Predicting student's test scores based on their first two assignment and quiz scores using the best model and parameters

Finn Massey

2023-11-14

Part 1 - Loading in appropriate libraries

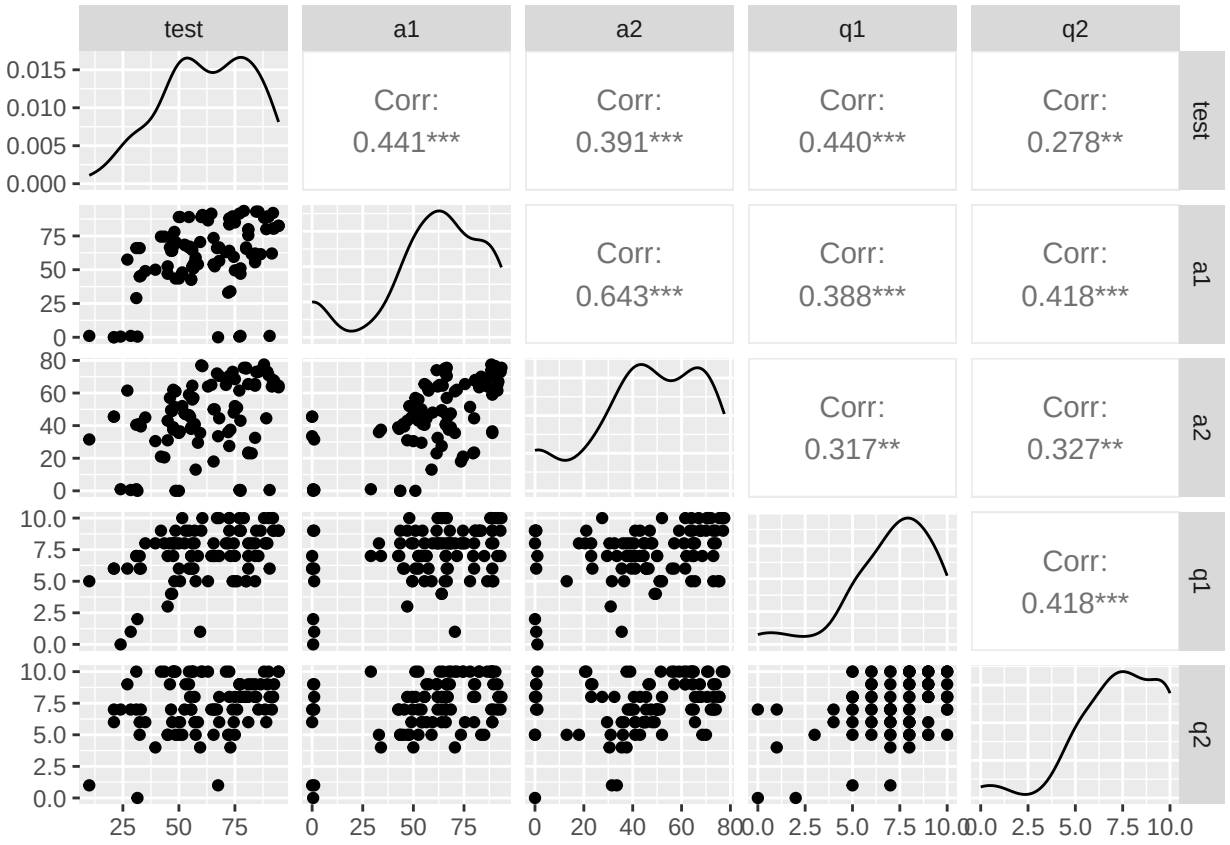
```
# Libraries being used
suppressPackageStartupMessages({
  library(tidyverse)
  library(ggplot2)
  library(GGally)
  library(dplyr)
  library(car)
  library(mgcv)
  library(MuMIn)
  library(insight)
  library(crossval)
  library(pROC)
})
```

Part 2 - Loading in the data

```
stats330 <- read.csv("stats330_grades.csv", stringsAsFactors=T)
```

Part 3 - A pairs plot of the data

```
ggpairs(stats330, columns = 2:6)
```



While almost all the variable combinations have a correlation of around 0.4, the relationship between a1 and a2 have a correlation of 0.643 which is the highest correlated pair of variables. According to the designated graph, this correlation represents a strong positive linear relationship between to two variables meaning that as a1 increases so does a2. There is an exception of a few outliers which occur when one of the two variables have a score of 0. The variable pair with the lowest correlation is q2 and test with a correlation of 0.278. The corresponding graph looks very scatter with no obvious visible trend.

