Data Science II Homework 3

Camille Okonkwo

Contents

```r
library(tidymodels)
library(caret)
library(earth)
library(pROC)
library(vip)
library(MASS)
set.seed(2)
```

Introduction

Background

We will develop a model to predict whether a given car gets high or low gas mileage based on the dataset `auto.csv`. The dataset contains 392 observations.

The response variable is `mpg_cat`, which indicates whether the miles per gallon of a car is high or low.

The predictors are:

- `cylinders`: Number of cylinders between 4 and 8
- `displacement`: Engine displacement (cu. inches)
- `horsepower`: Engine horsepower
- `weight`: Vehicle weight (lbs.)
- `acceleration`: Time to accelerate from 0 to 60 mph (sec.)
- `year`: Model year (modulo 100)
- `origin`: Origin of car (1. American, 2. European, 3. Japanese)

Split the dataset into two parts: training data (70%) and test data (30%)

```
auto = read_csv("data/auto.csv") |>
  drop_na() |>
  mutate(
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = forcats::fct_relevel(mpg_cat, c("low", "high"))
  )

set.seed(2)

# create a random split of 70% training and 30% test data
data_split <- initial_split(data = auto, prop = 0.7)

# partitioned datasets
training_data = training(data_split)
testing_data = testing(data_split)

head(training_data)
```

```
## # A tibble: 6 x 8
##   cylinders displacement horsepower weight acceleration  year origin mpg_cat
##       <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1         4           86         64   1875         16.4    81      1 high
## 2         6          225        100   3651         17.7    76      1 low
## 3         6          231        165   3445         13.4    78      1 low
## 4         5          131        103   2830         15.9    78      2 low
## 5         4           98         65   2380         20.7    81      1 high
## 6         4           97         75   2155         16.4    76      3 high
```

```
head(testing_data)
```

```
## # A tibble: 6 x 8
##   cylinders displacement horsepower weight acceleration  year origin mpg_cat
```

```
##       <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1         8          302        140   3449         10.5    70      1 low
## 2         8          390        190   3850          8.5    70      1 low
## 3         4          113         95   2372         15      70      3 high
## 4         6          200         85   2587         16      70      1 low
## 5         4           97         88   2130         14.5    70      3 high
## 6         4          107         90   2430         14.5    70      2 high
```

1a) Perform a logistic regression analysis using the training data. Are there redundant predictors in your model? If so, identify them. If none is present, please provide an explanation.

```
ctrl <- trainControl(method = "cv",
                     number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(2)

# logistic using glm
glm.fit <- glm(mpg_cat ~ .,
               data = training_data,
               family = binomial(link = "logit"))

coef(glm.fit)
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -2.054216e+01  2.717063e-02  9.626894e-04 -2.423505e-02 -4.363108e-03
##  acceleration          year        origin
##  1.453842e-01  4.225079e-01  2.459384e-01
```

```
# logistic using caret
set.seed(2)
model.glm <- train(x = training_data[1:7],
                   y = training_data$mpg_cat,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)

model.glm$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##  (Intercept)     cylinders  displacement    horsepower        weight
##   -2.054e+01     2.717e-02     9.627e-04    -2.424e-02    -4.363e-03
## acceleration          year        origin
##    1.454e-01     4.225e-01     2.459e-01
##
## Degrees of Freedom: 273 Total (i.e. Null);  266 Residual
## Null Deviance:       379.8
## Residual Deviance: 120.3     AIC: 136.3
```

```
#both models gave same coefficients

# penalized logistic model
glmnGrid.enet <- expand.grid(.alpha = seq(0, 1, length = 50),
                         .lambda = exp(seq(-5, 0, length = 50)))

set.seed(2)

model.glmn.enet <- train(x = training_data[1:7],
```

```
                    y = training_data$mpg_cat,
                    method = "glmnet",
                    tuneGrid = glmnGrid.enet,
                    metric = "ROC",
                    trControl = ctrl)

model.glmn.enet$bestTune
```

```
##          alpha      lambda
## 1452 0.5918367 0.007461796
```

```
coef(model.glmn.enet$finalModel)
```

