QMB 6358: Software Tools for Business Analytics
College of Business
University of Central Florida
Fall 2021

Final Examination

Due Thursday, December 2, 2021 at 9:59 PM
in your GitHub repo.

Instructions:

Complete this assignment within the space on your GitHub repo in a folder called `final_exam`. You may organize your files any way you like but leave your answers to all questions in this folder.

All of your responses can be completed using the language of your choice, as long as your solutions meet the specifications in each question. Store any printed output by writing or pasting into a document of your choice or pasting comments in your code. This output can also be automated by redirecting ouput from a script in Question 6.

When you are finished, submit your code and any other documents by pushing your changes to your GitHub repo, following the instructions in Question 7. Complete these exercises individualy and `git push` your own work.

Part A: Data Handling and Regression Modelling

Estimate the best regression model you can by solving as many of Questions 1 to 4 as you can. You do not necessarily have to solve them in order.

Question 1:

The folder `final_exam` contains three `.csv` files: `applications.csv`, `credit_bureau.csv`, and `demographic.csv`. The first dataset `applications.csv` contains the following variables.

`app_id`	=	a unique key for each customer who applied for credit
`ssn`	=	the social security number
`zip_code`	=	the the zip code in which the applicant resides
`income`	=	the applicant's reported income
`homeownership`	=	a categorical variable that indicates whether an applicant owns or rents a home
`purchases`	=	the monthly value of purchases on the account
`credit_limit`	=	the maximum amount that an applicant is approved to spend

Use this dataset to estimate a regression model to predict the monthly amount of `purchases` for each customer.

a) Read in the `applications.csv` dataset and store it in a data frame called `applications` in your workspace.

b) Calculate and store the printed output from either a `summary` of the data or `describe` the data, according to your choice of software. Use this to get familiar with the contents of the dataset.

c) Estimate a regression model to predict `purchases` as a function of the other variables in the dataset. Ignore the variables `app_id`, `ssn` and `zip_code`, which are keys for databases. Store the printed estimation output with the `print` and/or `summary` command, as appropriate.

Question 2:

Now use two files `applications.csv` and `credit_bureau.csv` in the folder `final_exam`. The dataset `credit_bureau.csv` contains the following variables.

`ssn`	=	the consumers unique social security number
`zip_code`	=	the zip code in which the consumer resides
`fico`	=	the consumer's credit score
`num_late`	=	the number of number of times a consumer has made a payment after the due date
`past_def`	=	the number of number of times a consumer has defaulted on a line of credit
`num_bankruptcy`	=	the number of number of times a consumer has filed for bankruptcy

Use the variables from both datasets to estimate a better regression model to predict the prices of airplanes.

a) Perform any pre-processing that needs to be done to the application data in `applications.csv` and the conumer data in `credit_bureau.csv` before joining them: clean them, `sort` them or `read` them, according to your strategy of choice.

b) Form a dataset `purch_app_bureau.csv` by `paste`ing, `join`ing, or `merge`ing the datasets, as needed.

c) If not already done in the above, `read` the new dataset and store it in a data frame called `purch_app_bureau` in your workspace.

d) Calculate and store the printed output from either a `summary` of the data or `describe` the data, according to your choice of software. Use this to get familiar with the contents of the dataset.

e) Estimate a regression model to predict `purchases` as a function of the other variables in the dataset. Ignore the variables `app_id`, `ssn` and `zip_code`, which are keys for databases. Store the printed estimation output with the `print` and/or `summary` command, as appropriate.

Question 3:

Now use all three files `applications.csv`, `credit_bureau.csv`, and `demographic.csv` in the folder `final_exam`. The dataset `demographic.csv` contains the following variables.

`zip_code`	=	the zip code to indicate each geographic region
`avg_income`	=	the average income in each zip code
`density`	=	the population density in each zip code

Use the variables from these datasets to estimate an even better regression model to predict the prices of airplanes.

a) Perform any pre-processing that needs to be done to the file `demographic.csv` before joining it to the others: clean, `sort` or `read`, according to your strategy of choice.

b) Form a dataset `purchase_full.csv` by `paste`ing, `join`ing, or `merge`ing the datasets, as needed.

c) If not already done in the above, `read` the new dataset and store it in a data frame called `purchase_full` in your workspace.

d) Calculate and store the printed output from either a `summary` of the new variables or `describe` the new variables, according to your choice of software. Use this to get familiar with the contents of the dataset.

e) Estimate a regression model to predict `purchases` as a function of the other variables in the dataset. Ignore the variables `app_id`, `ssn` and `zip_code`, which are keys for databases. Store the printed estimation output with the `print` and/or `summary` command, as appropriate.

