BACS - HW (Week 17)

This week, we will look at a dataset of US health insurance premium charges. We will build models that can *predict* what someone's insurance charges might be, given several factors about them. You download the dataset, and find more information about it, at the Kaggle platform where machine learning people like to host challenges and share datasets: https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset

Setup: Download the data, load it in your script, and omit any rows with missing values (NAs)

Note: The values of charges are large, so MSE values will be very large. This week let's use RMSE, or the Root-Mean-Square Error (*the square-root of MSE*), so we have smaller numbers.

Question 1) Create some explanatory models to learn more about charges:

a. Create an OLS regression model and report which factors are significantly related to charges
b. Create a decision tree (specifically, a *regression tree*) with default parameters to `rpart()`.
    i. Plot a visual representation of the tree structure
    ii. How *deep* is the tree (see nodes with "decisions" – ignore the leaves at the bottom)
    iii. How many leaf groups does it suggest to bin the data into?
    iv. What conditions (combination of decisions) describe each leaf group?

Question 2) Let's use LOOCV to see how how our models perform predictively overall

a. What is the $RMSE_{out}$ for the OLS regression model?
b. What is the $RMSE_{out}$ for the decision tree model?

For bagging and boosting, we will only use split-sample testing to save time: partition the data to create training and test sets using an 80:20 split. Use the regression model and decision tree you created in Question 1 for bagging and boosting.

Question 3) Let's see if bagging helps our models

a. Implement the `bagged_learn(…)` and `bagged_predict(…)` functions using the hints in the class notes and help from your classmates on Teams. Feel free to share your code on Teams to get feedback, or ask others for help.
b. What is the $RMSE_{out}$ for the bagged OLS regression?
c. What is the $RMSE_{out}$ for the bagged decision tree?

(see next page)

Question 4) Let's see if boosting helps our models. You can use a learning rate of 0.1 and adjust it if you find a better rate.

a. Write boosted_learn(…) and boosted_predict(…) functions using the hints in the class notes and help from your classmates on Teams. Feel free to share your code generously on Teams to get feedback, or ask others for help.
b. What is the $RMSE_{out}$ for the boosted OLS regression?
c. What is the $RMSE_{out}$ for the boosted decision tree?

Question 5) Let's engineer the best predictive decision trees. Let's repeat the bagging and boosting decision tree several times to see what kind of base tree helps us learn the fastest. But this time, split the data 70:20:10 — use 70% for training, 20% for fine-tuning, and use the last 10% to report the final $RMSE_{out}$.

a. Repeat the bagging of the decision tree, using a base tree of maximum depth 1, 2, … n, keep training on the 70% training set while the $RMSE_{out}$ of your 20% set keeps dropping; stop when the $RMSE_{out}$ has started increasing again (show prediction error at each depth). Report the final $RMSE_{out}$ using the final 10% of the data as your test set.
b. Repeat the boosting of the decision tree, using a base tree of maximum depth 1, 2, … n, keep training on the 70% training set while the $RMSE_{out}$ of your 20% set keeps dropping; stop when the $RMSE_{out}$ has started increasing again (show prediction error at each depth). Report the final $RMSE_{out}$ using the final 10% of the data as your test set.