DS2_HW3

Yixiao Sun

2024-03-22

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v infer        1.0.5     v tidyr        1.3.0
## v modeldata    1.2.0     v tune         1.1.2
## v parsnip      1.1.1     v workflows    1.1.3
## v purrr        1.0.2     v workflowsets 1.0.1
## v recipes      1.0.8     v yardstick    1.2.0
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()         masks scales::discard()
## x tidyr::expand()          masks Matrix::expand()
## x dplyr::filter()          masks stats::filter()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x tidyr::pack()            masks Matrix::pack()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x tidyr::unpack()          masks Matrix::unpack()
## x recipes::update()        masks Matrix::update(), stats::update()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(caTools)
data <- read.csv("~/Desktop/P8106 Data Science 2/ds2_hw3/auto.csv")
set.seed(123)
data <- data %>% mutate(mpg_cat = as.factor(mpg_cat), origin = as.factor(origin))
contrasts(data$mpg_cat)
```

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

```
ctrl1 <- trainControl(method = "cv", number = 10,summaryFunction = twoClassSummary,classProbs = TRUE)

data_split <- initial_split(data, prop = 0.7)
train_data <- training(data_split)
test_data <- testing(data_split)
```

Question 1

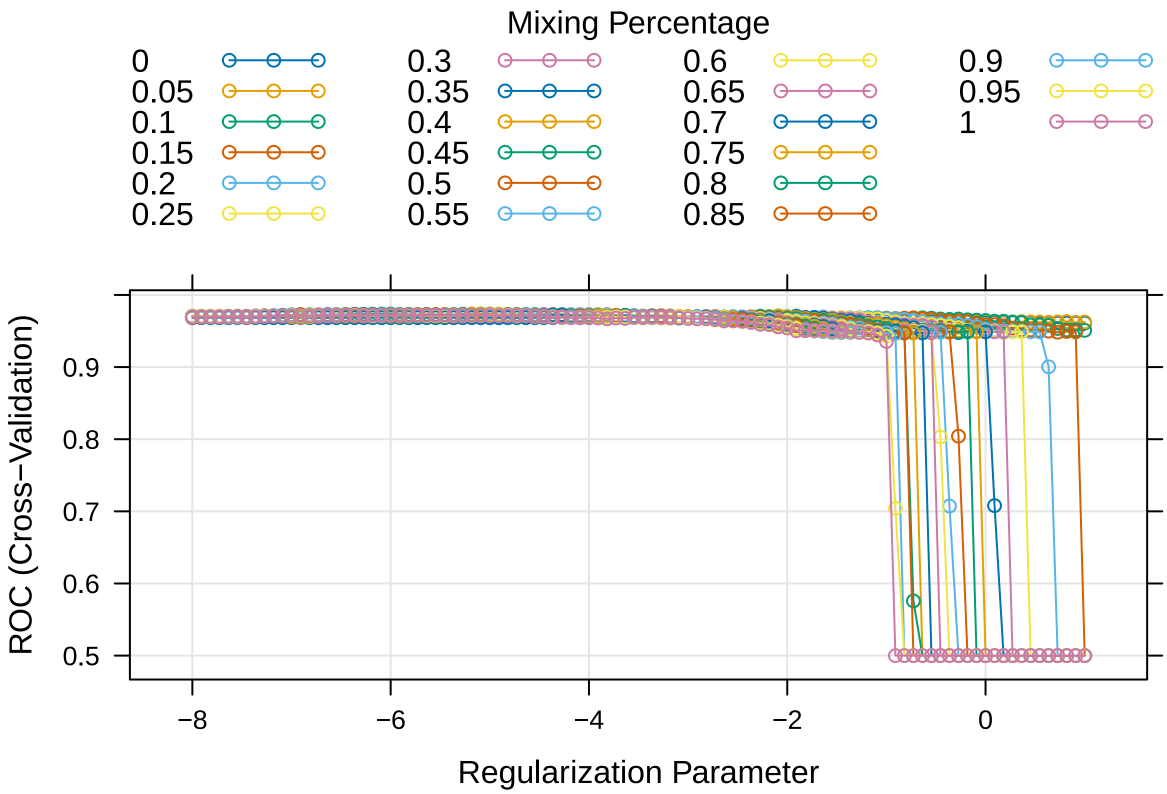
```
set.seed(123)
enet.fit <- train(mpg_cat ~ .,
                   data = train_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = seq(0,1,length = 21),
                                          lambda = exp(seq(1, -8, length = 100))),
                  metric = "ROC",
                  trControl = ctrl1)
enet.fit$bestTune
```

