DAEN 500-1 – Data Analytics Fundamentals

Spring 2021 Final Examination Exercise
3/5 – 3/6/2021

Final Submission Deadline: NLT 11:59PM (EST). Saturday, March 6, 2010

Failure to submit ON TIME will result in DAEN COURSE FAILURE

Name: Julie Ritenour **GMU G#: G01315806**

Student Signature (Honor Certification): Julie D Ritenour

This exam is **OPEN BOOK/OPEN NOTES**. You may consult any of the course texts, and the various reference materials recommended in the syllabus. ***The exam of course IS NOT "Open Web",*** especially in that you may NOT utilize expert "help" sites such as Stack Overflow, or other programming help or collaboration sites.

Additionally, you are restricted from discussing the substance of the questions on this exam with any other individual, until after you have submitted your final response for grading. The completed exam -- with your answers embedded in this docx document (add extra pages as necessary) should be submitted following instructions contained in the Final Exam Instructions BB site. If you have any trouble submitting and have extra parts of the answers you have trouble appending to this document, you may simply submit additional pages separately (the exam submission site is set for multiple submissions, just in case). Make certain all are submitted PRIOR TO THE DEADLINE!

HONOR CODE CERTIFICATION

Your signature above declares that you have followed the conditions of this exam, and that the work is yours alone. **Specifically:**

This must be your own work, authored and completed by you. As stated earlier, this is an "open source exam" – allowing books, notes or courseware, as well as *general* expert advice gained PRIOR to exam. YOU MAY NOT, HOWEVER, SEED OR USE ANY ADVICE ON HOW TO SOLVE THE QUESTION OR ANY CODE WRITTEN BY ANY OTHER INDIVIDUAL. *Any violation will result in an immediate failure in the exam and for the course, as well as referral to the GMU Honor Committee for determination of any other appropriate disciplinary consequences.*

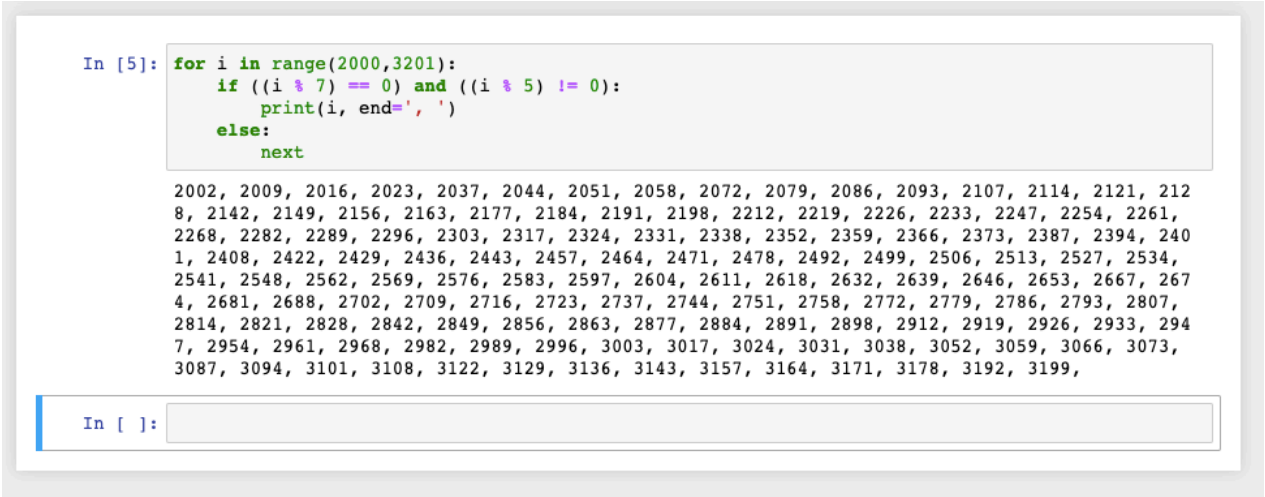
*NOTE: Your **submission** of any responses, files, programs, etc. in response to the DAEN500 final exam instructions, will also be your personal certification of your full compliance with the spirit and letter of the **GMU Honor Code** standards for take home and/or in-class exams.*

Problem 1: Python Programming Problem (15 Points Total)

- **Design and implement a Python program that is based on the following requirements: a) program will find all numbers which are divisible by 7 but are not a multiple of 5; and b) numbers between 2000 and 3200.**

- **INSERT (cut&paste) your Python code in space below and *then insert a screen shot in space below, showing code, your successful run, input and output.***

```
for i in range(2000,3201):
  if ((i % 7) == 0) and ((i % 5) != 0):
    print(i, end=', ')
  else:
    next
```



NOTE of alternative for help: To help test your code, you also may use a Python "programming window" found in the **Zybooks Section 35 Additional Material** OR any relevant IDE.

