Homework 3

Zhezheng Jin

Contents

```
library(tidyverse)
library(caret)
library(glmnet)
library(MASS)
library(tidymodels)
library(mlbench)
library(pROC)
library(pdp)
library(vip)
library(AppliedPredictiveModeling)
```

```
# Data Import
auto = read_csv("auto.csv") %>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    origin = as.factor(origin),
    cylinders = as.factor(cylinders)
    )
```

```
## Rows: 392 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): mpg_cat
## dbl (7): cylinders, displacement, horsepower, weight, acceleration, year, or...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
skimr::skim(auto)
```

Table 1: Data summary

Name	auto
Number of rows	392
Number of columns	8
Column type frequency:	
factor	3
numeric	5
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cylinders	0	1	FALSE	5	4: 199, 8: 103, 6: 83, 3: 4
origin	0	1	FALSE	3	1: 245, 3: 79, 2: 68
mpg_cat	0	1	FALSE	2	hig: 196, low: 196

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
displacement	0	1	194.41	104.64	68	105.00	151.0	275.75	455.0	
horsepower	0	1	104.47	38.49	46	75.00	93.5	126.00	230.0	
weight	0	1	2977.58	849.40	1613	2225.25	2803.5	3614.75	5140.0	
acceleration	0	1	15.54	2.76	8	13.78	15.5	17.02	24.8	
year	0	1	75.98	3.68	70	73.00	76.0	79.00	82.0	

```
# data partition
set.seed(5)
data_split <- initial_split(auto, prop = 0.7)
train <- training(data_split)
test <- testing(data_split)
```

The "auto" dataset contains 8 columns and 392 observations. Then we partition the dataset into two parts: training data (70%) and test data (30%), where the training data and test data contains 274 and 118 observations, respectively.

(a) Logistic Regression

```
contrasts(auto$mpg_cat)
```

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

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

# Using Penalized logistic regression (elastic net for Logistic)
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-6, 2, length = 50)))
set.seed(5)
model.glmn <- train(x = train[1:7],
                    y = train$mpg_cat,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)
model.glmn$bestTune
```

```
##     alpha     lambda
## 116   0.1 0.02869534
```

```
# Coefficients
coef(model.glmn$finalModel, model.glmn$bestTune$lambda)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                       s1
## (Intercept)   8.722967917
## cylinders     0.426298244
## displacement  0.007038546
## horsepower    0.015157530
## weight        0.001166032
## acceleration -0.036358680
## year         -0.210883361
## origin       -0.132528586
```

```
# Using caret for comparison
set.seed(5)
model.glm <- train(x = train [1:7],   # exclude the outcome
                   y = train$mpg_cat, # the same as mpg_cat ~ ., data = train
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
```

Based on the model coefficients matrix, there are no redundant predictors in the model as all of them have been assigned non-zero coefficients. non-zero coefficients indicate that after the penalization process, all have been deemed relevant to some extent for predicting the outcome variable.

We first consider the simple classifier with a cut-off of 0.5 and evaluate its performance on the test data.

(b) Performance Evaluation

```
test.pred.prob <- predict(model.glmn, newdata = test,
type = "prob") [,2]
test.pred <- rep("high", length(test.pred.prob))
test.pred[test.pred.prob>0.5] <- "low"

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   60   8
##       low     3  47
##
##                Accuracy : 0.9068
##                  95% CI : (0.8393, 0.9525)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8116
##
##  Mcnemar's Test P-Value : 0.2278
##
##             Sensitivity : 0.8545
```

