Machine Learning
for Trading

PROJECT 4: DEFEAT LEARNERS

Table of Contents

- Overview
- About the Project
- Your Implementation
- Contents of Report
- Testing Recommendations
- Submission Requirements
- Grading Information
- Development Guidelines
- Optional Resources

REVISIONS

This assignment is subject to change up until 3 weeks prior to the due date. We do not anticipate changes; any changes will be logged in this section.

1 OVERVIEW

In this assignment, you will generate data that you believe will work better for one learner than another. This will test your understanding of the strengths and weaknesses of various learners. You will submit the code for the project in Gradescope SUBMISSION. There is no report associated with this assignment.

1.1 Learning Objectives

The specific learning objectives for this assignment are focused on the following areas:

- **Supervised Learning – Learner Strengths and Weaknesses**: Demonstrate an understanding the relative strengths and weaknesses of a Decision Tree learner as compared to a Linear Regression Learner.

2 ABOUT THE PROJECT

In this project, you will evaluate the relative strengths of two types of supervised learners: a linear regression learner and a decision tree learner. Specifically, you will write a module that generates datasets that will be used by the learners. Your goal is to 1) produce datasets that enable a linear regression learner to consistently outperform a decision tree learner and 2) produce datasets that enable a decision tree learner to consistently outperform a linear regression learner.

Your data generation implementation should use a random number generator as part of its data generation process. We will pass your generators a random number seed. Whenever the seed is the same you should return exactly the same data set. Different seeds should result in different data sets. This project may require readings or additional research to ensure an understanding of the relative strengths and weaknesses of the different types of learners.

3 YOUR IMPLEMENTATION

You will create a Python program called gen_data.py, which contains three functions according to this API specification. The DTLearner and LinRegLearner specification are also provided in the API specification as a reference. This gen_data.py file must implement the three functions given in the API specification:

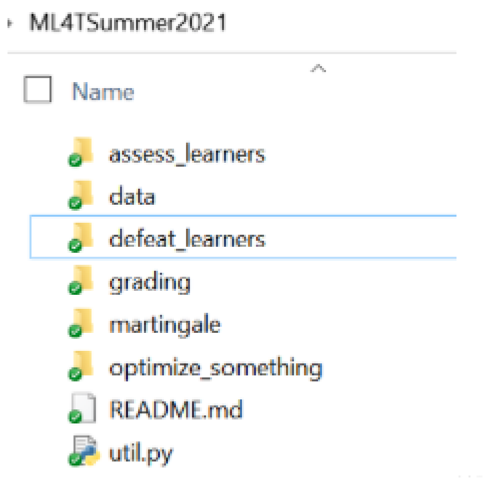
- best_4_lin_reg
- best_4_dt
- author

Your data generation should use a random number generator as part of its data generation process. We will pass your generators a random number seed (as shown below). Whenever the seed is the same you must return exactly the same data set. Different seeds must result in different data sets.

Before the deadline, make sure to pre-validate your submission using Gradescope TESTING. Once you are satisfied with the results in testing, submit the code to Gradescope SUBMISSION. **Only code submitted to Gradescope SUBMISSION will be graded. If you submit your code to Gradescope TESTING and have not also submitted your code to Gradescope SUBMISSION, you will receive a zero (0).**

To make it easier to get started on the project and focus on the concepts involved, you will be given a starter framework. This framework assumes you have already set up the local environment and ML4T Software. The framework for Project 4 can be obtained from: Defeat_Learners_2022Summer.zip.

Extract its contents into the base directory (e.g., ML4T_2022Summer). This will add a new folder called "defeat_learners" to the course directory structure.



ML4TSummer2021

Name

assess_learners
data
defeat_learners
grading
martingale
optimize_something
README.md
util.py

Within the defeat_learners folder are several files:

- defeat_learners/gen_data.py – An implementation of the code you are supposed to provide: It includes two functions that return a data set and a third function that returns a user ID. Note that the data sets those functions return DO NOT satisfy the requirements for the homework. But they do show you how you can generate a data set.
- defeat_learners/LinRegLearner.py – Our friendly, working, correct, linear regression learner. It is used by the grading script. Do not rely on local changes you make to this file, as you may only submit gen_data.py.
- defeat_learners/DTLearner.py – A working, but INCORRECT, Decision Tree learner. Replace it with your working, correct DTLearner.

- defeat_learners/testbest4.py – Code that calls the two data set generating functions and tests them against the two learners. Useful for debugging.
- defeat_learners/grade_best4.py – The local grading / pre-validation script. This is the same script that will be executed in the Gradescope TESTING Environment

3.2 Construct the Best for Linear Regression Learner and Decision Tree Learner Datasets

Implement the **best_4_lin_reg()** function such that when the generated dataset is passed to both learners, the LinRegLearner performs significantly better than the DTLearner (see rubric below for specific performance details).