```
##      alpha      lambda
## 1523  0.75 0.002478752
```

```
print(coef(enet.fit$finalModel,enet.fit$bestTune$lambda))
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  19.780123608
## cylinders     .
## displacement  .
## horsepower    0.038622487
## weight        0.004179216
## acceleration -0.009703077
## year         -0.455243363
## origin2      -1.092582796
## origin3      -0.534957446
```

```
plot(enet.fit, xTrans = log)
```



Based on the enet model which can be helpful to identify the redundant variable, we identify variables cylinders and displacement as redundant variable.

Question 2

```
set.seed(123)
predict_prob <- predict(enet.fit, newdata = test_data, type = "prob")[,2]
predicted_class <- ifelse(predict_prob > 0.5, "low", "high")
CM<-confusionMatrix(data = as.factor(predicted_class), reference = test_data$mpg_cat, positive = "low")
```

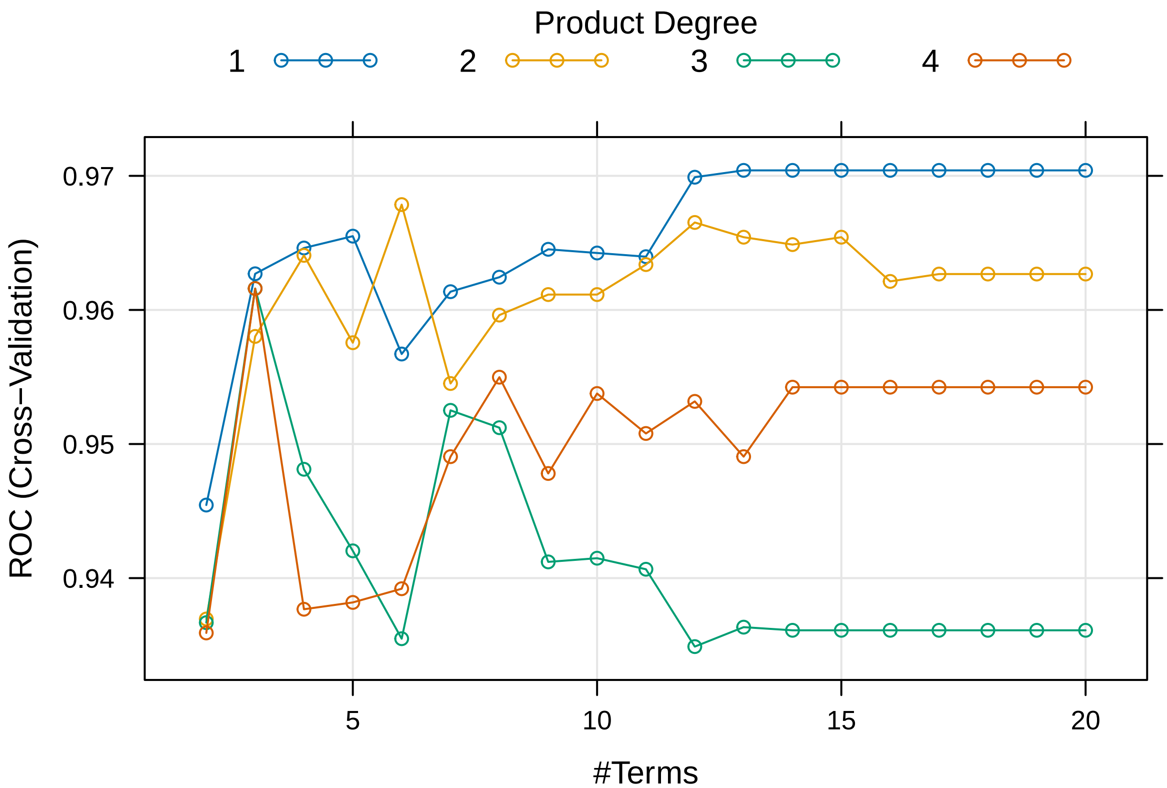
For the confusion matrix, we obtain an Accuracy of 0.8983, which indicates that our Elastic Net Model has an identify accuracy of 0.8983 .And we obtain a sensitivity of 0.9167, indicating that our Elatic Net Model classifies 91.67% of true "high" instances correctly. We get a specificity of 0.8793, indicating that our Elastic Net Model classifies 87.93% of true "low" instances correctly. Finally we get a Kappa of 0.7964, indicating that our model has a good performance in this classification.

Question 3

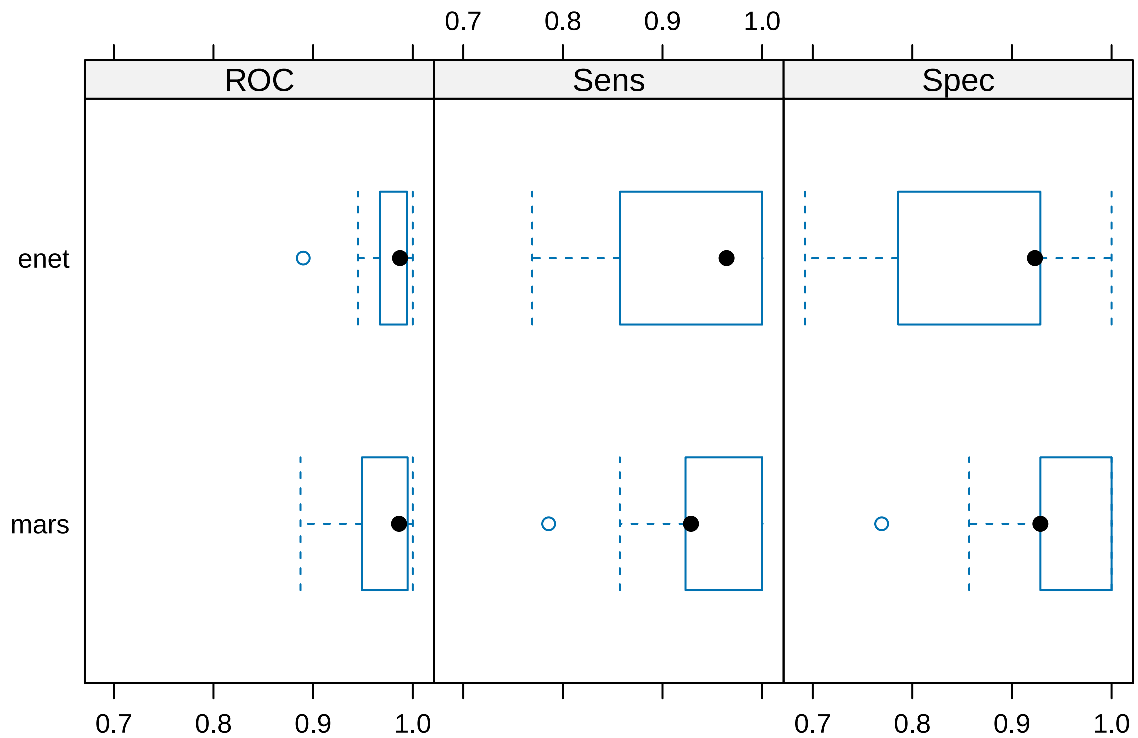
```
set.seed(123)
mars.fit <- train(mpg_cat ~.,
                    data = train_data,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                  metric = "ROC",
                  trControl = ctrl1)
```

