P8106 Data Science II Homework 5

Sarah Forrest - sef2183

5/5/2023

Contents

1. Predicting gas milage using the auto dataset

In this problem, we will apply support vector machines 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 is a binary variable that indicates whether the miles per gallon of a car is high or low. The predictors are `cylinders`, `displacement`, `horsepower`, `weight`, `acceleration`, `year`, and `origin`.

```
# read in data
auto = read.csv("data/auto.csv")
```

Set the `mpg_cat` variable to a factor.

```
auto$mpg_cat <- factor(auto$mpg_cat, c("high", "low"))
```

Create dummy variables for `origin` (1 = American, 2 = European, 3 = Japanese) so it will be treated as a character variable rather than a numeric variable. Two dummy variables are created: one for American cars (1 = American, 0 = otherwise) and one for European cars (1 = European, 0 = otherwise). Note that cars with Japanese origin have a value of 0 for both `origin_american` and `origin_european` dummy variables.

```
auto$origin_american <- ifelse(auto$origin == 1, 1, 0) # dummy variable for american origin (origin = 1
auto$origin_european <- ifelse(auto$origin == 2, 1, 0) # dummy variable for european origin (origin = 2

# remove original origin variable
auto$origin <- NULL
```

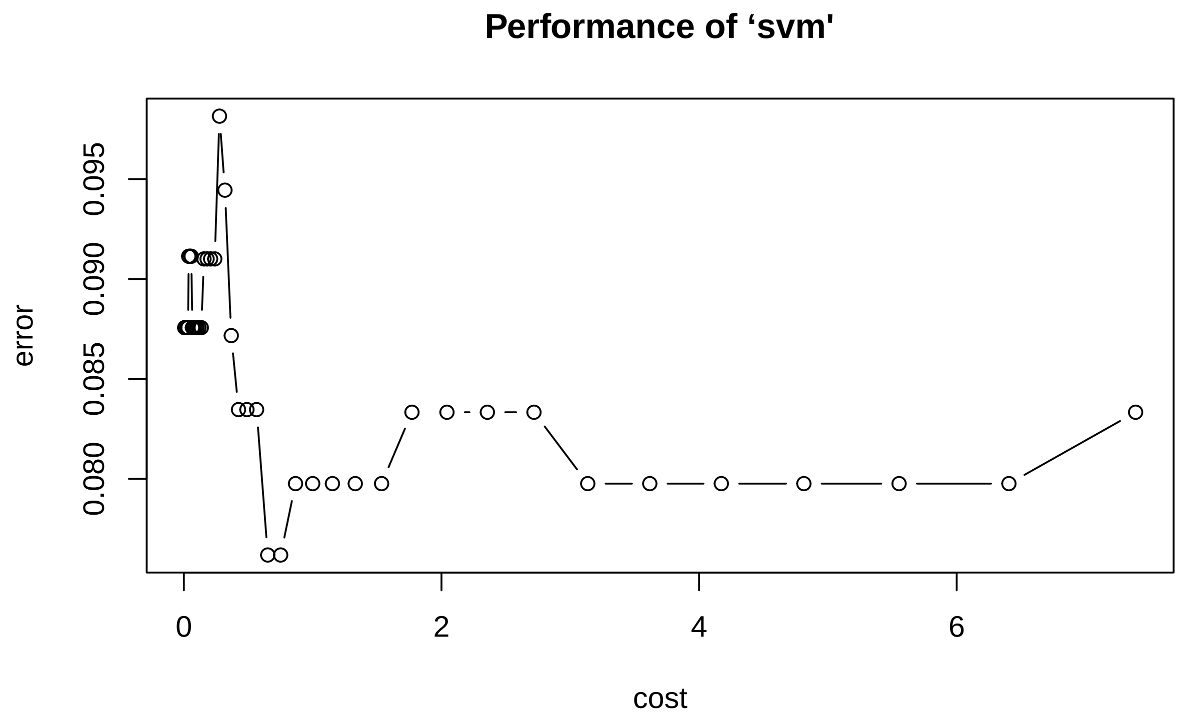
Split the dataset into two parts: training data (70%) and test data (30%)

```
set.seed(1) # for reproducibility

# specify rows of training data (70% of the dataset)
rowTrain <- createDataPartition(y = auto$mpg_cat,
                                p = .7,
                                list = F)

# create training dataset
auto_train <- auto[rowTrain, ]

# create test dataset
auto_test <- auto[-rowTrain, ]
```

