ECON 187: Project 1

Austin Pham (905318112)

For this project you will need to use two different datasets of your choice. One will be used for classification and the other for regularization. For the classification dataset make sure you have more than two classes. For your regularization dataset, since the methods focus on variable selection, please make sure to have as many predictors as possible (e.g., 10s or 100s).

Classification

```
body <- read_csv("bodyPerformance.csv")
body <- body %>% mutate(gender = factor(ifelse(gender == "M", 1, 0)))
sum(is.na(body))
```

```
## [1] 0
```

```
head(body)
```

```
## # A tibble: 6 x 12
##     age gender height_cm weight_kg ‘body fat_%‘ diastolic systolic gripForce
##   <dbl> <fct>      <dbl>     <dbl>        <dbl>     <dbl>    <dbl>     <dbl>
## 1    27 1           172.      75.2         21.3        80      130      54.9
## 2    25 1           165       55.8         15.7        77      126      36.4
## 3    31 1           180.      78           20.1        92      152      44.8
## 4    32 1           174.      71.1         18.4        76      147      41.4
## 5    28 1           174.      67.7         17.1        70      127      43.5
## 6    36 0           165.      55.4         22          64      119      23.8
## # ... with 4 more variables: ‘sit and bend forward_cm‘ <dbl>,
## #   ‘sit-ups counts‘ <dbl>, ‘broad jump_cm‘ <dbl>, class <chr>
```

```
plot1 <- body %>% ggpairs(columns = c(1:4,12), ggplot2::aes(col = class, fill = class))
plot2 <- body %>% ggpairs(columns = c(5:8,12), ggplot2::aes(col = class, fill = class))
plot3 <- body %>% ggpairs(columns = c(9:12), ggplot2::aes(col = class, fill = class))
plot1
plot2
plot2
```

Logistic Regression

QUESTION Do I train control over the entire dataset and then predict the entire datset using that as my MSE ??

```r
set.seed(42)
fit.control.cv <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
fit.control.boot <-  trainControl(method = "boot")
mn.fit.cv <- train(class ~ ., data = body, method = "multinom",
                   trControl = fit.control.cv, trace = FALSE,
                   preProcess = c("center", "scale"))
mn.fit.boot <- train(class ~ ., data = body, method = "multinom",
                     trControl = fit.control.boot, trace = FALSE,
                     preProcess = c("center", "scale"))
```

```r
mn.fit.cv$finalModel
```

```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay,
##     trace = FALSE)
##
## Coefficients:
##   (Intercept)       age  gender1   height_cm weight_kg ‘\\‘body fat_%\\‘‘
## B   1.6436941 -1.031072 1.228708 -0.07014251 0.8745149        0.029745697
## C   1.8396621 -1.891377 1.977937  0.11469195 1.2557692        0.005482127
## D   0.4281742 -2.824718 2.598717 -0.30909023 2.4963687        0.459483497
##    diastolic    systolic gripForce ‘\\‘sit and bend forward_cm\\‘‘
## B 0.05764353 -0.01869305 -0.920936                       -1.331405
## C 0.13202220 -0.07684313 -1.519526                       -2.285391
## D 0.25219705 -0.14232365 -2.101412                       -3.597352
##   ‘\\‘sit-ups counts\\‘‘ ‘\\‘broad jump_cm\\‘‘
## B              -1.423940            -0.6803174
## C              -2.603735            -1.1445756
## D              -4.161081            -1.2372507
##
## Residual Deviance: 23128.6
## AIC: 23200.6
```

```r
mn.fit.boot$finalModel
```

```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay,
##     trace = FALSE)
##
## Coefficients:
##   (Intercept)       age  gender1   height_cm weight_kg ‘\\‘body fat_%\\‘‘
## B   1.6436941 -1.031072 1.228708 -0.07014251 0.8745149        0.029745697
## C   1.8396621 -1.891377 1.977937  0.11469195 1.2557692        0.005482127
## D   0.4281742 -2.824718 2.598717 -0.30909023 2.4963687        0.459483497
##    diastolic    systolic gripForce ‘\\‘sit and bend forward_cm\\‘‘
## B 0.05764353 -0.01869305 -0.920936                       -1.331405
## C 0.13202220 -0.07684313 -1.519526                       -2.285391
## D 0.25219705 -0.14232365 -2.101412                       -3.597352
##   ‘\\‘sit-ups counts\\‘‘ ‘\\‘broad jump_cm\\‘‘
## B              -1.423940            -0.6803174
## C              -2.603735            -1.1445756
## D              -4.161081            -1.2372507
```

```
##
## Residual Deviance: 23128.6
## AIC: 23200.6
```

```
confusionMatrix(mn.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.4  5.9  2.0  0.3
##          B  6.1 11.1  5.4  1.3
##          C  0.5  7.1 12.9  4.1
##          D  0.0  0.8  4.7 19.2
##
##  Accuracy (average) : 0.6173
```

```
confusionMatrix(mn.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.4  6.0  2.0  0.3
##          B  6.1 11.1  5.4  1.3
##          C  0.5  7.0 13.0  4.1
##          D  0.0  0.8  4.6 19.2
##
##  Accuracy (average) : 0.6177
```

```
mn.fit.cv$results
```

```
##   decay  Accuracy     Kappa  AccuracySD    KappaSD
## 1 0e+00 0.6172122 0.4896154 0.009252803 0.01233862
## 2 1e-04 0.6172122 0.4896154 0.009252803 0.01233862
## 3 1e-01 0.6173367 0.4897813 0.009319778 0.01242802
```

```
mn.fit.boot$results
```

```
##   decay  Accuracy     Kappa  AccuracySD     KappaSD
## 1 0e+00 0.6177344 0.4903212 0.005948138 0.007938624
## 2 1e-04 0.6177344 0.4903212 0.005948138 0.007938624
## 3 1e-01 0.6177418 0.4903313 0.005897619 0.007870582
```

