Random Forest Regression and AdaBoost Classification

[Total Points: 25]

Objective:

The objective of this assignment is to implement a Random Forest Regressor for a regression task using the Boston Housing dataset and an AdaBoost Classifier for a classification task using the Wine dataset.

Dataset:

Boston Housing Dataset:
The Boston Housing dataset contains information collected by the U.S Census Service concerning housing in the area of Boston, Massachusetts. It comprises 506 samples with 13 features each, aiming to predict the median value of owner-occupied homes (in thousands of dollars) given various features such as crime rate, property tax rate, and accessibility to highways and employment centers. This dataset is commonly used for regression tasks in machine learning.

Breast_Cancer Dataset:
The Breast Cancer dataset from scikit-learn contains features computed from digitized images of breast mass fine needle aspirates. It includes 30 features describing characteristics like mean radius, texture, perimeter, area, smoothness, compactness, concavity, and concave points, among others. This dataset is labeled with two classes: malignant and benign, intended for binary classification tasks.

Random Forest Regression (Boston Housing Dataset):

Data preparation: (2 points)

- Load the Boston Housing dataset using load_boston from sklearn.datasets.
- Split the dataset into training, validation, and testing sets with ratios of 70%, 15%, and 15% respectively.
- Randomly shuffle the data before splitting to ensure a random distribution across sets.

Model training and hyperparameter tuning (5 points)

- Train Random Forest Regression models (implemented using sklearn.ensemble.RandomForestRegressor) by performing hyperparameter tuning for the number of trees and maximum depth using GridSearchCV as follows:
  - 'N_estimators': [50, 100, 150]
  - 'Max_depth': [None, 5, 10, 15] (Setting 'max_depth' to `None’ allows the tree to grow until each leaf node is pure, which might result in a highly complex and overfitted model, especially for datasets with noise or irrelevant features)
  - ‘Max_features’: log2(number_of_features)
- Use the ‘squared_error’ as the criterion to compute the best splits of nodes for the decision tree. (see sklearn.ensemble.RandomForestRegressor criterion parameter for reference)
- Identify the best combination of hyperparameters.

Evaluation with trained model corresponding to best hyperparameters: (3 points)

- Perform inference using the already trained trees corresponding to the best combination of hyperparameters on the testing set.
  - Report the mean squared error (MSE) between the predicted and actual target values on the testing set.
  - Create a scatter plot visualizing the predicted target values versus the target values on the testing set.

AdaBoost Classification (Breast_Cancer Dataset):

Data preparation: (2 points)

- Load the breast_cancer dataset using load_breast_cancer from sklearn.datasets.
- Split the dataset into training, validation, and testing sets with ratios of 70%, 15%, and 15%, respectively.
- Randomly shuffle the data before splitting to ensure a random distribution across sets.

Model training and hyperparameter tuning: (8 points)

- Train AdaBoost Classifier (implemented from scratch) by hyperparameter tuning for the number of weak learners as follows
- Default hyperparameters typically include:
  - 'n_estimators': [50, 100, 150]
- Identify the best number of weak learners.

Evaluation with trained model corresponding to best hyperparameters: (2 points)

- Perform inference using the model trained with the best number of weak learners from the previous step on the test set.
- Generate a classification report, including precision, recall, F1-score, and accuracy, and confusion matrix based on the predictions.

Comparison with sklearn's implementation of Adaboost Classifier: (3 points)

- Compare your implementation with sklearn.ensemble.AdaBoostClassifier. Perform hyperparameter tuning using GridSearchCV over the following parameter grid:
    - 'n_estimators': [50, 100, 150]
- Identify the best number of weak learners, and perform inference with the corresponding trained AdaBoost Classifier on the test set.
- Generate a classification report, including precision, recall, F1-score, and accuracy, and confusion matrix based on the predictions and compare these values against those obtained using the custom implementation.

Submission:

A .zip file containing the Python source code (.py file), an IPython Notebook (.ipynb), and a PDF report file. The final name should follow the template: Assign-X_YourName.zip. For example, if your name is John Doe and the assignment number is 6, the filename should be: Assign-6_JohnDoe.zip.