DAEN 500- DL1 – Data Analytics Fundamentals

Fall 2020 Final Examination Exercise
11/24 – 12/05/2020

Final Submission Deadline: NLT 11:59PM (EST). Saturday, Dec 5, 2020
Failure to submit ON TIME will result in DAEN COURSE FAILURE

Name: Scott Mayer GMU G#: G00497822

Student Signature (Honor Certification): *Scott Mayer*

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.*

FINAL EXAM PROBLEMS
COMPLETE ALL & INSERT ANSWERS BELOW QUESTIONS

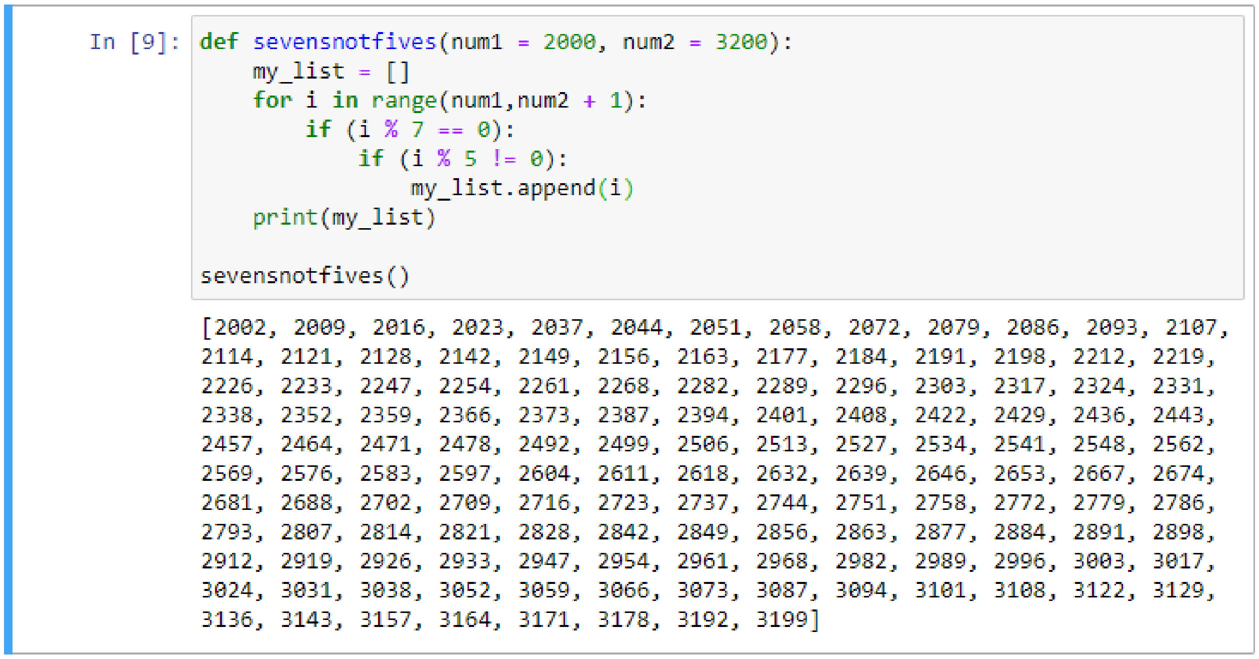
Problem 1: Python Programming Problem (15 Points Total)

- **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.***

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**.

```
def sevensnotfives(num1 = 2000, num2 = 3200):
    my_list = []
    for i in range(num1,num2 + 1):
        if (i % 7 == 0):
            if (i % 5 != 0):
                my_list.append(i)
    print(my_list)

sevensnotfives()
```



```
In [9]: def sevensnotfives(num1 = 2000, num2 = 3200):
            my_list = []
            for i in range(num1,num2 + 1):
                if (i % 7 == 0):
                    if (i % 5 != 0):
                        my_list.append(i)
            print(my_list)

        sevensnotfives()

        [2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107,
        2114, 2121, 2128, 2142, 2149, 2156, 2163, 2177, 2184, 2191, 2198, 2212, 2219,
        2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324, 2331,
        2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436, 2443,
        2457, 2464, 2471, 2478, 2492, 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562,
        2569, 2576, 2583, 2597, 2604, 2611, 2618, 2632, 2639, 2646, 2653, 2667, 2674,
        2681, 2688, 2702, 2709, 2716, 2723, 2737, 2744, 2751, 2758, 2772, 2779, 2786,
        2793, 2807, 2814, 2821, 2828, 2842, 2849, 2856, 2863, 2877, 2884, 2891, 2898,
        2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 2989, 2996, 3003, 3017,
        3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094, 3101, 3108, 3122, 3129,
        3136, 3143, 3157, 3164, 3171, 3178, 3192, 3199]
```