LDA

```r
lda.fit.cv <- train(class ~ ., data = body, method = "lda",
                    trControl = fit.control.cv, trace = FALSE,
                    preProcess = c("center", "scale"))
lda.fit.boot <- train(class ~ ., data = body, method = "lda",
                      trControl = fit.control.boot, trace = FALSE,
                      preProcess = c("center", "scale"))
```

```r
lda.fit.cv$finalModel
```

```
## Call:
## lda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age    gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
##
## Coefficients of linear discriminants:
##                                   LD1         LD2         LD3
## age                       -0.64678307  0.22309394  0.47011181
## gender1                    0.69584600 -1.28409193  1.15159573
## height_cm                 -0.05207736 -0.93671269 -0.35916540
## weight_kg                  0.53802507  1.12001910  0.15508149
## `body fat_%`               0.16993470  0.16486574  0.50244600
## diastolic                  0.06595464  0.07418465 -0.27129251
## systolic                  -0.04539913 -0.07760683  0.30887408
## gripForce                 -0.48748656  0.34068971 -0.19140457
## `sit and bend forward_cm` -0.74403894 -0.28310233  0.43563764
## `sit-ups counts`          -1.04324820  0.33643160  0.13732175
## `broad jump_cm`           -0.26279633  0.68788193  0.04374994
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.9785 0.0195 0.0019
```

```
lda.fit.boot$finalModel
```

```
## Call:
## lda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age     gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
##
## Coefficients of linear discriminants:
##                                   LD1         LD2         LD3
## age                       -0.64678307  0.22309394  0.47011181
## gender1                    0.69584600 -1.28409193  1.15159573
## height_cm                 -0.05207736 -0.93671269 -0.35916540
## weight_kg                  0.53802507  1.12001910  0.15508149
## `body fat_%`               0.16993470  0.16486574  0.50244600
## diastolic                  0.06595464  0.07418465 -0.27129251
## systolic                  -0.04539913 -0.07760683  0.30887408
## gripForce                 -0.48748656  0.34068971 -0.19140457
## `sit and bend forward_cm` -0.74403894 -0.28310233  0.43563764
## `sit-ups counts`          -1.04324820  0.33643160  0.13732175
## `broad jump_cm`           -0.26279633  0.68788193  0.04374994
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.9785 0.0195 0.0019
```

```
confusionMatrix(lda.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.2  6.1  2.1  0.3
```

```
##          B  6.2 11.0  5.4  1.4
##          C  0.6  7.4 14.1  5.0
##          D  0.0  0.5  3.4 18.3
##
##  Accuracy (average) : 0.6156
```

```
confusionMatrix(lda.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.2  6.2  2.1  0.3
##          B  6.4 10.9  5.5  1.6
##          C  0.6  7.3 13.9  4.9
##          D  0.0  0.5  3.4 18.3
##
##  Accuracy (average) : 0.6126
```

```
lda.fit.cv$results
```

```
##   parameter Accuracy     Kappa AccuracySD   KappaSD
## 1      none 0.615571 0.4874285 0.01318156 0.0175775
```

```
lda.fit.boot$results
```

```
##   parameter  Accuracy     Kappa  AccuracySD     KappaSD
## 1      none 0.6125766 0.4834807 0.004671325 0.006265792
```