Also implement the **best_4_dt()** function such that when the generated dataset is passed to both learners, the DTLearner performs significantly better than the LinRegLearner (see rubric below for specific performance details).

Each dataset must include no fewer than 2 and no more than 10 features (or “X”) columns. The dataset must contain 1 target (or “Y”) column. The Y column must contain real numbers. Y values may not be hard-coded and must be generated by the X value. Each dataset must contain no fewer than 10 and no more than 1000 examples (i.e., rows). While you are free to determine these sizes, they may not vary between generated datasets.

3.2.1 Example

```
X1, Y1 = best_4_lin_reg( seed = 5 )
X1, Y1 = best_4_dt( seed = 5 )
```

defeat_learner_example.py hosted with ❤ by **GitHub** **view raw**

3.3 Implement the author() function (Up to 10 point penalty)

You must implement a function called author() that returns your Georgia Tech user ID as a string. This is the ID you use to log into Canvas. It is not your 9 digit student number. Here is an example of how you might implement author():

```
def author():
  return 'tb34' # replace tb34 with your Georgia Tech username.
```

author_example.py hosted with ❤ by **GitHub** **view raw**

Implementing this method correctly does not provide any points, but there will be a penalty of up to –10 points for not implementing the author function.

3.4 Technical Requirements

The following technical requirements apply to this assignment

1. You will use your DTlearner from Project 3 and the provided LinRegLeaner during development, local testing, and any testing performed in the Gradescope TESTING environment.
2. The decision tree learner (DTLearner) will be instantiated with leaf_size=1. (Note: We expect you to fix your DTLearner implementation from Project 3 if it was incorrect since this code is required to run/test on Gradescope TESTING.)
3. You will use the LinRegLearner that is provided as part of the project setup .zip file.
4. The dataset must be a regression dataset (i.e., the target "Y" column must contain real numbers).
5. You should set the seed every time to ensure dataset reproducibility. The grader will pass in a seed, and if your code does not produce the same result for the same seed, and different results for different seeds, you will receive a points deduction.
6. No part of this project implementation may read data from files. All data must be generated by the functions.
7. Your code must run in less than 5 seconds per test case.
8. The code you submit should NOT generate any output: No prints, no charts, etc.
9. gen_data() must not import any learner.

3.5 Hints & Suggestions

If you are concerned about the performance of your DTLearner, you may develop additional tests in your local environment using the Scikit-learn decision tree learner. This learner will not be used within the Gradescope environment. We also encourage additional testing with different seeds beyond the grade scripts given. The private tester will use additional tests.

4 CONTENTS OF THE REPORT

There is no report associated with this assignment.

5 TESTING REQUIREMENTS

To test your code, you can modify the provided testbest4.py file. You are encouraged to perform any tests necessary to instill confidence that the code will run properly when submitted for grading and will produce the required results.

Additionally, we have provided the grade_best4.py file that can be used for your tests. This file is the same script that will be run when the code is submitted to Gradescope TESTING. This file is not a complete test suite and does not match the more stringent private grader that is used in Gradescope SUBMISSION. To run and test that the file will run from within the defeat_learners directory, use the command:

```
PYTHONPATH=../:. python grade_best4.py
```

grade_best_4_pythonpath hosted with ❤ by GitHub view raw

In addition to testing on your local machine, you are encouraged to submit your files to Gradescope TESTING, where some basic pre-validation tests will be performed against the code. No credit will be given for coding assignments that do not pass this pre-validation. **Gradescope TESTING does not grade your assignment.** The Gradescope TESTING script is not a complete test suite and does not match the more stringent private grader that is used in Gradescope SUBMISSION. Thus, the maximum Gradescope TESTING score of 100, while instructional, does not represent the minimum score one can expect when the assignment is graded using the private grading script. You are encouraged to develop additional tests to ensure that all project requirements are met.

NOTE: **When submitting to Gradescope TESTING, you will also need to submit your DTLearner.py file in addition to the gen_data.py file**.

You are allowed **unlimited** resubmissions to Gradescope **TESTING**. Please refer to the Gradescope Instructions for more information.

6 SUBMISSION REQUIREMENTS

This is an individual assignment. All work you submit should be your own. Make sure to cite any sources you reference and use quotes and in-line citations to mark any direct quotes.