Problem 2: Python Programming Problem (15 Points Total)

- **Design and implement a Python program that is based on the following 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.**

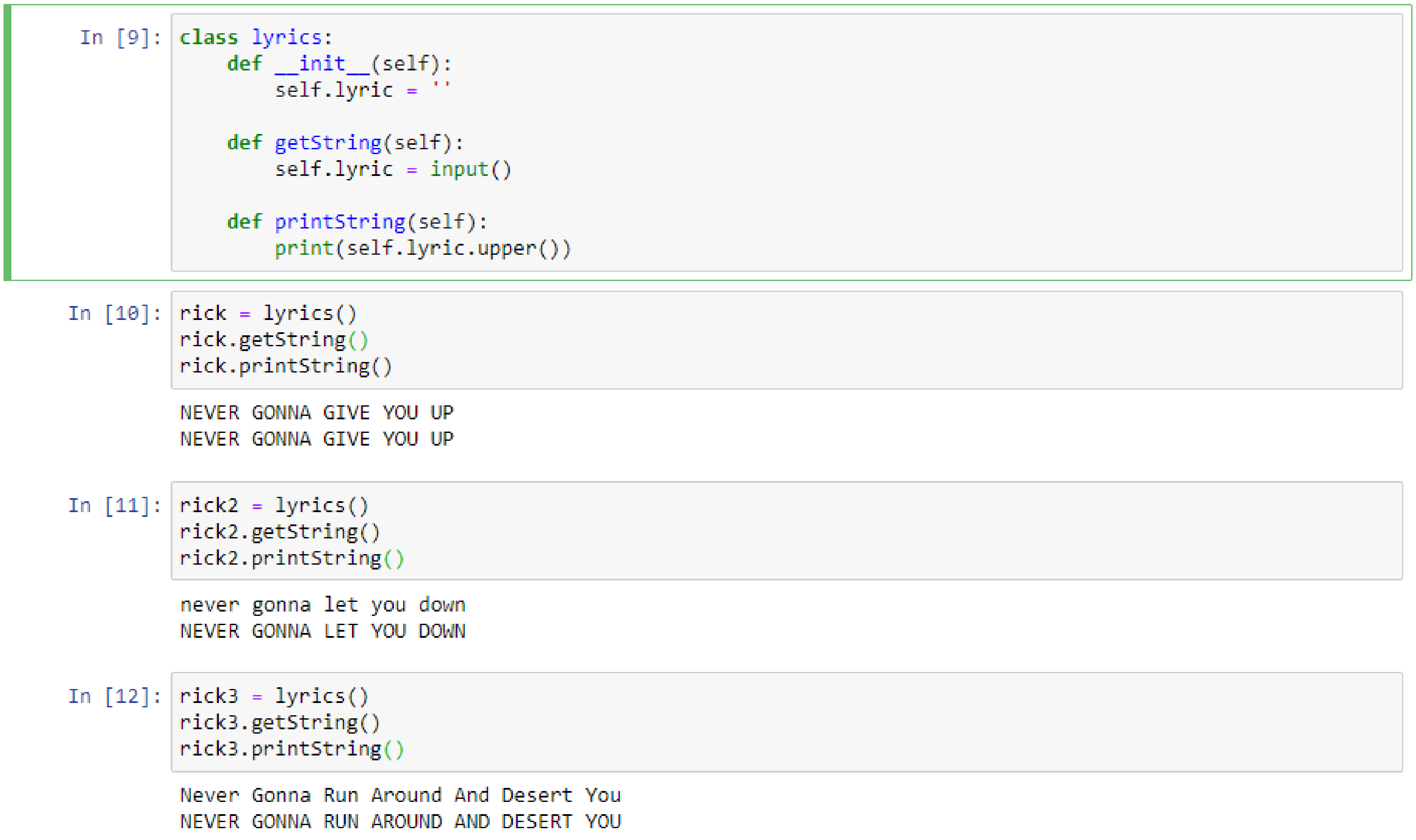
  b) demonstrate code works using three different test input strings

