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:___Dawa O'Sullivan_____ GMU G# _G00828230_____
Student Signature (Honor Certification): _____

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:**

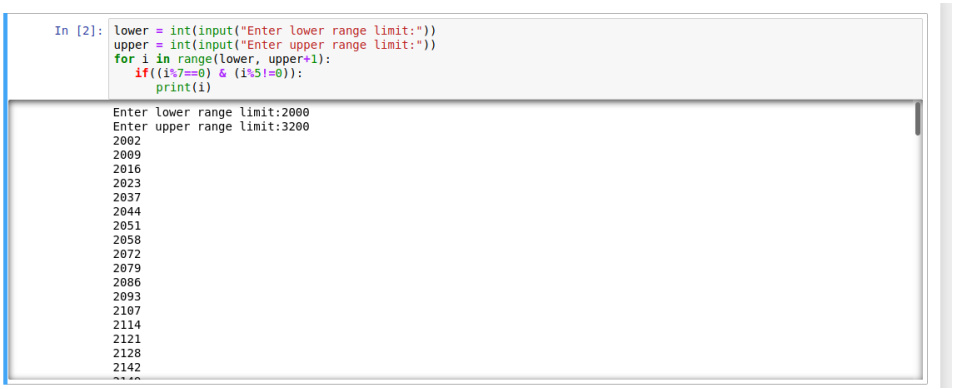
This must be your own work, authored and completed by you. As stated earlier, this is an "open source exam" – allowing books, notes or courseware, as well as *general* expert advice gained PRIOR to exam. YOU MAY NOT, HOWEVER, SEED OR USE ANY ADVICE ON HOW TO SOLVE THE QUESTION OR ANY CODE WRITTEN BY 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.***

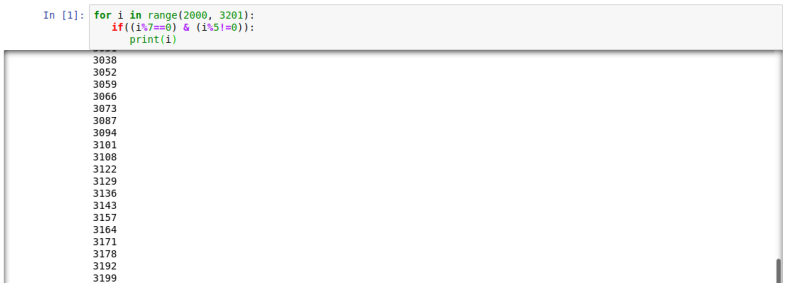
```
lower = int(input("Enter lower range limit:"))
upper = int(input("Enter upper range limit:"))
for i in range(lower, upper+1):
   if((i%7==0) and (i%5!=0)):
      print(i)
```



Screenshot cuts off values

Code not requiring user inputs

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



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 My_String:
    def __init__(self):
        self.words = ""
    def get_string(self):
        self.string = input()
    def print_string(self):
        print(self.string.upper())
string = My_String()
string.get_string()
string.print_string()
```

```
In [1]: class My_String:
            def __init__(self):
                self.words = ""
            def get_string(self):
                self.string = input()
            def print_string(self):
                print(self.string.upper())
        string = My_String()
        string.get_string()
        string.print_string()

