ECP 3004: Python for Business Analytics
Department of Economics
College of Business
University of Central Florida
Spring 2021

Assignment 3

Due Sunday, February 14, 2021 at 11:59 PM
in your GitHub repository

Instructions:

Complete this assignment within the space on your *private* GitHub repo (not a fork of the course repo ECP3004S21!) in a folder called `assignment_03`. In this folder, save your answers to Questions 1 and 2 in a file called `my_A3_functions.py`, following the sample script in the folder `assignment_03` in the course repository. When you are finished, submit it by uploading your files to your GitHub repo using any one of the approaches outlined in Question 3. You are free to discuss your approach to each question with your classmates but you must upload your own work.

Please note: In computer programming, many small details are very important. A file with the wrong name in the wrong folder will not run, even if the functions work perfectly.

Question 1:

Follow the function design recipe to define functions for all of the following Exercises. For each function, create three examples to test your functions. Record the definitions in the sample script `my_A3_functions.py`

Exercise 1 Write a python function `quad_roots_1()` that will calculate the real-valued roots of the quadratic equation, i.e. the values of x such that $ax^2 + bx + c = 0$. The return value should be a list of two roots, according to the positive and negative roots of the quadratic formula. Your function should have four arguments: the first is the value on the horizontal axis, x, the rest are the coefficients (a, b, c) in the quadratic equation, in that order.

Exercise 2 The discriminant of the quadratic equation is the value $b^2 - 4ac$. What happens to your answer in Example 1 when $b^2 - 4ac < 0$? Write a new python function `quad_roots_real()` that will not throw an error for any value of $b^2 - 4ac$. Use the value `None` to indicate that there are no real roots.

Exercise 3 In Assignment 2, you wrote a function `utility()` that calculated the value of the Cobb-Douglass utility function $u(x, y; \alpha) = x^{\alpha} y^{1-\alpha}$. In this function, the first two arguments are x and y, respectively, and the third is α. Write an augmented version of the function called `utility_positive()` that returns the same value as `utility()` when x, y and α (and $1 - \alpha$) are non-negative numbers but returns the value `None` otherwise. For each case of negative

numbers, make your function print a message that tells the user what is wrong with the inputs.

Exercise 4 The likelihood function of the logistic regression model is used to estimate coefficients in logistic regression. Logistic regression is used to model binary events, i.e. whether or not an event occurred. For each observation i, the observation y_i equals 1 if the event occurred and 0 if it did not. Build on the function from Assignment 2 and write a python function `logit_like()` that calculates the log-likelihood of observation (y_i, x_i). That is, it returns the log of the function $\ell(x_i; \beta_0, \beta_1)$ if $y_i = 1$ or the log of the function $(1 - \ell(x_i; \beta_0, \beta_1))$ if $y_i = 0$. For reference, the logit link function is defined as

$$\ell(x_i; \beta_0, \beta_1) = Prob(y = 1|x) = \frac{e^{x_i\prime\beta}}{1 + e^{x_i\prime\beta}} = \frac{e^{\beta_0 + x_i\beta_1}}{1 + e^{\beta_0 + x_i\beta_1}}.$$

This function will have four arguments, $(y_i, x_i; \beta_0, \beta_1)$, in that order.

Question 2:

For all of the Exercises in Question 1, use your examples to test the functions you defined. Make sure to record the examples in the docstring within each function and don't forget the leading string `>>>` . Complete the code at the bottom of your `my_A3_functions.py` script so that it will make the comparisons between your expected answers and the output from your functions. When you run the whole block of code at the bottom, it should show the results of all your comparisons.

Question 3:

Push your completed files to your GitHub repository following one of these three methods.

Method 1: In a Browser

Upload your code to your GitHub repo using the interface in a browser.

1. Browse to your `assignment_0X` folder in your repository (the "X" corresponds to Assignment X.).
2. Click on the "Add file" button and select "Upload files" from the drop-down menu.
3. Revise the generic message "Added files via upload" to leave a more specific message. You can also add a description of what you are uploading in the field marked "Add an optional extended description..."
4. Press the button "'Commit changes," leaving the buton set to "Commit directly to the `main` branch."

Method 2: With GitHub Desktop

Upload your code to your GitHub repo using the interface in GitHub Desktop.

1. Save your file within the folder in your repository within the folder referenced in GitHub Desktop.

2. When you see the changes in GitHub Desktop, add a description of the changes you are making in the bottom left panel.

3. Press the button "Commit to main" to commit those changes.

4. Press the button "Push origin" to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.

Method 3: At the Command Line

Push your code directly to the repository from the command line in a terminal window, such as GitBash on a Windows machine or Terminal on a Mac.

1. Open GitBash or Terminal and navigate to the folder inside your local copy of your git repo containing your assignments. Any easy way to do this is to right-click and open GitBash within the folder in Explorer. A better way is to navigate with UNIX commands, such as `cd`.

2. Enter `git add .` to stage all of your files to commit to your repo. You can enter `git add my_filename.ext` to add files one at a time, such as `my_functions.py` in this Assignment.

3. Enter `git commit -m "Describe your changes here"`, with an appropriate description, to commit the changes. This packages all the `add`ed changes into a single unit and stages them to `push` to your online repo.

4. Enter `git push origin main` to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.