Question 4:

Now calculate new variables to estimate a model for monthly purchase volume using some different functional forms. Use the variables from your best model from Questions 1 to 3.

a) Create a new variable `utilization`, which is defined as the ratio of `purchases` to the consumer's `credit_limit`.

b) Calculate and store the printed output from either a `summary` of the new variable or `describe` the new variable, according to your choice of software. Use this to get familiar with the nature of this variable and check that it is well-defined.

c) Estimate a regression model to predict `utilization` as a function of the other relevant variables in the dataset. Store the printed estimation output with the `print` and/or `summary` command, as appropriate.

d) Create a new variable `log_odds_util`, the log-odds ratio, which is defined as the logarithm of the ratio of `utilization` over one minus `utilization`. Use the logarithm function `log()` in `R` or `math.log()` in `Python`.

e) Inspect the new variable and estimate a regression model to predict `utilization` as a function of the other relevant variables in the dataset. Store the printed estimation output with the `print` and/or `summary` command, as appropriate.

Part B: Function Design and Optimization

Question 5:

Estimate $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_k)'$ by minimizing the sum of squared residuals, defined as

$$SSR(\beta; y, x_1, \ldots, x_k) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{1i} - \ldots - \beta_k x_{ki})^2$$

a) Define a function `SSR(beta; ...)` that calculates the sum of squared residuals. Your function should be compatible with the best model from Part A. In particular, it should allow for all k explanatory variables that are used in your model.

b) Test your function by comparing the value to the `SSR` obtained from your best model from Part A. Take the value of `beta` from the estimated coefficients to calculate `SSR(beta; ...)`. Compare this value with `sum(my_lm_model$residuals^2)` in `R` or `sum(reg_model_sm.resid**2)` using the `stats.models` module in `Python`, for example.

c) Use a numerical optimization function to minimize your `SSR(beta; ...)` function.

d) Verify the accuracy of your calculation by printing your optimal parameter values and comparing them with the values in your estimated model from Part A. Validate the optimized value of the `SSR(beta; ...)` function against the values from part (b).

e) Now consider an estimation method called *regularization*. Using this method, a penalty term is applied to the parameter β to avoid highly variable estimates far from zero. Write a new objective function `SSR_reg` for this estimation method, which is defined as

$$SSR\_reg(\beta, \lambda; y, x_1, \ldots, x_k) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{1i} - \ldots - \beta_k x_{ki})^2 + \lambda \left(\sum_{j=1}^{k} \beta_j^2 \right)$$

f) Test your function by comparing the value to the `SSR` obtained from your best model from Part A. For this function, you would expect a higher value of `SSR_reg` compared to `SSR` from the penalty term. A higher value of `lambda`, i.e. λ, will apply a larger penalty.

g) Use a numerical optimization function to minimize your `SSR_reg(beta, lambda; ...)` function.

h) Verify the accuracy of your calculation by printing your optimal parameter values and comparing them with the values in your estimated model from Part A. With the penalty for regularization, you would expect similar values of the coefficients in β, except that the absolute values of the larger coefficients will be smaller, i.e. closer to zero.

Part C: Software Management and Version Control

Question 6:

Create a UNIX shell script called `final_exam.sh` that runs all the software to answer Questions 1 to 5 in Parts A and B.

a) Use commands such as `Rscript`, `python3`, or `sqlite3` to run your software.

b) Redirect the output of each script to appropriately-named `.txt` or `.out` files, using the "`>`" operator, to save your output.

c) You can test your script by running `./final_exam.sh`.

Question 7:

Push your completed files to your GitHub repository following these steps. See the `README.md` and the `GitHub_Quick_Reference.md` in the folder `demo_03_version_control` in the QMB6358F21 course repository for more instructions.

1. Open GitBash 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.

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_filename.ext`. in this example.

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.