Part 4 - Appropriate numeric summaries for all the variables.

```
summary(stats330)
```

```
##                                     id          test             a1
##  withdrawn                           : 3   Min.   :10.00   Min.   : 0.00
##  017e9b40-e806-423e-8211-9f029e7507e7: 1   1st Qu.:48.12   1st Qu.:49.62
##  0575c20a-515c-4f5f-b8b9-f1399487cb10: 1   Median :63.75   Median :64.25
##  0c9cf32d-8fa4-4c85-89de-5f318e301d11: 1   Mean   :62.41   Mean   :60.26
##  1238a6e3-9929-45f9-9e35-29a9728c24b3: 1   3rd Qu.:78.62   3rd Qu.:80.00
##  13ef667c-e9bc-47c7-9210-68d5c0698a78: 1   Max.   :94.50   Max.   :93.50
##  (Other)                              :94   NA's   :4       NA's   :4
##        a2              q1               q2
##  Min.   : 0.00   Min.   : 0.000   Min.   : 0.000
```

```
##  1st Qu.:35.50   1st Qu.: 6.000   1st Qu.: 6.000
##  Median :47.25   Median : 8.000   Median : 8.000
##  Mean   :46.26   Mean   : 7.162   Mean   : 7.374
##  3rd Qu.:64.88   3rd Qu.: 9.000   3rd Qu.: 9.000
##  Max.   :77.50   Max.   :10.000   Max.   :10.000
##  NA's   :4       NA's   :3        NA's   :3
```

From this summary we can see that there are a number of missing entries in all of the variables which will need to be removed before we fit any models to the data. We also learnt that the max score for the two test were 10/10, 94.5/100 for assignment 1 and 77.5 for assignment 2, and 94.5 for the test. Quiz 2 had the highest mean with 7.374 / 10 and assignment 2 had the lowest mean with 46.26 / 80. Another interesting fact is that the only variable where the minimum isn't zero is the test.

Part 5 - Cleaning the data

```
# Remove all entries with at least one empty value
stats330_clean <- stats330[rowSums(is.na(stats330)) == 0,]
```

Part 6 - The variance inflation factor

```
stats330.fit <- glm(test ~ a1 + a2 + q1 + q2, data=stats330_clean)
vif(stats330.fit)
```

```
##       a1       a2       q1       q2
## 1.928425 1.727756 1.244883 1.278805
```

The variance inflation factor explains the level of multicollinearity that exists between the variable and the other independent variables in a regression model.as well as the extent to which the variance of the estimated coefficient is increased due to multicollinearity. Since the VIF values for all variables are less than 5 best still fairly close to 1, we can say that it suggests a moderate level of multicollinearity. While there most likely still is some correlation with other variables it is not substantial enough to cause any issues.

Part 7 - Fitting a GAM model

Part A - Fitting the model

```
stats330_gam.fit = gam(test ~ q1 + q2 + a1 + a2, family="gaussian", data = stats330_clean)
```

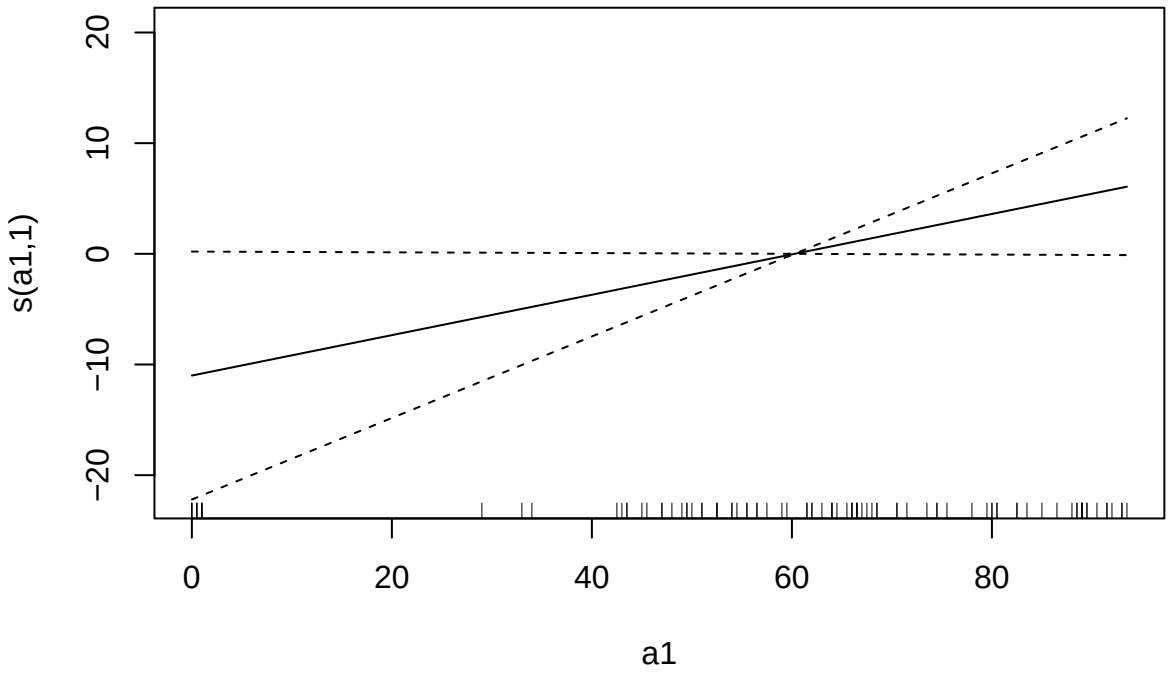
I have chosen the Gaussian family (normal distribution) because its accepted that most data collected which involves human behavior has a normal distribution and this variable appear to be no different as shown by the distribution graph for test in the pairs plot. The family shouldn't be binomial as the test column isn't binary, and it shouldn't be poisson because the test column doesn't contain counts. I felt that the Gaussian/normal distribution best described the test column due to it having scores created by human students.

