Programming Assignment-4 (Homework): Ensemble Learning Algorithms (Bagging and Boosting)

Objective

This assignment aims to provide hands-on experience in implementing and understanding fundamental ensemble learning algorithms from scratch. You will be implementing a simple ensemble method on a multiclass classification problem and dataset to grasp the core concepts of combining multiple models for improved predictive performance.

Note: Use DecisionTreeClassifier from scikit-learn library. (Kindly avoid implementing a decision tree from scratch as it was already implemented in Tutorial-3. Only importing base models from scikit-learn is allowed. Do not import the Bagging or Boosting Classifier module directly from scikit-learn library)

DATASET-1 : Multiple Class with Multiple features - car_evaluation.csv The Car Evaluation Database contains examples with the structural information i.e., directly relates CAR to the six input attributes: buying, maint, doors, persons, lug_boot, safety.

Information about Attribute Information:

Class Values*(Decision about Car purchase):* unacc, acc, good, vgood (*Unacceptable, Acceptable, Good, Very Good*)

Attributes (Features)*: Depending upon the following attribute decision has to be made.*

1. buying: vhigh, high, med, low. (*Buying Price*)
2. maint: vhigh, high, med, low. (*Maintenance Cost*)
3. doors: 2, 3, 4, 5more. (*Number of Doors*)
4. persons: 2, 4, more. (number of persons)
5. lug_boot: small, med, big (size of luggage boot)
6. safety: low, med, high (safety)

Problem-1: Bagging (Bootstrap Aggregating) Classifier

Implement the Bagging algorithm from scratch using a simple base estimator (e.g., Decision Tree) on a multiclass classification problem.

Problem Description

In this section, you will implement Bagging. You will use a **DATASET-1** provided above for multiclass classification where multiple base models will be trained on different bootstrap samples of the data, and their predictions will be combined.

Instructions

Complete the following steps for implementing and evaluating the Bagging algorithm:

1. **Dataset Generation:**
    - Use **DATASET** provided for this problem.
    - Represent your data using Pandas DataFrames.
2. **Data Splitting:**
    - Split the dataset into training and testing sets (e.g., 80% training, 20% testing) using `sklearn.model_selection.train_test_split`.
3. **Bagging Algorithm Implementation:**
    - Implement the Bagging algorithm from scratch. Your implementation should include:
        - A method to create `n_estimators=10` bootstrap samples from the training data. For each bootstrap sample, randomly sampled with replacement from the original training set (overlap between samples is allowed).
        - A method to train a base estimator (e.g., a simple Decision Tree from `sklearn.tree.DecisionTreeClassifier` with `max_depth=3`) on each bootstrap sample.
        - A prediction method that combines the predictions of individual estimators (e.g., using majority voting for classification).
4. **Model Training:**
    - Train your custom Bagging model and Decision Tree Classifier on the training data.
5. **Prediction:**
    - Make predictions on the test set using your trained Bagging model and single Decision Tree.
6. **Evaluation:**
    - Calculate and report the accuracy score on the test set.

- Compare the performance of your Bagging model to a single base estimator (e.g., a single Decision Tree trained on the full training set). Discuss the observed differences.

- Discuss about performance of single decision tree vs Bagging model. Did the Bagging model perform better than a single decision tree? Why or Why not?

Problem-2: Boosting (AdaBoost) Classifier

Goal

Implement the AdaBoost algorithm from scratch using a simple base estimator (e.g., Decision Stump) on a multiclass classification problem.

Problem Description

In this section, you will implement AdaBoost, a sequential ensemble method. You will work with a **DATASET-1** provided for multiclass classification, iteratively training weak learners and adjusting sample weights.

Instructions

Complete the following steps for implementing and evaluating the AdaBoost algorithm:

1. **Dataset Generation:**
   - Use **DATASET** provided for this problem.
   - Represent your data using Pandas DataFrames.
2. **Data Splitting:**
   - Split the dataset into training and testing sets (e.g., 80% training, 20% testing) using `sklearn.model_selection.train_test_split`.
3. **AdaBoost Algorithm Implementation:**
   - Implement the AdaBoost algorithm from scratch. Your implementation should include:
     - Initialization of sample weights.
     - An iterative training process for `n_estimators=10` weak learners (e.g., a Decision Tree with `max_depth=1`, also known as a Decision Stump).
     - For each iteration:
       - Train a weak learner on the data with current sample weights.
       - Calculate the weighted error of the weak learner.

        - Calculate the "amount of say" (alpha) for the weak learner.
        - Update the sample weights based on correct and incorrect predictions.
        - Normalize the sample weights.
    - A prediction method that combines the predictions of individual weak learners, weighted by their "amount of say".
4. **Model Training:**
    - Train your custom AdaBoost model and Decision Tree on the training data.
5. **Prediction:**
    - Make predictions on the test set using your trained AdaBoost model and Decision Tree.
6. **Evaluation:**
    - Calculate and report the accuracy score on the test set for both Adaboost Model and Decision Tree.
    - Discuss how the sequential nature of AdaBoost differs from Bagging and its implications for performance.

- Discuss about performance of single decision tree vs Boosting model vs Bagging model. Did the AdaBoost model perform better than a single decision tree or Bagging model? Why or Why not?

Submission Guidelines

- Submit a single Jupyter Notebook with all your code.
- Clearly separate Problem-1 (Bagging) and Problem-2 (AdaBoost) with markdown headings.
- Ensure all code blocks are executed and produce the expected outputs.
- Add comments to explain your code, especially for custom implementations and complex logic.
- Ensure your code is well-structured and easy to read.
- For any plots, include appropriate titles, axis labels, and legends.

Evaluation Criteria

Your assignment will be evaluated based on the following:

- **Correctness:** Proper implementation of Bagging and AdaBoost algorithms from scratch.
- **Functionality:** The code runs without errors and produces expected outputs.

- **Evaluation Metrics:** Accurate calculation and reporting of accuracy scores.
- **Code Quality:** Readability, comments, and adherence to Python best practices.
- **Understanding:** Demonstrated understanding of the underlying concepts of both algorithms, including their differences and how they combine weak learners.

Due Date

Please submit your completed Jupyter Notebook by Oct 25, 2025 11:45 PM GMT+5:30.