DAEN 500- DL1 – Data Analytics Fundamentals

Summer 2020 Final Examination
6/1 – 7/27/2020

Final Submission Deadline: NLT 11:59PM (EST). Monday, 27 July 2020
Failure to submit ON TIME will result in DAEN COURSE FAILURE

Name: Ferdinand Percentie **GMU G#** G01291432

Student Signature (Honor Certification): Ferdinand Percentie (Signature attached)

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!

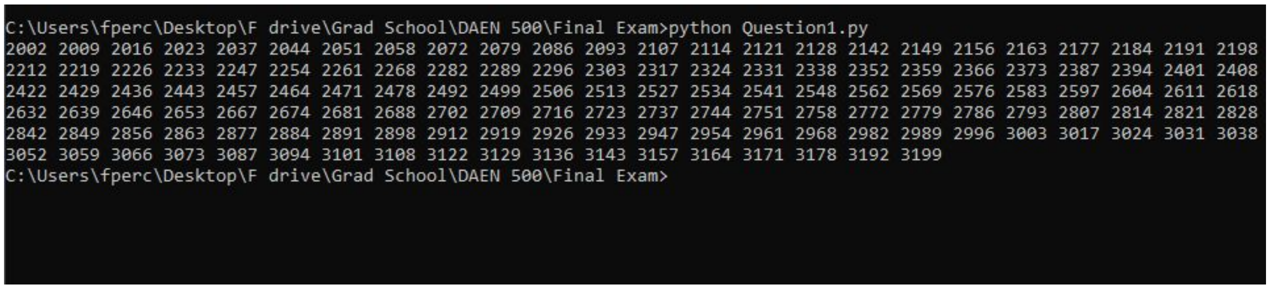
FINAL EXAM PROBLEMS
COMPLETE ALL & INSERT ANSWERS BELOW QUESTIONS

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 your successful run 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**.

```
#Script: Question1.py
#Author: Ferdinand Percentie
#Date: July 22 2020

'''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
                    b) numbers between 2000 and 3200'''

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



```
C:\Users\fperc\Desktop\F drive\Grad School\DAEN 500\Final Exam>python Question1.py
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
C:\Users\fperc\Desktop\F drive\Grad School\DAEN 500\Final Exam>
```

Problem 2: Python Programming Problem (15 Points Total)

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

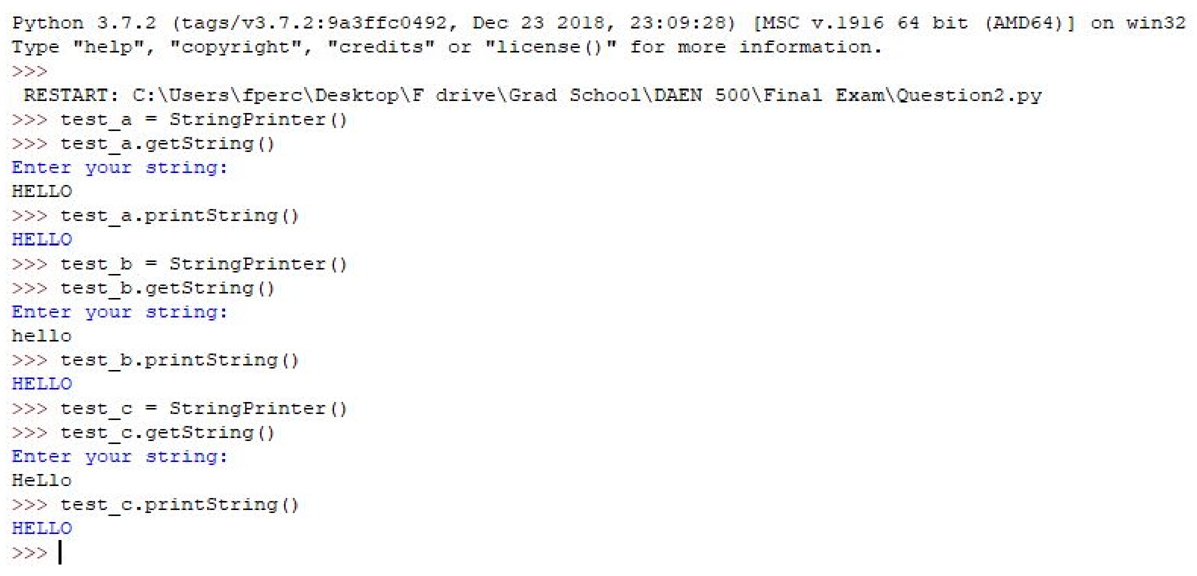
a) define a class which has *at least two* methods

- **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).***

