p8106_hw3_qz2266

Qing Zhou

2023-03-23

Data import

In this problem, 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.

```
auto = read.csv("data/auto.csv") %>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    mpg_cat = fct_relevel(mpg_cat, c("low", "high")),
    year = factor(year),
    origin = as.factor(origin))
```

Note: Here I mutated `year` as a factor variable, since I don't assume there's a linear relationship between model year and gas millage.

Data spliting

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

```
set.seed(1)

# partition
trainRows <- createDataPartition(y = auto$mpg_cat, p = 0.7,list = FALSE)
train_data = auto[trainRows, ]
test_data = auto[-trainRows, ]

x = model.matrix(mpg_cat ~ ., train_data)[,-1]
y = train_data$mpg_cat
x_test = model.matrix(mpg_cat ~ ., test_data)[,-1]
y_test = test_data$mpg_cat
```

(a) Logistic regression model

1. Perform a logistic regression using the training data:

```
set.seed(1)
contrasts(auto$mpg_cat)
```

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

```r
# model fitting
logit_fit <- glm(mpg_cat ~ .,
               data = auto,
               subset = trainRows,
               family = binomial(link = "logit"))
summary(logit_fit)
```

```
##
## Call:
## glm(formula = mpg_cat ~ ., family = binomial(link = "logit"),
##     data = auto, subset = trainRows)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.11544  -0.08650  -0.00003   0.06287   3.11968
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.456e+01  5.404e+00   2.694  0.00706 **
## cylinders    -2.843e-01  6.741e-01  -0.422  0.67320
## displacement  2.122e-02  1.781e-02   1.191  0.23359
## horsepower   -2.117e-02  3.353e-02  -0.631  0.52782
## weight       -7.902e-03  2.031e-03  -3.890  0.00010 ***
## acceleration  3.038e-01  2.112e-01   1.439  0.15027
## year71       -5.227e-02  1.955e+00  -0.027  0.97867
## year72       -1.796e+00  1.387e+00  -1.295  0.19539
## year73       -1.812e+00  1.593e+00  -1.138  0.25527
## year74        1.473e+00  1.792e+00   0.822  0.41127
## year75        1.936e+00  1.410e+00   1.373  0.16991
## year76        2.179e+00  1.601e+00   1.361  0.17355
## year77        1.025e+00  1.751e+00   0.585  0.55847
## year78        1.435e+00  1.605e+00   0.894  0.37117
## year79        4.929e+00  1.716e+00   2.872  0.00408 **
## year80        5.072e+00  1.876e+00   2.704  0.00686 **
## year81        5.212e+00  1.709e+00   3.049  0.00229 **
## year82        2.113e+01  1.944e+03   0.011  0.99133
## origin2       2.215e+00  1.051e+00   2.107  0.03508 *
## origin3       2.720e+00  1.164e+00   2.338  0.01939 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.617  on 275  degrees of freedom
## Residual deviance:  82.751  on 256  degrees of freedom
## AIC: 122.75
##
## Number of Fisher Scoring iterations: 18
```

Predictors in the logistic model that are statistically significant at the 5% level of significance are listed as below: - `weight` (vehicle weight (lbs.)) - `year79` (model year 79) - `year80` (model year 80) - `year81` (model year 81) - `origin2` (European origin) - `origin3` (Japanese origin).

2. Set the probability threshold to 0.5 to determine class labels and compute the confusion matrix using the test data:

```
# forecast new observations in the testing set
test.pred.prob <- predict(logit_fit, newdata = test_data,
                          type = "response")
test.pred <- rep("low", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] = "high"

#confusion matrix
logit_cm = confusionMatrix(data = factor(test.pred, levels = c("low", "high")),
                                         reference = y_test,
                                         positive = "high")
logit_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   50    4
##       high   8   54
##
##                Accuracy : 0.8966
##                  95% CI : (0.8263, 0.9454)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7931
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.9310
##             Specificity : 0.8621
##          Pos Pred Value : 0.8710
##          Neg Pred Value : 0.9259
##              Prevalence : 0.5000
##          Detection Rate : 0.4655
##    Detection Prevalence : 0.5345
##       Balanced Accuracy : 0.8966
##
##        'Positive' Class : high
##
```

```
# extract overall accuracy of the model
logit_cm$byClass["Balanced Accuracy"]
```

```
## Balanced Accuracy
##         0.8965517
```

- The confusion matrix shows the number of correct and incorrect predictions per class. It helps in understanding the classes that are being confused by model as other class. The rows refer to predicted class, while the columns indicate the actual class. Therefore, the number of true lows, true highs, false lows, and false highs are 50, 54, 4, and 8, respectively. Here true lows means the model has 50 correct predictions as having low gas mileage, and true high means the model has 54 correct predictions as having high gas mileage. False high indicate there are 8 count of number of low gas mileage that were misclassified as high gas mileage, and false low indicate there are 4 count of number of high gas mileage that were misclassified as low gas mileage.

