
Features are selected based on their relevance to the target variable. In this exercise,

features like Age, Gender\_encode, Department\_encode, Job\_Title\_encode,

Years\_at\_Company, Satisfaction\_Level, Average\_Monthly\_Hours,

Promotion\_Last\_5Years, and Salary were chosen based on their correlation with the

target variable Attrition.

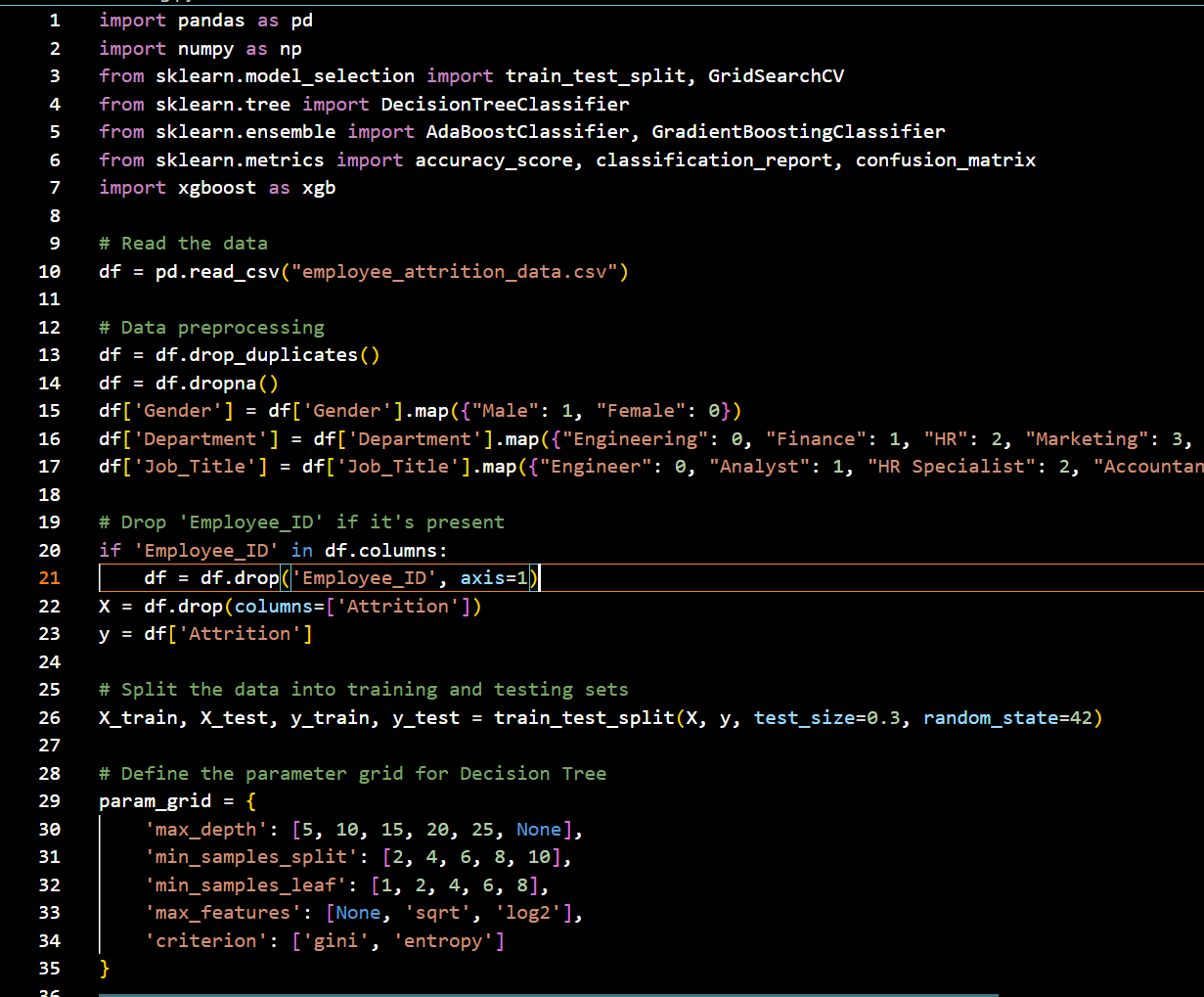
**When Can Overfitting Occur?**  
Overfitting can occur in Boosting if the model becomes too complex, particularly if too