QDA

```
qda.fit.cv <- train(class ~ ., data = body, method = "qda",
                    trControl = fit.control.cv, trace = FALSE,
                    preProcess = c("center", "scale"))
qda.fit.boot <- train(class ~ ., data = body, method = "qda",
                      trControl = fit.control.boot, trace = FALSE,
                      preProcess = c("center", "scale"))
```

```
qda.fit.cv$finalModel
```

```
## Call:
## qda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
```

```
## Group means:
##           age     gender1    height_cm   weight_kg ‘body fat_%‘   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce ‘sit and bend forward_cm‘ ‘sit-ups counts‘
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   ‘broad jump_cm‘
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
```

```
qda.fit.boot$finalModel
```

```
## Call:
## qda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##           age     gender1    height_cm   weight_kg ‘body fat_%‘   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce ‘sit and bend forward_cm‘ ‘sit-ups counts‘
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   ‘broad jump_cm‘
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
```

```
confusionMatrix(qda.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 19.0  6.0  1.9  0.3
##          B  5.5 12.2  5.7  1.6
```

```
##          C  0.4  6.2 15.8  4.4
##          D  0.1  0.6  1.6 18.6
##
##  Accuracy (average) : 0.6572
```

```
confusionMatrix(qda.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.8  6.1  1.9  0.4
##          B  5.7 11.9  5.5  1.7
##          C  0.5  6.3 15.7  4.4
##          D  0.1  0.6  1.7 18.6
##
##  Accuracy (average) : 0.6506
```

```
qda.fit.cv$results
```

```
##   parameter  Accuracy     Kappa AccuracySD    KappaSD
## 1      none 0.6571846 0.5429133 0.01127552 0.01503324
```

```
qda.fit.boot$results
```

```
##   parameter  Accuracy    Kappa AccuracySD    KappaSD
## 1      none 0.6505498 0.534103 0.01300762 0.01732938
```

kNN

```
knn.fit.cv <- train(class ~ ., data = body, method = "knn",
                    trControl = fit.control.cv, preProcess = c("center", "scale"))
knn.fit.boot <- train(class ~ ., data = body, method = "knn",
                      trControl = fit.control.boot, preProcess = c("center", "scale"))
```

```
knn.fit.cv$finalModel
```

```
## 9-nearest neighbor model
## Training set outcome distribution:
##
##    A    B    C    D
## 3348 3347 3349 3349
```

```
knn.fit.boot$finalModel
```

```
## 9-nearest neighbor model
## Training set outcome distribution:
##
##    A    B    C    D
## 3348 3347 3349 3349
```

```
confusionMatrix(knn.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 19.9  8.2  2.8  0.6
##          B  4.5 11.7  7.4  2.0
##          C  0.6  4.4 13.6  5.7
##          D  0.1  0.6  1.2 16.6
##
##  Accuracy (average) : 0.6179
```

```
confusionMatrix(knn.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.5  8.2  2.9  0.7
##          B  5.1 11.0  7.6  2.2
##          C  1.1  5.0 12.5  5.9
##          D  0.2  0.9  1.9 16.4
##
##  Accuracy (average) : 0.584
```

```
knn.fit.cv$results
```

```
##   k  Accuracy     Kappa AccuracySD    KappaSD
## 1 5 0.6045941 0.4727952 0.01216431 0.01622369
## 2 7 0.6142261 0.4856382 0.01132453 0.01510491
## 3 9 0.6178839 0.4905149 0.01147146 0.01529822
```

```
knn.fit.boot$results
```

```
##   k  Accuracy     Kappa  AccuracySD     KappaSD
## 1 5 0.5599129 0.4132660 0.006783944 0.009123717
## 2 7 0.5754512 0.4339888 0.006548934 0.008788444
## 3 9 0.5840070 0.4454149 0.006918906 0.009270053
```

k-Means

Based on your fits, identify whether a linear or non-linear model is more appropriate. Make sure to discuss your results (including plots and tables), and to use CV and/or bootstrap to evaluate your models' performance.

Regularization

LASSO

Ridge

Elastic Net

PCA

Based on your fits, identify the best model taking into consideration the bias-variance tradeoff. Make sure to discuss your results (including plots and tables), and to use CV and/or bootstrap to evaluate your models' performance.

Sources

https://remiller1450.github.io/s230f19/caret3.html https://dataaspirant.com/knn-implementation-r-using-caret-package/ https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train