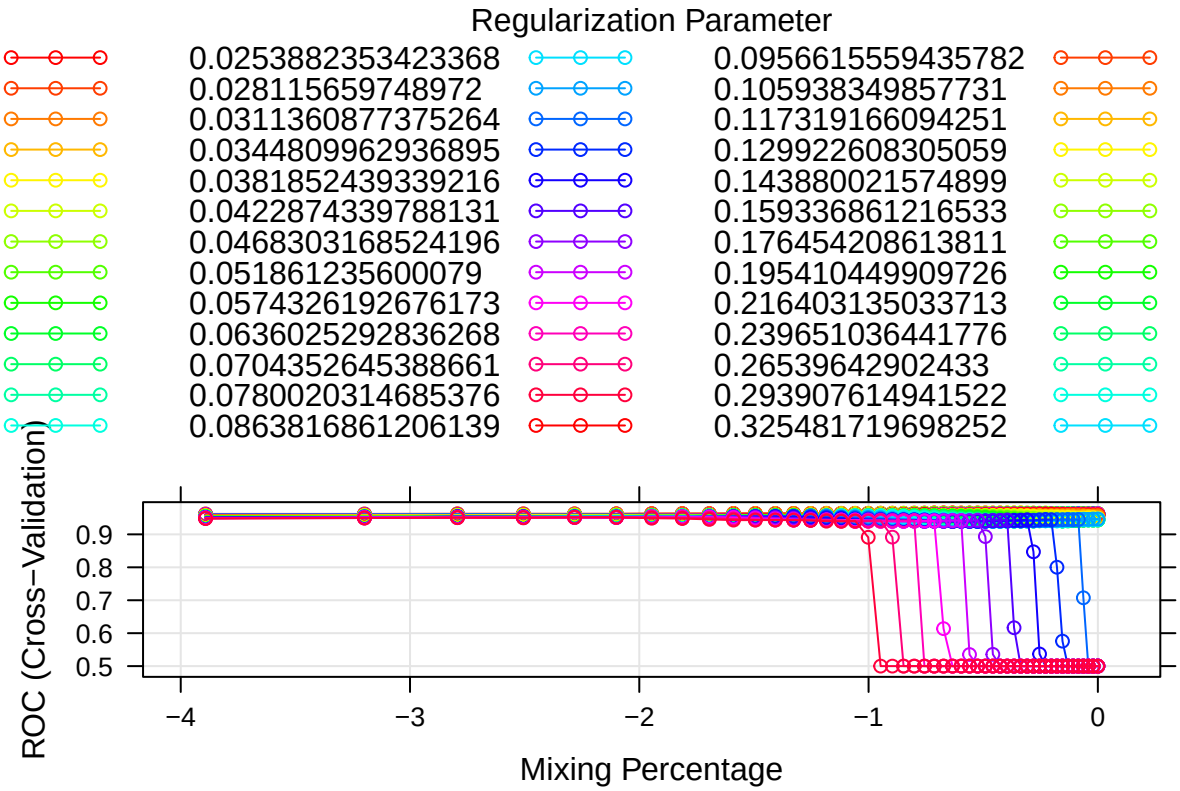
```
## 8 x 86 sparse Matrix of class "dgCMatrix"
##
## (Intercept)  0.0145988  2.772761e-01  5.628382e-01  0.8378696134  1.103030550
## cylinders    .         -8.365051e-03 -2.723286e-02 -0.0452641540 -0.062321570
## displacement .         -2.464543e-04 -5.450895e-04 -0.0008312087 -0.001106769
## horsepower   .          .             .             .             .
## weight       .         -5.648051e-05 -9.795455e-05 -0.0001383856 -0.000178174
## acceleration .          .             .             .             .
## year         .          .             .             .             .
## origin       .          .             .             .             .
##
## (Intercept)   1.360103731  1.6187757487  1.8713701454  2.117317251
## cylinders    -0.078766645 -0.0926414855 -0.1056029536 -0.117911514
## displacement -0.001372569 -0.0015928446 -0.0018010347 -0.002003707
## horsepower    .           -0.0003732899 -0.0007904975 -0.001198438
## weight       -0.000217242 -0.0002517148 -0.0002854069 -0.000319104
## acceleration  .            .             .            .
## year          .            .             .            .
## origin        .            .             .            .
##
## (Intercept)   2.3573997097  2.2089389623  1.7695483969  1.3260285084
## cylinders    -0.1295562265 -0.1399624098 -0.1493482788 -0.1581112706
## displacement -0.0022015493 -0.0023765184 -0.0025339521 -0.0026872642
## horsepower   -0.0016008411 -0.0019301720 -0.0022071486 -0.0024885490
## weight       -0.0003529223 -0.0003869075 -0.0004213629 -0.0004565996
## acceleration  .             .             .            .
## year          .             0.0048568031  0.0133535777  0.0218691380
## origin        .             .             .            .
##
## (Intercept)   0.8784376284  0.4269573269 -0.027868662 -0.4861912425
## cylinders    -0.1662400199 -0.1737238923 -0.180579052 -0.1867429830
## displacement -0.0028367689 -0.0029826934 -0.003125877 -0.0032651054
## horsepower   -0.0027780730 -0.0030792061 -0.003395443 -0.0037294094
## weight       -0.0004927178 -0.0005298086 -0.000567869 -0.0006071392
## acceleration  .             .             .            .
## year          0.0304118721  0.0389877932  0.047597935  0.0562502888
## origin        .             .             .            .
##
## (Intercept)  -0.9472912898 -1.4241226691 -1.9124537517 -2.3994461159
## cylinders    -0.1922289673 -0.1966251655 -0.2000243374 -0.2027884936
## displacement -0.0034011201 -0.0035217704 -0.0036285787 -0.0037329519
```

```
## horsepower   -0.0040841794 -0.0044698386 -0.0048891772 -0.0053381977
## weight       -0.0006476041 -0.0006884023 -0.0007298207 -0.0007726896
## acceleration  .             .             .             .
## year          0.0649422290  0.0736574945  0.0823977919  0.0911716277
## origin        .             0.0052614627  0.0142976663  0.0224347333
##
## (Intercept)  -2.8846937719 -3.3674989443 -3.8472363361 -4.3232767728
## cylinders    -0.2049145429 -0.2063955740 -0.2072268711 -0.2074053261
## displacement -0.0038340896 -0.0039322873 -0.0040274895 -0.0041196097
## horsepower   -0.0058176051 -0.0063286134 -0.0068717958 -0.0074473286
## weight       -0.0008170935 -0.0008630396 -0.0009105441 -0.0009596125
## acceleration  .             .             .             .
## year          0.0999750660  0.1088009651  0.1176417389  0.1264888176
## origin        0.0297382393  0.0362444246  0.0420044306  0.0470739929
##
## (Intercept)  -4.794915699 -5.261677060 -5.722875833 -6.177908513 -6.626185062
## cylinders    -0.206932575 -0.205807441 -0.204032713 -0.201613641 -0.198558230
## displacement -0.004208963 -0.004294556 -0.004376600 -0.004454871 -0.004529116
## horsepower   -0.008055276 -0.008694464 -0.009364210 -0.010063218 -0.010789901
## weight       -0.001010210 -0.001062374 -0.001116054 -0.001171212 -0.001227800
## acceleration  .             .             .             .             .
## year          0.135332414  0.144162935  0.152968997  0.161738966  0.170460710
## origin        0.051498845  0.055368545  0.058737003  0.061672139  0.064242543
##
## (Intercept)  -7.067131831 -7.500133889 -7.924781607 -8.340508273 -8.746836222
## cylinders    -0.194877470 -0.190604188 -0.185727838 -0.180282017 -0.174292730
## displacement -0.004599053 -0.004664848 -0.004725215 -0.004780281 -0.004829696
## horsepower   -0.011542418 -0.012318922 -0.013116737 -0.013933755 -0.014767543
## weight       -0.001285762 -0.001344981 -0.001405467 -0.001467094 -0.001529765
## acceleration  .             .             .             .             .
## year          0.179121741  0.187709713  0.196211354  0.204614132  0.212905559
## origin        0.066516472  0.068540875  0.070417867  0.072191001  0.073917534
##
## (Intercept)  -9.143319870 -9.529549298 -9.905153577 -10.343982343 -10.791681075
## cylinders    -0.167790482 -0.160810253 -0.153391489  -0.145639553  -0.136604586
## displacement -0.004873103 -0.004910148 -0.004940487  -0.004963679  -0.004973410
## horsepower   -0.015615606 -0.016475419 -0.017344442  -0.017965907  -0.018521802
## weight       -0.001593376 -0.001657814 -0.001722959  -0.001793643  -0.001869675
## acceleration  .             .             .              0.003497789   0.007826140
## year          0.221073496  0.229106280  0.236992831   0.244821719   0.252586172
## origin        0.075649925  0.077435368  0.079315417   0.081423338   0.083906449
##
## (Intercept)  -11.224987973 -11.647620590 -12.059339201 -12.459937610
## cylinders     -0.127729041  -0.118626101  -0.109357233  -0.099984088
## displacement  -0.004977516  -0.004973218  -0.004960057  -0.004937672
## horsepower    -0.019069713  -0.019599751  -0.020109886  -0.020598356
## weight        -0.001945008  -0.002021145  -0.002097938  -0.002175226
## acceleration   0.012000959   0.016177898   0.020355247   0.024530679
## year           0.260158685   0.267564237   0.274794959   0.281843820
## origin         0.086416068   0.089071100   0.091882650   0.094856610
##
## (Intercept)  -12.848104494 -13.225962515 -13.592344541 -13.947164044
## cylinders     -0.090326539  -0.080879566  -0.071521074  -0.062312401
## displacement  -0.004909340  -0.004868845  -0.004818338  -0.004757709
```

```
## horsepower    -0.021056656  -0.021497933  -0.021913677  -0.022302891
## weight        -0.002252423  -0.002330193  -0.002407954  -0.002485520
## acceleration   0.028678541   0.032842430   0.036995350   0.041132255
## year           0.288662261   0.295328992   0.301798972   0.308067892
## origin         0.098020243   0.101303828   0.104744211   0.108334123
##
## (Intercept)  -14.290388128 -14.622010110 -14.942298715 -15.250830628
## cylinders     -0.053313065  -0.044579115  -0.036531546  -0.028497911
## displacement  -0.004686888  -0.004605885  -0.004507933  -0.004406357
## horsepower    -0.022664946  -0.022999461  -0.023303173  -0.023582120
## weight        -0.002562712  -0.002639351  -0.002715286  -0.002790318
## acceleration   0.045248218   0.049337741   0.053396579   0.057415886
## year           0.314132935   0.319992041   0.325652016   0.331096376
## origin         0.112064006   0.115922416   0.119950187   0.124033608
##
## (Intercept)  -15.547868850 -15.833488563 -16.107779105 -1.636979e+01
## cylinders     -0.020860552  -0.013659676  -0.006922514 -4.083554e-04
## displacement  -0.004295273  -0.004175007  -0.004046056 -3.924827e-03
## horsepower    -0.023833924  -0.024059097  -0.024258422 -2.444240e-02
## weight        -0.002864285  -0.002937022  -0.003008372 -3.077317e-03
## acceleration   0.061389381   0.065309685   0.069169316  7.290652e-02
## year           0.336332596   0.341361341   0.346183713  3.508017e-01
## origin         0.128203500   0.132443448   0.136737046  1.407508e-01
##
## (Intercept)  -16.625382411 -16.870413711 -17.104647397 -17.328179681
## cylinders             .             .             .             .
## displacement  -0.003694268  -0.003455235  -0.003216005  -0.002977866
## horsepower    -0.024562263  -0.024659079  -0.024737135  -0.024798032
## weight        -0.003145088  -0.003211351  -0.003275932  -0.003338696
## acceleration   0.076587564   0.080198520   0.083729735   0.087173257
## year           0.355374835   0.359748661   0.363913570   0.367872735
## origin         0.145500322   0.150328992   0.155169608   0.159994356
##
## (Intercept)  -17.541132793 -17.743655507 -17.935921667 -18.118128605
## cylinders             .             .             .             .
## displacement  -0.002742068  -0.002509751  -0.002281947  -0.002059577
## horsepower    -0.024843390  -0.024874814  -0.024893877  -0.024902107
## weight        -0.003399518  -0.003458290  -0.003514920  -0.003569336
## acceleration   0.090521594   0.093767984   0.096906438   0.099931776
## year           0.371629930   0.375189428   0.378555953   0.381734636
## origin         0.164776062   0.169490025   0.174114091   0.178628604
##
## (Intercept)  -18.290496548 -18.453267723 -18.605642728 -18.749954394
## cylinders             .             .             .             .
## displacement  -0.001843444  -0.001634236  -0.001439550  -0.001246084
## horsepower    -0.024900969  -0.024891849  -0.024883223  -0.024862305
## weight        -0.003621479  -0.003671309  -0.003718086  -0.003763195
## acceleration   0.102839686   0.105626774   0.108239398   0.110775400
## year           0.384730977   0.387550793   0.390197838   0.392682521
## origin         0.183016482   0.187263239   0.191196004   0.195121207
##
## (Intercept)  -18.885593300 -1.901281e+01 -1.913058e+01 -1.924182e+01
## cylinders             .             .             .             .
## displacement  -0.001060434 -8.832591e-04 -7.224465e-04 -5.630676e-04
```

```
## horsepower    -0.024836429 -2.480706e-02 -2.478343e-02 -2.475025e-02
## weight        -0.003806017 -3.846533e-03 -3.883934e-03 -3.919888e-03
## acceleration   0.113189752  1.154794e-01  1.175845e-01  1.196239e-01
## year           0.395009808  3.971862e-01  3.992133e-01  4.011075e-01
## origin         0.198887415  2.024804e-01  2.057243e-01  2.089537e-01
##
## (Intercept)  -19.345761865 -19.442648422 -1.953130e+01 -1.961494e+01
## cylinders      .             .             .             .
## displacement  -0.000411654  -0.000268758 -1.421048e-04 -1.652439e-05
## horsepower    -0.024715043  -0.024679166 -2.465213e-02 -2.461736e-02
## weight        -0.003953751  -0.003985517 -4.014351e-03 -4.042050e-03
## acceleration   0.121549707   0.123359654  1.249903e-01  1.265718e-01
## year           0.402871632   0.404512155  4.060279e-01  4.074400e-01
## origin         0.212021391   0.214915410  2.174674e-01  2.200085e-01
##
## (Intercept)  -19.685192791 -19.75297424 -19.816632146 -19.872266877
## cylinders      .             .            0.002512119   0.006803420
## displacement   .             .            .             .
## horsepower    -0.024584742  -0.02453298  -0.024495634  -0.024489769
## weight        -0.004057179  -0.00407090  -0.004088408  -0.004107116
## acceleration   0.127670640   0.12877198   0.129988970   0.131137255
## year           0.408606313   0.40971050   0.410713970   0.411579808
## origin         0.219926290   0.21931754   0.219510798   0.220435375
##
## (Intercept)  -19.923427117 -19.970565974 -20.011561098 -20.050967500
## cylinders      0.010799743   0.014471992   0.017612402   0.020702238
## displacement   .             .             .             .
## horsepower    -0.024482879  -0.024475596  -0.024483931  -0.024480502
## weight        -0.004124383  -0.004140299  -0.004154123  -0.004167496
## acceleration   0.132200772   0.133183753   0.133988762   0.134805607
## year           0.412371397   0.413099989   0.413764435   0.414378738
## origin         0.221301369   0.222092690   0.222802682   0.223478275
##
## (Intercept)  -20.087687499 -20.121559798
## cylinders      0.023558176   0.026180852
## displacement   .             .
## horsepower    -0.024472974  -0.024464530
## weight        -0.004179925  -0.004191382
## acceleration   0.135578695   0.136296020
## year           0.414944102   0.415463524
## origin         0.224087081   0.224639936
```

```
# visualizing AUC and regularization parameters
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
superpose.line = list(col = myCol))
plot(model.glmn.enet, par.settings = myPar, xTrans = function(x) log(x))
```