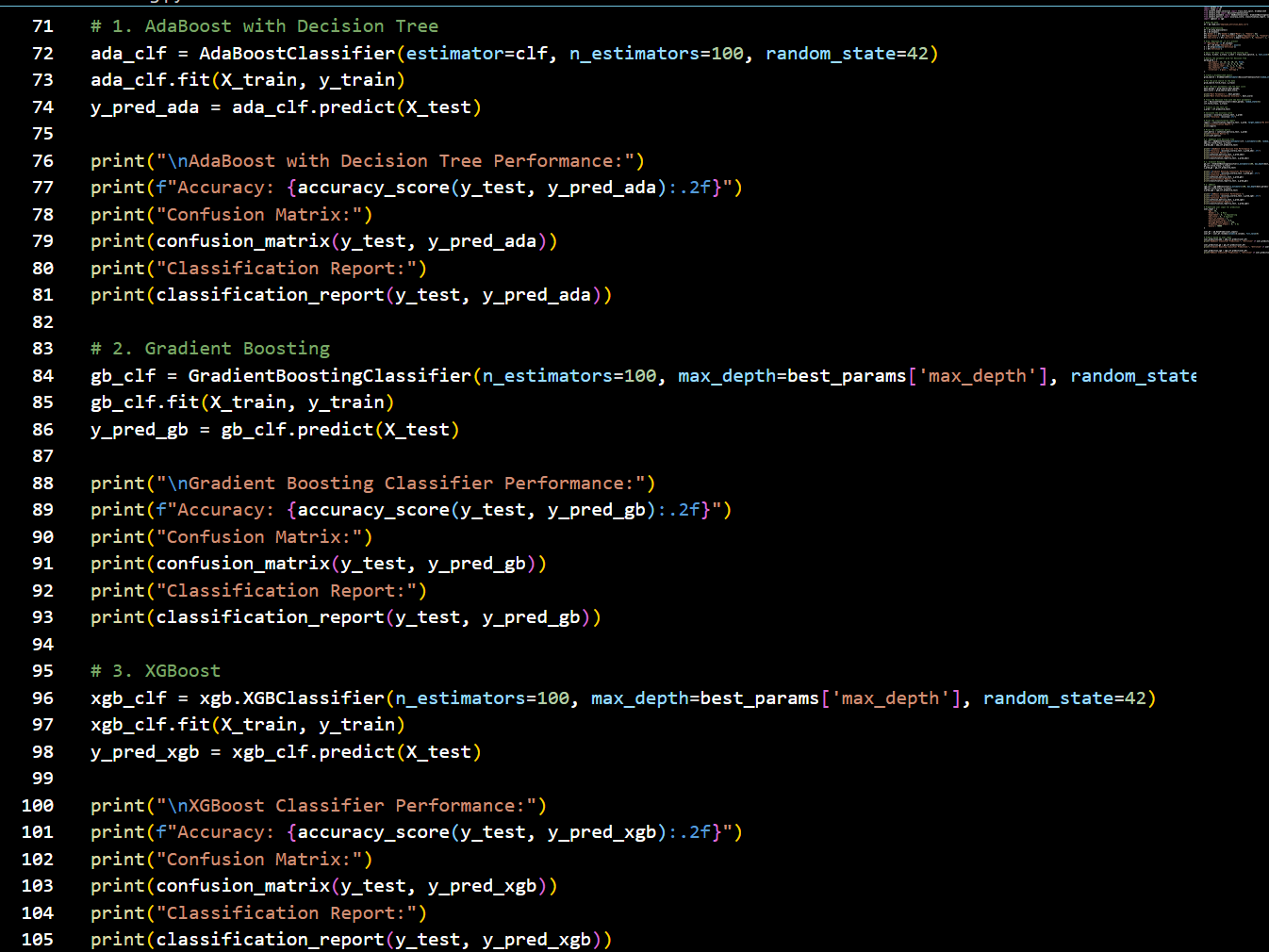
many boosting rounds are used, or if the base estimators are too deep. However,

regularization techniques such as limiting tree depth, learning rate adjustments, and