```
#Script: Question2.py
#Author: Ferdinand
#Date: July 22 2020

class StringPrinter:
        '''Class that has two methods: getString and printString'''
        def __init__(self):
                self.string = ''
        def getString(self):
                '''Get a string from console input.'''
                self.string = input('Enter your string:\n')
        def printString(self):
                ''To print the string in uppercase.'''
                print(self.string.upper())
```



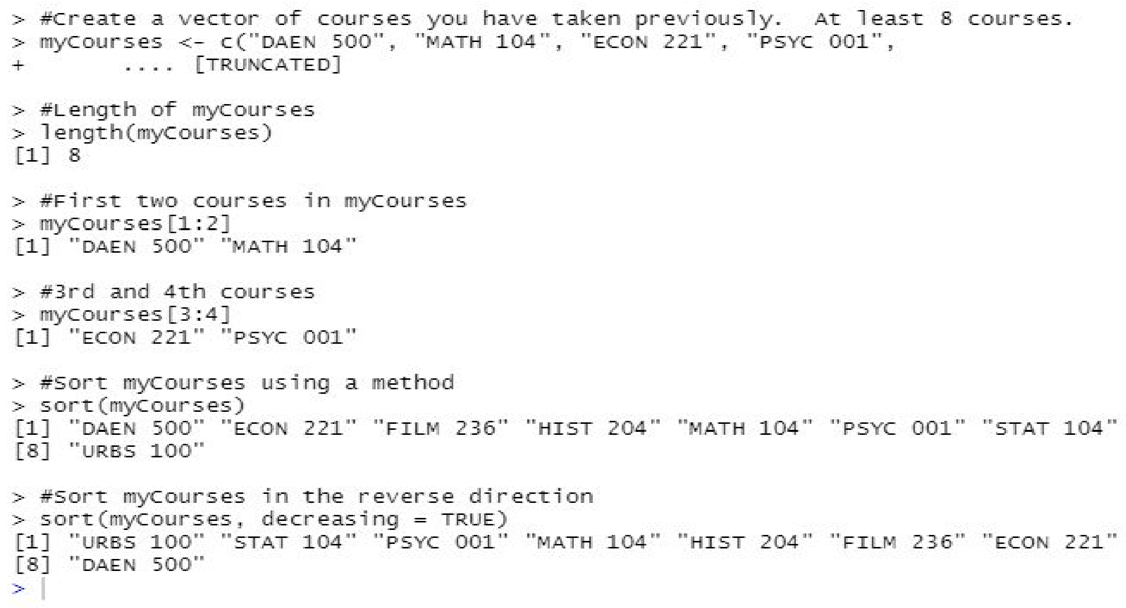
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\fperc\Desktop\F drive\Grad School\DAEN 500\Final Exam\Question2.py
>>> test_a = StringPrinter()
>>> test_a.getString()
Enter your string:
HELLO
>>> test_a.printString()
HELLO
>>> test_b = StringPrinter()
>>> test_b.getString()
Enter your string:
hello
>>> test_b.printString()
HELLO
>>> test_c = StringPrinter()
>>> test_c.getString()
Enter your string:
HeLlo
>>> test_c.printString()
HELLO
>>> 
```

