MLHW5\_JF

February 21, 2022

# QI: You should create and compare three different models. Remember to remove the ID variable as you do not want to include that in your analysis.

set.seed(100)  
  
alc = read.csv("/Users/judyfordjuoh/Desktop/Machine Learning/ML\_hw5/alcohol\_use.csv")  
  
#Strip off ID Variable  
alc <- alc[,2:9]  
  
#Make alcohol use into a alcohol consumption (0 for current use and 1 for current use) #From OH  
alc$alc\_consumption <- factor(alc$alc\_consumption, levels = c("NotCurrentUse", "CurrentUse"))  
  
#Check distributions, missing data etc.#From OH  
summary(alc)

## neurotocism\_score extroversion\_score openness\_score   
## Min. :-3.464360 Min. :-3.273930 Min. :-3.273930   
## 1st Qu.:-0.678250 1st Qu.:-0.695090 1st Qu.:-0.717270   
## Median : 0.042570 Median : 0.003320 Median :-0.019280   
## Mean : 0.000047 Mean :-0.000163 Mean :-0.000534   
## 3rd Qu.: 0.629670 3rd Qu.: 0.637790 3rd Qu.: 0.723300   
## Max. : 3.273930 Max. : 3.273930 Max. : 2.901610   
## agreeableness\_score conscientiousness\_score impulsiveness\_score  
## Min. :-3.464360 Min. :-3.464360 Min. :-2.555240   
## 1st Qu.:-0.606330 1st Qu.:-0.652530 1st Qu.:-0.711260   
## Median :-0.017290 Median :-0.006650 Median :-0.217120   
## Mean :-0.000245 Mean :-0.000386 Mean : 0.007216   
## 3rd Qu.: 0.760960 3rd Qu.: 0.584890 3rd Qu.: 0.529750   
## Max. : 3.464360 Max. : 3.464360 Max. : 2.901610   
## sens\_seeking\_score alc\_consumption  
## Min. :-2.078480 NotCurrentUse: 881   
## 1st Qu.:-0.525930 CurrentUse :1004   
## Median : 0.079870   
## Mean :-0.003292   
## 3rd Qu.: 0.765400   
## Max. : 1.921730

#Omit those with missing data #From OH  
alc <- na.omit(alc)  
  
#tidyverse way to create data partition (70/30)  
#training.data<-chr$life\_exp %>% createDataPartition(p=0.7, list=F)  
train.indices <- createDataPartition(y = alc$alc\_consumption,p = 0.7,list = FALSE)  
train.data <- alc[train.indices, ]  
test.data <- alc[-train.indices, ]

#REGULARIZED REGRESSION: ELASTIC NET  
set.seed(123)  
  
en.model <- train(  
 alc\_consumption ~., data = train.data, method = "glmnet",  
 trControl = trainControl("cv", number = 10), preProc = c("center", "scale"), tuneLength = 10  
 )  
#Print the values of alpha and lambda that gave best prediction  
en.model$bestTune %>% knitr::kable() # 0.6(alpha)| 0.2641193(lambda)| 0.8545881|(accuracy)