```
## 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(mars.fit)
```



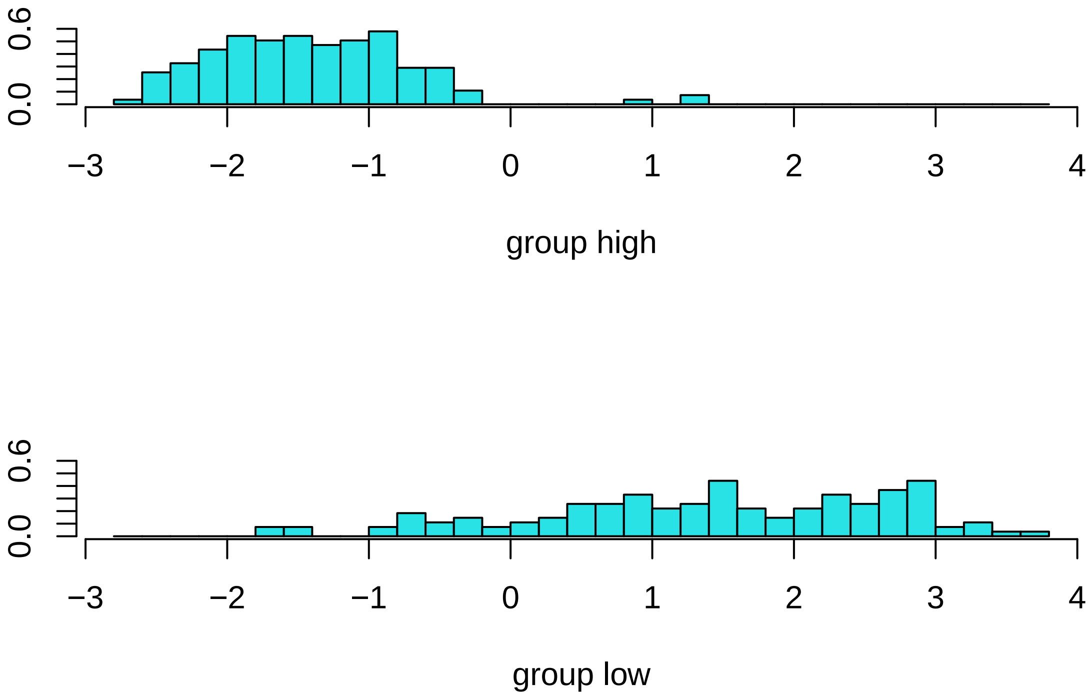