- ***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 lyrics:
    def __init__(self):
        self.lyric = ''

    def getString(self):
        self.lyric = input()

    def printString(self):
        print(self.lyric.upper())
```



```
In [9]: class lyrics:
            def __init__(self):
                self.lyric = ''

            def getString(self):
                self.lyric = input()

            def printString(self):
                print(self.lyric.upper())
```

```
In [10]: rick = lyrics()
         rick.getString()
         rick.printString()
```

```
NEVER GONNA GIVE YOU UP
NEVER GONNA GIVE YOU UP
```

```
In [11]: rick2 = lyrics()
         rick2.getString()
         rick2.printString()
```

```
never gonna let you down
NEVER GONNA LET YOU DOWN
```

```
In [12]: rick3 = lyrics()
         rick3.getString()
         rick3.printString()
```

```
Never Gonna Run Around And Desert You
NEVER GONNA RUN AROUND AND DESERT YOU
```

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("DAEN 500", "CS 504", "OR 531", "AIT 580", "STAT 515", "BUS 300", "ECON 301", "MATH 203")

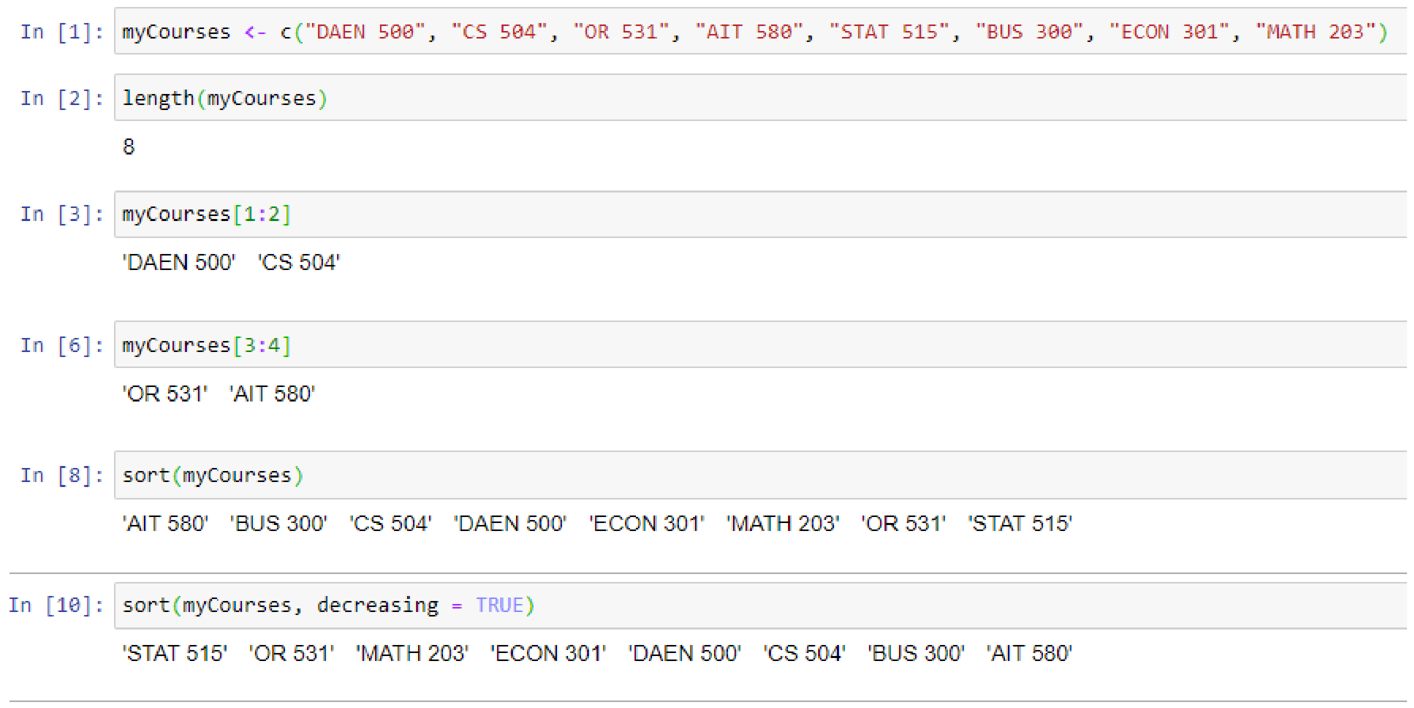
length(myCourses)

myCourses[1:2]

myCourses[3:4]

sort(myCourses)

sort(myCourses, decreasing = TRUE)