|  |  |  |
| --- | --- | --- |
|  | alpha | lambda |
| 54 | 0.6 | 0.2641193 |

#Print all of the options examined. Bc this is a logistic regression we are using the Accuracy. If it was linear regression it would be MSE/RMSE.  
en.model$results %>% knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| alpha | lambda | Accuracy | Kappa | AccuracySD | KappaSD |
| 0.1 | 0.0003256 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.1 | 0.0007522 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.1 | 0.0017377 | 0.8121337 | 0.6226894 | 0.0383508 | 0.0761426 |
| 0.1 | 0.0040144 | 0.8128971 | 0.6240812 | 0.0361381 | 0.0718582 |
| 0.1 | 0.0092737 | 0.8136489 | 0.6253079 | 0.0316963 | 0.0630953 |
| 0.1 | 0.0214235 | 0.8113762 | 0.6204617 | 0.0311078 | 0.0619218 |
| 0.1 | 0.0494911 | 0.8068308 | 0.6110650 | 0.0289489 | 0.0580292 |
| 0.1 | 0.1143309 | 0.7977568 | 0.5925978 | 0.0365522 | 0.0729450 |
| 0.1 | 0.2641193 | 0.7977625 | 0.5917474 | 0.0397048 | 0.0793364 |
| 0.2 | 0.0003256 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.2 | 0.0007522 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.2 | 0.0017377 | 0.8113818 | 0.6212322 | 0.0389140 | 0.0771914 |
| 0.2 | 0.0040144 | 0.8121453 | 0.6225468 | 0.0376246 | 0.0747286 |
| 0.2 | 0.0092737 | 0.8128971 | 0.6238930 | 0.0330158 | 0.0657870 |
| 0.2 | 0.0214235 | 0.8113648 | 0.6206302 | 0.0290875 | 0.0579446 |
| 0.2 | 0.0494911 | 0.8030372 | 0.6034791 | 0.0300228 | 0.0601443 |
| 0.2 | 0.1143309 | 0.7947435 | 0.5865095 | 0.0381881 | 0.0763618 |
| 0.2 | 0.2641193 | 0.8075711 | 0.6109129 | 0.0335677 | 0.0669260 |
| 0.3 | 0.0003256 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.3 | 0.0007522 | 0.8128970 | 0.6242900 | 0.0376811 | 0.0745977 |
| 0.3 | 0.0017377 | 0.8121394 | 0.6228235 | 0.0392968 | 0.0777223 |
| 0.3 | 0.0040144 | 0.8121453 | 0.6226453 | 0.0400864 | 0.0795349 |
| 0.3 | 0.0092737 | 0.8121396 | 0.6224212 | 0.0346651 | 0.0689879 |
| 0.3 | 0.0214235 | 0.8098381 | 0.6176467 | 0.0304741 | 0.0607125 |
| 0.3 | 0.0494911 | 0.7992490 | 0.5961889 | 0.0302826 | 0.0602192 |
| 0.3 | 0.1143309 | 0.8022908 | 0.6015708 | 0.0336270 | 0.0672568 |
| 0.3 | 0.2641193 | 0.8257592 | 0.6465454 | 0.0317192 | 0.0642225 |
| 0.4 | 0.0003256 | 0.8121394 | 0.6228235 | 0.0392968 | 0.0777223 |
| 0.4 | 0.0007522 | 0.8121394 | 0.6228235 | 0.0392968 | 0.0777223 |
| 0.4 | 0.0017377 | 0.8121394 | 0.6228235 | 0.0392968 | 0.0777223 |
| 0.4 | 0.0040144 | 0.8121453 | 0.6226068 | 0.0376246 | 0.0747777 |
| 0.4 | 0.0092737 | 0.8128914 | 0.6239638 | 0.0341973 | 0.0680904 |
| 0.4 | 0.0214235 | 0.8075768 | 0.6132475 | 0.0317688 | 0.0630210 |
| 0.4 | 0.0494911 | 0.7992433 | 0.5962907 | 0.0315590 | 0.0626365 |
| 0.4 | 0.1143309 | 0.8038232 | 0.6046356 | 0.0342528 | 0.0682093 |
| 0.4 | 0.2641193 | 0.8265283 | 0.6477486 | 0.0287950 | 0.0581172 |
| 0.5 | 0.0003256 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.5 | 0.0007522 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.5 | 0.0017377 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.5 | 0.0040144 | 0.8113934 | 0.6211546 | 0.0363160 | 0.0719792 |
| 0.5 | 0.0092737 | 0.8128914 | 0.6239638 | 0.0341973 | 0.0680904 |
| 0.5 | 0.0214235 | 0.8045579 | 0.6072490 | 0.0318067 | 0.0630905 |