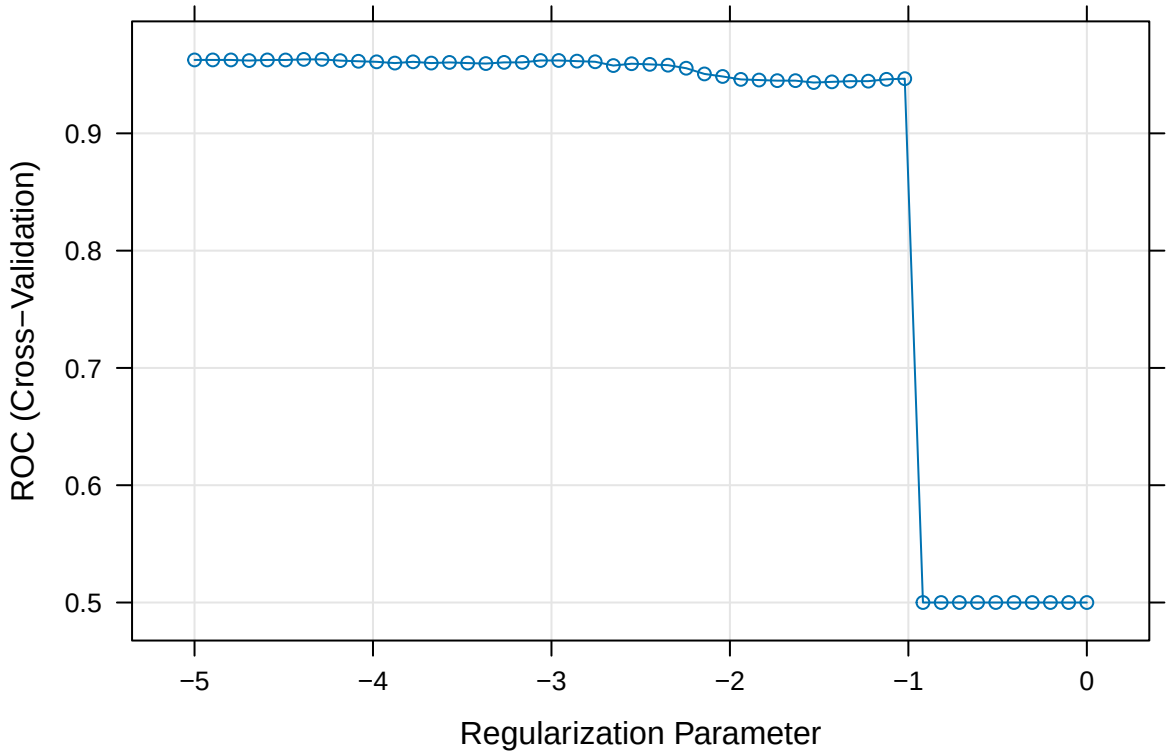



```
best_lambda_enet <- model.glmn.enet$bestTune$lambda

best_lambda_coef_enet <- coef(model.glmn.enet$finalModel, s = best_lambda_enet)
best_lambda_coef_enet
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)  -14.194503875
## cylinders     -0.055827150
## displacement  -0.004706672
## horsepower    -0.022563801
## weight        -0.002541147
## acceleration   0.044098368
## year           0.312438584
## origin         0.111022013
```

```
# lasso
glmnGrid.lasso <- expand.grid(.alpha =  1,
                              .lambda = exp(seq(-5, 0, length = 50)))

set.seed(2)

model.glmn.lasso <- train(x = training_data[1:7],
                    y = training_data$mpg_cat,
                    method = "glmnet",
                    tuneGrid = glmnGrid.lasso,
                    metric = "ROC",
                    trControl = ctrl)
```

```r
plot(model.glmn.lasso, par.settings = myPar, xTrans = function(x) log(x))
```




```r
best_lambda_lasso <- model.glmn.lasso$bestTune$lambda

best_lambda_coef_lasso <- coef(model.glmn.lasso$finalModel, s = best_lambda_lasso)
best_lambda_coef_lasso
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  -1.260768e+01
## cylinders    -4.852480e-02
## displacement -3.977988e-04
## horsepower   -1.095520e-02
## weight       -3.074227e-03
## acceleration  5.727472e-02
## year          2.860600e-01
## origin        4.339447e-02
```