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.

```
#Create a vector of courses you have taken previously.  At least 8 courses.
myCourses <- c("DAEN 500", "MATH 104", "ECON 221", "PSYC 001",
        "URBS 100", "HIST 204", "STAT 104", "FILM 236")

#Length of myCourses
length(myCourses)

#First two courses in myCourses
myCourses[1:2]

#3rd and 4th courses
myCourses[3:4]

#Sort myCourses using a method
sort(myCourses)

#Sort myCourses in the reverse direction
sort(myCourses, decreasing = TRUE)
```



```
> #Create a vector of courses you have taken previously.  At least 8 courses.
> myCourses <- c("DAEN 500", "MATH 104", "ECON 221", "PSYC 001",
+        .... [TRUNCATED]

> #Length of myCourses
> length(myCourses)
[1] 8

> #First two courses in myCourses
> myCourses[1:2]
[1] "DAEN 500" "MATH 104"

> #3rd and 4th courses
> myCourses[3:4]
[1] "ECON 221" "PSYC 001"

> #Sort myCourses using a method
> sort(myCourses)
[1] "DAEN 500" "ECON 221" "FILM 236" "HIST 204" "MATH 104" "PSYC 001" "STAT 104"
[8] "URBS 100"

> #Sort myCourses in the reverse direction
> sort(myCourses, decreasing = TRUE)
[1] "URBS 100" "STAT 104" "PSYC 001" "MATH 104" "HIST 204" "FILM 236" "ECON 221"
[8] "DAEN 500"
> 
```

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)**

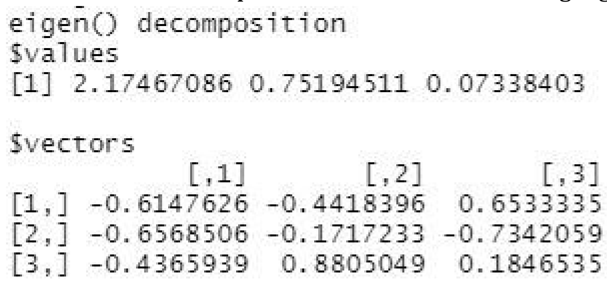
1) A component is a variable formed by combining more than one predictor variables, typically those that are highly correlated or interdependent. Typically correlation values are used to weight the variables in the newly constructed component. (i.e. Z = v1*x1 + v2*x2, where Z is the principal component, the v's are the respective weights and the x's are the predictor variable)

2) Principal Component Analysis is the process of using these components, combining correlated or interdependent predictor variables, to lessen the number of variables being analyzed and simplify analysis, in order to maximize the data variability along as few components as possible.

3) For example, if you wanted to analyze the prices of a house in a neighborhood, you could combine highly correlated variables into components to simplify the analysis. The square footage of a house may be highly correlated to the number of bedrooms, number of bathrooms, and the property tax bill for the house. Combining these predictor variables into a single component can filter out the noise that comes from having a lot of highly correlated variables and will help from overweighting factors relating to the size of the house in your analysis on price and can help bring out other variables (in this example, quality of the home, safety of the neighborhood, quality of the schools, etc.).
To perform Principal Component Analysis,, you would first arrange a correlation table of the predictor variables. Then, you would convert that correlation table to a correlation matrix, calculate the eigenvalues for that matrix, and use that to calculate eigenvectors for each column. You can then produce the weighted components of the principal components.
Example with Correlation Table using hypothetical variables #Bed, #Bath, Tax:

	# Bed	# Bath	Tax
# Bed	1	0.9	0.3
# Bath	0.9	1	0.5
Tax	0.3	0.5	1

The calculations are performed with the following eigenvalues and vectors:



```
eigen() decomposition
$values
[1] 2.17467086 0.75194511 0.07338403

$vectors
           [,1]       [,2]       [,3]
[1,] -0.6147626 -0.4418396  0.6533335
[2,] -0.6568506 -0.1717233 -0.7342059
[3,] -0.4365939  0.8805049  0.1846535
```

The principal components produced as a result of this example would result in principal components of::
Z1 = -0.6147626*X1 -0.4418396*X2 +0.6533335*X3
Z2 = -0.6568506*X1 -0.1717233*X2 -0.7342059*X3
Z3 = -0.4365939*X1 +0.8805049*X2 +0.1846535*X3

If these components were too related (i.e. orthogonal to one another when charted), you could simplify further analysis by removing one of those components.