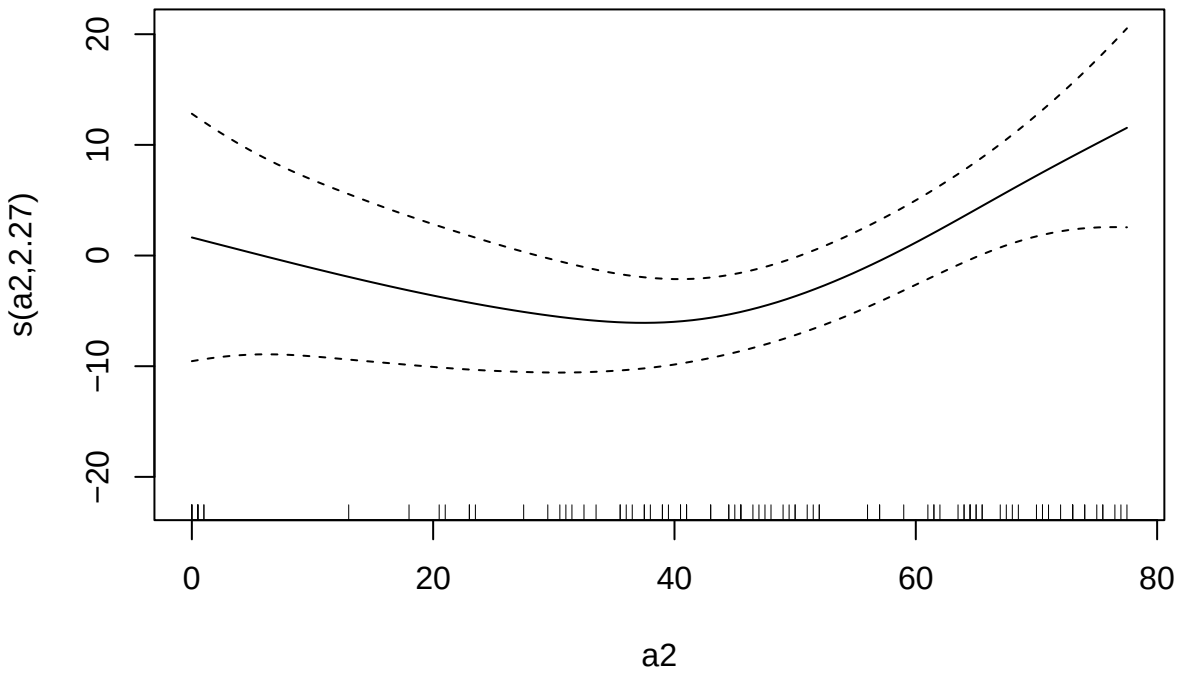
Part B - Smoothing the appropriate numerical variables

```
stats330_gam.fit = gam(test ~ q1 + q2 + s(a1) + s(a2), family="gaussian", data = stats330_clean)
```

Part C - Output plots

```
plot(stats330_gam.fit)
```





Part D - Comment on the plots

As observed from the 2 graphs above, we'll probably need a quadratic term for a2 as you probably couldn't fit a straight line through the graph without touching the dotted lines and the sample line is very curved, and you'd also potentially need a quadratic term for a1 as the dotted lines appear to cross over meaning it would be impossible for a line to go through the graph and not touch a dotted line.