Per the elastic net penalized logistic model, there are no redundant predictors. Lasso is more likely to shrink coefficients to zero, so when using an alpha = 1, no coefficients shrank in either model, telling us that all predictors are meaningful in the model. It's important to note in the matrix of coefficients for each respective model, some lambda values did shrink certain coefficients, but the lambda in both final models included all coefficients.

1b) Based on the model in (a), set a probability threshold to determine the class labels and compute the confusion matrix using the test data. Briefly interpret what the confusion matrix reveals about your model's performance.

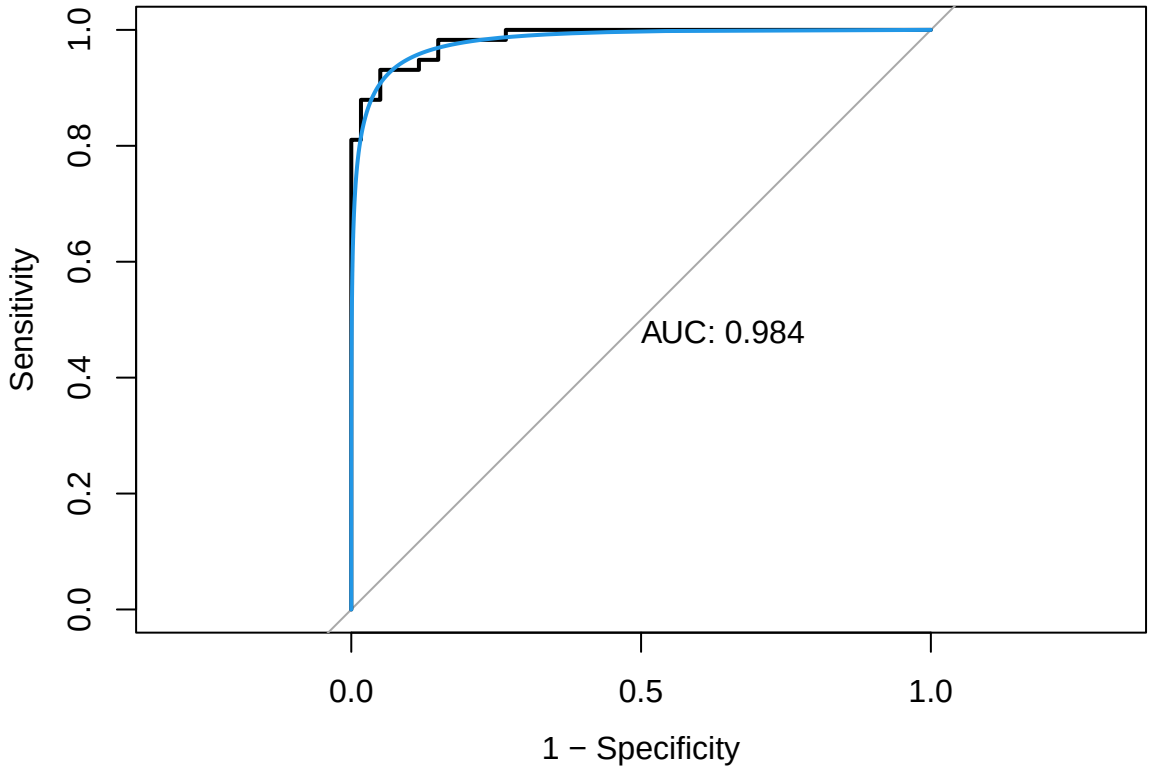
```
# checking coding
contrasts(auto$mpg_cat)
```

```
##      high
## low     0
## high    1
```

```
# predict class labels using the logistic regression model and testing data
test.pred.prob <- predict(glm.fit,
                          newdata = testing_data,
                          type = "response")

 # setting a probability threshold of 0.5
test.pred <- rep("low", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] <- "high"


confusionMatrix(data = as.factor(test.pred),
                reference = testing_data$mpg_cat,
                positive = "high")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   57    5
##       high   3   53
##
##                Accuracy : 0.9322
##                  95% CI : (0.8708, 0.9703)
##     No Information Rate : 0.5085
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8643
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9138
##             Specificity : 0.9500
##          Pos Pred Value : 0.9464
##          Neg Pred Value : 0.9194
##              Prevalence : 0.4915
##          Detection Rate : 0.4492
##    Detection Prevalence : 0.4746
##       Balanced Accuracy : 0.9319
##
##        'Positive' Class : high
##
```

```r
# plotting test ROC curve
roc.glm <- roc(testing_data$mpg_cat, test.pred.prob)
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE) # smooth ROC curve
```



The matrix reveals the accuracy of the model is 93.22%[87.08%-97.03%], meaning the model correctly predicts the class (low or high mileage car) 93.22% of the time on the testing data and we are 95% confident the accuracy ranges from 87.98% to 93.03%. A p-value less than 0.05 indicates to us there's sufficient evidence that the model's accuracy is better than simply predicting the most frequent class. A Kappa of 0.8643 shows there is high inter-rater agreement between the observed label and predicted label (by chance). A sensitivity of 0.9138 indicates that the model correctly identifies 91.38% of the high mileage cars, and a specificity of 0.95 indicates that the model correctly identifies 95% of the low mileage cars.