Problem 5: Multiple vs. Logistic (30 points)

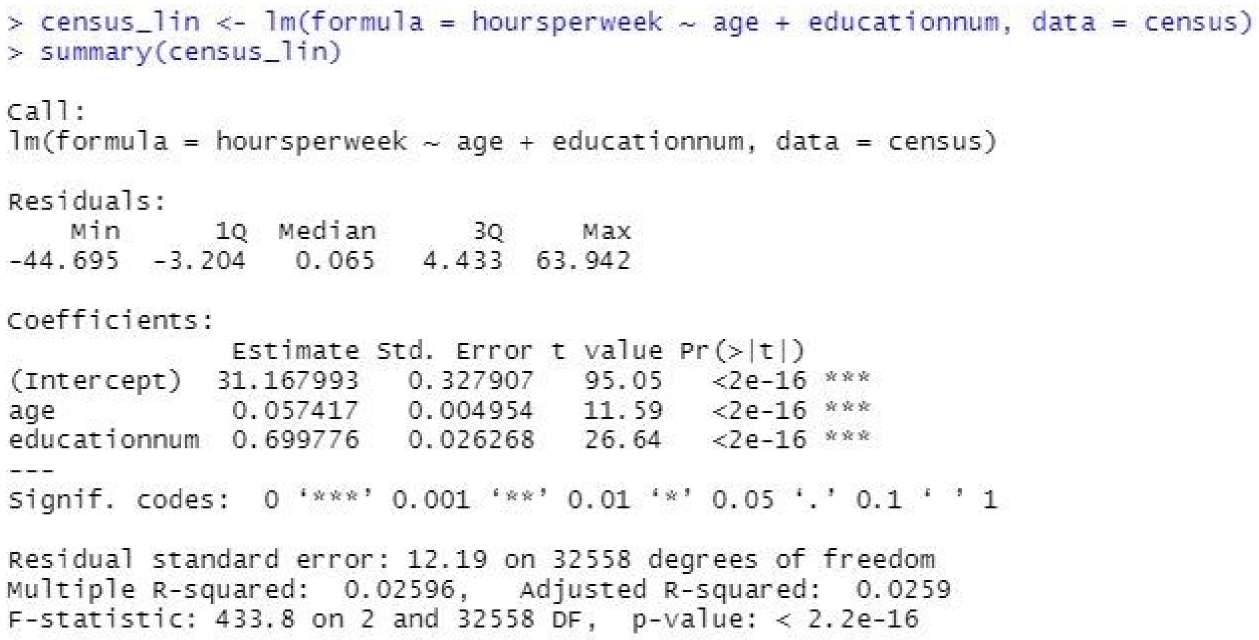
- (a) Describe: What is difference between Multiple Regression and Logistic Regression? What circumstances might determine which to use? (10 points)
- (b) Demonstrate: Using any data, and any tool set you've learned about, show differences (20 points)
- SUGGESTION: may be solved using RapidMiner, or 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) Multiple Regression is the tool used to determine the linear relationship between one response variable and one or more predictor variables. Depending on whether the regression equation is representing a sample or population, the equation ($Y=\beta0+\beta1X1+\beta2X2+\ldots$) may have an added error term (e) if the data comes from a sample.