```
##             Specificity : 0.9524
##          Pos Pred Value : 0.9400
##          Neg Pred Value : 0.8824
##              Prevalence : 0.4661
##          Detection Rate : 0.3983
##    Detection Prevalence : 0.4237
##       Balanced Accuracy : 0.9035
##
##        'Positive' Class : low
##
```

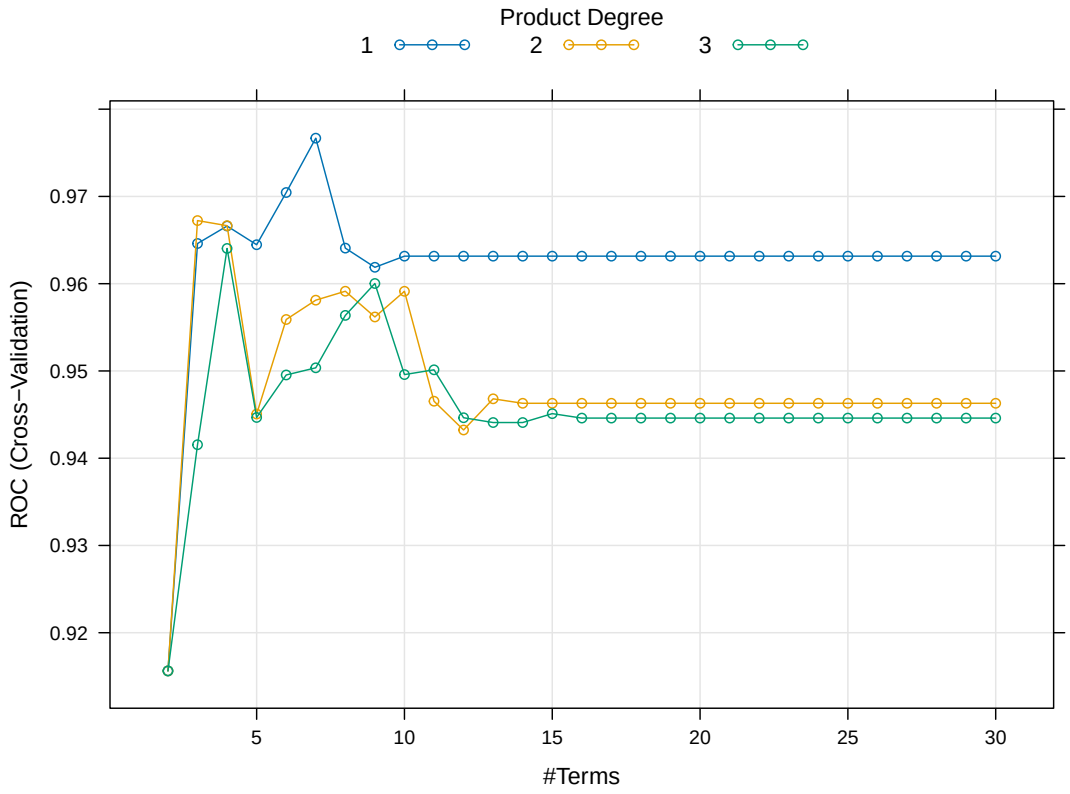
Based on our confusion matrix analysis, our model's accuracy when applied to test data is 90.68% (95% CI: 83.83% to 95.25%). No information rate is 53.39%, which represents the accuracy if we made the same class prediction for all observations without any information. The p-value is close to 0 which means the accuracy is statistically significantly better than our no information rate. Our sensitivity (true positives of all actual positives) and specificity (true negatives of all actual negatives) are 85.45% and 95.24%, respectively, with a positive predictive value (true positives of all predicted positives) and negative predictive value (true negatives of all predicted negatives) of 94% and 88.24%, respectively. Additionally, our model demonstrates a balanced accuracy of 90.35%, calculated as the average of our sensitivity and specificity, which indicates good performance in detecting both true positives and true negatives. The high kappa value of 0.8116 suggests a strong inter-rater agreement, even accounting for the possibility of chance agreement.

(c) MARS

```
set.seed(5)

model.mars <- train(x = train[1:7],
                    y = train$mpg_cat,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:3,nprune = 2:30),
                    metric = "ROC",
                    trControl = ctrl)
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