In sum, the confusion matrix tells us the model performs well in distinguishing between high and low gas mileage cars. It has high accuracy, sensitivity, specificity, as well as positive and negative predictive values (0.9464 & 0.9194, respectively), suggesting that it is effective in making predictions.

1c) Train a multivariate adaptive regression spline (MARS) model. Does the MARS model improve the prediction performance compared to logistic regression?

```
set.seed(2)

# log stats
coef(model.glm$finalModel)
```

```
##   (Intercept)     cylinders  displacement    horsepower        weight
## -2.054216e+01  2.717063e-02  9.626894e-04 -2.423505e-02 -4.363108e-03
##  acceleration          year        origin
##  1.453842e-01  4.225079e-01  2.459384e-01
```

```
summary(model.glm)
```

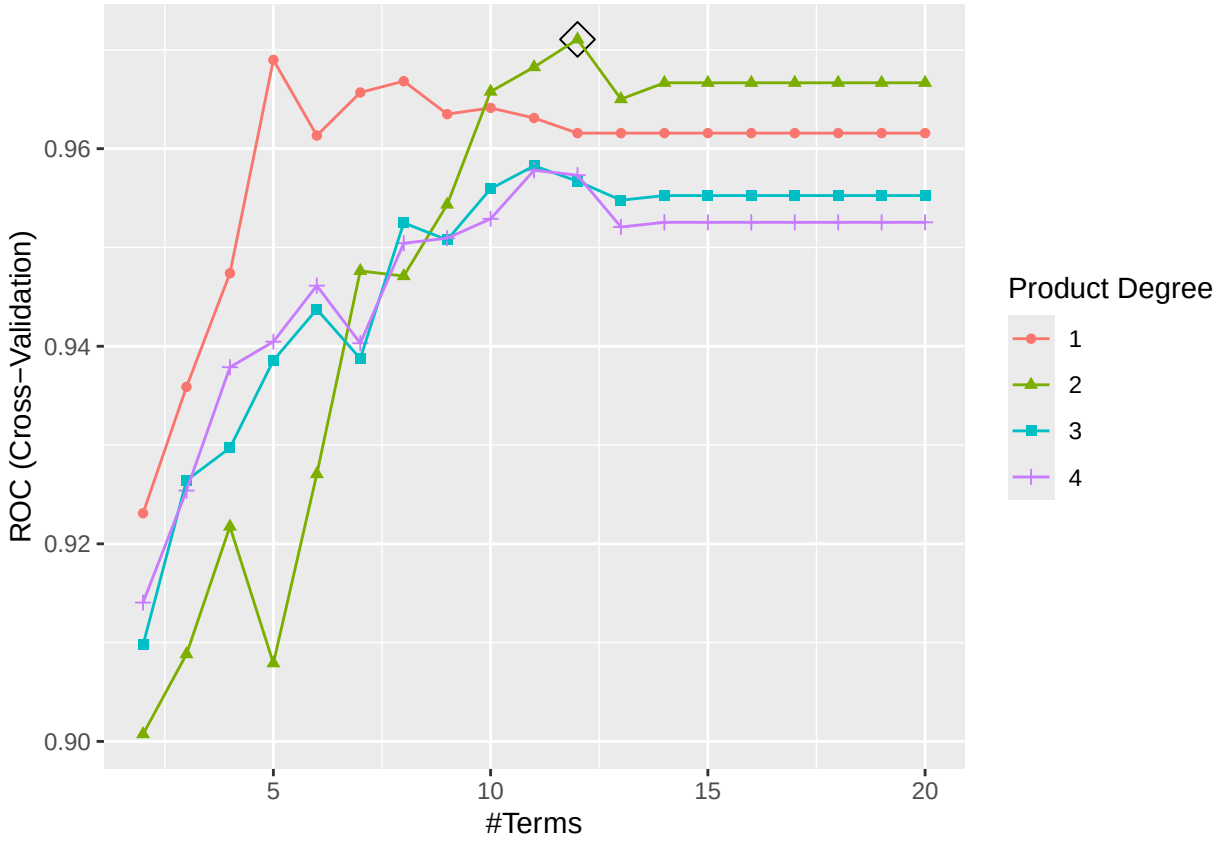
```
##
## Call:
## NULL
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.054e+01  6.889e+00  -2.982 0.002866 **
## cylinders     2.717e-02  4.756e-01   0.057 0.954446
## displacement  9.627e-04  1.305e-02   0.074 0.941212
## horsepower   -2.423e-02  2.554e-02  -0.949 0.342653
## weight       -4.363e-03  1.226e-03  -3.558 0.000373 ***
## acceleration  1.454e-01  1.593e-01   0.913 0.361485
## year          4.225e-01  8.732e-02   4.838 1.31e-06 ***
## origin        2.459e-01  4.057e-01   0.606 0.544401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 379.83  on 273  degrees of freedom
## Residual deviance: 120.29  on 266  degrees of freedom
## AIC: 136.29
##
## Number of Fisher Scoring iterations: 7
```

```
model.glm$fin
```

```
##
## Call:  NULL
##
## Coefficients:
##  (Intercept)     cylinders  displacement    horsepower        weight
##   -2.054e+01     2.717e-02     9.627e-04    -2.424e-02    -4.363e-03
## acceleration          year        origin
##    1.454e-01     4.225e-01     2.459e-01
##
## Degrees of Freedom: 273 Total (i.e. Null);  266 Residual
## Null Deviance:       379.8
```

```
## Residual Deviance: 120.3     AIC: 136.3
```

```
# MARS
set.seed(2)

model.mars <- train(x = training_data[1:7],
                    y = training_data$mpg_cat,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl)

ggplot(model.mars, highlight = TRUE)
```




```
coef(model.mars$finalModel)
```