| 0.5 | 0.0494911 | 0.8007758 | 0.5992832 | 0.0313997 | 0.0623267 |
| 0.5 | 0.1143309 | 0.8038232 | 0.6049099 | 0.0349732 | 0.0693433 |
| 0.5 | 0.2641193 | 0.8447108 | 0.6826592 | 0.0271640 | 0.0556344 |
| 0.6 | 0.0003256 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.6 | 0.0007522 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.6 | 0.0017377 | 0.8113818 | 0.6212646 | 0.0389140 | 0.0769527 |
| 0.6 | 0.0040144 | 0.8113934 | 0.6211546 | 0.0363160 | 0.0719792 |
| 0.6 | 0.0092737 | 0.8113762 | 0.6209471 | 0.0335065 | 0.0665158 |
| 0.6 | 0.0214235 | 0.8038118 | 0.6058786 | 0.0324502 | 0.0642298 |
| 0.6 | 0.0494911 | 0.8015562 | 0.6012607 | 0.0337181 | 0.0666295 |
| 0.6 | 0.1143309 | 0.8000695 | 0.5980700 | 0.0376543 | 0.0745373 |
| 0.6 | 0.2641193 | 0.8545881 | 0.7022163 | 0.0251889 | 0.0521354 |
| 0.7 | 0.0003256 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.7 | 0.0007522 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.7 | 0.0017377 | 0.8106242 | 0.6197086 | 0.0383449 | 0.0757870 |
| 0.7 | 0.0040144 | 0.8113877 | 0.6211354 | 0.0356496 | 0.0707012 |
| 0.7 | 0.0092737 | 0.8113877 | 0.6210390 | 0.0332430 | 0.0659816 |
| 0.7 | 0.0214235 | 0.8030600 | 0.6044280 | 0.0347819 | 0.0689000 |
| 0.7 | 0.0494911 | 0.7977910 | 0.5942562 | 0.0381801 | 0.0749248 |
| 0.7 | 0.1143309 | 0.7871672 | 0.5736683 | 0.0416713 | 0.0821020 |
| 0.7 | 0.2641193 | 0.8545881 | 0.7022163 | 0.0251889 | 0.0521354 |
| 0.8 | 0.0003256 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.8 | 0.0007522 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.8 | 0.0017377 | 0.8106242 | 0.6197086 | 0.0383449 | 0.0757870 |
| 0.8 | 0.0040144 | 0.8098724 | 0.6181079 | 0.0366935 | 0.0727552 |
| 0.8 | 0.0092737 | 0.8091207 | 0.6164379 | 0.0338365 | 0.0672677 |
| 0.8 | 0.0214235 | 0.7992892 | 0.5969406 | 0.0339631 | 0.0669907 |
| 0.8 | 0.0494911 | 0.7932338 | 0.5858872 | 0.0403490 | 0.0791514 |
| 0.8 | 0.1143309 | 0.7757978 | 0.5525061 | 0.0353881 | 0.0688779 |
| 0.8 | 0.2641193 | 0.8545881 | 0.7022163 | 0.0251889 | 0.0521354 |
| 0.9 | 0.0003256 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.9 | 0.0007522 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 0.9 | 0.0017377 | 0.8091091 | 0.6167123 | 0.0366102 | 0.0723516 |
| 0.9 | 0.0040144 | 0.8098724 | 0.6181079 | 0.0366935 | 0.0727552 |
| 0.9 | 0.0092737 | 0.8060960 | 0.6105643 | 0.0371064 | 0.0735486 |
| 0.9 | 0.0214235 | 0.8000352 | 0.5985007 | 0.0340150 | 0.0669999 |
| 0.9 | 0.0494911 | 0.7811123 | 0.5625389 | 0.0404021 | 0.0789974 |
| 0.9 | 0.1143309 | 0.7773129 | 0.5556870 | 0.0352056 | 0.0685651 |
| 0.9 | 0.2641193 | 0.8545881 | 0.7022163 | 0.0251889 | 0.0521354 |
| 1.0 | 0.0003256 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 1.0 | 0.0007522 | 0.8106242 | 0.6197732 | 0.0380108 | 0.0751364 |
| 1.0 | 0.0017377 | 0.8091091 | 0.6167123 | 0.0366102 | 0.0723516 |
| 1.0 | 0.0040144 | 0.8098782 | 0.6180338 | 0.0364925 | 0.0723843 |
| 1.0 | 0.0092737 | 0.8060960 | 0.6105643 | 0.0371064 | 0.0735486 |
| 1.0 | 0.0214235 | 0.7977624 | 0.5940871 | 0.0365203 | 0.0718927 |
| 1.0 | 0.0494911 | 0.7765553 | 0.5540731 | 0.0361096 | 0.0702951 |
| 1.0 | 0.1143309 | 0.7773129 | 0.5556870 | 0.0352056 | 0.0685651 |
| 1.0 | 0.2641193 | 0.7152834 | 0.4054590 | 0.0508691 | 0.1083319 |