        ALL UPPERCASE
        ALL UPPERCASE
```

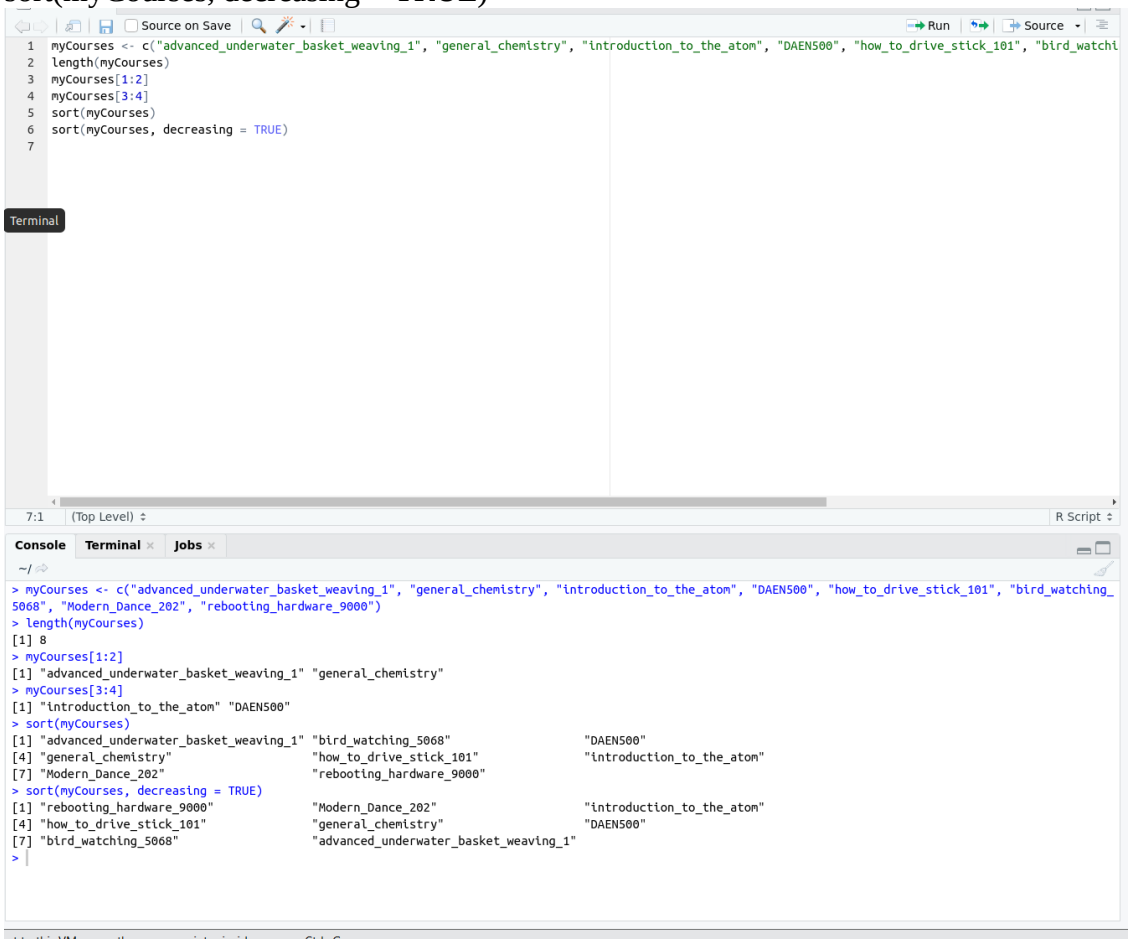
```
In [2]: class My_String:
            def __init__(self):
                self.words = ""
            def get_string(self):
                self.string = input()
            def print_string(self):
                print(self.string.upper())
        string = My_String()
        string.get_string()
        string.print_string()

        all lowercase
        ALL LOWERCASE
```

```
In [3]: class My_String:
            def __init__(self):
                self.words = ""
            def get_string(self):
                self.string = input()
            def print_string(self):
                print(self.string.upper())
        string = My_String()
        string.get_string()
        string.print_string()