```
In [1]: myCourses <- c("DAEN 500", "CS 504", "OR 531", "AIT 580", "STAT 515", "BUS 300", "ECON 301", "MATH 203")

In [2]: length(myCourses)

        8

In [3]: myCourses[1:2]

        'DAEN 500'   'CS 504'

In [6]: myCourses[3:4]

        'OR 531'   'AIT 580'

In [8]: sort(myCourses)

        'AIT 580'   'BUS 300'   'CS 504'   'DAEN 500'   'ECON 301'   'MATH 203'   'OR 531'   'STAT 515'

In [10]: sort(myCourses, decreasing = TRUE)

        'STAT 515'   'OR 531'   'MATH 203'   'ECON 301'   'DAEN 500'   'CS 504'   'BUS 300'   'AIT 580'
```

Problem 4: Principal Component Analysis (25 points)

Provide a description of the following:

1) What is a component – Provide a description (5 points)
2) Principal Component Analysis – Provide a description.(5 points)
3) **Provide an specific example of Principal Component Analysis(15 points)**

1) What is a component – Provide a description (5 points)

A component, by definition, is a part of a whole. Within the context of data, a component could be a variable within a model, or an attribute of a record. A principal component is a new component that is derived from correlated variables in the process of principal component analysis (PCA).

2) Principal Component Analysis – Provide a description (5 points)

Principal component analysis (PCA) is a method of dimensionality reduction. Dimensionality refers to the number of attributes, a.k.a. features (or dimensions), that describe a record, or instance. Often, datasets will have many dimensions, and therefore a high dimensionality.

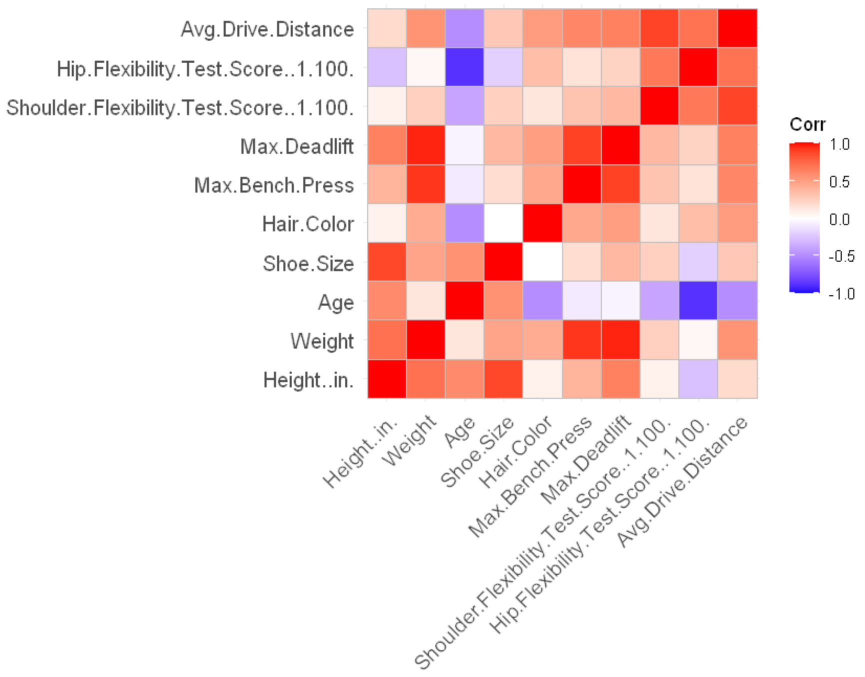
While it's nice to have thorough datasets, there are some complications that come with high dimensionality. It can be costly, computationally, to store and process. It can be difficult to interpret the many relationships between large numbers of features. Often, there are correlated features that create redundancy. Accuracy of predictive models can suffer, as well.