# Model coefficients  
coef(en.model$finalModel, en.model$bestTune$lambda)

## 8 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 0.1364653  
## neurotocism\_score .   
## extroversion\_score .   
## openness\_score .   
## agreeableness\_score .   
## conscientiousness\_score .   
## impulsiveness\_score 0.4252278  
## sens\_seeking\_score .

#Confusion Matrix  
confusionMatrix(en.model)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 32.2 0.0  
## CurrentUse 14.5 53.3  
##   
## Accuracy (average) : 0.8545

#LOGISTIC REGRESSION  
  
logistic\_control1 <- trainControl(method = "cv", number = 3, savePredictions = T)  
  
set.seed(1000)  
logistic <- train(alc\_consumption ~ ., data = train.data, method = "glm", family = "binomial", trControl = logistic\_control1)  
  
summary(logistic)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.4828 -0.6646 0.2114 0.7660 1.7753   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.26409 0.07282 3.627 0.000287 \*\*\*  
## neurotocism\_score 0.20172 0.08758 2.303 0.021261 \*   
## extroversion\_score 0.39653 0.09307 4.261 2.04e-05 \*\*\*  
## openness\_score 0.01032 0.08327 0.124 0.901417   
## agreeableness\_score 0.07913 0.07816 1.012 0.311306   
## conscientiousness\_score -0.01821 0.08670 -0.210 0.833627   
## impulsiveness\_score 1.93509 0.13105 14.767 < 2e-16 \*\*\*  
## sens\_seeking\_score 0.11863 0.10336 1.148 0.251075   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1824.3 on 1319 degrees of freedom  
## Residual deviance: 1184.6 on 1312 degrees of freedom  
## AIC: 1200.6  
##   
## Number of Fisher Scoring iterations: 5

confusionMatrix(logistic)

## Cross-Validated (3 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 37.1 8.9  
## CurrentUse 9.6 44.4  
##   
## Accuracy (average) : 0.8152

confusionMatrix(table((logistic$pred)$pred,(logistic$pred)$obs))

## Confusion Matrix and Statistics  
##   
##   
## NotCurrentUse CurrentUse  
## NotCurrentUse 490 117  
## CurrentUse 127 586  
##   
## Accuracy : 0.8152   
## 95% CI : (0.7931, 0.8358)  
## No Information Rate : 0.5326   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.6284   
##   
## Mcnemar's Test P-Value : 0.5645   
##   
## Sensitivity : 0.7942   
## Specificity : 0.8336   
## Pos Pred Value : 0.8072   
## Neg Pred Value : 0.8219   
## Prevalence : 0.4674   
## Detection Rate : 0.3712   
## Detection Prevalence : 0.4598   
## Balanced Accuracy : 0.8139   
##   
## 'Positive' Class : NotCurrentUse   
##

#LASSO  
  
#NTS: first create a grid to search lambda  
lambda <- 10^seq(-3,3, length = 100)  
  
set.seed(100)  
  