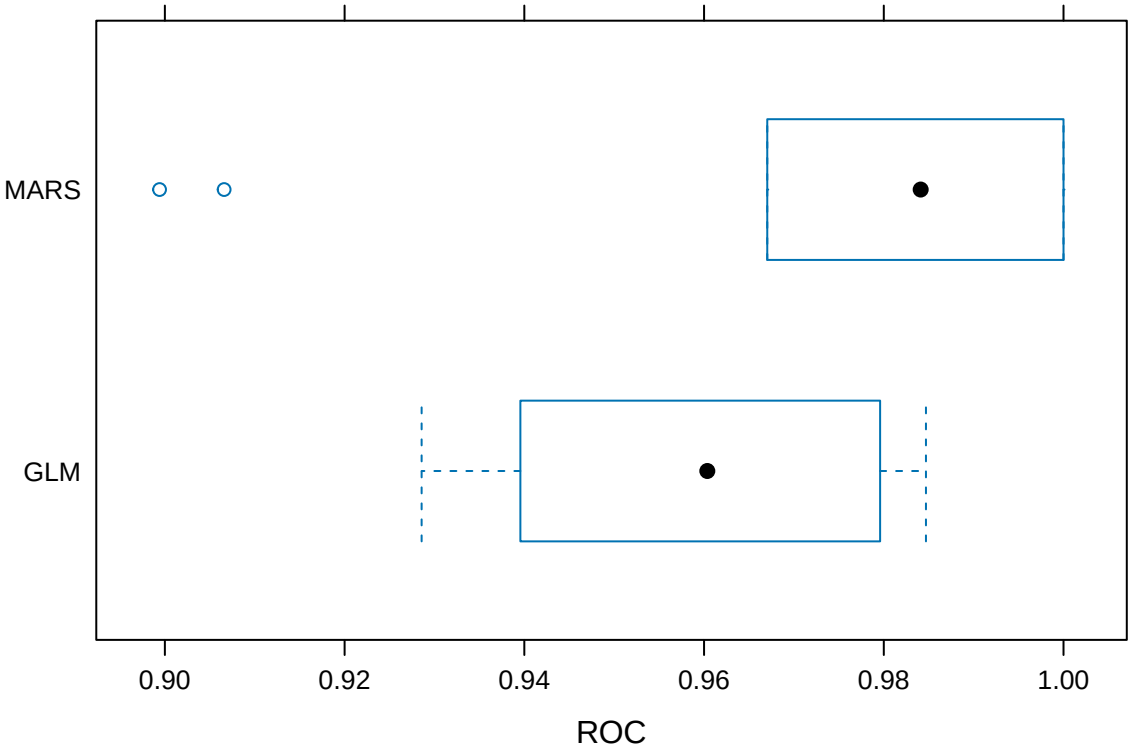
```
##                                (Intercept)
##                              -7.7330069243
##                      h(displacement-232)
##                               -0.9356130852
##                      h(232-displacement)
##                                0.0847147717
##                                  h(year-72)
##                                0.7579418640
##      h(4-cylinders) * h(232-displacement)
##                               -0.6891088932
##          h(displacement-173) * h(year-72)
```

```
##                               -0.0257570538
##           h(173-displacement) * h(year-72)
##                               -0.0115097096
##       h(232-displacement) * h(weight-2648)
##                               -0.0001223507
##           h(displacement-232) * h(year-75)
##                                0.2998569916
##          h(acceleration-14.5) * h(year-72)
##                                0.2342242317
##                h(4-cylinders) * h(year-72)
##                               16.3097304420
## h(232-displacement) * h(acceleration-14.1)
##                               -0.0037792287
```

```
set.seed(2)


res <- resamples(list(GLM = model.glm,
                      MARS = model.mars))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, MARS
## Number of resamples: 10
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM  0.9285714 0.9395604 0.9603611 0.9593316 0.9795918 0.9846939    0
## MARS 0.8994083 0.9676217 0.9841052 0.9710633 1.0000000 1.0000000    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM  0.6923077 0.7857143 0.8846154 0.8527473 0.9271978 0.9285714    0
## MARS 0.7857143 0.8008242 0.8846154 0.8829670 0.9285714 1.0000000    0
##
## Spec
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM  0.7692308 0.8736264 0.9285714 0.9049451 0.9285714    1    0
## MARS 0.6923077 0.9285714 0.9285714 0.9329670 1.0000000    1    0
```

```
bwplot(res, metric = "ROC")
```

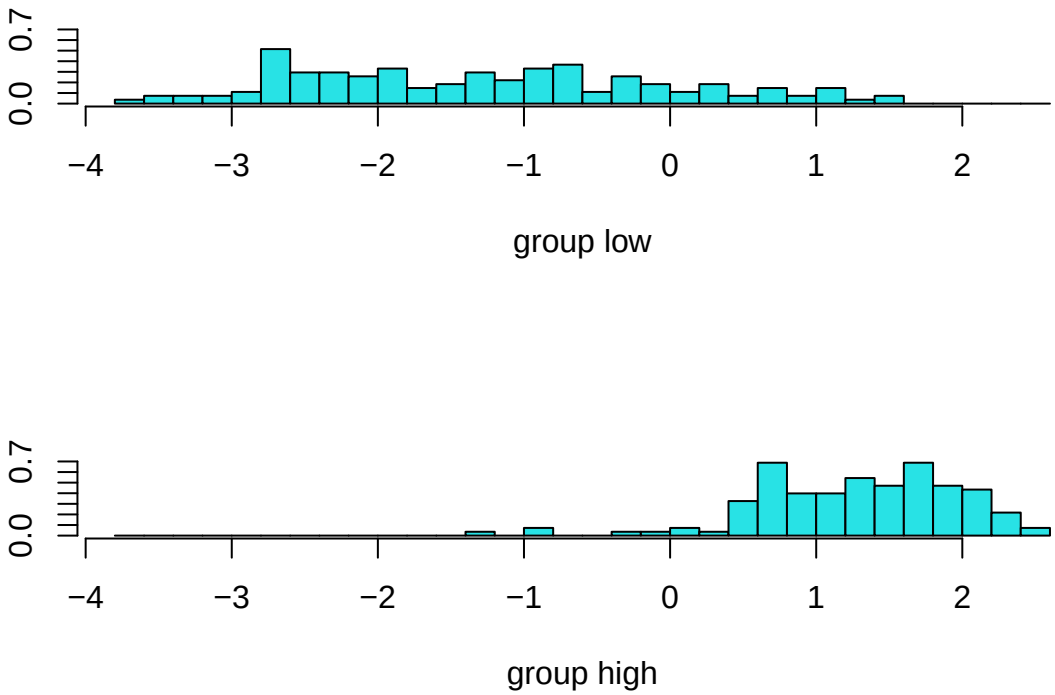



```
# predicted probabilities with testing data
glm.pred <- predict(model.glm, newdata = testing_data, type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = testing_data, type = "prob")[,2]

# ROC curves
roc.glm <- roc(testing_data$mpg_cat, glm.pred)
roc.mars <- roc(testing_data$mpg_cat, mars.pred)
```

MARS has the highest mean and median ROC values, already pointing us that this model will have a better discriminatory power compared to the logistic model. However, in order to properly determine if MARS has better **predictive power**, we must consider the predicted probabilites using the testing data and compare the area under the curve values for each respective ROC curve for the log model and the MARS model. Based on the AUC values, the MARS model does marginally improve predictive performance compared to logistic regression.