```
bwplot(resamples(list(enet = enet.fit, mars = mars.fit)), matrix = "ROC")
```



After creating the mars model, we conclude that the Elastic Net Model is still better for the prediction performance for producing a higher ROC value.

Question 4

```
set.seed(123)
library(MASS)
lda.fit <- train(mpg_cat ~ .,
                   data = train_data,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl1)
lda<-lda(mpg_cat ~ ., data = train_data)
plot(lda)
```




```r
lda.model = lda(mpg_cat~., data = train_data)
lda.model$scaling
```

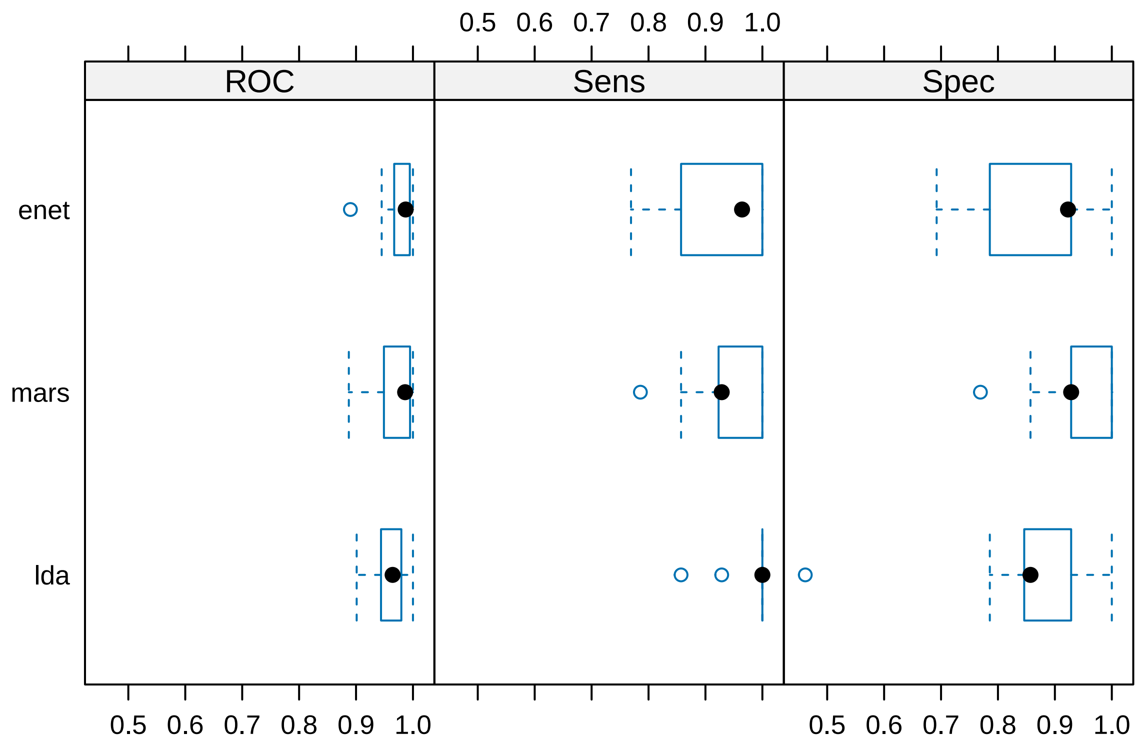
```
##                        LD1
## cylinders      0.429265756
## displacement -0.001426629
## horsepower   -0.008892402
## weight        0.001238912
## acceleration -0.015901121
## year         -0.142295080
## origin2      -0.532898297
## origin3      -0.430714726
```

```r
head(predict(lda.model)$x)
```

```
##          LD1
## 1 -1.1648422
## 2  0.9117385
## 3 -1.0172224
## 4 -0.3713083
## 5 -1.1847868
## 6  1.3295526
```

Question 5

```r
bwplot(resamples(list(enet = enet.fit, mars = mars.fit, lda = lda.fit)), matrix = "RMSE")
```




