Homework 4 (due 10/14)

DS-210 @ Boston University

Fall 2022

Before you start. . .

Collaboration policy: You may verbally collaborate on required homework problems. However, you must write your solutions independently without showing them to other students. If you choose to collaborate on a problem, you are allowed to discuss it with at most 2 other students currently enrolled in the class.

The header of each assignment you submit must include the field "Collaborators:" with the names of the students with whom you have had discussions concerning your solutions. If you didn't collaborate with anyone, write "Collaborators: none." A failure to list collaborators may result in credit deduction.

You may use external resources such as software documentation, textbooks, lecture notes, and videos to supplement your general understanding of the course topics. You may use references such as books and online resources for well known facts. However, you must always cite the source.

You may not look up answers to a homework assignment in the published literature or on the web. You may not share written work with anyone else.

Submitting: Solutions should be submitted via Gradescope. Please submit a solution to this homework as a single IPython notebook (.ipynb) as much as this is possible.

Grading: Whenever we ask for a solution, you may receive partial credit if your solution is not sufficiently efficient or close to optimal. For instance, if we ask you to solve a specific problem that has a polynomial– time algorithm that is easy to implement, but the solution you provide is exponentially slower, you are likely to receive partial credit.

Late submission policy: No extensions, except for extraordinary circumstances that are pre-approved.

Questions

1. (8 points) It is important—especially for the second part of this course—that you know how to edit text files and use source control. Install an appropriate editor that allows you to edit and save files and the *git* source control package. Explain which editor you have

decided on and how to save files from the editor and find them using a command terminal on your computer. For Windows users installing the git for Windows source control which comes with the git bash shell is recommended though other options are possible. Include the output of the command *git --version* in your writeup.

2. (8 points) Install Rust, which we will use in the second part of the course. You can do it by following instructions at https://www.rust-lang.org/tools/install. (There may exist alternate methods for your operating system. You can find a discussion of some of them in the official documentation.)
   After you are done installing Rust, you should be able to run the following commands from the command line: rustc and cargo. In particular, run both *rustc --version* and *cargo --version* and include their output in your homework solution to prove that you have successfully installed Rust and to receive credit for this part of the homework.
   Note: https://doc.rust-lang.org/book/ch01-01-installation.html also discusses various options for installing Rust.
3. (12 points) In this problem, you are asked to use decision trees for regression with two different target loss functions: minimum square error and minimum absolute error. Design a function f : [0, 1] → [0, 1] such that when you sample input points X_i from [0, 1] and use the sequence (X_i , f(X_i)) as your input, you are likely to see the difference between the two loss functions with six leafs (i.e., with max_leaf_nodes=6).
   Among other things, your solution should contain:
    a. a plot of f and the two functions resulting from the training process under different loss functions,
    b. an explanation of the differences and how they are a result of differences between the two loss functions.

   Note: Make sure you read the documentation for *sklearn.tree.DecisionTreeRegressor* including usage examples
   https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html#sphx-glr-auto-examples-tree-plot-tree-regression-py
4. (12 points) Recall that *numpy.polyfit* can be used to find a bounded degree polynomial that fits data well. Show clear examples of overfitting and underfitting occurring when learning a function f from [0, 1] to R. Support your examples with error estimates. Your examples can feature noisy data points.