The goal of PCA is simplification by reducing the number of dimensions. PCA removes the redundancy of correlated variables and creates new, uncorrelated variables called principal components. These principal components are better (more efficient) at explaining variability in the dataset in fewer dimensions.

3) Provide a specific example of Principal Component Analysis (15 points)

Let's say we have the following dataset about PGA Tour golfers. Note that the values for each attribute are entirely made-up for example's sake.

Name	Height (in)	Weight	Age	Shoe Size	Hair Color	Max Bench Press	Max Deadlift	Shoulder Flexibility Test Score (1-100)	Hip Flexibility Test Score (1-100)	Avg Drive Distance
Tiger Woods	73	185	44	10	0	275	390	85	80	300
Phil Mickelson	75	200	50	12	1	285	375	75	70	295
Bryson Dechambeau	73	240	27	10	2	350	500	90	95	340
Justin Thomas	70	160	27	9	2	250	350	80	90	305
Dustin Johnson	76	200	36	13	1	275	420	95	95	335
Rickie Fowler	69	150	31	9	0	250	300	80	90	295
Rory McIlroy	69	165	31	10	1	275	335	95	90	330



We want to explore how the various attributes affect the target variable - average drive distance. PCA evaluates the different attributes to explore correlations. It takes correlated variables and combines them into new variables, called principal components, according to how big of an effect they have on the target variable. So in this case, PCA might identify a correlation between max bench press, max deadlift, height, weight, and perhaps even shoe size (the correlation matrix above helps illustrate this). It would then combine them into a principal component that we can think of as “size and strength factors”. Similarly, PCA might identify correlation between shoulder flexibility and hip flexibility test scores, then create a new principal component we can think of as “flexibility factors”. PCA would report on how much each principal component explains variance in the target variable. Let’s say, in this case, that these two principal components contribute to 90% of the variance in average drive distance.