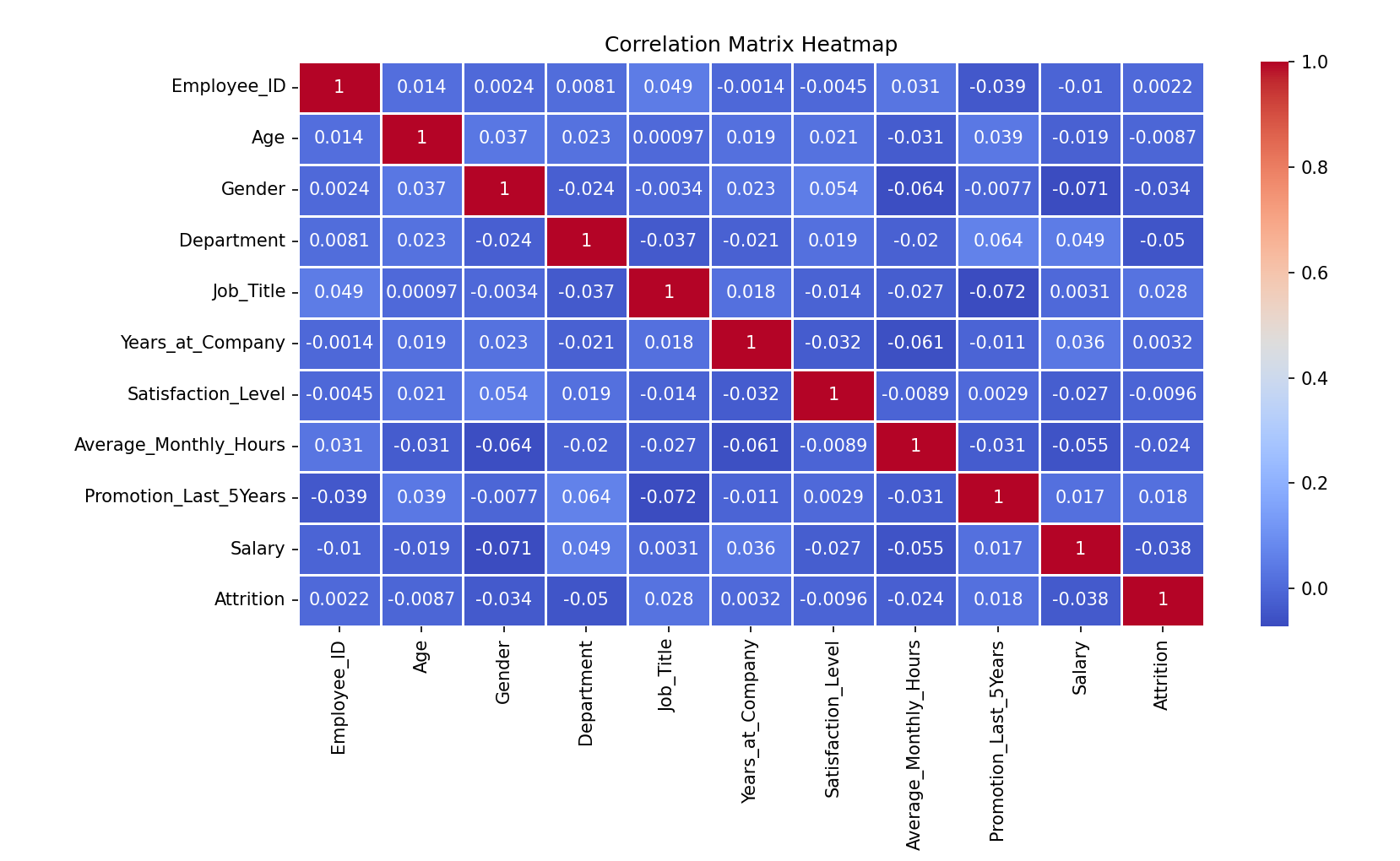
early stopping can help mitigate overfitting in Boosting mode