#NTS: replace tuneLength with tuneGrid and alpha is 1 because we are doing lasso. If we were doing rigid it would be 0.   
lasso\_m <- train(  
 alc\_consumption ~., data = train.data, method = "glmnet", trControl = trainControl("cv", number = 10), preProc = c("center", "scale"), tuneGrid = expand.grid(alpha = 1, lambda = lambda)  
)  
  
#Print the values of alpha and lambda that gave best prediction  
lasso\_m$bestTune %>% knitr::kable() # 1(alpha)|0.23101(lambda)|0.8538 (Accuracy)

|  |  |  |
| --- | --- | --- |
|  | alpha | lambda |
| 40 | 1 | 0.231013 |

#Print all of the options examined  
lasso\_m$results %>% knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| alpha | lambda | Accuracy | Kappa | AccuracySD | KappaSD |
| 1 | 0.0010000 | 0.8114169 | 0.6209944 | 0.0379351 | 0.0768656 |
| 1 | 0.0011498 | 0.8106594 | 0.6195058 | 0.0373518 | 0.0757112 |
| 1 | 0.0013219 | 0.8106594 | 0.6195058 | 0.0373518 | 0.0757112 |
| 1 | 0.0015199 | 0.8106536 | 0.6195596 | 0.0378891 | 0.0768169 |
| 1 | 0.0017475 | 0.8091269 | 0.6164206 | 0.0368831 | 0.0747590 |
| 1 | 0.0020092 | 0.8091269 | 0.6164206 | 0.0368831 | 0.0747590 |
| 1 | 0.0023101 | 0.8083750 | 0.6148647 | 0.0367233 | 0.0744688 |
| 1 | 0.0026561 | 0.8083750 | 0.6148647 | 0.0367233 | 0.0744688 |
| 1 | 0.0030539 | 0.8091326 | 0.6164321 | 0.0359768 | 0.0729835 |
| 1 | 0.0035112 | 0.8106477 | 0.6194080 | 0.0338185 | 0.0686510 |
| 1 | 0.0040370 | 0.8114053 | 0.6209021 | 0.0348300 | 0.0706336 |
| 1 | 0.0046416 | 0.8121687 | 0.6224115 | 0.0357756 | 0.0724573 |
| 1 | 0.0053367 | 0.8106419 | 0.6193311 | 0.0349681 | 0.0708864 |
| 1 | 0.0061359 | 0.8113938 | 0.6207914 | 0.0345406 | 0.0701231 |
| 1 | 0.0070548 | 0.8098787 | 0.6178122 | 0.0350915 | 0.0712491 |
| 1 | 0.0081113 | 0.8091211 | 0.6162736 | 0.0334835 | 0.0678177 |
| 1 | 0.0093260 | 0.8091211 | 0.6162736 | 0.0334835 | 0.0678177 |
| 1 | 0.0107227 | 0.8076174 | 0.6132552 | 0.0317272 | 0.0643281 |
| 1 | 0.0123285 | 0.8091327 | 0.6162884 | 0.0316494 | 0.0642530 |
| 1 | 0.0141747 | 0.8053619 | 0.6090790 | 0.0350203 | 0.0703755 |
| 1 | 0.0162975 | 0.8038467 | 0.6060317 | 0.0370407 | 0.0744578 |
| 1 | 0.0187382 | 0.8015740 | 0.6015675 | 0.0387854 | 0.0780415 |
| 1 | 0.0215443 | 0.7985379 | 0.5956605 | 0.0415791 | 0.0834783 |
| 1 | 0.0247708 | 0.7985379 | 0.5956605 | 0.0415791 | 0.0834783 |
| 1 | 0.0284804 | 0.7954959 | 0.5899952 | 0.0411056 | 0.0822470 |
| 1 | 0.0327455 | 0.7894466 | 0.5784752 | 0.0469388 | 0.0937202 |
| 1 | 0.0376494 | 0.7826167 | 0.5652049 | 0.0422930 | 0.0846511 |
| 1 | 0.0432876 | 0.7818591 | 0.5642021 | 0.0448733 | 0.0895764 |
| 1 | 0.0497702 | 0.7788346 | 0.5584476 | 0.0408501 | 0.0818617 |
| 1 | 0.0572237 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.0657933 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.0756463 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.0869749 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.1000000 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.1149757 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.1321941 | 0.7773252 | 0.5555093 | 0.0404140 | 0.0810523 |
| 1 | 0.1519911 | 0.8311656 | 0.6575425 | 0.0522593 | 0.1016554 |
| 1 | 0.1747528 | 0.8545536 | 0.7020098 | 0.0284574 | 0.0593010 |
| 1 | 0.2009233 | 0.8545536 | 0.7020098 | 0.0284574 | 0.0593010 |
| 1 | 0.2310130 | 0.8545536 | 0.7020098 | 0.0284574 | 0.0593010 |
| 1 | 0.2656088 | 0.7144219 | 0.4036276 | 0.0428567 | 0.0920662 |
| 1 | 0.3053856 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.3511192 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.4037017 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.4641589 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.5336699 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.6135907 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.7054802 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.8111308 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 0.9326033 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1.0722672 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1.2328467 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1.4174742 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1.6297508 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1.8738174 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 2.1544347 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 2.4770764 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 2.8480359 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 3.2745492 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 3.7649358 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 4.3287613 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 4.9770236 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 5.7223677 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 6.5793322 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 7.5646333 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 8.6974900 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 10.0000000 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 11.4975700 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 13.2194115 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 15.1991108 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 17.4752840 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 20.0923300 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 23.1012970 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 26.5608778 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 30.5385551 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 35.1119173 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 40.3701726 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 46.4158883 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 53.3669923 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 61.3590727 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 70.5480231 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 81.1130831 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 93.2603347 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 107.2267222 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 123.2846739 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 141.7474163 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 162.9750835 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 187.3817423 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 215.4434690 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 247.7076356 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 284.8035868 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 327.4549163 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 376.4935807 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 432.8761281 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 497.7023564 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 572.2367659 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 657.9332247 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 756.4633276 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 869.7490026 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |
| 1 | 1000.0000000 | 0.5325765 | 0.0000000 | 0.0026546 | 0.0000000 |