        Mix UPPER and lower-CaSe
        MIX UPPER AND LOWER-CASE
```

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("advanced_underwater_basket_weaving_1", "general_chemistry", "introduction_to_the_atom", "DAEN500", "how_to_drive_stick_101", "bird_watching_5068", "Modern_Dance_202", "rebooting_hardware_9000")

length(myCourses)

myCourses[1:2]

myCourses[3:4]

sort(myCourses)

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)
    - When creating a model there are typically multiple predictor variables. A component is a new variable formed by a combination of predictor variables. There will be
2) Principal Component Analysis – Provide a description.(5 points)
    - Principle component analysis is a method of reducing the dimensions of a data set by analyzing components and dropping the components that are not good predictors for our model. If we have 10 input variables, we get 10 components, the variables are now independent of each other and we can select a number of the resulting components
    - Drawbacks include:
        - Loss of accuracy for the sake of reduced dimensionality/simplification
        - Independent variables become less easy to interpret at a glance
3) **Provide a specific example of Principal Component Analysis(15 points)**

- Using the concrete dataset from problem 5 (might want to read through that one first as I put all the explanation there and did this problem last)
- Only selecting for samples that have a 28 day curation time and not including the resulting compressive strength leaves us with 7 variables which map to 7 principal components
    - Don't want to include our dependent variable of compressive strength as the goal of PCA is to find covariance in our input variables
- This is done by the software, in this case rapidminer

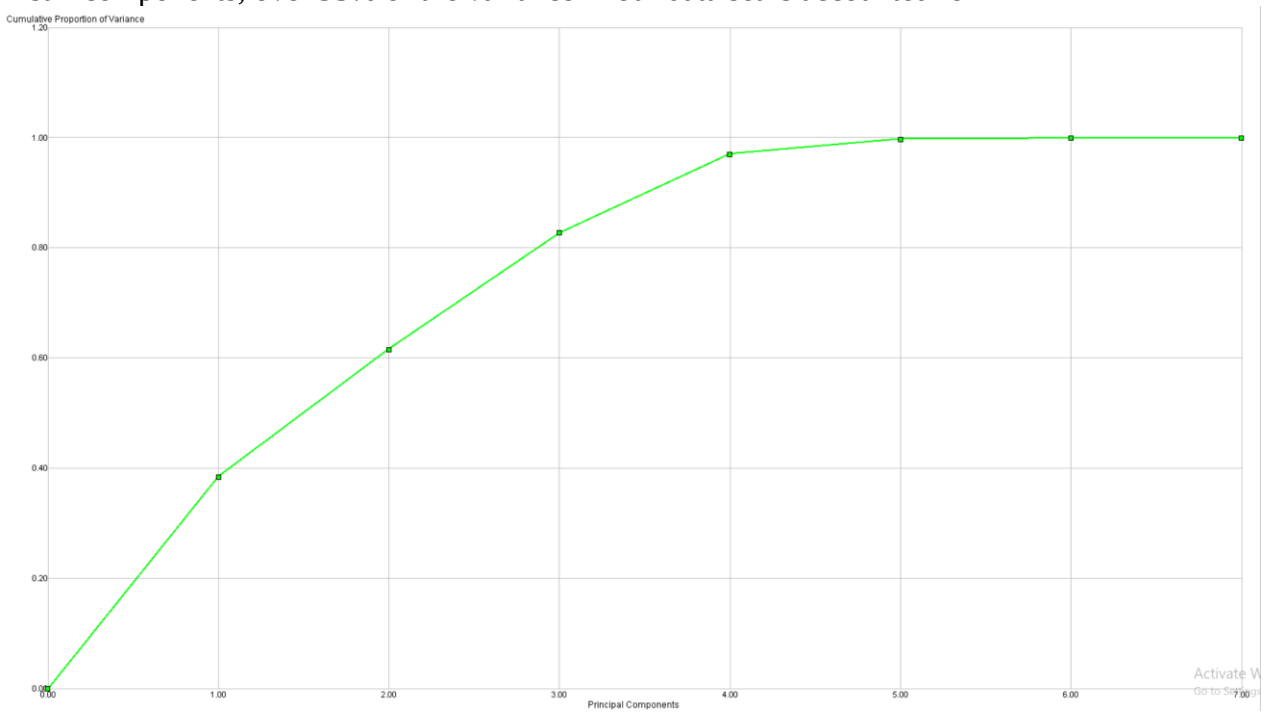