##
## Attaching package: 'plotrix'

## The following object is masked from 'package:scales':
##
##     rescale

## Loading required package: TeachingDemos
```

```
plot(model.mars)
```



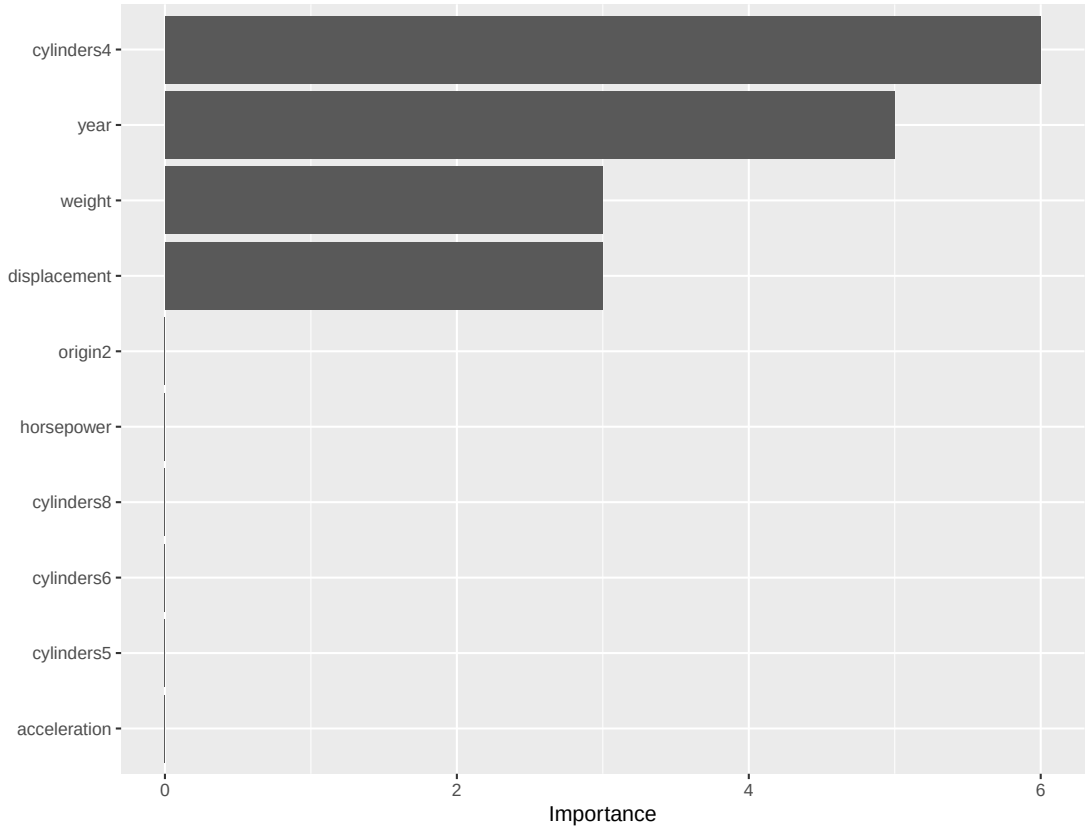

```
model.mars$bestTune %>%
  knitr::kable()
```

	nprune	degree
6	7	1

```
coef(model.mars$finalModel) %>%
  knitr::kable(col.names = "Coefficient")
```

	Coefficient
(Intercept)	0.4848229
cylinders4	-3.2265952
h(displacement-232)	-0.0592572
h(displacement-122)	-1.4688226
h(displacement-119)	1.2686563
h(displacement-140)	0.2514909
h(year-78)	-1.5432048

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




```
# Confusion Matrix Comparison
mars.pred.prob <- predict(model.mars, newdata = test, type = "prob")[,2]
mars.pred <- rep("high", length(mars.pred.prob))
mars.pred[mars.pred.prob>0.5] <- "low"