Problem 2: Python Programming Problem (15 Points Total)

- **Design and implement a Python program that is based on thefollowing requirements:**

  a) define a class which has *at least two* methods

  - **Method 1 – getString: to get a string from console input; and,**
  - **Method 2 - printString: to print the string in upper case.**

  b) demonstrate code works using three different test input strings

- ***INSERT** code below* **and** *INSERT* **a screen shot of the program and successfully run output that** *includes test input for input strings (test strings must include (a) all upper case, (b) all lower case, **and** (c) mix of upper and lower case).*

```
class reviseString:
   def __init__(self):
      self.user_input = ''

   def getString(self):
      self.user_input = input('Insert String: ')

   def printString(self):
      print(self.user_input.upper())

user_string = reviseString()
user_string.getString()
user_string.printString()
```

```
In [3]: class reviseString:
            def __init__(self):
                self.user_input = ''

            def getString(self):
                self.user_input = input('Insert String: ')

            def printString(self):
                print(self.user_input.upper())

        user_string = reviseString()
        user_string.getString()
        user_string.printString()

        Insert String: this is all lower case
        THIS IS ALL LOWER CASE
```

In [4]:
```
class reviseString:
    def __init__(self):
        self.user_input = ''

    def getString(self):
        self.user_input = input('Insert String: ')

    def printString(self):
        print(self.user_input.upper())

user_string = reviseString()
user_string.getString()
user_string.printString()
```

```
Insert String: THIS IS ALL UPPER CASE
THIS IS ALL UPPER CASE
```

In [ ]:

In [5]:
```
class reviseString:
    def __init__(self):
        self.user_input = ''

    def getString(self):
        self.user_input = input('Insert String: ')

    def printString(self):
        print(self.user_input.upper())

user_string = reviseString()
user_string.getString()
user_string.printString()
```

```
Insert String: This is a Mix of Upper and Lower Case
THIS IS A MIX OF UPPER AND LOWER CASE
```

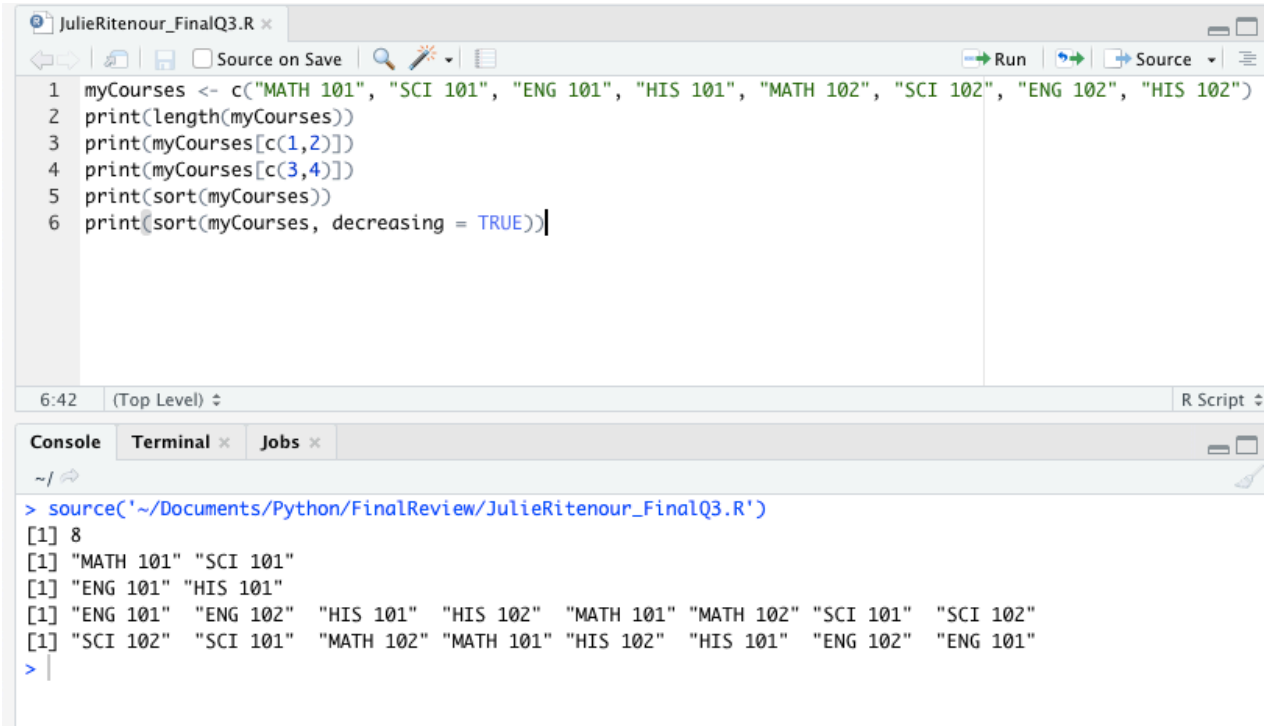
In [ ]:

Problem 3: R Programming Problem (20 Points Total)

- **Perform the following problems using R:**
  - Create a vector of courses (e.g., MATH 101) you have taken previously. Make sure you have at least 8 courses. Name the vector myCourses
  - Get the length of the vector myCourses
  - Get the first two courses from myCourses
  - Get the 3rd and 4th courses from myCourses
  - Sort myCourses using a method
  - Sort myCourse in the reverse direction

- *INSERT code below* and *INSERT* a screen shot of the program and successfully run output.

```
myCourses <- c("MATH 101", "SCI 101", "ENG 101", "HIS 101", "MATH 102", "SCI 102", "ENG 102", "HIS 102")
print(length(myCourses))
print(myCourses[c(1,2)])
print(myCourses[c(3,4)])
print(sort(myCourses))
print(sort(myCourses, decreasing = TRUE))
```



Problem 4: Principal Component Analysis (25 points)

Provide a description of the following:

1) What is a component – Provide a description (5 points)
    - A component is a new variable that is comprised of a weighted combination of correlated predictor variables. The correlation value between predictor variables is used to determine the weighted value of each predictor variable in the new component.
2) Principal Component Analysis – Provide a description (5 points)
    - A Principle Component Analysis is the process of creating new variables, called components, to capture as much variability in the data as possible while reducing the number of total parameters thereby simplifying the analysis.
3) **Provide an specific example of Principal Component Analysis (15 points)**
    - One situation where a Principal Component Analysis could be applied is in a university's analysis of the academic performance of an incoming freshman class based on a number of predictor variables. The analysis utilizes the following predictor variables: grade point average, scores on standardized tests, volunteer hours, and participation in extracurricular activities. There is a desire to simplify the analysis by reducing the number of attributes used in the analysis. A correlation analysis is performed, and it determines there is a high correlation between a student's grade point average and scores on standardized tests, as well as between volunteer hours and participation in extracurricular activities. The high correlation between the two sets predictor variables indicates that the variables in each set act similarly. Therefore, a principle component analysis could be performed to combine the variables in each set to create two new components thereby simplifying the analysis without reducing the variability of the data and negatively impacting the results.

Problem 5: Multiple vs. Logistic (30 points)

(a) Describe: What is difference between Multiple Regression and Logistic Regression? What circumstances might determine which to use? (10 points)

A Multiple Regression Model is used to determine how changes in any number of independent variables impact one dependent variable, whereas a Logistic Regression Model is preformed to predict a categorical dependent variable (also known as a binary response variable) based on a number of independent variables. Determining which model to use is based on the dependent variable. If the dependent variable is a numerical continuous value, then a Multiple Regression Model should be used. If the dependent variable is categorical, and comprised of only two options, then a Logistic Regression Model should be used.

(b) Demonstrate: Using any data, and any tool set you've learned about, show differences (20 points)

SUGGESTION: may be solved using RapidMiner, or other toolsets, BOTH TO ANALYZE AND TO VISUALIZE REGRESSION DIFFERENCES.

Step 1: Perform a quick search of the UCIS public data archive, a well-curated site which you already have seen as part of your introductory RapidMiner training.

Step 2: Pick a dataset you find interesting, input dataset into regression tools you've chosen.

Step 3: Run regression, .and use visualizations to demonstrate the conceptual answers you provided for 5.(a).