# Model coefficients  
coef(lasso\_m$finalModel, lasso\_m$bestTune$lambda)

## 8 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 0.1335513  
## neurotocism\_score .   
## extroversion\_score .   
## openness\_score .   
## agreeableness\_score .   
## conscientiousness\_score .   
## impulsiveness\_score 0.3038980  
## sens\_seeking\_score .

#Confusion Matrix  
confusionMatrix(lasso\_m)

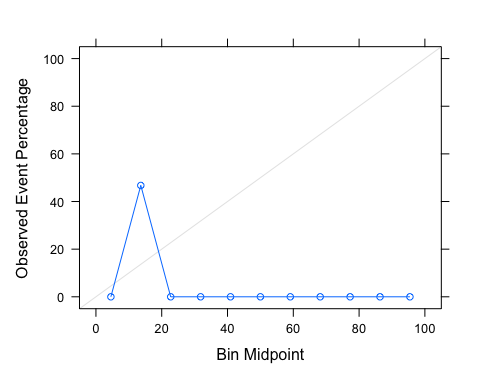
## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 32.2 0.0  
## CurrentUse 14.5 53.3  
##   
## Accuracy (average) : 0.8545

# Q2: Decide which model you would choose as your final model

#Q2 Task: You should tune and compare the performance of all three models within the training set using cross-validation and then decide which model you would choose as your final model. Provide justification for your choice.  
  