PCA might find that age was uncorrelated with the other features, yet it still explained 5% of the variance of average drive distances. So age could be used to create a third principal component that we are interested in. PCA might also identify hair color as being uncorrelated to the other variables. It would probably also tell us that hair color does not explain a significant variance between average drive distances, so we can choose to ignore it.

In this example, PCA took the dataset from 9 potentially explanatory variables to 3 principal components that explain 95% of the variance in average drive distance. Using these principal components, we can build a model that accurately predicts the average driver distance of new PGA Tour golfers - without being overwhelmed with tons of variables.

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)

(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).

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

Multiple regression is an extension of simple linear regression. Linear regression models the relationship between two variables - response variable Y as a function of an explanatory variable X. Multiple regression adds in one or more explanatory variables to that model, which results in response variable Y as a function of X1 and X2 (optionally, X3 … Xn). Each additional explanatory variable adds a new dimension to the model. Multiple regression models, given values for each explanatory variable, can predict the value of the response variable for a new datapoint.

For the most part, it only makes sense to use linear/multiple regression when the response variable is continuous.

Logistic regression also models the relationship between a response variable and one or more explanatory variables, but this time the response variable Y is not continuous, but binary. The condition Y is either present (1) or absent (0). Logistic regression models are used to predict, based on the input values of the explanatory variables, the likelihood that Y will be present.