We get the resulting table of Eigenvalues (we mostly care about the proportion of variance as it tells us how much of the variance in our input variables is captured by each component)

	Std. Dev.	Proportion of Variance	Cumulative variance
PC 1	117.452	0.385	0.385
PC 2	90.997	0.231	0.616
PC 3	87.143	0.212	0.828
PC 4	71.534	0.143	0.971
PC 5	30.927	0.027	0.998
PC 6	8.532	0.002	1.000
PC 7	3.397	0.000	1.000

As well as this table of Eigenvectors

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Cement	-0.824	0.339	-0.262	0.070	0.305	0.197	-0.025
Blast Furnace Slag	0.491	0.701	-0.001	-0.166	0.432	0.230	-0.027
Fly Ash	0.196	-0.432	-0.149	0.601	0.549	0.301	-0.010
Water	0.037	0.039	-0.018	0.079	-0.521	0.807	-0.260
Superplasticizer	0.006	-0.009	-0.022	0.017	0.089	-0.254	-0.963
Coarse Aggregate	-0.201	-0.121	0.902	-0.140	0.263	0.202	-0.052
Fine Aggregate	0.013	-0.437	-0.309	-0.762	0.266	0.246	-0.043

From first table of Eigenvalues we can make a cumulative variance chart and see that after the first 4 components, over 95% of the variance in our data set is accounted for



Now we can use components the first 4 components in our analysis instead of all 7 as is the case in problem 5.

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).

Problem 5: Multiple vs. Logistic

Describe the difference, what circumstances might determine which to use:

A

Both logistic and multiple regression models are used to predict outcomes of a single dependent variable based on models created from input data. Additionally, the data should be normally distributed.

Logistic regression predicts binary outcomes (dependent variable is either pass/fail, yes/no, etc.) and assigns a probability to the outcome. This can be done with respect to one input variable or with multiple. Use cases are when testing for a binary outcome, did the student pass or fail course, based on one or more predictor variables (study time, lecture attendance, optional assignment completion).