matrix <- confusionMatrix(data = as.factor(mars.pred),
                          reference = test$mpg_cat,
                          positive = "low")
matrix
```

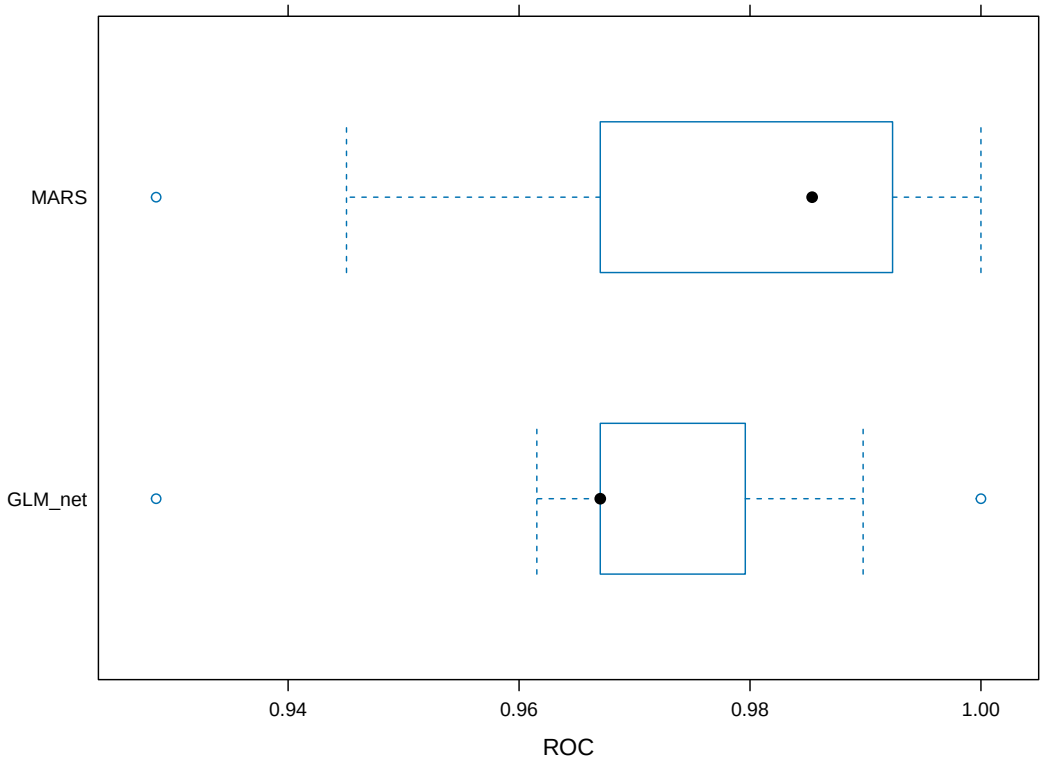
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   60   9
##       low     3  46
##
##                Accuracy : 0.8983
##                  95% CI : (0.8291, 0.9463)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7942
```

```
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.8364
##             Specificity : 0.9524
##          Pos Pred Value : 0.9388
##          Neg Pred Value : 0.8696
##              Prevalence : 0.4661
##          Detection Rate : 0.3898
##    Detection Prevalence : 0.4153
##       Balanced Accuracy : 0.8944
##
##        'Positive' Class : low
##
```

```
# ROC comparison
res <- resamples(list(MARS = model.mars,
                      GLM_net = model.glmn))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: MARS, GLM_net
## Number of resamples: 10
##
## ROC
##              Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## MARS    0.9285714 0.9701465 0.9853807 0.9766771 0.9915130    1    0
## GLM_net 0.9285714 0.9670330 0.9670330 0.9707117 0.9795657    1    0
##
## Sens
##              Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## MARS    0.8461538 0.9230769 0.9230769 0.9318681 0.9807692    1    0
## GLM_net 0.8461538 0.9230769 0.9285714 0.9472527 1.0000000    1    0
##
## Spec
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## MARS    0.8571429 0.9285714 0.9285714 0.9290476 0.9285714 1.0000000    0
## GLM_net 0.7142857 0.8571429 0.8571429 0.8719048 0.9285714 0.9333333    0
```