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

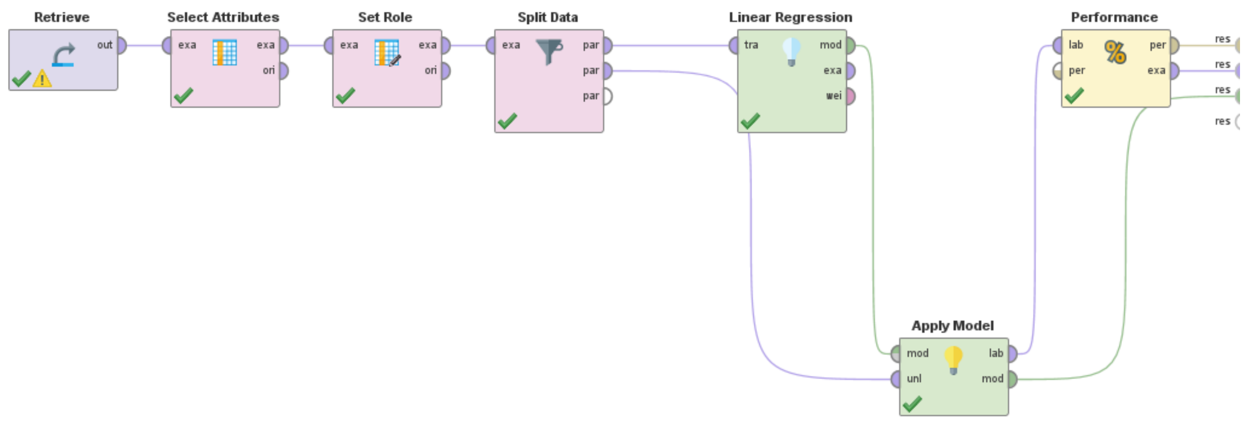
For this section, I used a dataset that contains data on PGA Tour player performance in the 2017 FedEx Cup season. The dataset was obtained from Kaggle.com (https://www.kaggle.com/grantruedy/pga-tour-golf-data-2017-season) and the data was scraped from the official PGA Tour statistics website. It contains 195 player records and 69 attributes.

To show an example of a multiple regression model, I will use the "POINTS" attribute as the target, or dependent variable. This refers to the total number of FedEx Cup points that a player earned in the 2017 season. There was one outlier that I corrected after researching the correct number for the corresponding player, Bryson Dechambeau. POINTS values in the dataset ranged from 10 to 3289. My goal is to build a model that can accurately predict the number of points a player will earn based off performance variables.

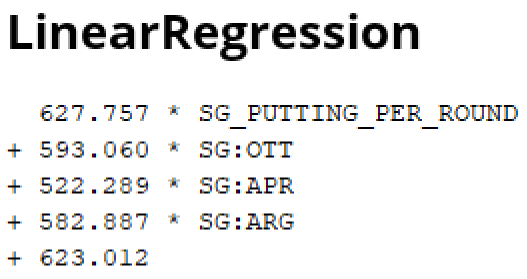
Using RapidMiner Studio, I designed a model that selected the target variable POINTS and 4 explanatory variables: SG:OTT (shots gained off the tee), SG:APR (shots gained on approach shots), SG:ARG (shots gained around the green), and SG:PUTTING_PER_ROUND. These "shots gained" metrics measure a player's performance, relative to his competitors, in the specified aspects of golf.

I then split the data into a training set (70%) and test set (30%). The training set was used to build a multiple linear regression model. Then, the model was applied to the test set. Performance was measured by how accurately the model was able to predict the true POINTS values in the test set.

Here is a visual of the process in RapidMiner Studio:

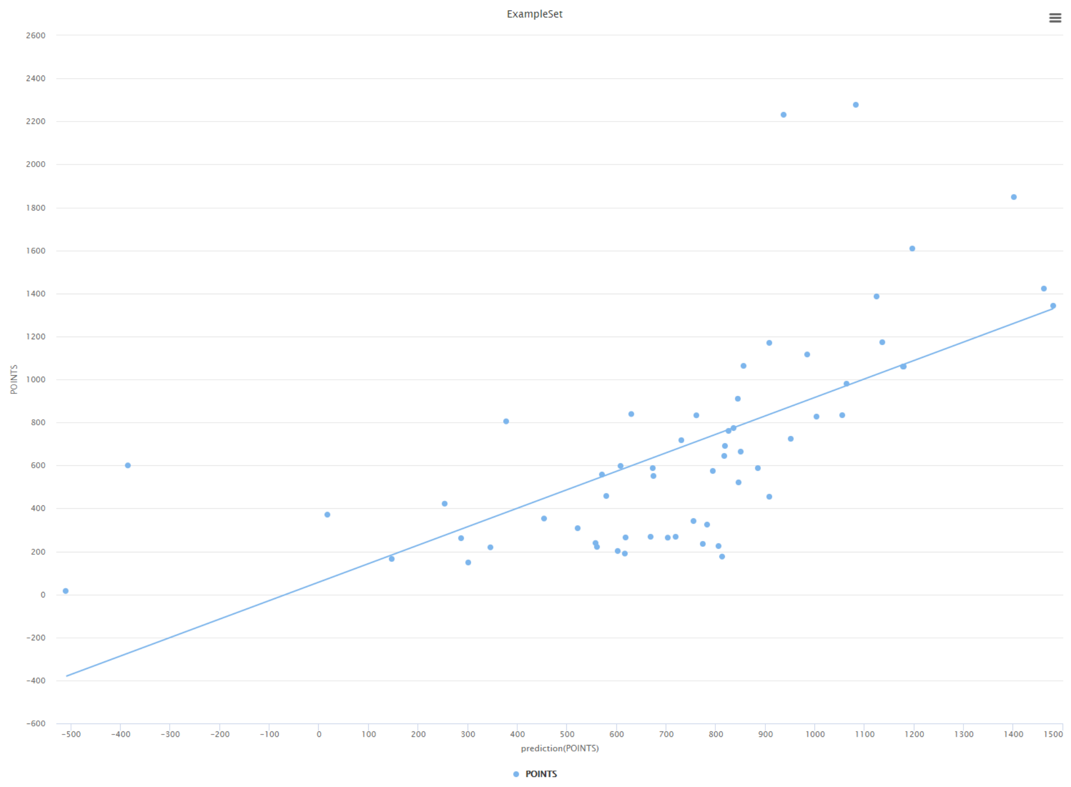


The resulting model is shown below:

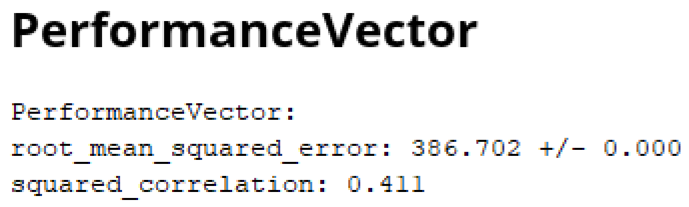


LinearRegression