Part 8 - Fitting a model using dredge()

```
options(na.action = "na.fail")
lm_initial = glm(test ~ q1 + q2 + a1 + I(a1^2) + a2 + I(a2^2), family="gaussian", data = stats330_clean)
dredge(lm_initial, rank = "AIC")
```

```
## Fixed term is "(Intercept)"
```

```
## Global model call: glm(formula = test ~ q1 + q2 + a1 + I(a1^2) + a2 + I(a2^2), family = "gaussian",
##     data = stats330_clean)
## ---
## Model selection table
##    (Intrc)        a1       a1^2      a2     a2^2    q1       q2 df   logLik
## 30   36.12  0.193000            -0.6625 0.010090 2.593           6 -410.737
## 31   37.47            0.0016810 -0.5508 0.008879 2.745           6 -411.319
```

```
## 32   35.71  0.258000 -0.0006961 -0.6969 0.010560 2.568          7 -410.690
## 62   37.12  0.198100            -0.6719 0.010220 2.632 -0.20340  7 -410.710
## 29   37.97                     -0.5097 0.009772 3.084          5 -413.204
## 63   37.70            0.0016900 -0.5521 0.008902 2.755 -0.04388  7 -411.318
## 27   28.78            0.0015250         0.002619 2.794          5 -413.653
## 64   36.81  0.269500 -0.0007564 -0.7106 0.010760 2.611 -0.23240  8 -410.655
## 26   26.82  0.135300                   0.002840 2.760          5 -413.913
## 25   29.82                            0.003861 3.100          4 -415.149
## 61   36.52                     -0.5023 0.009594 3.012  0.28090  6 -413.152
## 59   28.05            0.0014960         0.002590 2.759  0.15930  6 -413.637
## 28   28.83 -0.003131  0.0015540         0.002618 2.796          6 -413.653
## 58   26.20  0.132300                   0.002816 2.730  0.14450  6 -413.900
## 19   28.88            0.0026590                  3.062          4 -415.977
## 57   27.77                            0.003719 2.989  0.43340  5 -415.029
## 14   50.68  0.260300            -0.7228 0.011200                 5 -415.191
## 23   26.96            0.0020440  0.1182          2.935          5 -415.274
## 60   28.15 -0.007542  0.0015660         0.002589 2.763  0.16300  7 -413.637
## 18   25.26  0.251100                             3.044          4 -416.805
## 51   27.00            0.0025530                  2.966  0.40610  5 -415.875
## 22   24.19  0.180000             0.1334          2.931          5 -415.939
## 20   29.04 -0.010400  0.0027560                  3.069          5 -415.976
## 16   49.60  0.385600 -0.0013550 -0.7887 0.012110                 6 -415.028
## 46   48.80  0.250900            -0.7069 0.010970         0.31440  6 -415.130
## 55   25.52            0.0019740  0.1154          2.862  0.32290  6 -415.209
## 24   27.72 -0.052300  0.0025120  0.1221          2.964          6 -415.245
## 15   53.82            0.0022520 -0.5724 0.009668                 5 -416.349
## 50   23.50  0.239300                             2.950  0.42300  5 -416.698
## 54   22.76  0.171600             0.1309          2.856  0.34800  6 -415.865
## 52   27.30 -0.021460  0.0027500                  2.977  0.41620  6 -415.870
## 21   26.26                      0.2529          3.380          4 -417.882
## 48   48.12  0.370500 -0.0012760 -0.7718 0.011860         0.25830  7 -414.987
## 47   50.24            0.0021190 -0.5540 0.009335         0.56100  6 -416.152
## 56   26.29 -0.060700  0.0025120  0.1197          2.889  0.34810  7 -415.170
## 53   23.00                      0.2358          3.176  0.74110  5 -417.541
## 11   45.08            0.0021010         0.003172                 4 -418.628
## 10   41.50  0.201700                   0.003329                 4 -418.654
## 43   40.57            0.0019250         0.003003         0.76590  5 -418.272
## 42   37.75  0.183600                   0.003184         0.70140  5 -418.362
## 13   57.32                     -0.5187 0.011050                 4 -419.496
## 12   43.14  0.099460  0.0011560         0.003169                 5 -418.524
## 45   50.22                     -0.4902 0.010280         1.05100  5 -418.806
## 44   39.38  0.074590  0.0012260         0.003010         0.72110  6 -418.214
## 9    49.14                            0.005045                 3 -421.271
## 41   41.58                            0.004536         1.19400  4 -420.404
## 7    43.54            0.0026930  0.1589                         4 -420.595
## 3    47.13            0.0035710                                 3 -421.746
## 35   40.45            0.0032020                          1.10900  4 -421.026
## 39   37.95            0.0024350  0.1472                  0.97100  5 -420.036
## 6    39.37  0.255500             0.1652                         4 -421.093
## 49   25.21                                           3.755  1.34700  4 -421.246
## 2    41.44  0.348000                                            3 -422.296
## 17   31.81                                           4.230          3 -422.347
## 38   34.37  0.227100             0.1562                  0.95690  5 -420.565
## 8    42.75  0.044640  0.0022880  0.1553                         5 -420.575
```

```
## 4    45.12  0.102800  0.0025920                                4 -421.641
## 34   35.72  0.310600                           1.07100  4 -421.646
## 36   39.40  0.065690  0.0025900                1.07000  5 -420.983
## 40   37.73  0.014640  0.0023040  0.1461        0.96350  6 -420.034
## 37   36.49                      0.3041        1.59100  4 -423.304
## 5    46.04                      0.3539                 3 -424.793
## 33   42.84                                    2.62600  3 -428.971
## 1    62.41                                             2 -432.917
##      AIC delta weight
## 30 833.5  0.00  0.261
## 31 834.6  1.16  0.146
## 32 835.4  1.91  0.101
## 62 835.4  1.95  0.099
## 29 836.4  2.93  0.060
## 63 836.6  3.16  0.054
## 27 837.3  3.83  0.038
## 64 837.3  3.84  0.038
## 26 837.8  4.35  0.030
## 25 838.3  4.82  0.023
## 61 838.3  4.83  0.023
## 59 839.3  5.80  0.014
## 28 839.3  5.83  0.014
## 58 839.8  6.33  0.011
## 19 840.0  6.48  0.010
## 57 840.1  6.58  0.010
## 14 840.4  6.91  0.008
## 23 840.5  7.07  0.008
## 60 841.3  7.80  0.005
## 18 841.6  8.14  0.004
## 51 841.7  8.28  0.004
## 22 841.9  8.40  0.004
## 20 842.0  8.48  0.004
## 16 842.1  8.58  0.004
## 46 842.3  8.79  0.003
## 55 842.4  8.94  0.003
## 24 842.5  9.02  0.003
## 15 842.7  9.22  0.003
## 50 843.4  9.92  0.002
## 54 843.7 10.26  0.002
## 52 843.7 10.27  0.002
## 21 843.8 10.29  0.002
## 48 844.0 10.50  0.001
## 47 844.3 10.83  0.001
## 56 844.3 10.87  0.001
## 53 845.1 11.61  0.001
## 11 845.3 11.78  0.001
## 10 845.3 11.83  0.001
## 43 846.5 13.07  0.000
## 42 846.7 13.25  0.000
## 13 847.0 13.52  0.000
## 12 847.0 13.57  0.000
## 45 847.6 14.14  0.000
## 44 848.4 14.95  0.000
## 9  848.5 15.07  0.000
```

```
## 41 848.8 15.33  0.000
## 7  849.2 15.72  0.000
## 3  849.5 16.02  0.000
## 35 850.1 16.58  0.000
## 39 850.1 16.60  0.000
## 6  850.2 16.71  0.000
## 49 850.5 17.02  0.000
## 2  850.6 17.12  0.000
## 17 850.7 17.22  0.000
## 38 851.1 17.66  0.000
## 8  851.2 17.68  0.000
## 4  851.3 17.81  0.000
## 34 851.3 17.82  0.000
## 36 852.0 18.49  0.000
## 40 852.1 18.59  0.000
## 37 854.6 21.13  0.000
## 5  855.6 22.11  0.000
## 33 863.9 30.47  0.000
## 1  869.8 36.36  0.000
## Models ranked by AIC(x)
```