Multiple regression predicts a continuous outcome (dependent variable is a range of results) based on multiple input variables. If there was only one input variable then we have a linear regression. Use cases are when predicting outcomes that are continuous in nature, what % score did the student get in the class, based on several predictor variables (study time, lecture attendance, optional assignment completion).

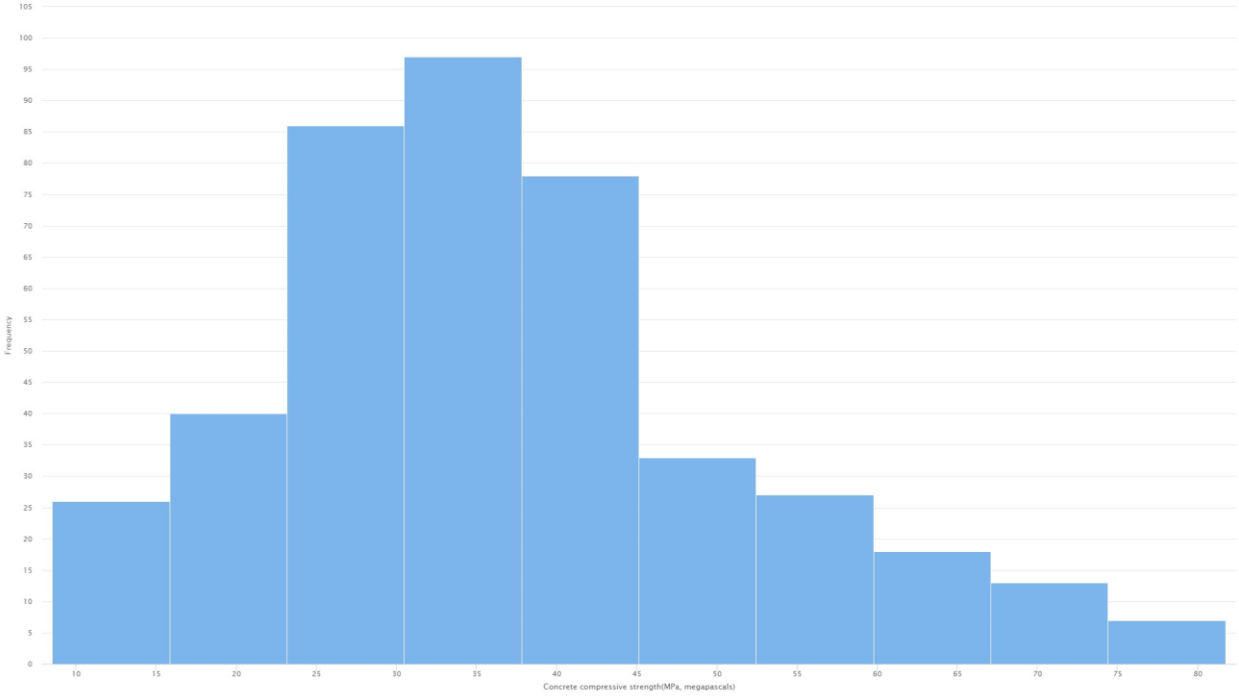
Concrete Mixtures and Compressive Strength

Some relevant background information:

- One method of measuring concrete strength is how much of a compressive load a cylinder of concrete can bear before breaking (usually using a press of some sort)
- The benchmark used in this analysis for a passing grade (logistic model) is 35 megapascals (roughly 5000 PSI) which is a common standard for compressive strength for concrete in roadways
- Only data points at 28 days of aging were used as concrete takes time to fully set and 28 days is the standard for concrete that is ready for load bearing
- Models created with a 70% partition of data and applied to remaining 30% (random shuffling)
- All figures uploaded in separate file for easier viewing

Check for Normal Distribution

Figure 1. Concrete Compressive Strength in Megapascals (MPa) Histogram



Number of bins = 10
Can see a skewed right distribution that we will consider close enough for the purposes of this exercise.

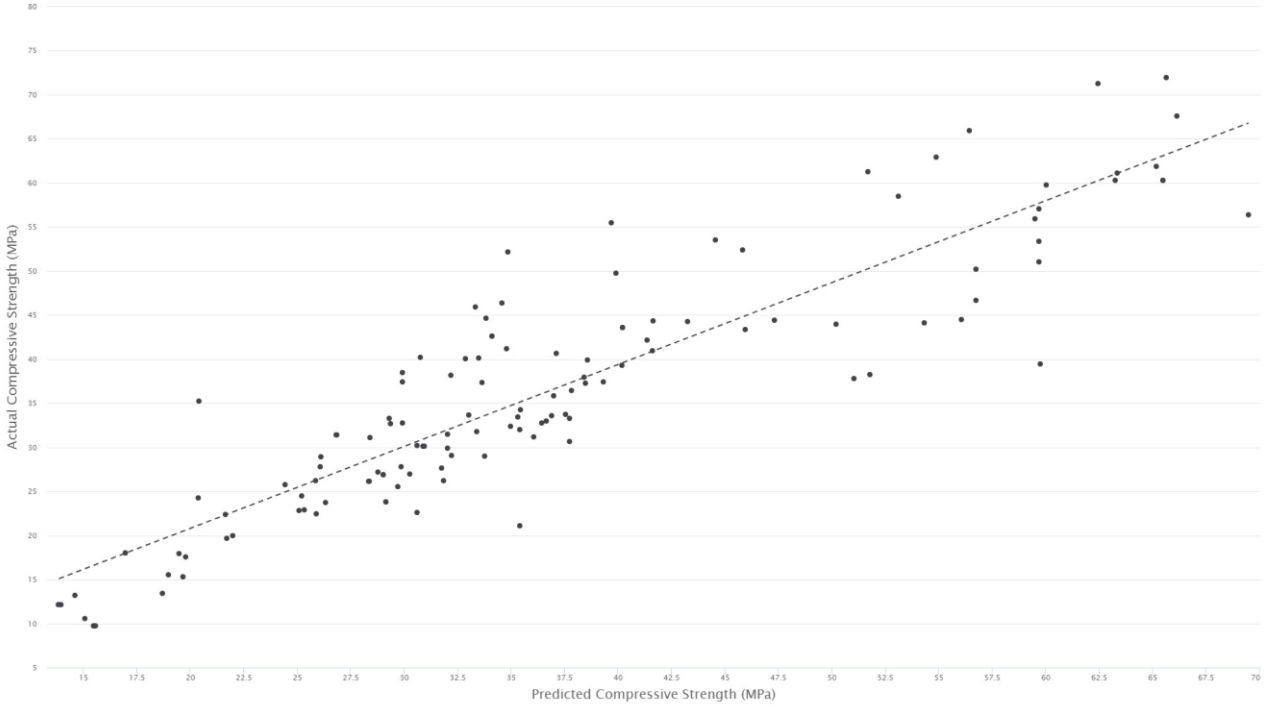
Multiple Regression Compressive Strength of Concrete (MPa)

0.169 * Cement (component 1)(kg in a m^3 mixture)
+ 0.144 * Blast Furnace Slag (component 2)(kg in a m^3 mixture)
+ 0.103 * Fly Ash (component 3)(kg in a m^3 mixture)
- 0.088 * Water (component 4)(kg in a m^3 mixture)
+ 0.034 * Coarse Aggregate (component 6)(kg in a m^3 mixture)
+ 0.050 * Fine Aggregate (component 7)(kg in a m^3 mixture)
- 81.069

Table 1. Multiple Regression Data

All Components in kg per m^3 mixture	Coefficient	Std. Error	Std. Coefficient	Tolerance	t-stat	p-value