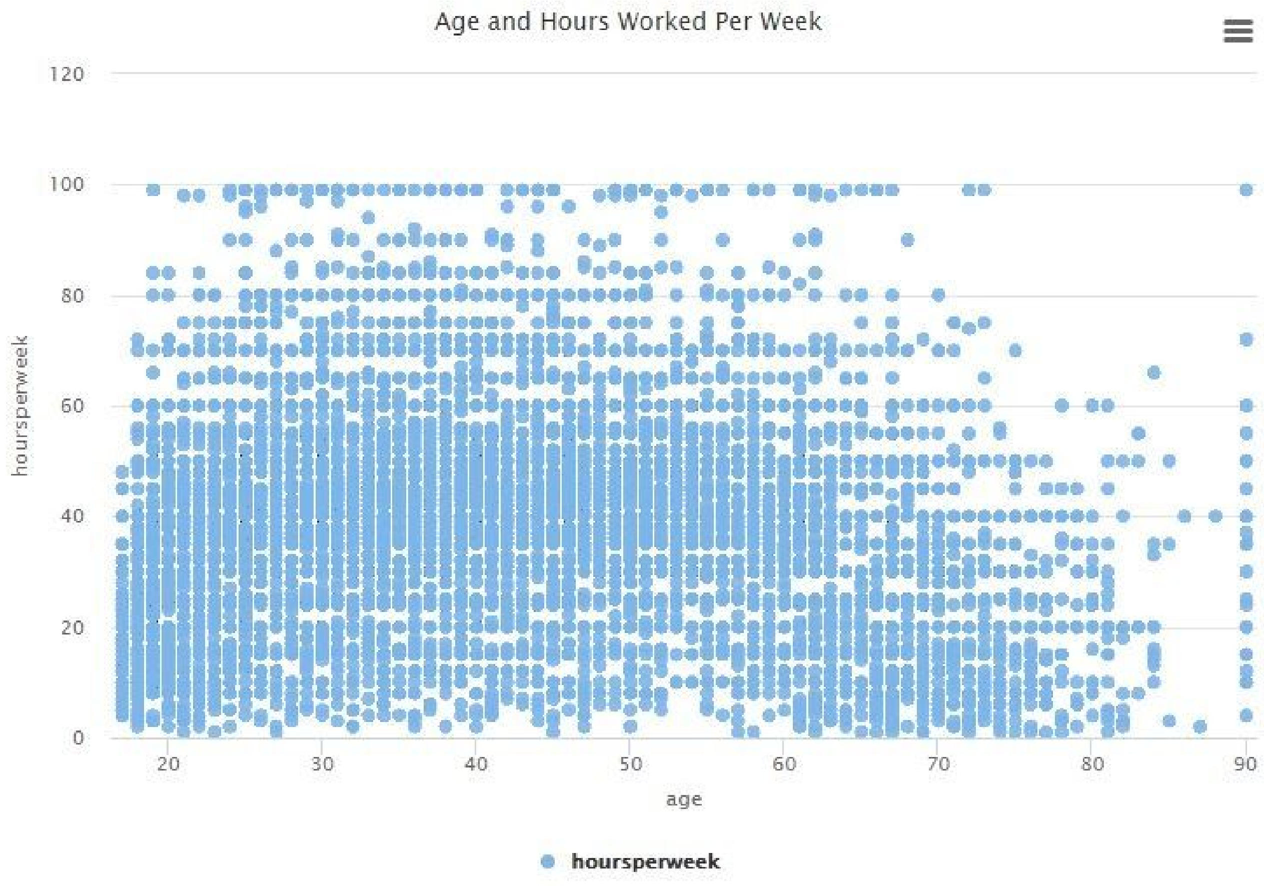
Logistic Regression is the tool used to model a regression where the response variable has a binary outcome. Whereas multiple regression would fit a straight line to the data, the logisitic regression model fits a "S" -shaped curve and is represented as an exponential function divided by another exponential function. ($E(Y)=\pi(X)=(e^\wedge(\beta0+\beta1X))/(1+e^\wedge(\beta0+\beta1X))$). Like the multiple regression function, the logistic regression may have an additional error term (e), if the data come from a sample.

b) Using the adults data set (https://archive.ics.uci.edu/ml/datasets/Adult) from UCIS, I've used the hoursperweek column as the response variable and age and educationnum (a ranking of education levels) columns as the predictor variables. For multiple linear regression, I have to use multiple linear regression formula: $Y=\beta0+\beta1X1+\beta2X2\ldots$ The coefficients in the multiple regression model suggest that both predictor variables have somewhat of a positive effect on the hours worked per week when these variables are modeled together. The scatter plots in the graphs below would be used to apply the linear regression line based on the formula above (however, the chart made in RapidMiner refused to produce a regression line for the number of rows of data in the adult data set).