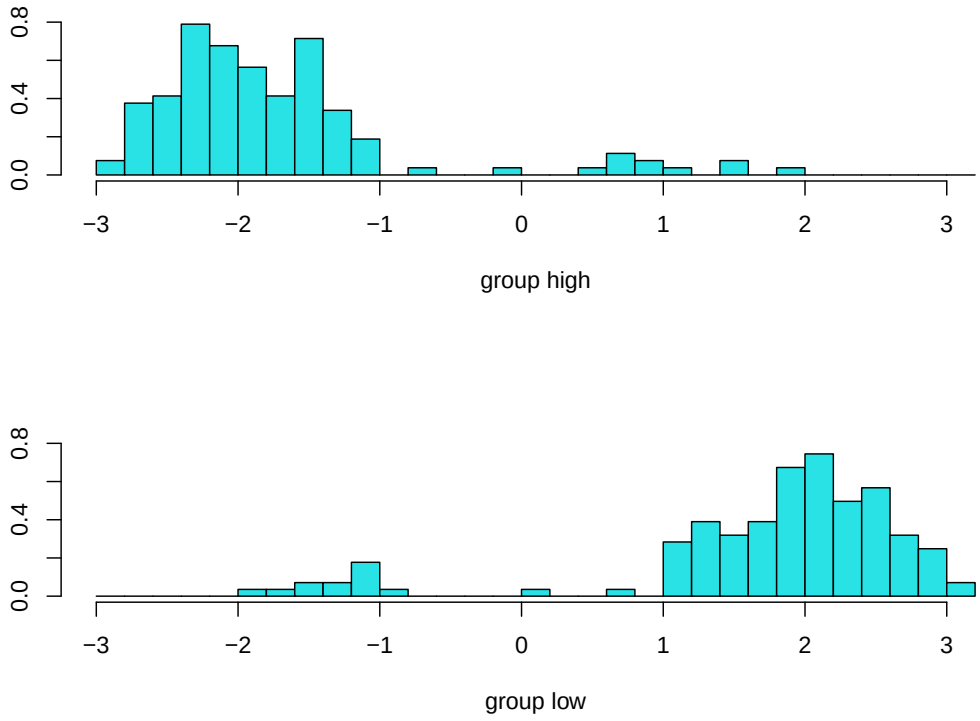
```
bwplot(res, metric = "ROC")
```



Based on the confusion matrix comparison, the penalized logistic regression model slightly outperforms the MARS model in predicting the performance for the dataset in question. The improvements are marginal across several key performance metrics, suggesting that while the logistic regression model is preferable in this instance, the difference is not overwhelmingly large. However, based on the ROC comparsion, the MARS has a higher mean ROC value, which could indeed overturn the initial conclusion based on accuracy, kappa, and other metrics tied to a specific threshold. This would suggest that while the penalized logistic regression may perform slightly better at the particular threshold chosen for classification (leading to higher accuracy, kappa, etc.), the MARS model is more robust overall, with a better capability to distinguish between classes across various thresholds.

(d) LDA

```
lda.fit <- lda(mpg_cat~., data = train)

# Plot the linear discriminants in LDA
plot(lda.fit)
```




```
# using caret for LDA
set.seed(5)
model.lda <- train(mpg_cat ~ .,
                   data =  train,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
```

Since we have two classes, we only have one linear discriminant, which allows us to generate a linear discriminant plot displaying our transformed predictors for each class in a histogram.