1d) Perform linear discriminant analysis using the training data. Plot the linear discriminant variable(s).

```
set.seed(2)

# LDA using caret
model.lda <- train(x = training_data[, 1:7],
                   y = training_data$mpg_cat,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)


# prediction
lda.pred2 <- predict(model.lda, newdata = testing_data, type = "prob")

# LDA using MASS
lda.fit <- lda(mpg_cat~.,
               data = training_data)

# linear discriminant variables
plot(lda.fit)
```

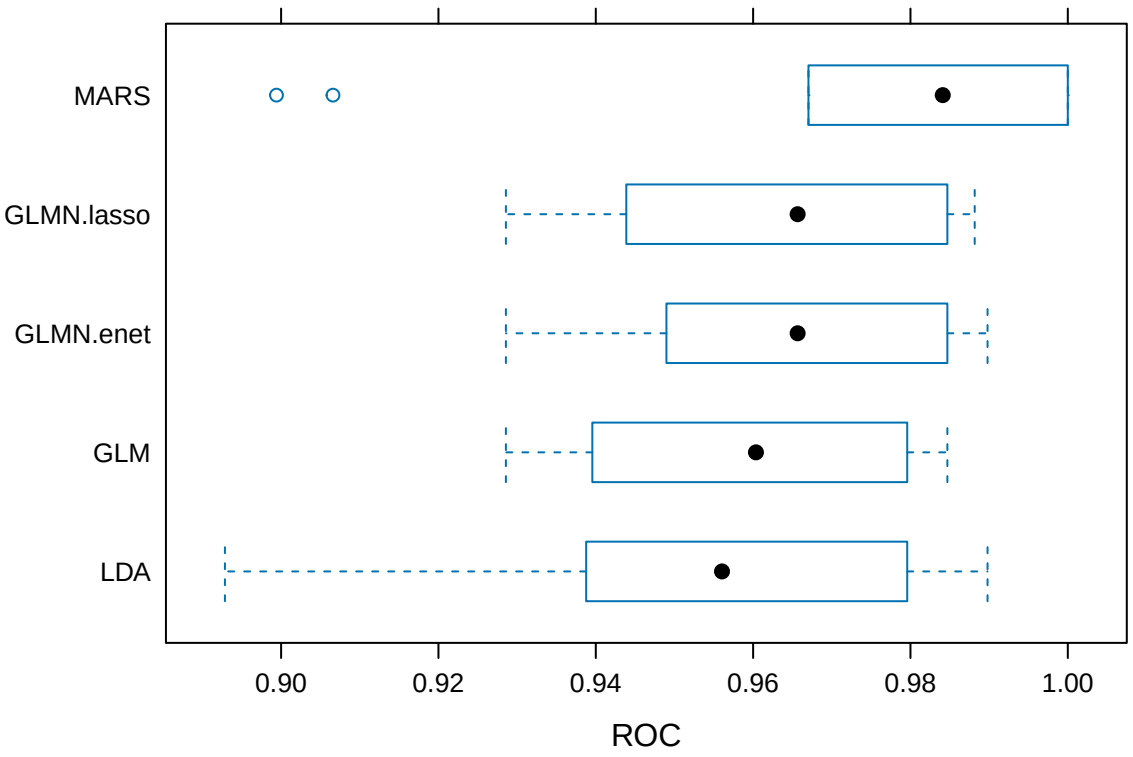


1e) Which model will you use to predict the response variable? Plot its ROC curve using the test data. Report the AUC and the misclassification error rate.

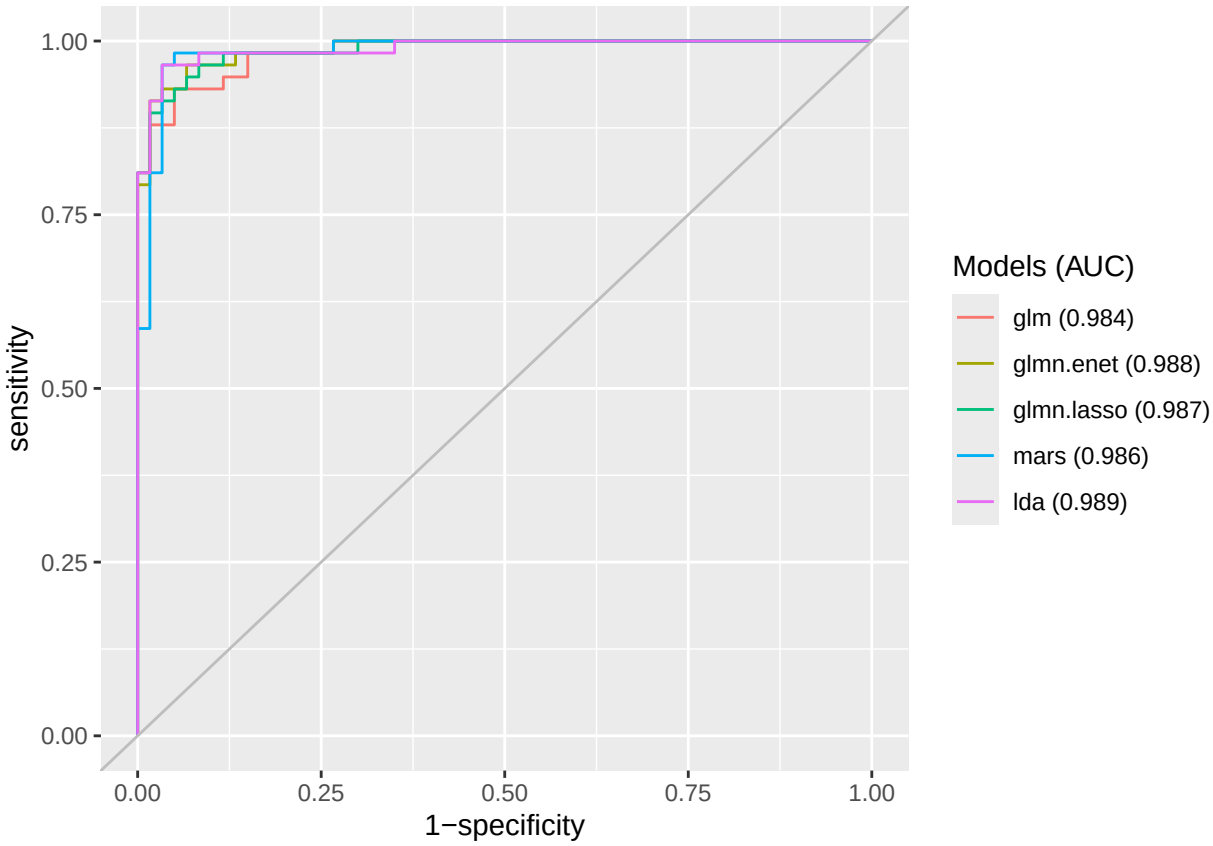
```
# resampling comparison
res = resamples(list(
                     GLM = model.glm,
                     GLMN.enet = model.glmn.enet,
                     GLMN.lasso = model.glmn.lasso,
                     MARS = model.mars,
                     LDA = model.lda))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMN.enet, GLMN.lasso, MARS, LDA
## Number of resamples: 10
##
## ROC
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM        0.9285714 0.9395604 0.9603611 0.9593316 0.9795918 0.9846939    0
## GLMN.enet  0.9285714 0.9507457 0.9656593 0.9651823 0.9846939 0.9897959    0
## GLMN.lasso 0.9285714 0.9441719 0.9656593 0.9630630 0.9846939 0.9881657    0
## MARS       0.8994083 0.9676217 0.9841052 0.9710633 1.0000000 1.0000000    0
## LDA        0.8928571 0.9389717 0.9560440 0.9516000 0.9783163 0.9897959    0
##
## Sens
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM        0.6923077 0.7857143 0.8846154 0.8527473 0.9271978 0.9285714    0
## GLMN.enet  0.6923077 0.7857143 0.8846154 0.8527473 0.9271978 0.9285714    0
## GLMN.lasso 0.6923077 0.7857143 0.8846154 0.8527473 0.9271978 0.9285714    0
## MARS       0.7857143 0.8008242 0.8846154 0.8829670 0.9285714 1.0000000    0
## LDA        0.6923077 0.7733516 0.8516484 0.8230769 0.8571429 0.9285714    0
##
## Spec
##                 Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## GLM        0.7692308 0.8736264 0.9285714 0.9049451 0.9285714    1    0
## GLMN.enet  0.8461538 0.8736264 0.9285714 0.9126374 0.9285714    1    0
## GLMN.lasso 0.8461538 0.8736264 0.9285714 0.9197802 0.9285714    1    0
## MARS       0.6923077 0.9285714 0.9285714 0.9329670 1.0000000    1    0
## LDA        0.9230769 0.9285714 0.9285714 0.9560440 1.0000000    1    0
```