Cement	0.169	0.012	1.156	0.565	14.269	0.0
Blast Furnace Slag	0.144	0.014	0.846	0.996	10.412	0.0
Fly Ash	0.103	0.018	0.457	0.872	5.832	0.000
Water	-0.088	0.045	-0.113	0.832	-1.967	0.050
Coarse Aggregate	0.034	0.013	0.191	0.952	2.692	0.008
Fine Aggregate	0.050	0.015	0.242	0.961	3.390	0.001

Figure 2. Actual vs Predicted Compressive Strength (MPa) for Multiple Regression Model



Performance Vector

root_mean_squared_error: 6.245 +/- 0.000
squared_correlation: 0.807

Observations:

From the performance vector we can see a squared correlation of 0.807 which is decent fit and the line show in Figure 2 appears a good fit at first glance. Looking into the variables we can see that both the water content and course aggregate content both have p values above the standard threshold of 0.005 and further modeling could be done where those two values are not included.

Logistic Regression for Compressive Strength of Concrete (MPa)

Using the same concrete compressive strength data, we can convert the compressive strength to a binary value of a passing or failing rating based on a threshold, in this case 35MPa as it is a standard value for compressive strength in roadways.

Figure 3. Logistic Regression Test Model for 35MPa Compressive Strength

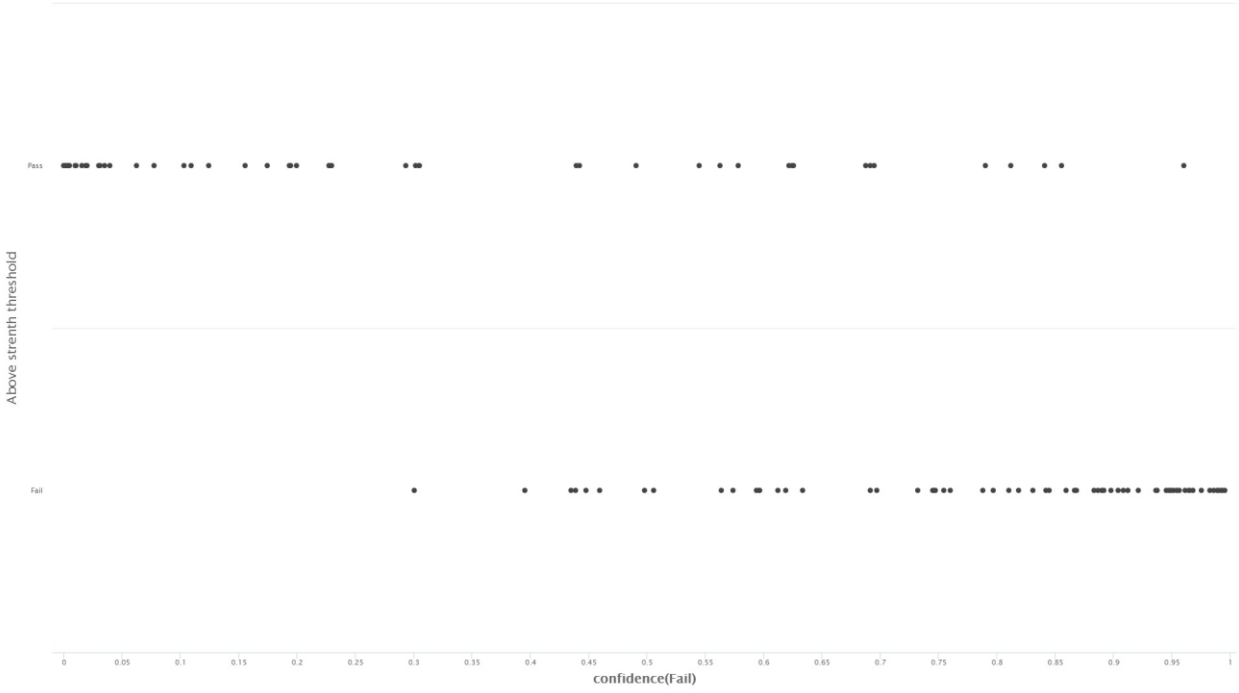


Table 1. Logistic Regression with no Threshold.

	true Pass	true Fail	class precision
pred. Pass	47	7	87.04%
pred. Fail	14	59	80.82%
class recall	77.05%	89.39%	

Overall Accuracy: 83.46%

If we only care if the road is strong enough, we can weight the logistic regression to value the true fail to reduce the chances of having a collapse at the expense of categorizing true passes as fails.

Table 2. Logistic Regression with Fail Threshold Set to 0.60

	true Pass	true Fail	class precision
pred. Pass	44	1	97.78%
pred. Fail	17	65	79.27%
class recall	72.13%	98.48%	

Overall Accuracy: 85.04