- The overall prediction accuracy could be calculated as $\frac{50+54}{50+8+4+54} = 0.897$, with 95% CI (0.8263, 0.9454). Balance accuracy is also 0.897. The No Information Rate (NIR) is 0.5. The no information rate is 0.5, which means if we made the same class prediction for all observations given no information, our model would be 50% accurate, which is not very ideal. P-value is $<$ 2e-16. So the accuracy is significantly different from no information rate.

- The kappa coefficient is 0.7931, which measures the agreement between classification and truth values. A kappa value of 1 represents perfect agreement, while a value of 0 represents no agreement. So here the kappa value indicates substantial agreement.

- Sensitivity(the percentage of true positives among the positive observations)is 0.931 while specificity(the percentage of true negatives among the negative observations) is 0.8621. Both are high. Positive Predictive Value (PPV) measures the ratio of true positive predictions considering all positive predictions. Negative Predictive Value (NPV) measures the ratio of true negative predictions considering all negative predictions.Here PPV is 0.871 while NPV is 0.9259.

3. We can also fit a logistic regression using caret:

```r
# logistic model using caret
set.seed(1)
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

model.glm = train(x = auto[trainRows, 1:7],
                  y = auto$mpg_cat[trainRows],
                  method = "glm",
                  metric = "ROC",
                  trControl = ctrl)
summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.11544  -0.08650  -0.00003   0.06287   3.11968
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.456e+01  5.404e+00   2.694  0.00706 **
## cylinders    -2.843e-01  6.741e-01  -0.422  0.67320
## displacement  2.122e-02  1.781e-02   1.191  0.23359
## horsepower   -2.117e-02  3.353e-02  -0.631  0.52782
```

```
## weight       -7.902e-03  2.031e-03  -3.890  0.00010 ***
## acceleration  3.038e-01  2.112e-01   1.439  0.15027
## year71       -5.227e-02  1.955e+00  -0.027  0.97867
## year72       -1.796e+00  1.387e+00  -1.295  0.19539
## year73       -1.812e+00  1.593e+00  -1.138  0.25527
## year74        1.473e+00  1.792e+00   0.822  0.41127
## year75        1.936e+00  1.410e+00   1.373  0.16991
## year76        2.179e+00  1.601e+00   1.361  0.17355
## year77        1.025e+00  1.751e+00   0.585  0.55847
## year78        1.435e+00  1.605e+00   0.894  0.37117
## year79        4.929e+00  1.716e+00   2.872  0.00408 **
## year80        5.072e+00  1.876e+00   2.704  0.00686 **
## year81        5.212e+00  1.709e+00   3.049  0.00229 **
## year82        2.113e+01  1.944e+03   0.011  0.99133
## origin2       2.215e+00  1.051e+00   2.107  0.03508 *
## origin3       2.720e+00  1.164e+00   2.338  0.01939 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382.617  on 275  degrees of freedom
## Residual deviance:  82.751  on 256  degrees of freedom
## AIC: 122.75
##
## Number of Fisher Scoring iterations: 18
```