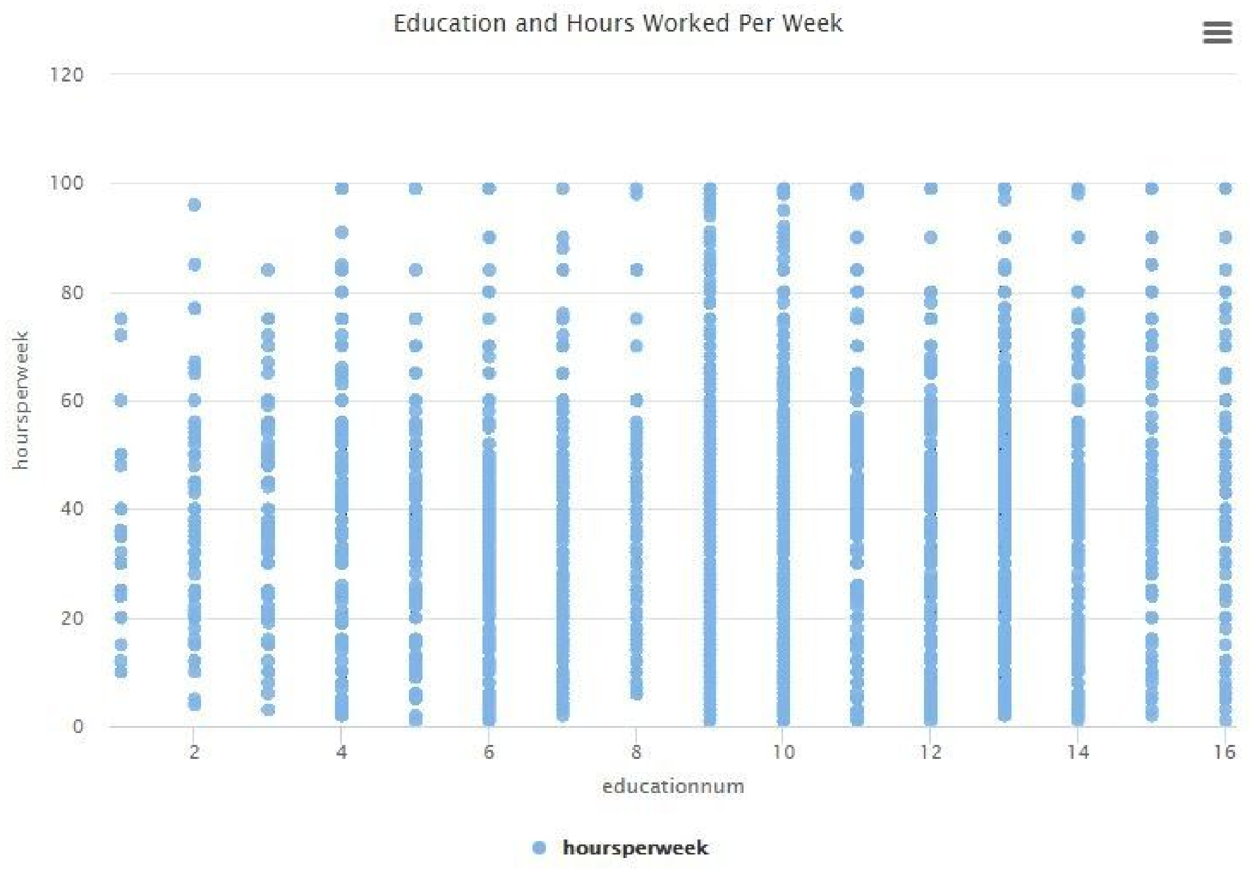


```
> census_lin <- lm(formula = hoursperweek ~ age + educationnum, data = census)
> summary(census_lin)

Call:
lm(formula = hoursperweek ~ age + educationnum, data = census)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.695  -3.204   0.065   4.433  63.942 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  31.167993   0.327907   95.05   <2e-16 ***
age           0.057417   0.004954   11.59   <2e-16 ***
educationnum  0.699776   0.026268   26.64   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.19 on 32558 degrees of freedom
Multiple R-squared:  0.02596,	Adjusted R-squared:  0.0259 
F-statistic: 433.8 on 2 and 32558 DF,  p-value: < 2.2e-16
```