```
dredge(lm_initial, rank = "BIC")
```

```
## Fixed term is "(Intercept)"

## Global model call: glm(formula = test ~ q1 + q2 + a1 + I(a1^2) + a2 + I(a2^2), family = "gaussian",
##     data = stats330_clean)
## ---
## Model selection table
##    (Intrc)        a1       a1^2      a2     a2^2    q1       q2 df   logLik
## 25   29.82                            0.003861 3.100           4 -415.149
## 30   36.12  0.193000           -0.6625 0.010090 2.593           6 -410.737
## 29   37.97                     -0.5097 0.009772 3.084           5 -413.204
## 31   37.47            0.0016810 -0.5508 0.008879 2.745           6 -411.319
## 27   28.78            0.0015250         0.002619 2.794           5 -413.653
## 19   28.88            0.0026590                  3.062           4 -415.977
## 26   26.82  0.135300                     0.002840 2.760           5 -413.913
## 18   25.26  0.251100                              3.044           4 -416.805
## 57   27.77                               0.003719 2.989  0.43340  5 -415.029
## 14   50.68  0.260300           -0.7228 0.011200                  5 -415.191
## 23   26.96            0.0020440  0.1182          2.935           5 -415.274
## 32   35.71  0.258000 -0.0006961 -0.6969 0.010560 2.568           7 -410.690
## 62   37.12  0.198100           -0.6719 0.010220 2.632 -0.20340  7 -410.710
## 61   36.52                     -0.5023 0.009594 3.012  0.28090  6 -413.152
## 21   26.26                      0.2529          3.380           4 -417.882
## 51   27.00            0.0025530                  2.966  0.40610  5 -415.875
## 63   37.70            0.0016900 -0.5521 0.008902 2.755 -0.04388  7 -411.318
## 59   28.05            0.0014960         0.002590 2.759  0.15930  6 -413.637
## 22   24.19  0.180000             0.1334          2.931           5 -415.939
## 28   28.83 -0.003131  0.0015540         0.002618 2.796           6 -413.653
## 20   29.04 -0.010400  0.0027560                  3.069           5 -415.976
## 58   26.20  0.132300                     0.002816 2.730  0.14450  6 -413.900
## 11   45.08            0.0021010         0.003172                  4 -418.628
```

```
## 15   53.82           0.0022520 -0.5724 0.009668                 5 -416.349
## 10   41.50  0.201700                   0.003329                 4 -418.654
## 9    49.14                             0.005045                 3 -421.271
## 50   23.50  0.239300                                 2.950  0.42300  5 -416.698
## 3    47.13           0.0035710                                      3 -421.746
## 13   57.32                     -0.5187 0.011050                     4 -419.496
## 16   49.60  0.385600 -0.0013550 -0.7887 0.012110                     6 -415.028
## 46   48.80  0.250900           -0.7069 0.010970         0.31440     6 -415.130
## 55   25.52           0.0019740  0.1154          2.862  0.32290     6 -415.209
## 64   36.81  0.269500 -0.0007564 -0.7106 0.010760 2.611 -0.23240     8 -410.655
## 24   27.72 -0.052300  0.0025120  0.1221          2.964              6 -415.245
## 53   23.00                      0.2358          3.176  0.74110     5 -417.541
## 2    41.44  0.348000                                               3 -422.296
## 17   31.81                                      4.230              3 -422.347
## 41   41.58                             0.004536         1.19400     4 -420.404
## 54   22.76  0.171600            0.1309          2.856  0.34800     6 -415.865
## 52   27.30 -0.021460  0.0027500                 2.977  0.41620     6 -415.870
## 60   28.15 -0.007542  0.0015660         0.002589 2.763  0.16300     7 -413.637
## 43   40.57           0.0019250         0.003003         0.76590     5 -418.272
## 7    43.54           0.0026930  0.1589                              4 -420.595
## 42   37.75  0.183600                   0.003184         0.70140     5 -418.362
## 47   50.24           0.0021190 -0.5540 0.009335         0.56100     6 -416.152
## 12   43.14  0.099460  0.0011560         0.003169                    5 -418.524
## 35   40.45           0.0032020                          1.10900     4 -421.026
## 6    39.37  0.255500            0.1652                              4 -421.093
## 45   50.22                     -0.4902 0.010280         1.05100     5 -418.806
## 49   25.21                                      3.755  1.34700     4 -421.246
## 4    45.12  0.102800  0.0025920                                      4 -421.641
## 34   35.72  0.310600                                    1.07100     4 -421.646
## 48   48.12  0.370500 -0.0012760 -0.7718 0.011860         0.25830     7 -414.987
## 56   26.29 -0.060700  0.0025120  0.1197          2.889  0.34810     7 -415.170
## 39   37.95           0.0024350  0.1472                  0.97100     5 -420.036
## 5    46.04                      0.3539                              3 -424.793
## 44   39.38  0.074590  0.0012260         0.003010         0.72110     6 -418.214
## 38   34.37  0.227100            0.1562                  0.95690     5 -420.565
## 8    42.75  0.044640  0.0022880  0.1553                              5 -420.575
## 36   39.40  0.065690  0.0025900                          1.07000     5 -420.983
## 37   36.49                      0.3041                  1.59100     4 -423.304
## 40   37.73  0.014640  0.0023040  0.1461                  0.96350     6 -420.034
## 33   42.84                                              2.62600     3 -428.971
## 1    62.41                                                          2 -432.917
##      BIC delta weight
## 25 848.6  0.00  0.176
## 30 849.0  0.35  0.148
## 29 849.3  0.70  0.125
## 31 850.1  1.51  0.083
## 27 850.2  1.59  0.080
## 19 850.3  1.66  0.077
## 26 850.8  2.11  0.061
## 18 852.0  3.31  0.034
## 57 853.0  4.34  0.020
## 14 853.3  4.67  0.017
## 23 853.5  4.83  0.016
## 32 853.5  4.84  0.016
```

```
## 62 853.5  4.88  0.015
## 61 853.8  5.18  0.013
## 21 854.1  5.46  0.011
## 51 854.7  6.04  0.009
## 63 854.7  6.09  0.008
## 59 854.8  6.15  0.008
## 22 854.8  6.16  0.008
## 28 854.8  6.18  0.008
## 20 854.9  6.24  0.008
## 58 855.3  6.67  0.006
## 11 855.6  6.96  0.005
## 15 855.6  6.98  0.005
## 10 855.6  7.01  0.005
## 9  856.3  7.66  0.004
## 50 856.3  7.68  0.004
## 3  857.2  8.61  0.002
## 13 857.3  8.69  0.002
## 16 857.6  8.93  0.002
## 46 857.8  9.13  0.002
## 55 857.9  9.29  0.002
## 64 858.0  9.35  0.002
## 24 858.0  9.36  0.002
## 53 858.0  9.37  0.002
## 2  858.3  9.71  0.001
## 17 858.4  9.81  0.001
## 41 859.1 10.51  0.001
## 54 859.2 10.60  0.001
## 52 859.2 10.61  0.001
## 60 859.4 10.73  0.001
## 43 859.5 10.83  0.001
## 7  859.5 10.89  0.001
## 42 859.6 11.01  0.001
## 47 859.8 11.18  0.001
## 12 860.0 11.33  0.001
## 35 860.4 11.75  0.000
## 6  860.5 11.89  0.000
## 45 860.5 11.90  0.000
## 49 860.8 12.19  0.000
## 4  861.6 12.98  0.000
## 34 861.6 12.99  0.000
## 48 862.1 13.43  0.000
## 56 862.4 13.80  0.000
## 39 863.0 14.36  0.000
## 5  863.3 14.70  0.000
## 44 863.9 15.30  0.000
## 38 864.1 15.42  0.000
## 8  864.1 15.44  0.000
## 36 864.9 16.25  0.000
## 37 864.9 16.31  0.000
## 40 867.6 18.94  0.000
## 33 871.7 23.06  0.000
## 1  875.0 26.37  0.000
## Models ranked by BIC(x)
```