(b). MARS model

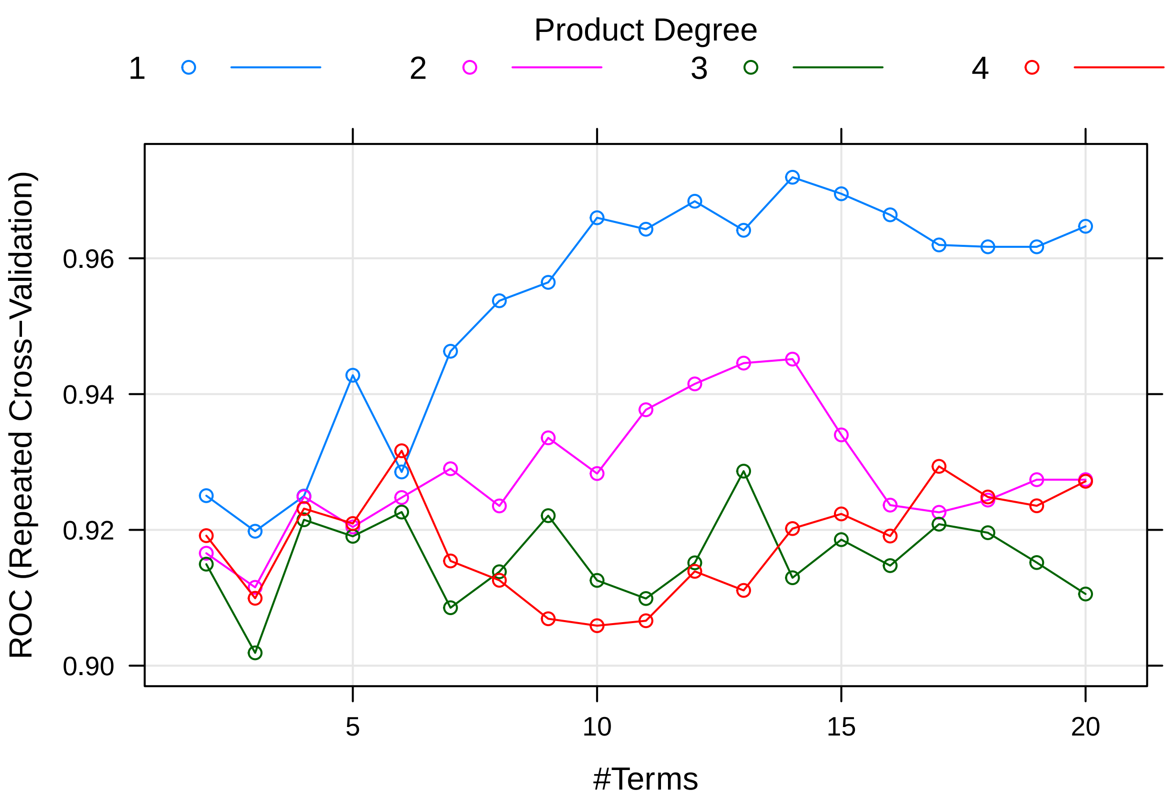
Train a multivariate adaptive regression spline (MARS) model using the training data:

```
set.seed(1)

ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)


mars_fit <- train(x,
                  y,
                  method = "earth",
                  tuneGrid = expand.grid(degree = 1:4,
                                         nprune = 2:20),
                  metric = "ROC",
                  trControl = ctrl)
summary(mars_fit)
```

```
## Call: earth(x=matrix[276,19], y=factor.object, keepxy=TRUE,
##             glm=list(family=function.object, maxit=100), degree=1, nprune=14)
##
## GLM coefficients
##                             high
## (Intercept)           24.7173171
```

```
## year79                  4.1236606
## year80                  6.6689496
## year81                  5.9373049
## year82                 18.8702838
## h(cylinders-4)        -12.2477760
## h(6-cylinders)         -9.5405565
## h(cylinders-6)         14.2195295
## h(displacement-119)    -3.3515335
## h(displacement-122)     3.7646000
## h(displacement-146)    -0.7640305
## h(displacement-151)     0.3398935
## h(weight-2914)         -0.0267003
## h(weight-3085)          0.0250052
##
## GLM (family binomial, link logit):
##  nulldev  df       dev  df   devratio     AIC iters converged
##  382.617 275   50.3427 262      0.868   78.34    18         1
##
## Earth selected 14 of 24 terms, and 7 of 19 predictors (nprune=14)
## Termination condition: Reached nk 39
## Importance: cylinders, displacement, weight, year80, year81, year79, ...
## Number of terms at each degree of interaction: 1 13 (additive model)
## Earth GCV 0.05285177    RSS 11.87269    GRSq 0.7901221    RSq 0.827932
```

```
plot(mars_fit)
```