(a) Fit a support vector classifier (linear kernel) to the training data.

Linear Boundary

```
set.seed(1)

# tuning parameter cost for linear boundary
linear.tune <- tune.svm(mpg_cat ~ . ,
                        data = auto_train,
                        kernel = "linear",
                        cost = exp(seq(-5,2,len = 50)), # specify a grid of cost parameters with a leng
                        scale = TRUE) # must scale predictors when running svm model
plot(linear.tune)
```




```
# summary(linear.tune)
linear.tune$best.parameters
##         cost
## 33 0.6514391
```

The optimal value for the cost tuning parameter is 0.6514391.

Fit optimal support vector classifier (linear kernel) using the best cost parameter

```
svm_model_lin <- linear.tune$best.model

# print the model summary
summary(svm_model_lin)
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto_train, cost = exp(seq(-5, 2,
##     len = 50)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.6514391
##
## Number of Support Vectors:  64
##
##  ( 31 33 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

Training error rate

```
# predict the support vector classifier (linear kernel) on the training data
train_pred_lin <- predict(svm_model_lin, newdata = auto_train)

# confusion matrix
confusionMatrix(data = train_pred_lin,
                reference = auto_train$mpg_cat)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  132  13
##       low     6 125
##
##                Accuracy : 0.9312
##                  95% CI : (0.8946, 0.958)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8623
##
##  Mcnemar's Test P-Value : 0.1687
##
##             Sensitivity : 0.9565
##             Specificity : 0.9058
##          Pos Pred Value : 0.9103
##          Neg Pred Value : 0.9542
##              Prevalence : 0.5000
##          Detection Rate : 0.4783
##    Detection Prevalence : 0.5254
##       Balanced Accuracy : 0.9312
##
##        'Positive' Class : high
##

# compute the training error rate
train_error_rate_lin <- mean(train_pred_lin != auto_train$mpg_cat)
train_error_rate_lin
## [1] 0.06884058
```

Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (N). Therefore, the training error rate = (6 + 13) / 276 = 0.0688. This is also equivalent to 1 minus the accuracy = 1 - 0.9312 = **0.0688**

Test error rate

```
# predict the support vector classifier (linear kernel) on the test data
test_pred_lin <- predict(svm_model_lin, newdata = auto_test)

# confusion matrix
confusionMatrix(data = test_pred_lin,
                reference = auto_test$mpg_cat)
## Confusion Matrix and Statistics
##
##             Reference
## Prediction high low
##       high   53   9
##       low     5  49
##
##                Accuracy : 0.8793
##                  95% CI : (0.8058, 0.9324)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7586
##
##  Mcnemar's Test P-Value : 0.4227
##
##             Sensitivity : 0.9138
##             Specificity : 0.8448
##          Pos Pred Value : 0.8548
##          Neg Pred Value : 0.9074
##              Prevalence : 0.5000
##          Detection Rate : 0.4569
##    Detection Prevalence : 0.5345
##       Balanced Accuracy : 0.8793
##
##        'Positive' Class : high
##

# compute the test error rate
test_error_rate_lin <- mean(test_pred_lin != auto_test$mpg_cat)
test_error_rate_lin
## [1] 0.1206897
```

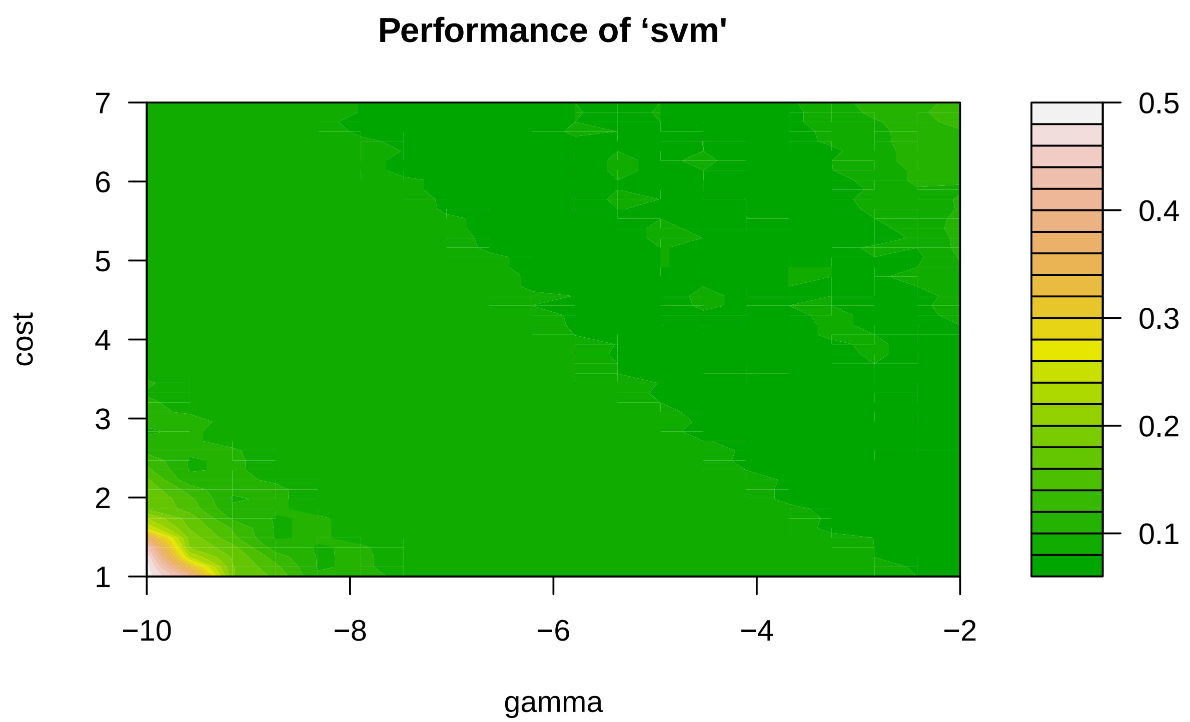
The test error rate = (5 + 9) / 116 = 0.1207. This is also equivalent to 1 minus the accuracy = 1 - 0.8793 = **0.1207**

(b) Fit a support vector machine with a radial kernel to the training data.

Non-Linear Boundary

```
set.seed(1)

# tuning parameter cost and gamma
radial.tune <- tune.svm(mpg_cat ~ . ,
```

```
                        data = auto_train,
                        kernel = "radial",
                        cost = exp(seq(1,7,len = 50)), # specify a grid of cost parameters with a lengt
                        gamma = exp(seq(-10,-2,len = 20))) # specify a grid of gamma parameters with a

plot(radial.tune, transform.y = log, transform.x = log,
     color.palette = terrain.colors)
```




```
# summary(radial.tune)
radial.tune$best.parameters
##          gamma     cost
## 717 0.03826736 197.4952
```

The optimal value for the cost tuning parameter is 197.4952 and the optimal value for the gamma tuning parameter is 0..03826736.

Fit optimal support vector classifier (radial kernel) using the best parameters

```
svm_model_rad <- radial.tune$best.model

# print the model summary
summary(svm_model_rad)
##
## Call:
## best.svm(x = mpg_cat ~ ., data = auto_train, gamma = exp(seq(-10,
##     -2, len = 20)), cost = exp(seq(1, 7, len = 50)), kernel = "radial")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  197.4952
##
## Number of Support Vectors:  50
##
##  ( 22 28 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

Training error rate

```
# predict the support vector machine (radial kernel) on the training data
train_pred_rad <- predict(svm_model_rad, newdata = auto_train)

# confusion matrix
confusionMatrix(data = train_pred_rad,
                reference = auto_train$mpg_cat)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high  135   5
##       low     3 133
##
##                Accuracy : 0.971
##                  95% CI : (0.9437, 0.9874)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.942
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9783
##             Specificity : 0.9638
##          Pos Pred Value : 0.9643
##          Neg Pred Value : 0.9779
##              Prevalence : 0.5000
##          Detection Rate : 0.4891
##    Detection Prevalence : 0.5072
##       Balanced Accuracy : 0.9710
##
##        'Positive' Class : high
##
```

```r
# compute the training error rate
train_error_rate_rad <- mean(train_pred_rad != auto_train$mpg_cat)
train_error_rate_rad
## [1] 0.02898551
```

The training error rate = (3 + 5) / 276 = 0.029. This is also equivalent to 1 minus the accuracy = 1 - 0.971 = **0.029**

Test error rate

```r
# predict the support vector machine (radial kernel) on the test data
test_pred_rad <- predict(svm_model_rad, newdata = auto_test)

# confusion matrix
confusionMatrix(data = test_pred_rad,
                reference = auto_test$mpg_cat)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high low
##       high   49  10
##       low     9  48
##
##                Accuracy : 0.8362
##                  95% CI : (0.7561, 0.8984)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 4.315e-14
##
##                   Kappa : 0.6724
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8448
##             Specificity : 0.8276
##          Pos Pred Value : 0.8305
##          Neg Pred Value : 0.8421
##              Prevalence : 0.5000
##          Detection Rate : 0.4224
##    Detection Prevalence : 0.5086
##       Balanced Accuracy : 0.8362
##
##        'Positive' Class : high
##

# compute the test error rate
test_error_rate_rad <- mean(test_pred_rad != auto_test$mpg_cat)
test_error_rate_rad
## [1] 0.1637931
```

The test error rate = (9 + 10) / 116 = 0.1638. This is also equivalent to 1 minus the accuracy = 1 - 0.8362 = **0.1638**

2. Hierarchical clustering on the states using the USArrests dataset

In this problem, we perform hierarchical clustering on the states using the USArrests data in the ISLR package. For each of the 50 states in the United States, the dataset contains the number of arrests per 100,000 residents for each of three crimes: Assault, Murder, and Rape. The dataset also contains the percent of the population in each state living in urban areas, UrbanPop. The four variables will be used as features for clustering and are scaled.

```
# read in data
arrests <- data.frame(USArrests)
```