Assignment due dates in your time zone can be found by looking at the Project in the Assignment menu item in Canvas (ensure your Canvas time zone settings are set up properly). This date is 23:59 AOE converted to your time zone. Late submissions are allowed for a penalty. The times and penalties are as follows:

- -10% Late Penalty: +1 Hour late: submitted by 00:59 AOE (next day)
- -25% Late Penalty: +12 Hours Late: submitted by 11:59 AOE (next day)
- -50% Late Penalty: +24 Hours Late: submitted by 23:59 AOE (next day)
- -100% Late Penalty: > 24+ Late: submitted after 23:59 AOE (next day)

Assignments received after Monday at 23:59 AOE (even if only by a few seconds) are not accepted without advanced agreement except in cases of medical or family emergencies. In the case of such an emergency, please contact the Dean of Students.

6.1 Report Submission

There is no report submission associated with this assignment.

6.2 Code Submission

This class uses Gradescope, a server-side auto-grader, to evaluate your code submission. No credit will be given for code that does not run in this environment and students are encouraged to leverage Gradescope TESTING prior to submitting an assignment for grading. **Only code submitted to Gradescope SUBMISSION will be graded. If you submit your code to Gradescope TESTING and have not also submitted your code to Gradescope SUBMISSION, you will receive a zero (0).**

Please submit the following files to Gradescope **SUBMISSION**:

gen_data.py

Do not submit any other files.

Important: You are allowed a MAXIMUM of three (3) code submissions to Gradescope SUBMISSION.

7 GRADING INFORMATION

The submitted code (which is worth 100% of your grade) is run as a batch job after the project deadline. The code represents 100% of the assignment grade will be graded using a rubric design to mirror the implementation details above. Deductions will be applied for unmet implementation requirements or code that fails to run.

We do not provide an explicit set timeline for returning grades, except that all assignments and exams will be graded before the institute deadline (end of the term). As will be the case throughout the term, the grading team will work as quickly as possible to provide project feedback and grades.

Once grades are released, any grade-related matters must follow the Assignment Follow-Up guidelines and process alone. Regrading will only be undertaken in cases where there has been a genuine error or misunderstanding. Please note that requests will be denied if they are not submitted using the Summer 2022 form or do not fall within the timeframes specified on the Assignment Follow-Up page.

7.1 Grading Rubric

7.1.1 Report [0 points]

There is no report associated with this project.

7.1.2 Code

There is no separate code section associated with this project.

7.1.3 Auto-Grader (Private Grading Script) [100 points]

Deductions will be applied if any of the following occur:

- If either dataset contains fewer or more than the allowed number of samples. (-20 points each)
- If either dataset contains fewer or more than the allowed number of dimensions in X. (-20 points each)
- If, when the seed is the same, the best_4_lin_reg dataset generator does not return the same data. (-20 points)
- If, when the seed is the same, the best_4_dt dataset generator does not return the same data. (-20 points)
- If, when the seed is different, the best_4_lin_reg dataset generator does not return different data. (-20 points)
- If, when the seed is different, the best_4_dt dataset generator does not return different data. (-20 points)
- If the author() method is not implemented. (-10 points)
- If the code attempts to import a learner. (-10 points up to -100 if the code crashes)

We will test your code against the following cases. Each case will be deemed “correct” if:

For best_4_lin_reg (1 test case)

- We will call best_4_lin_reg 15 times, and select the 10 best datasets. For each successful test +5 points (total of 50 points)

- For each test case, we will randomly select 60% of the data for training and 40% for testing.
- Success for each case is defined as: RMSE LinReg < RMSE DT * 0.9

For best_4_dt (1 test case)

- We will call best_4_dt 15 times, and select the 10 best datasets. For each successful test +5 points (total of 50 points)
- For each test case, we will randomly select 60% of the data for training and 40% for testing.
- Success for each case is defined as: RMSE DT < RMSE LinReg * 0.9

8 DEVELOPMENT GUIDELINES (ALLOWED & PROHIBITED)

See the Course Development Recommendations, Guidelines, and Rules for the complete list of requirements applicable to all course assignments. **The Project Technical Requirements are grouped into three sections: Always Allowed, Prohibited with Some Exceptions, and Always Prohibited**.

The following exemptions to the Course Development Recommendations, Guidelines, and Rules apply to this project:

- You may not read any files or use any routines or functions to read files.
- You may use the decision tree implementation from the scikit-learn package for additional testing in your **local environment** alone.

9 OPTIONAL RESOURCES

Although the use of these or other resources is not required; some may find them useful in completing the project or in providing an in-depth discussion of the material.

- Mitchell, Machine Learning (Chapter 3)
- James, D. Witten, T. Hastie, R. Tibshirani (2017), An Introduction to Statistical Learning (Chapters 2, 3, and 8)

Videos:

- Decision Tree Videos, Charles Isbell and Michael Littman, Georgia Tech ML 7641

- Decision Tree Video (Part 1, staring around 0:40 minutes), Tom Mitchell, CMU 601
- Decision Tree Video (Part 2), Tom Mitchell, CMU 601