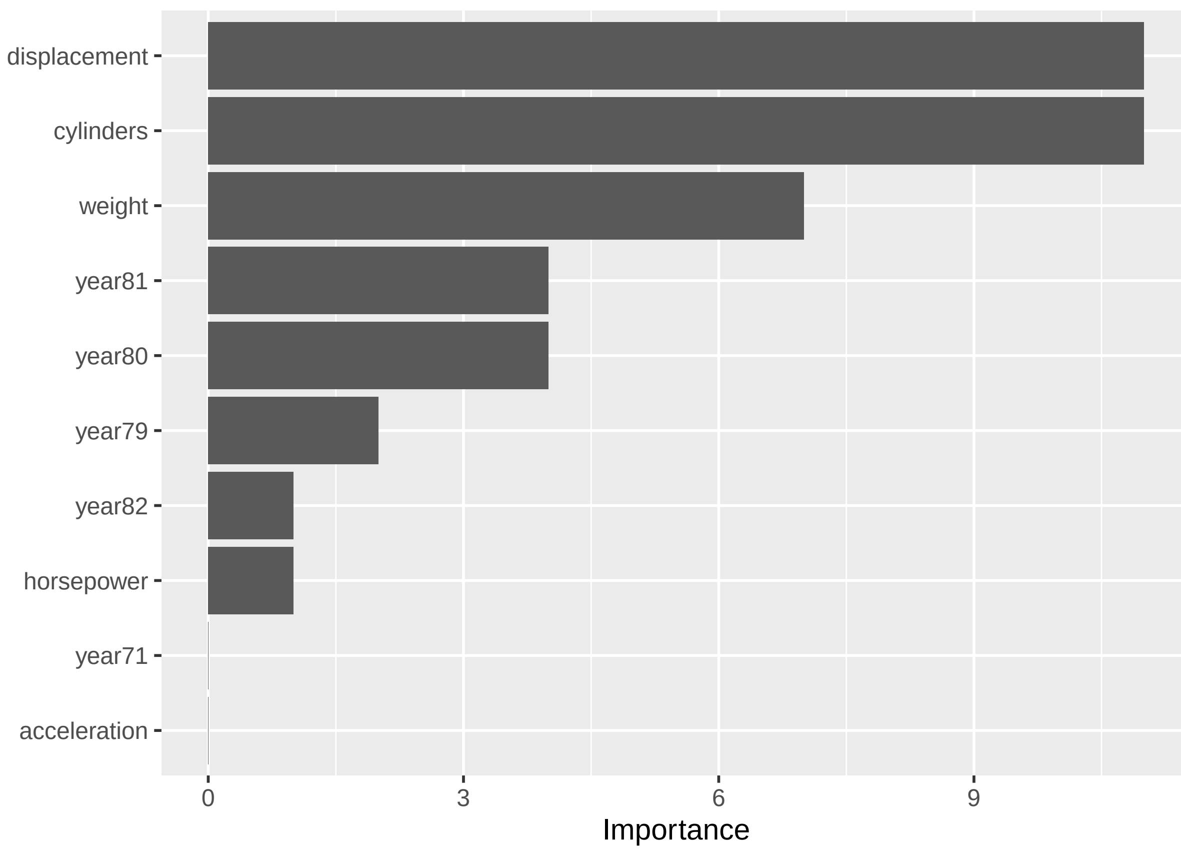
```r
# extract the optimal tuning parameters
kable(mars_fit$bestTune, "simple")
```

	nprune	degree
13	14	1

```r
coef(mars_fit$finalModel)
```

```
##          (Intercept) h(displacement-151)      h(weight-3085)      h(cylinders-6)
##          24.71731713          0.33989353          0.02500520         14.21952947
##       h(6-cylinders)      h(cylinders-4)      h(weight-2914) h(displacement-122)
##          -9.54055650        -12.24777603         -0.02670028          3.76460003
## h(displacement-119) h(displacement-146)              year81              year80
##          -3.35153350         -0.76403053          5.93730488          6.66894959
##               year79              year82
##           4.12366056         18.87028378
```

```r
vip(mars_fit$finalModel)
```



- From the first plot, we can see that when degree = 1 with 14 terms, the curve achieve its highest point. We got the same result from the extracted best tunes from the final MARS model.

- From the variable importance measurement plot (VIP), we found only `displacement`, `cylinders`, `weight`, `year79-82` and `horsepower` contributes to the model.
- The final MARS model has RSS = 11.873, R-squared = 0.8279, which is pretty high.

(c). LDA

1. Perform LDA using the training data:

```
## LDA fit
set.seed(1)
lda.fit = lda(mpg_cat~., data = auto,
               subset = trainRows)


lda.fit$scaling
```

```
##                        LD1
## cylinders    -0.426908230
## displacement  0.000918401
## horsepower    0.010708775
## weight       -0.001113534
## acceleration  0.051520119
## year71        0.149857720
## year72       -0.335837331
## year73       -0.247508917
## year74        0.563065711
## year75        0.071317413
## year76        0.170905092
## year77        0.210067050
## year78       -0.155093189
## year79        1.087107392
## year80        1.265894240
## year81        1.401233485
## year82        1.653369428
## origin2       0.615523222
## origin3       0.669705292
```

```
head(predict(lda.fit)$x)
```

```
##           LD1
## 1  -2.1457347
## 7  -2.1480040
## 10 -1.9925840
## 13 -2.2611263
## 14 -0.6300608
## 15  1.0937072
```

```
mean(predict(lda.fit)$x) ## seen as 0, means data is centered
```

```
## [1] 2.484551e-17
```

2. Plot the linear discriminants in LDA

```
plot(lda.fit)
```