Part 9 - A function that will enable cross validation

```
predfun.lm <- function(train.x, train.y, test.x, test.y) {
  lm1.fit <- lm(train.y ~ a1 + a2 + I(a2^2) + q1, data=train.x)
  ynew <- predict(lm1.fit, test.x)
  out1 <- mean((ynew - test.y)^2)

  lm2.fit <- lm(train.y ~ I(a1^2) + a2 + I(a2^2) + q1, data=train.x)
  ynew <- predict(lm2.fit, test.x)
  out2 <- mean((ynew - test.y)^2)

  lm3.fit <- lm(train.y ~ a1 + I(a1^2) + a2 + I(a2^2) + q1, data=train.x)
  ynew <- predict(lm3.fit, test.x)
  out3 <- mean((ynew - test.y)^2)

  lm4.fit <- lm(train.y ~ I(a2^2) + q1, data=train.x)
  ynew <- predict(lm4.fit, test.x)
  out4 <- mean((ynew - test.y)^2)

  lm5.fit <- lm(train.y ~ a1 + a2 + I(a2^2) + q1, data=train.x)
  ynew <- predict(lm5.fit, test.x)
  out5 <- mean((ynew - test.y)^2)

  lm6.fit <- lm(train.y ~ a2 + I(a2^2) + q1, data=train.x)
  ynew <- predict(lm6.fit, test.x)
  out6 <- mean((ynew - test.y)^2)

  lm7.fit <- lm(train.y ~ a1*a2 + a1*q1 + a1*q2 + a2*q1 + a2*q2 + q1*q2, data=train.x)
  ynew <- predict(lm7.fit, test.x)
  out7 <- mean((ynew - test.y)^2)

  c(out1, out2, out3, out4, out5, out6, out7)
}
```