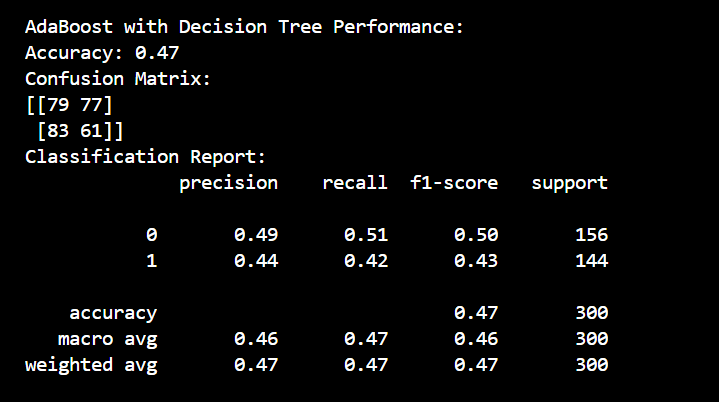
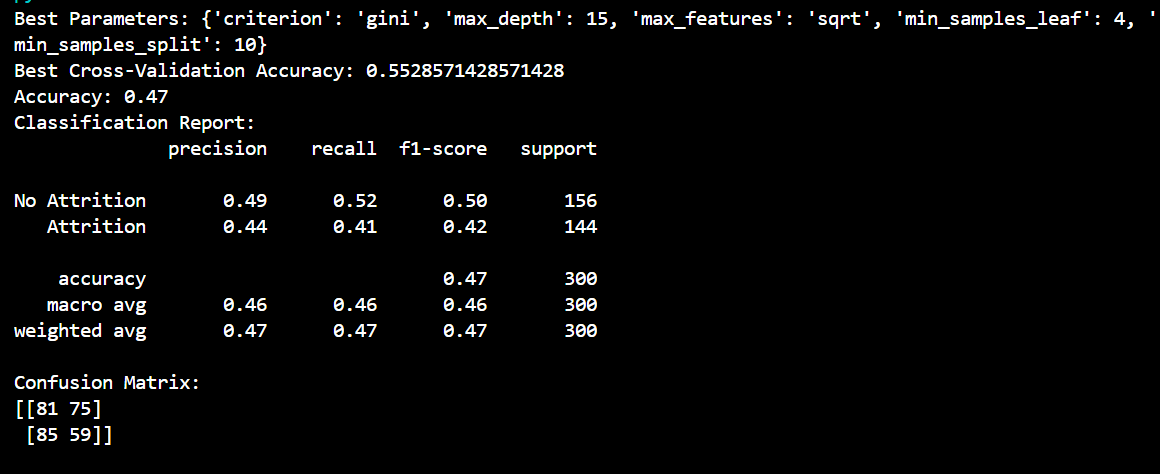
Code:

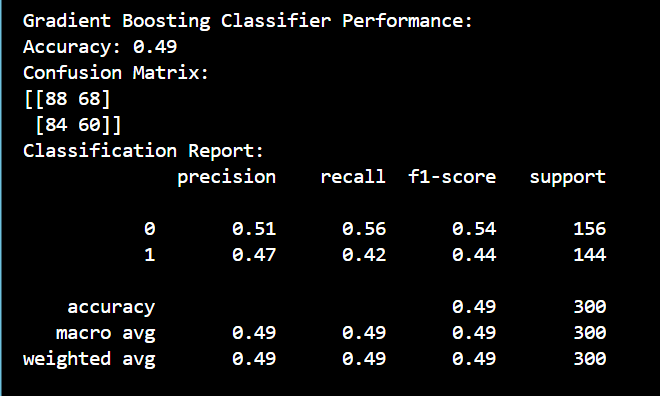
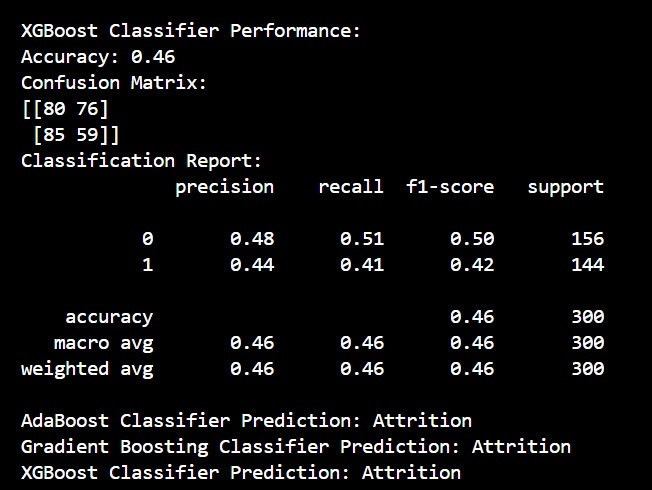


Output:





**Conclusion**:

Boosting with Decision Trees is a powerful technique that enhances model

performance by focusing on correcting errors made by previous models. By sequentially

improving the model, Boosting often results in higher accuracy and robustness

compared to single models or even Bagging. However, careful tuning of

hyperparameters is essential to avoid overfitting and ensure the model generalizes well

to unseen data.