```
  627.757 * SG_PUTTING_PER_ROUND
+ 593.060 * SG:OTT
+ 522.289 * SG:APR
+ 582.887 * SG:ARG
+ 623.012
```



When applied to the test data, the model performance was:



PerformanceVector

```
PerformanceVector:
root_mean_squared_error: 386.702 +/- 0.000
squared_correlation: 0.411
```

Squared_correlation, or R^2, is measured on a scale of 0-1 and measures how much of the variation in the dependent variable is explained by the selected independent variables. In this case, the four "shots gained" metrics accounted for 41.1% of the variance in FedEx Cup points. Considering how many variables go into a golfer's performance across an entire year of tournaments, this model's performance is not bad at all. The RMSE refers to how far off, on average, the linear regression model was from the test points. Considering the range of POINTS values in the original dataset, this is not bad either. All in all, though, this model does an OK job of predicting a dependent variable that is quite hard to predict due to so much uncertainty.

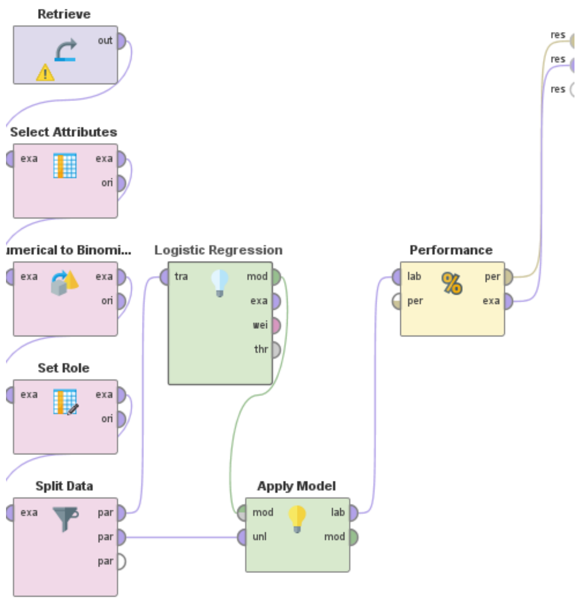
Next, I built a logistic regression model using the same dataset. POINTS was once again used as the target variable, but because we are using logistic regression, I converted POINTS into a binary variable.

A quick side note about the rules of the PGA Tour – a player can only compete on the Tour once he has earned his Tour card. One of the ways that he can retain his tour card for the following year is by placing in the top 125 of the final FedEx Cup standings. In this dataset, the 125th ranked player had a POINTS values of 405. Accordingly, I created a binary variable that describes whether a PGA Tour player earned a Tour card for the next season (POINTS >= 405) or not (POINTS < 405).

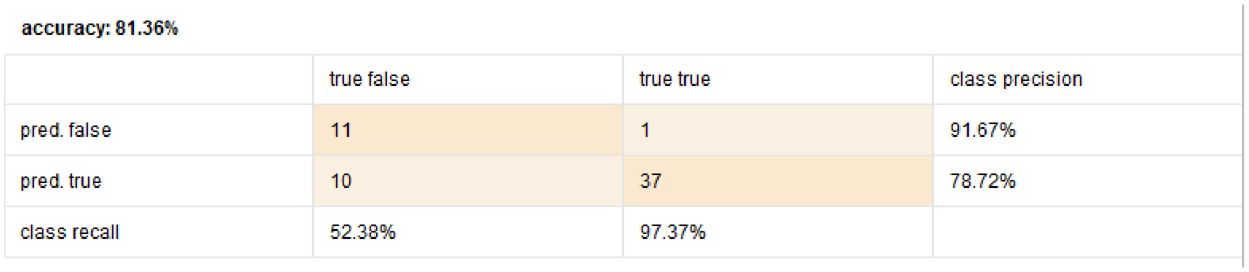
This time, I selected new explanatory variables AVG_Driving_DISTANCE, EVENTS_PLAYED, and FAIRWAY_HIT_%. Note that FAIRWAY_HIT_% is a measure of driving accuracy.

Basically, the question I'm asking with this model is – based on a player's performance off the tee and number of events played, how accurately can we predict whether or not that player earns enough points to retain their PGA Tour card?

Here's what the process looked like in RapidMiner:



Once again, the test data was split into a training set (70%) and a test set (30%). The training set was used to build the logistic regression model, which was then applied to the test set. The resulting performance is shown below:



accuracy: 81.36%

	true false	true true	class precision
pred. false	11	1	91.67%
pred. true	10	37	78.72%
class recall	52.38%	97.37%	

We see that the model correctly predicted 37 out of the 38 players in the test set who earned enough points to retain their Tour cards. The model also predicted that 11 players would not retain their tour cards, while the true number was 21. Overall, 48 correct predictions were made out of 59 total instances, resulting in a model accuracy of 81.36%. This model could easily be improved by adding more dependent variables that cover a larger scope of golf skills. But the performance is quite good when you consider that the explanatory variables only speak to performance off the tee and events played.

As you can see, the key difference between the multiple regression and logistic regression model is the type of response variable. Multiple regression models work well with a numeric (often continuous) response variable. In the demonstration above, we built a model that could predict the value of Y, in this case the number of FedEx Cup points a player earned. Logistic regression models work well with binomial response variables. In the demonstration above, we used the model to predict the presence/absence of Y, in this case whether or not a player earned enough FedEx Cup points to retain his tour card in 2017.