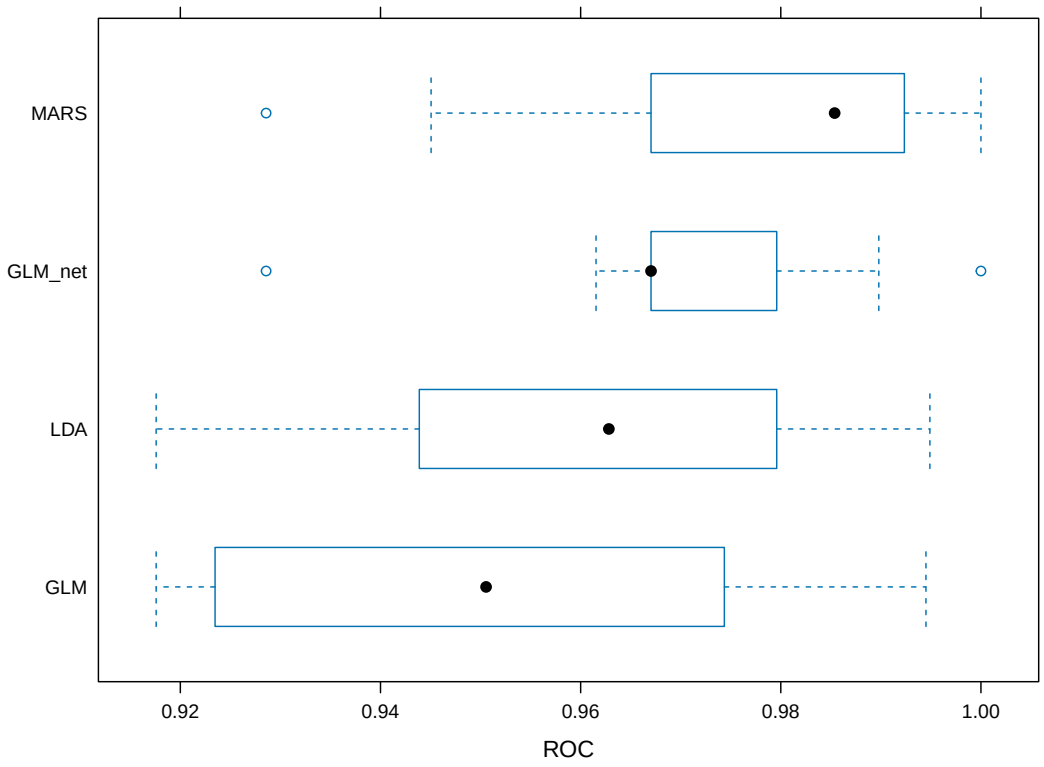
(e) Model Comparison

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

```
##
## Call:
## summary.resamples(object = res)
##
## Models: LDA, MARS, GLM, GLM_net
```

```
## Number of resamples: 10
##
## ROC
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA     0.9175824 0.9455455 0.9628205 0.9607745 0.9764521 0.9948980    0
## MARS    0.9285714 0.9701465 0.9853807 0.9766771 0.9915130 1.0000000    0
## GLM     0.9175824 0.9274922 0.9505495 0.9514783 0.9697802 0.9945055    0
## GLM_net 0.9285714 0.9670330 0.9670330 0.9707117 0.9795657 1.0000000    0
##
## Sens
##              Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## LDA     0.7692308 0.9230769 0.9230769 0.9164835 0.9285714    1    0
## MARS    0.8461538 0.9230769 0.9230769 0.9318681 0.9807692    1    0
## GLM     0.7692308 0.9230769 0.9285714 0.9395604 1.0000000    1    0
## GLM_net 0.8461538 0.9230769 0.9285714 0.9472527 1.0000000    1    0
##
## Spec
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA     0.7857143 0.8571429 0.9285714 0.9076190 0.9321429 1.0000000    0
## MARS    0.8571429 0.9285714 0.9285714 0.9290476 0.9285714 1.0000000    0
## GLM     0.7857143 0.8571429 0.9285714 0.9004762 0.9321429 1.0000000    0
## GLM_net 0.7142857 0.8571429 0.8571429 0.8719048 0.9285714 0.9333333    0
```

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



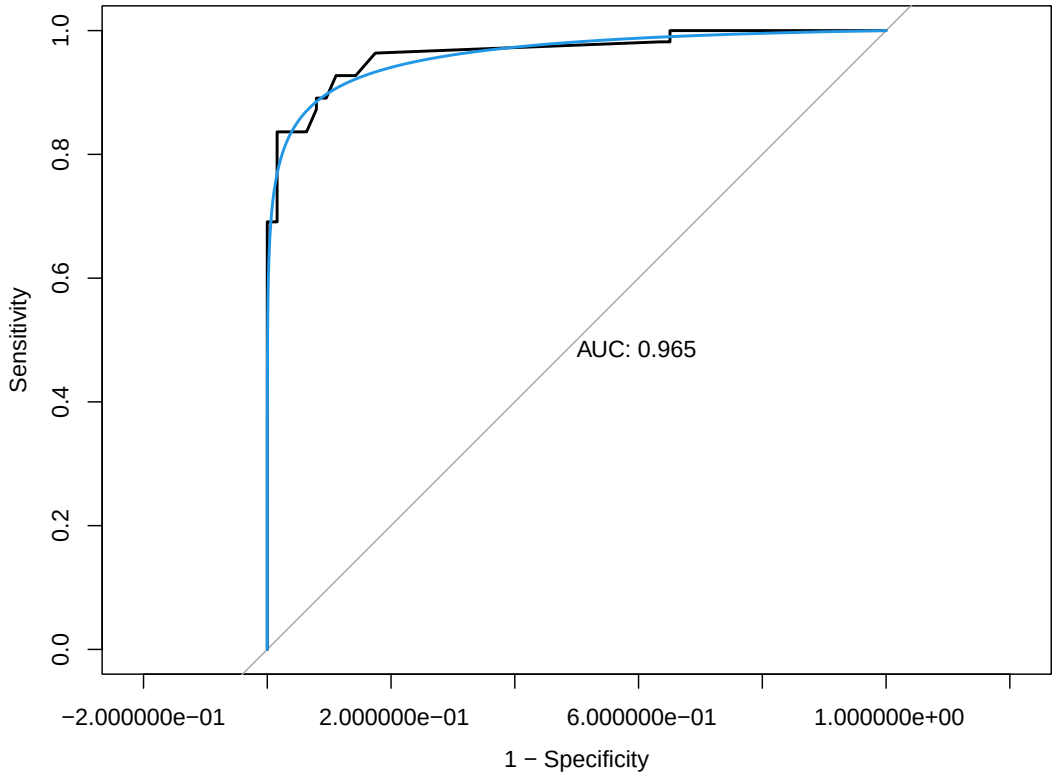
Since MARS model has the highest mean ROC, based on the resampling results from how our models perform on the training data, I would use MARS model to predict the response variable `mpg_cat`.

```
# roc
roc.mars <- roc(test$mpg_cat, mars.pred.prob)
```

```
## Setting levels: control = high, case = low
```

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

```
plot(roc.mars, legacy.axes = TRUE,  print.auc = TRUE)
plot(smooth(roc.mars), col = 4, add = TRUE)
```




```
# Compute the misclassification error rates:
round((1 - (matrix$overall["Accuracy"])),4)
```

```
## Accuracy
##   0.1017
```

From the plot above, the AUC for MARS model is 0.965.The misclassification error rate is about 10.17%.