Part 10 - Total number of parameters

The all-interactions model has 11 parameters, test, a1, a2, q1, q2, a1:a2, a1:q1, a1:q2, a2:q1, a2:q2, and q1:q2.

Part 11 - Performing cross validation

```
set.seed(592)
cv.out = crossval(predfun.lm, X = stats330_clean[, 3:6], Y = stats330_clean[, 2], K = 10, B = 500, verbo
round(cv.out$stat, 2)
```

```
## [1] 290.74 291.22 299.61 296.91 290.74 294.36 322.31
```

The models with the lowest mean squared prediction error was model 1 and 5 which are actually the same model: test ~ a1 + a2 + I(a2^2) + q1. This model had the lowest AIC score and the 2nd lowest BIC score.

While it was in this case, the model with the lowest AIC/BIC or the most predictors isn't necessarily also the best predictive models because it might be over fitting to the data. If a model has a larger amount of predictors, it may have a low AIC or BIC score due to its ability to fit the training data well, but it not it's ability to generalize new and unseen data and making predictions on new data. A model with a large amount of predictors would also increase in its complexity which would give it a higher probability to detect noise and may have higher variance than other models. This could make models with low BIC or AIC not actually be the most appropriate models which is why it is also important to do further tests on your model rather than just rely on the AIC and BIC scores.

Part 12 - Testing the model by predicting my own test score

```
# Prediction model
prediction_model <- lm(test ~ a1 + a2 + I(a2^2) + q1, data=stats330_clean)
# Update your scores here.
my_data <- data.frame(a1 = 49.5, a2 = 31, q1 = 8, q2 = 4)
# This creates a confidence interval for the mean test score for
# students with these assessments scores
predict(prediction_model, newdata = my_data, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 55.58267 50.09178 61.07355
```

```
# This creates a prediction interval for an individual with
# these assessment scores
predict(prediction_model, newdata = my_data, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 55.58267 22.51868 88.64666
```

Part 13 - Comment on these results

The prediction interval is wider than the confidence interval because while the confidence interval only takes into account the range of uncertainty of the estimate, the prediction interval also takes into account the variability of the data. The confidence interval is used to estimate the population mean whereas the prediction interval is used to predict any specific point meaning that the range of uncertainty of any specific predicted value is bound to be wider than the range of uncertainty of just the population mean.

Part 14 - A confidence interval using non-parametric bootstrapping

```
set.seed(592)
my_vector <- unlist(my_data)

n_iterations <- 1000
bootstrap_means <- numeric(n_iterations)
true_score <- 40

i <- 1
while (i <= n_iterations) {
```

```
  bootstrap_sample <- sample(my_vector, replace = TRUE)
  bootstrap_means[i] <- mean(bootstrap_sample)
  i <- i + 1
}
c(2*true_score-quantile(bootstrap_means, 0.975, na.rm = TRUE), 2*true_score-quantile(bootstrap_means, 0
```

```
## 97.5%  2.5%
## 39.75 74.00
```

This confidence interval means that we are 95% confident that the true mean of test scores with my assessment scores will lie within the range 39.75 - 74. Out of the 1000 iterations if non-parametric bootstrapping we did 95% of those test score means would be within the range of 39.75 - 74.

Part 15 - Creating the 'pass' variable

```
stats330_clean <- stats330_clean %>%
  mutate(pass = ifelse(test >= 50, 1, 0)) %>%
  glimpse()
```

```
## Rows: 98
## Columns: 7
## $ id   <fct> 564349d0-ec7b-4bc5-9be5-3d39a28a9938, a104b926-8a12-45f5-bcdf-6dd~
## $ test <dbl> 47.5, 75.0, 68.0, 57.5, 48.0, 56.0, 84.0, 43.5, 71.0, 31.0, 47.5,~
## $ a1   <dbl> 71.5, 49.5, 66.5, 59.0, 78.0, 64.5, 62.0, 74.5, 63.0, 29.0, 67.5,~
## $ a2   <dbl> 62.0, 52.0, 70.5, 13.0, 51.5, 64.5, 32.5, 20.5, 65.0, 1.0, 37.5, ~
## $ q1   <int> 7, 5, 10, 5, 5, 8, 8, 8, 8, 7, 8, 8, 4, 7, 8, 4, 9, 6, 10, 6, 6, ~
## $ q2   <int> 10, 8, 8, 5, 10, 10, 8, 10, 10, 10, 10, 5, 6, 1, 9, 7, 8, 10, 7, ~
## $ pass <dbl> 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1,~
```

Part 16 - Fitting a model predicting 'pass'

```
passing_model.fit <- glm(pass ~ q1, family="binomial", data=stats330_clean)
```