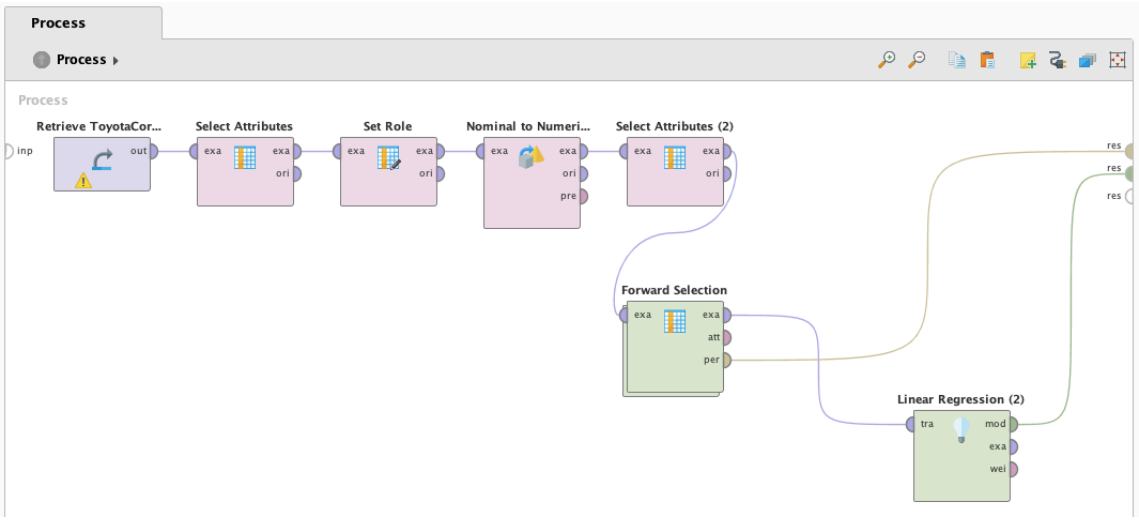
Analysis:

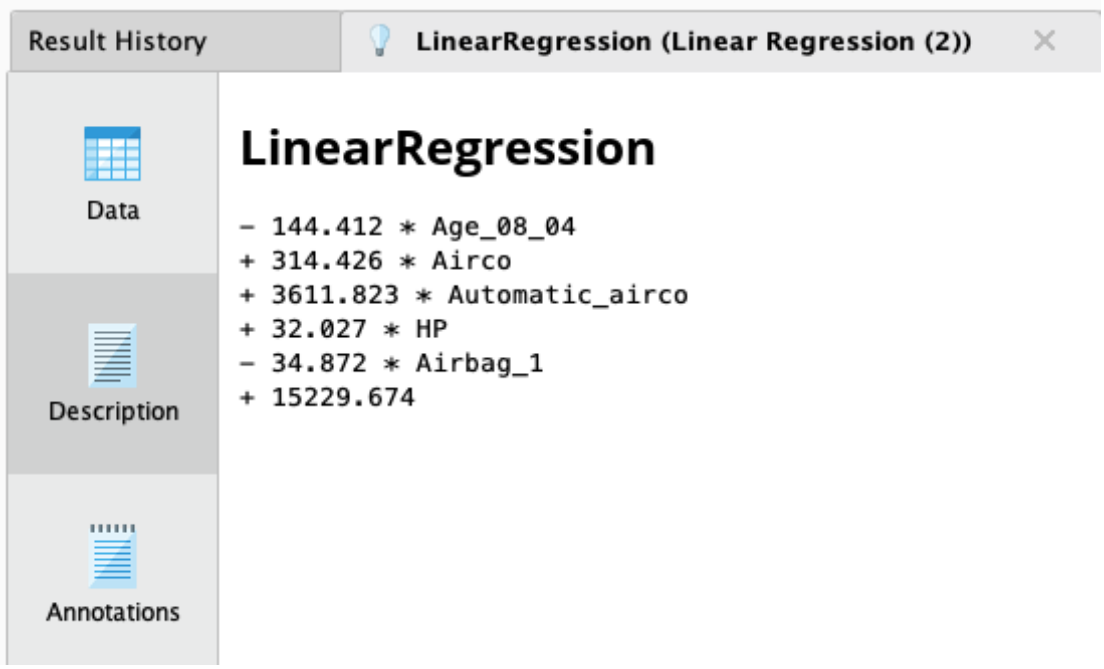
To show the difference between the two models I will use two different datasets. For the multiple regression model, the dataset ToyotaCorolla.csv contains over 1400 observations of characteristics and price of Toyota Corolla's. I used RapidMiner to build a multiple regression model that would use the car's characteristics to predict the price. Therefore, the car's characteristics would be the independent variables and the car's price would be the dependent variable. Since the dependent variable, price, is continuous, a Multiple Regression Model is appropriate. The Multiple Regression Model created a formula that would predict the car's price if given the car's characteristics (see

screen shots below).

For the Logistic Regression Model, I used a dataset that contains almost 200 observations of characteristics of pregnant women, as well as a categorical attribute specifying if the weight of each woman's baby was high or low at birth. I used RapidMiner to build a model that would predict if a baby's birth weight would be classified as high or low at birth. A Logistic Regression mMdel is appropriate in this case because the model is using the independent variables to choose between the dependent variable's two options, high (HI) or low (LO). In the case of birth weights, the logistic regression model was able to predict a baby's birth weight category with an accuracy of almost 72% (see screen shots below).

Multiple Regression Screen Shots:





Logistic Regression Screen Shots: