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))
```

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), and with a No Information Rate (NIR) of 0.5.

- 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. 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.

(b). MARS model