#Make a calibration plot to see which one is the best  
  
  
#EN Calibration  
fitted.results\_modelen <- en.model %>% predict(train.data)  
  
error\_model\_4 <- mean(fitted.results\_modelen !=train.data$alc\_consumption, na.rm = T)  
  
print(paste('Accuracy [en.model]', 1-error\_model\_4))

## [1] "Accuracy [en.model] 0.854545454545454"

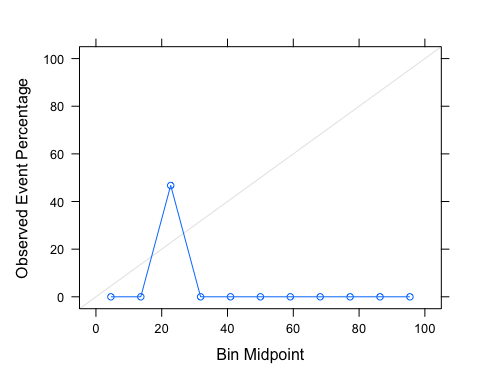
testProb <- data.frame(obs = train.data$alc\_consumption,  
 pred.logit = error\_model\_4)  
  
calPlotData\_model4<- calibration(obs~as.numeric(pred.logit), data = testProb)  
  
xyplot(calPlotData\_model4, auto.key = list(columns = 2))



#Logistic regression Calibration  
fitted.results\_model2 <- logistic %>% predict(test.data)  
  
error\_model2 <- mean(fitted.results\_model2 !=test.data$alc\_consumption, na.rm = T)  
  
print(paste('Accuracy [logistic]', 1-error\_model2))

## [1] "Accuracy [logistic] 0.79646017699115"

testProb2 <- data.frame(obs = test.data$alc\_consumption,  
 pred.logit2 = error\_model2)  
  
calPlotData\_model2<- calibration(obs~as.numeric(pred.logit2), data = testProb2)  
  
xyplot(calPlotData\_model2, auto.key = list(columns = 2))



#LASSO Calibration  
fitted.results\_model5 <- lasso\_m %>% predict(train.data)  
  
error\_model5 <- mean(fitted.results\_model5 !=test.data$alc\_consumption, na.rm = T)

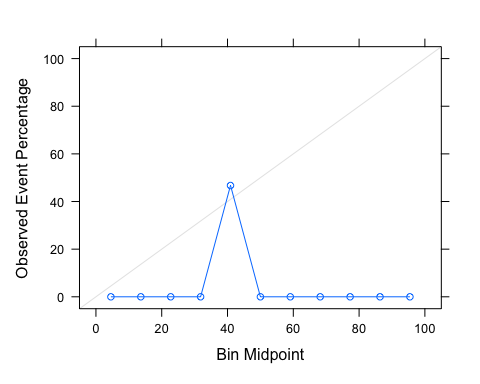
## Warning in `!=.default`(fitted.results\_model5, test.data$alc\_consumption):  
## longer object length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of  
## shorter object length

print(paste('Accuracy [lasso\_m]', 1-error\_model5))

## [1] "Accuracy [lasso\_m] 0.559848484848485"

testProb5 <- data.frame(obs = test.data$alc\_consumption,  
 pred.logit5 = error\_model5)  
  
calPlotData\_model5<- calibration(obs~as.numeric(pred.logit5), data = testProb5)  
  
xyplot(calPlotData\_model5, auto.key = list(columns = 2))



In the elastic net model, the average accuracy was 0.8545. The intercept was 0.1365 and the remaining variables went to zero, except for Measure of Impulsivity (impulsiveness\_score = 0.4253). The best predicting alpha and lambda was 0.6(alpha) and 0.2641(lambda), which resulted in an accuracy of 0.8545.

In the logistic regression model the average accuracy was 0.8152 and the sensitivity and specificity was 0.7942 and 0.8336, respectively.

In the LASSO model, the average accuracy was 0.8538. The intercept was 0.1336 and the remaining variables went to zero, except for Measure of Impulsivity (impulsiveness\_score = 0.3038).The best predicting alpha and lambda was 1(alpha) and 0.2310(lambda), which resulted in an accuracy of 0.8545.