```r
resamp<-resamples(list(enet = enet.fit, mars = mars.fit, lda = lda.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, mars, lda
## Number of resamples: 10
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## enet 0.8901099 0.9697802 0.9872449 0.9733909 0.9933281    1    0
## mars 0.8873626 0.9534929 0.9862637 0.9704082 0.9947998    1    0
## lda  0.9010989 0.9441719 0.9642857 0.9590659 0.9770408    1    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## enet 0.7692308 0.8750000 0.9642857 0.9269231 1.0000000    1    0
## mars 0.7857143 0.9244505 0.9285714 0.9280220 0.9821429    1    0
```

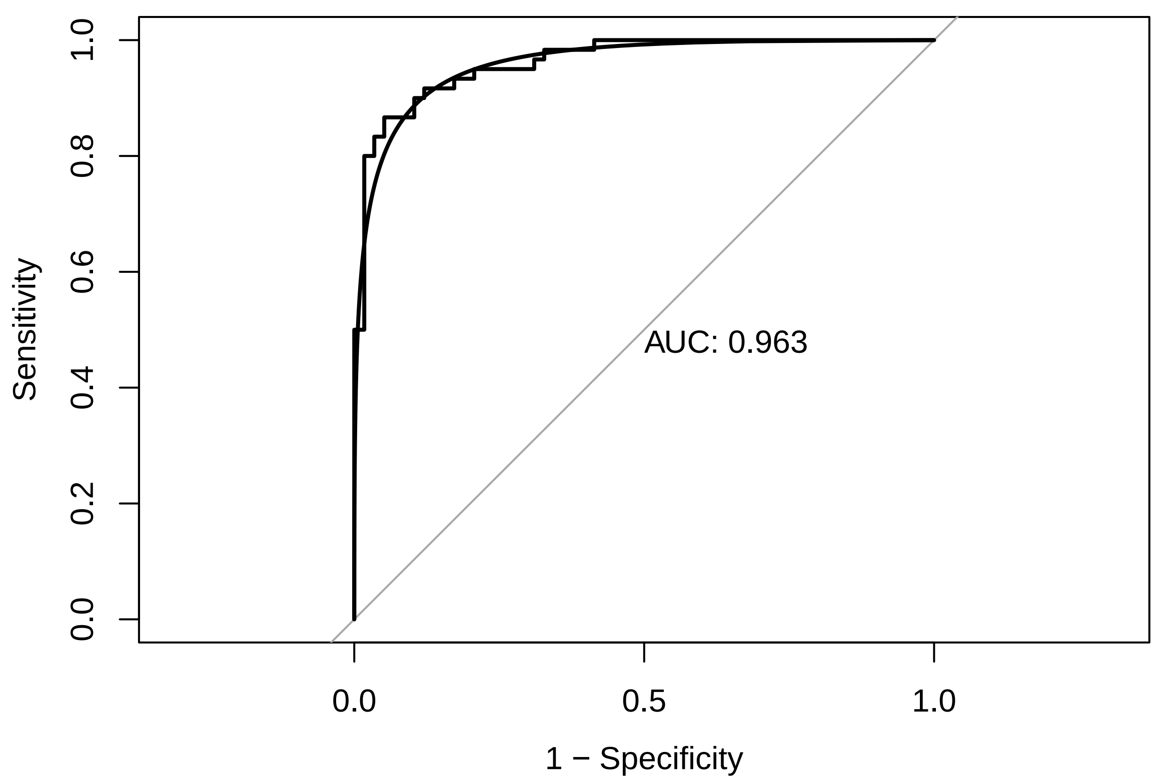
```
## lda  0.8571429 1.0000000 1.0000000 0.9785714 1.0000000    1    0
##
## Spec
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## enet 0.6923077 0.8035714 0.9230769 0.8752747 0.9285714    1    0
## mars 0.7692308 0.9285714 0.9285714 0.9340659 1.0000000    1    0
## lda  0.4615385 0.8489011 0.8571429 0.8450549 0.9107143    1    0
```

```r
library(pROC)
roc_response <- roc(response = test_data$mpg_cat, predictor = as.numeric(predict_prob))
```

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

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

```r
plot(roc_response,print.auc = T, legacy.axes = T)
plot(smooth(roc_response),add = T)
```




```r
auc(roc_response)
```

```
## Area under the curve: 0.9626
```

```r
error_rate <- 1- CM$overall
print(error_rate)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##     0.10169492     0.20356527     0.17090325     0.05366117     0.49152542
## AccuracyPValue  McnemarPValue
##     1.00000000     0.22717001
```

Based on the comparison, we choose Elastic Net Model to predict the response variable, The AUC value is 0.9626, and the calculated misclassification error rate is 0.10169492.