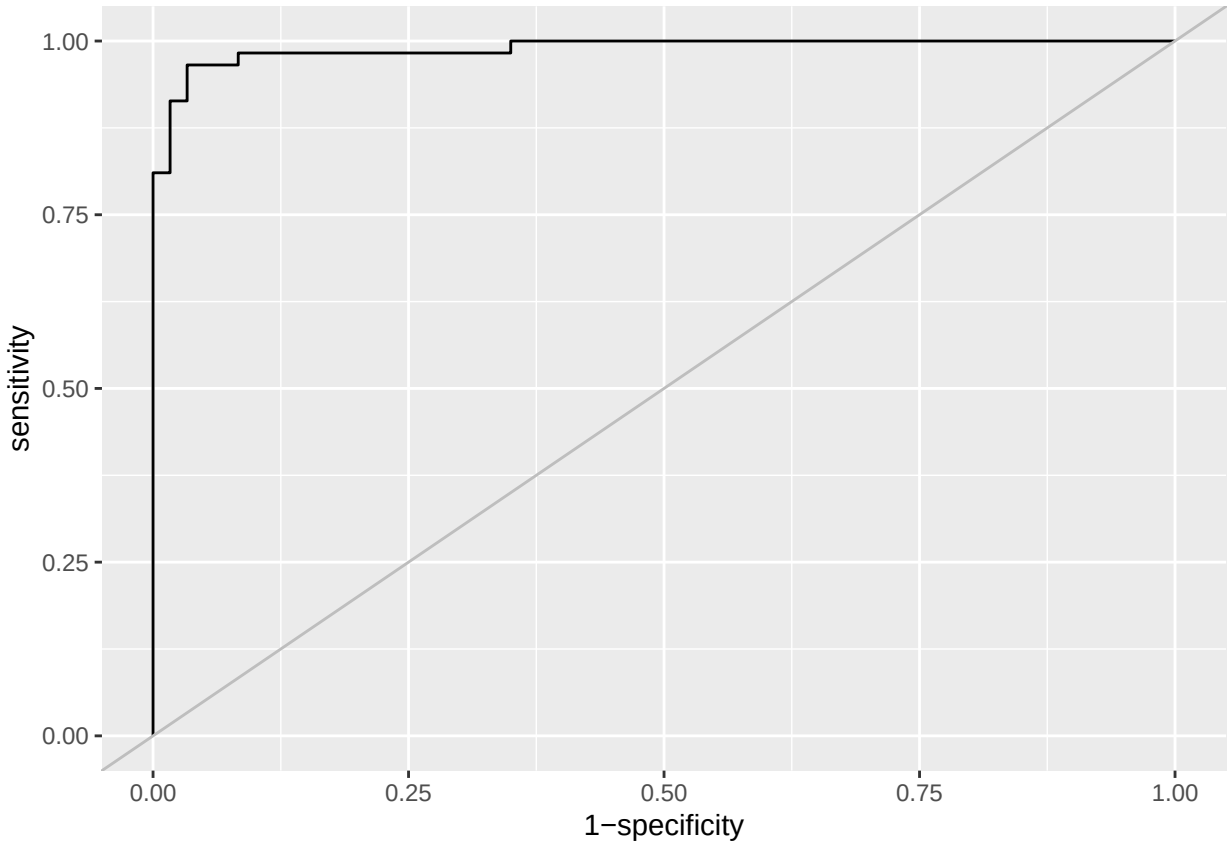
```
bwplot(res, metric = "ROC")
```




```
# predicition
glm.pred <- predict(model.glm, newdata = testing_data, type = "prob")[,2]
glmn.enet.pred <- predict(model.glmn.enet, newdata = testing_data, type = "prob")[,2]
glmn.lasso.pred <- predict(model.glmn.lasso, newdata = testing_data, type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = testing_data, type = "prob")[,2]
lda.pred <- predict(model.lda, newdata = testing_data, type = "prob")[,2]

# ROC curves (for AUC)
roc.glm <- roc(testing_data$mpg_cat, glm.pred)
roc.glmn.enet <- roc(testing_data$mpg_cat, glmn.enet.pred)
roc.glmn.lasso <- roc(testing_data$mpg_cat, glmn.lasso.pred)
roc.mars <- roc(testing_data$mpg_cat, mars.pred)
roc.lda <- roc(testing_data$mpg_cat, lda.pred)

# AUC values
auc <- c(roc.glm$auc[1], roc.glmn.enet$auc[1], roc.glmn.lasso$auc[1], roc.mars$auc[1], roc.lda$auc[1])

modelNames <- c("glm", "glmn.enet", "glmn.lasso", "mars","lda")

# combined ROC curves
ggroc(list(roc.glm, roc.glmn.enet, roc.glmn.lasso, roc.mars, roc.lda),
      legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") + geom_abline(intercept = 0, slope = 1, color = "grey")
```




```
# ROC for best model only (LDA)
ggroc(roc.lda, legacy.axes = T) +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```




```
# AUC for LDA
roc.lda$auc
```

```
## Area under the curve: 0.9891
```

```
misclass = predict(model.lda, newdata = testing_data, type = "raw")
misclass2 = predict(model.mars, newdata = testing_data, type = "raw")
# Convert character labels to binary
misclass_binary <- ifelse(misclass == "low", 0, 1)
misclass_binary2 <- ifelse(misclass2 == "low", 0, 1)
# take the mean of the logical vector
mean(misclass_binary)
```

```
## [1] 0.5338983
```

```
mean(misclass_binary2)
```

```
## [1] 0.4915254
```

From the re-sampling summary, we can see the MARS model has the highest mean and median ROC so I would use the MARS model to predict miles per gallon. The LDA model, however, has a greater AUC indicating better predictive performance in comparison to the MARS model. Considering the context of the data, while having high sensitivity and specificity (and therefore higher ROC values) are important for understanding and interpreting model performance, I will prefer to choose a model with higher discriminatory power and a better overall performance in correctly classifying high and low mileage cars. Therefore, I will choose the LDA model over the MARS. The AUC of the LDA model is 0.9890805. The mis-classification error rate is 0.5338983.