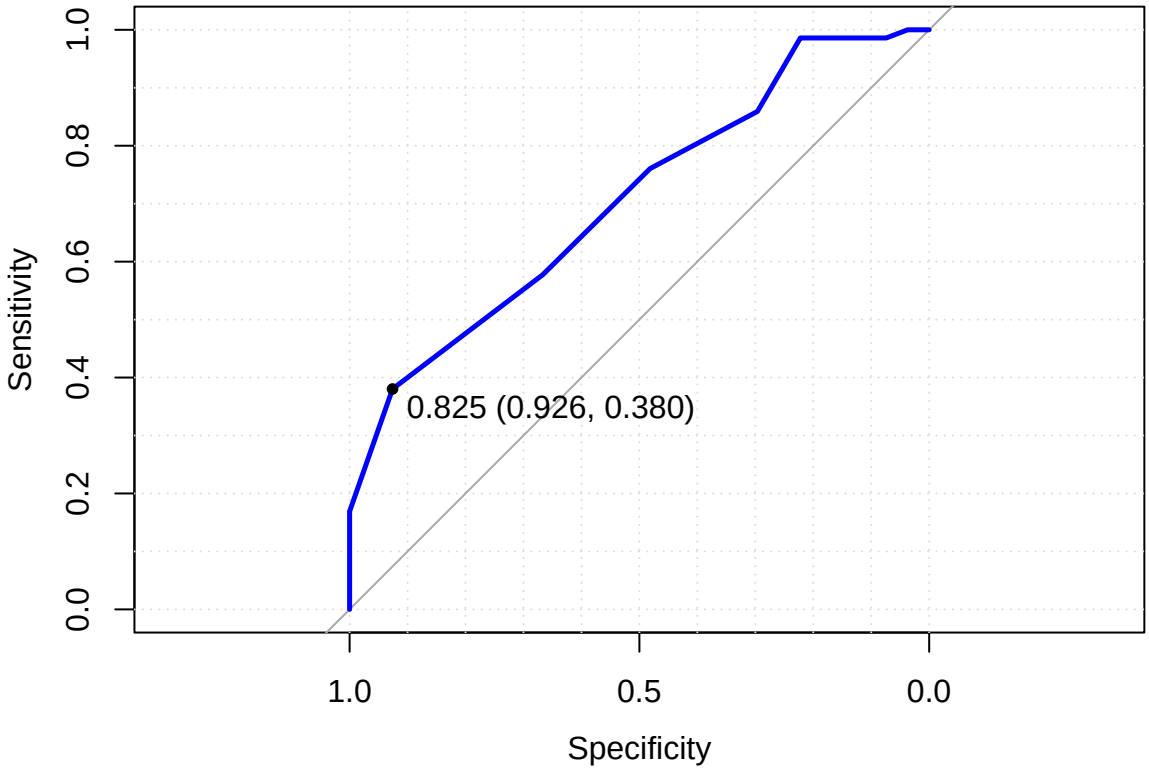
Part 17 - An ROC curve

```
passing_model.roc <- roc(response = stats330_clean$pass, predictor = fitted.values(passing_model.fit))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(passing_model.roc, col = "blue", grid = TRUE, lwd=2.5, print.thres = "best")
```



Part 18 - The AUC value

```
auc_value <- auc(passing_model.roc)
auc_value
```

```
## Area under the curve: 0.7066
```

Part 19 - The threshold grade

```
optimal_cutoff <- coords(passing_model.roc, "best")
optimal_cutoff
```

```
##   threshold specificity sensitivity
## 1 0.8251556   0.9259259   0.3802817
```

This shows that the optimal cutoff is it at the point which maximises both specificity and sensitivity which is 0.9259259, 0.3802817. The threshold for q1 to reach this maximised point is 0.8251556.

Part 20 - The use of this threshold grade

The optimal threshold for q1 0.8251556 but since q1 scores can only be in whole numbers we have to choose whether to round down or up. If we round the threshold up we would be prioritising specificity more than sensitivity because we would be trying to minimise false positives so that only students who truly need support receive the intervention email. If we round the threshold down we would be prioritising sensitivity more than specificity because we would be trying to minimise false negatives so that all students who need support will receive the intervention email. I think in this situation sensitivity is more important than specificity because if we're trying to predict which students will fail the test this early, it would be less harmful to send the letter to someone who doesn't need to then to not send it to someone who does. Due to this we should round the optimal threshold down to 0.8. Instructions: Send the intervention letter to any student who scores a 8 or less in q1.

Part 21 - A confusion matrix

```
confusion_matrix <- table(actual = stats330_clean$pass, pred = round(fitted(passing_model.fit)))
confusion_matrix
```

```
##       pred
## actual  0  1
##      0  6 21
##      1  1 70
```

```
fp <- confusion_matrix[3]
tp <- confusion_matrix[4]
fn <- confusion_matrix[2]
tn <- confusion_matrix[1]
fpr <- fp/(fp+tn)
tpr <- tp/(tp+fn)
pred_error <- (fp+fn)/(fp+fn+tn+tp)
```

Here is the false positive rate:

```
fpr
```

```
## [1] 0.7777778
```

Here is the true positive rate:

```
tpr
```

```
## [1] 0.9859155
```

Here is the prediction error:

```
pred_error
```

```
## [1] 0.2244898
```

Part 22 - Comment on results

There is a chance this model might have over fit to this dataset meaning that when it is exposed to new data the strong patterns and generalisations it made thoroughly on this dataset might not translate to the new dataset therefore resulting in a much higher prediction error. Another reason could be that this dataset isn't actually very big with only 98 rows in the cleaned dataset, which means it might not have had enough data to properly learn all the patterns need to properly predict on new data. We could get a more concrete estimate of the prediction error by performing cross validation or training on more independent data before testing on new data.