Homework 3: Boosting, Bagging, and Random Forests from Scratch (100 points)

Prof. Amine Aboussalah
Due Date: Monday 17th November 2025.
Submission will be done on Brightspace.

Datasets

- **Boosting (Gradient Boosting Regression)**: California Housing Dataset (Scikit-learn)
- **Random Forests:** UCI Bank Marketing Dataset
- **Bagging**: California Housing Dataset (Scikit-learn)

Overview

This homework will help better understand supervised learning models and ensemble methods. You will implement and evaluate three main algorithms:

- Boosting
- Bagging
- Random Forests

All implementations must be from scratch using `numpy` for linear algebra (and optionally `pandas`, `matplotlib`). You may not use machine learning libraries like scikit-learn, TensorFlow, or PyTorch for model implementation. Metrics should also be coded from scratch.

Task 1: Gradient Boosting with Decision Trees (35 points)

Objective

Implement a gradient boosting regressor using shallow decision trees as weak learners.

Instructions

Implementation (20 points)

- Implement regression trees with:
    - MSE-based splitting (5 points)
    - Support for max depth and minimum samples per split (3 points)
    - Recursive tree building and prediction (5 points)
- Implement gradient boosting with:
    - Residual fitting at each stage (3 points)
    - Learning rate control and iterative updates (2 points)
    - Subsampling (2 points)

Application & Evaluation (15 points)

- Load the California Housing Dataset (subset if needed) (2 points)
- Split into 70/30 train/test sets (1 point)
- Train models with different numbers of boosting rounds (e.g., 5, 10, 50, 100) (3 points)
- Evaluate performance using RMSE, MAE, and R^2 and interpret results (3 points)
- Plot:
    - Predicted vs. true values (scatter plot) (2 points)
    - Training error across boosting rounds (2 points)

- Compare with scikit-learn's `GradientBoostingRegressor` and discuss (2 points)

Task 2: Random Forest Implementation (35 points)

Objective

Build a random forest classifier using your decision tree implementation from Homework 2 as the base learner.

Instructions

Implementation (20 points)

- Implement the random forest algorithm:
  - Implement bootstrap sampling (5 points)
  - Random feature selection for each split (3 points)
  - Train an ensemble of decision trees (5 points)
  - Aggregate predictions using majority voting (classification) (4 points)
  - Include model parameters: number of trees, max depth, number of features per split (3 points)

Application & Evaluation (15 points)

- Load and preprocess the **Bank Marketing dataset** (binary classification: subscribe vs. not) (2 points)
- Train and test your random forest (80/20 split) with varying number of trees (2, 5, 10) (4 points)
- Evaluate performance using:
  - Accuracy, precision, recall, F1-score, ROC/AUC (3 points)
  - Comparison with a single decision tree (3 points)
  - Comparison with scikit-learn's built-in random forest classifier (3 points)

Task 3: Bagging with K-Nearest Neighbors Regression (30 points)

Objective

Use bagging (bootstrap aggregation) to improve the performance of a high-variance regression model K-Nearest Neighbors (KNN). Compare it to a single KNN model as a baseline.

Instructions

Implementation (16 points)

1. **KNN Regression (Baseline)**
   - Implement KNN regression (or use scikit-learn) **(3 points)**
   - Support hyperparameter k (number of neighbors) **(1 point)**
   - **Clarification:** KNN (K-Nearest Neighbors) regression is a supervised machine learning algorithm that predicts a continuous value for a new data point by averaging the values of its k nearest neighbors from the training set. It works by first finding the k data points in the training data that are closest to the new point using a distance metric like Euclidean distance, and then calculating the mean or median of those k neighbors' target values to make the prediction.
2. **Bagging Ensemble**
   - Bootstrap sampling from the training set **(2 points)**
   - Train multiple KNN regressors on different bootstrap samples **(2 points)**
   - Average predictions over all base regressors **(2 points)**
   - Support hyperparameters: number of models, sampling ratio, k for KNN **(2 points)**
   - Compare results with different distance metrics (Euclidean, Manhattan) **(2 points)**
   - Discuss effect of k on bias and variance **(2 points)**

Application & Evaluation (14 points)

1. **Dataset**
   - California Housing dataset (or any tabular regression dataset) **(1 point)**
   - Train-test split: 70/30 **(1 point)**
2. **Models to Compare (3 points)**
   - Single KNN regressor
   - Bagged ensemble of KNN regressors (vary number of models: 2, 5, 10)
3. **Evaluation Metrics**
   - RMSE, MAE, R^2, interpret the metrics in details **(3 points)**

4. **Visualization**
   - Scatter plot: predicted vs. true values **(2 points)**

- Prediction intervals across models (standard deviation of predictions) **(2 points)**

5. **Analysis**

    - Compare performance of single vs. bagged KNN and discuss the effect of ensemble size on variance reduction and overall performance **(2 points)**

General Requirements (up to -10 points penalty)

- Submit one **Jupyter Notebook (LASTNAME_FIRSTNAME)**
- All code must be **original**
  Use markdown cells to document your work clearly
- Comment your code and handle potential exceptions
- Report all chosen hyperparameters and justify your decisions