Plot of Age and Hours Worked Per Week



Plot of Education and Hours Worked Per Week

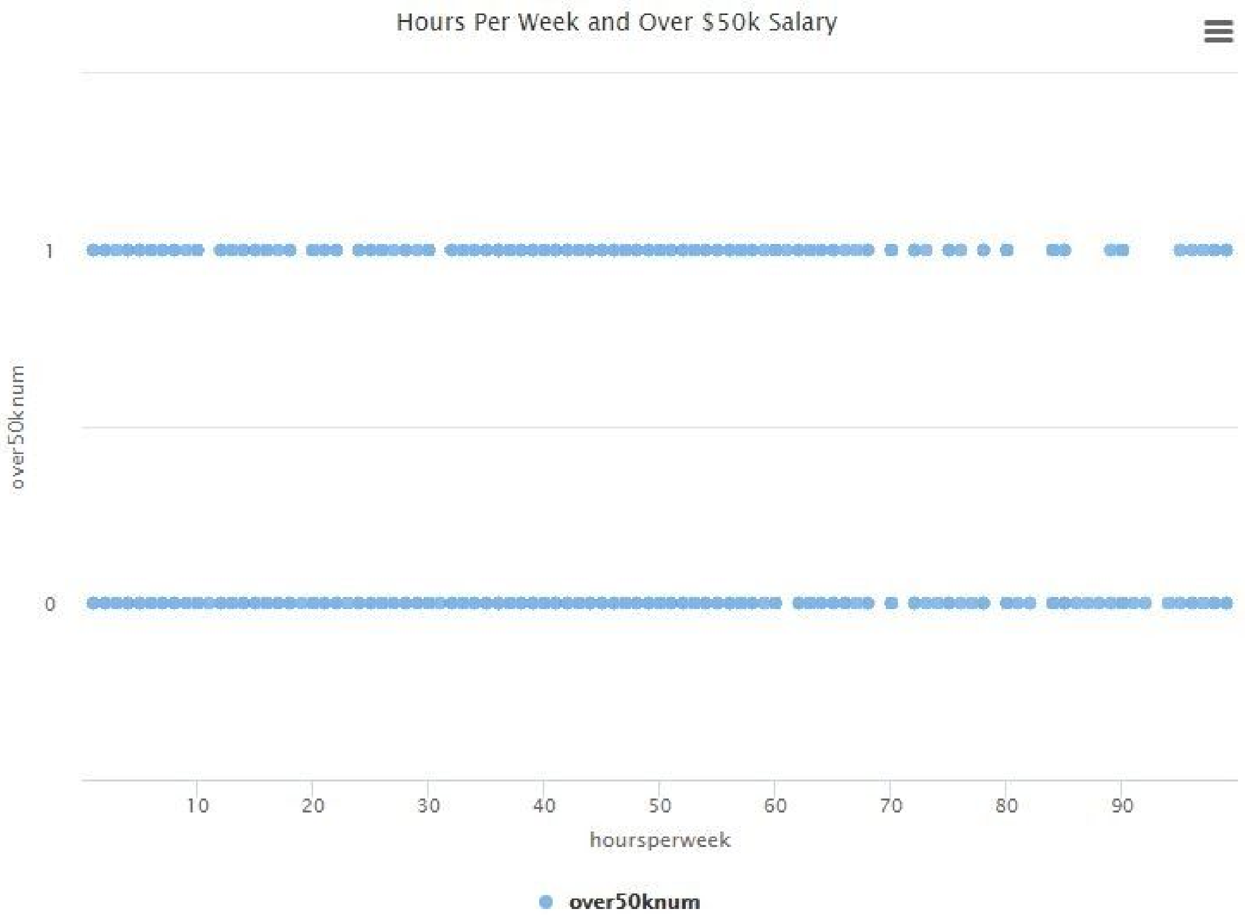


Logistic Regression

Logistic regression however is best used for data where the outcome is a binary response. In the adult data set, we have a binary response variable in the “Over 50K” column (whether or not this row is predictive of an income above $50k). I convert it into a binary variable (0, 1) and use it as the response variable and use hoursperweek as the predictor variable. The regression summary below displays the regression coefficient of 0.04645 for age which suggests that hours per week has a slight effect on the chances of having an income over 50k when the two variables are modeled together. The graph at the bottom of the page is an example of the data, which would be fitted with a S-shaped curve (no example in the Zybooks or Lynda materials) based on the formula: E(Y)= π (X)=(e^(β 0+ β 1X))/(1+e^(β 0+ β 1X)), where B0 and B1 are our coefficient estimates below.

.

```
Call:
glm(formula = over50knum ~ hoursperweek, family = binomial, data = census)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-1.8440  -0.7123  -0.7123  -0.3557   2.4899  

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  -3.100074   0.052898  -58.60   <2e-16 ***
hoursperweek  0.046450   0.001184   39.22   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 35948  on 32560  degrees of freedom
Residual deviance: 34187  on 32559  degrees of freedom
AIC: 34191

Number of Fisher Scoring iterations: 4
```


Hours Per Week and Over $50k Salary
over50knum
hoursperweek
● over50knum