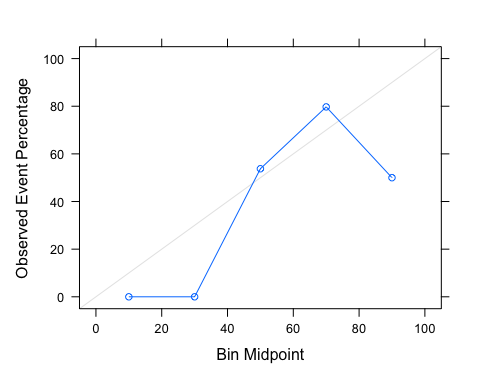
Both the elastic net model and the LASSO model had the same average accuracy, which was higher than the logistic regression, as well as the same calibration plot. Although it may seem like I can use either the elastic net or the lasso model for my final model, I am choosing the elastic net model because elastic Net combines characteristics of both lasso and ridge. Elastic Net reduces the impact of different features while not eliminating all of the features. The elastic net model also had a larger beta for impulsiveness\_score compared to the LASSO model which can mean there is a relationship between this feature and the outcome which is worth further exploring. Also the coefficients in the elastic net model was slightly larger than the coefficients in the LASSO model.

# Q3: Apply your final model in the test set and report your final evaluation metrics

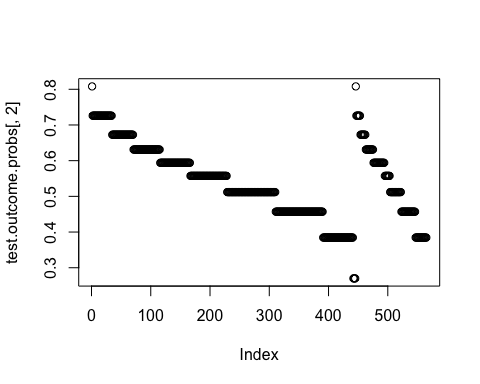
#Using the test data to make predictions  
  
en\_pred2 <- en.model %>% predict(test.data)  
confusionMatrix(en\_pred2,test.data$alc\_consumption, positive = "CurrentUse")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NotCurrentUse CurrentUse  
## NotCurrentUse 178 0  
## CurrentUse 86 301  
##   
## Accuracy : 0.8478   
## 95% CI : (0.8155, 0.8764)  
## No Information Rate : 0.5327   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.688   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.6742   
## Pos Pred Value : 0.7778   
## Neg Pred Value : 1.0000   
## Prevalence : 0.5327   
## Detection Rate : 0.5327   
## Detection Prevalence : 0.6850   
## Balanced Accuracy : 0.8371   
##   
## 'Positive' Class : CurrentUse   
##

#Obtain predicted probabilities  
test.outcome.probs<-predict(en.model, test.data, type="prob")  
  
testProbs.rmodel <- data.frame(obs = test.data$alc\_consumption,  
 pred.en=test.outcome.probs[,2])  
  
#Create calibration plot  
alc\_PlotData.rmodel<-calibration(obs ~ pred.en, data = testProbs.rmodel, class="CurrentUse", cuts=5)  
  
xyplot(alc\_PlotData.rmodel, auto.key = list(columns = 2))



plot(test.outcome.probs[,2])

 After applying my final model in the test set, the average accuracy was 0.8478 and the sensitivity and specificity was 1 and 0.6742, respectively. The positive predictive value and the negative predictive value were 0.778 and 1, respectively. Based on the calibration plot, we see that there are points where the two lines are very close, showing that the model has a good fit.

# Q5

#Q5 Task: What research questions could this analysis either a) directly address or b) indirectly help to address by providing information that could be used in subsequent analyses? Limit this response to no more than 1 paragraph. Be sure to use complete sentences.

This research can be used for a plethora of research questions. This research can be used to directly address: does an individual’s measure of impulsiveness affect their current use of alcohol? This research can be used to further indirectly help to address reaserch concerning the relationship between impulsiveness and sensation-seeking behaviors amongst NYC young adults (18-25 years old) and the rate of drunk driving deaths caused by NYC young adults.