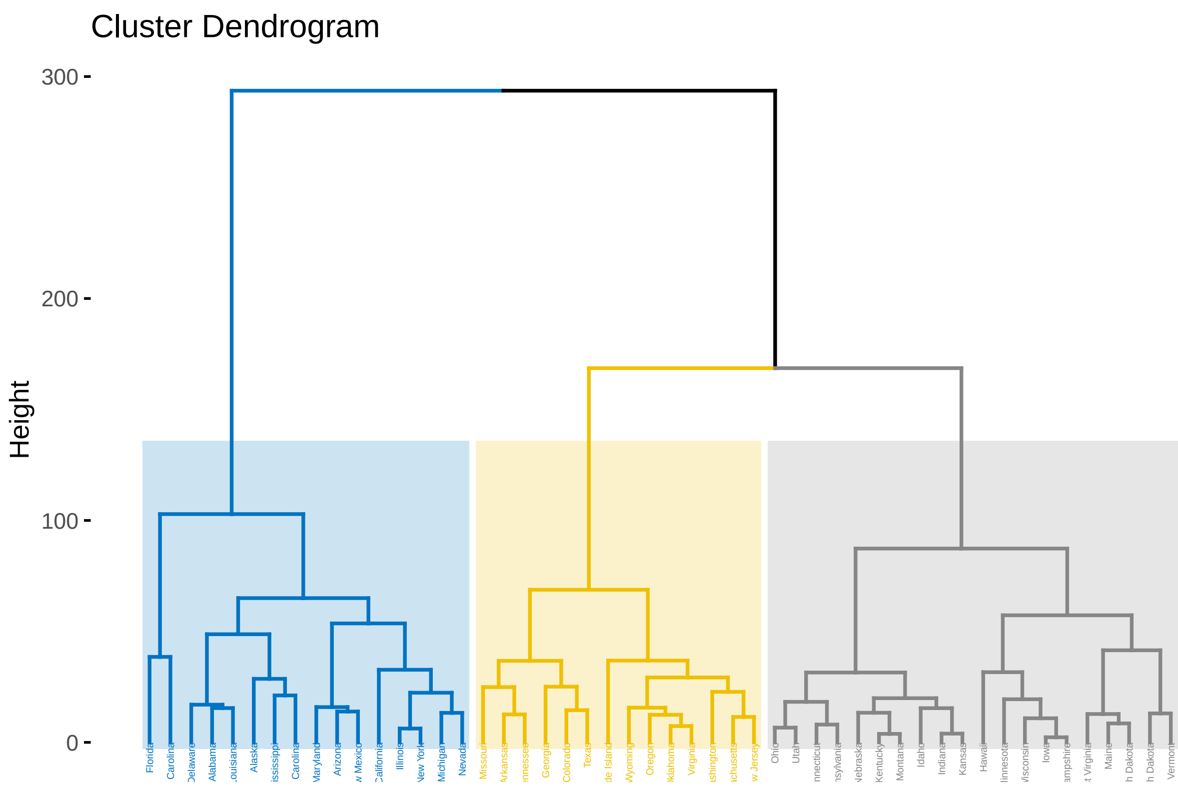
(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states. Cut the dendrogram at a height that results in three distinct clusters.

Complete linkage and Euclidean distance is specified.

```
hc.complete <- hclust(dist(arrests), method = "complete")
```

The function `fviz_dend()` is applied to visualize the dendrogram.

```
 set.seed(1)

fviz_dend(hc.complete, k = 3, # 3 clusters
          cex = 0.3,
          palette = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE, rect_fill = TRUE, rect_border = "jco",
          labels_track_height = 2.5)
```




```
# cut the dendrogram at a height that results in three distinct clusters
ind3.complete <- cutree(hc.complete, 3)
```

Cluster 1

```
arrests[ind3.complete == 1,]
##                Murder Assault UrbanPop Rape
## Alabama          13.2     236       58 21.2
## Alaska           10.0     263       48 44.5
## Arizona           8.1     294       80 31.0
## California        9.0     276       91 40.6
## Delaware          5.9     238       72 15.8
## Florida          15.4     335       80 31.9
## Illinois         10.4     249       83 24.0
## Louisiana        15.4     249       66 22.2
## Maryland         11.3     300       67 27.8
## Michigan         12.1     255       74 35.1
## Mississippi      16.1     259       44 17.1
## Nevada           12.2     252       81 46.0
## New Mexico       11.4     285       70 32.1
## New York         11.1     254       86 26.1
## North Carolina   13.0     337       45 16.1
## South Carolina   14.4     279       48 22.5
```

The states in cluster 1 include: Alabama, Alaska, Arizona, California, Delaware, Florida, Illinois, Louisiana, Maryland, Michigan, Mississippi, Nevada, New Mexico, New York, North Carolina, and South Carolina.

Cluster 2

```
arrests[ind3.complete == 2,]
##                Murder Assault UrbanPop Rape
## Arkansas          8.8     190       50 19.5
## Colorado          7.9     204       78 38.7
## Georgia          17.4     211       60 25.8
## Massachusetts     4.4     149       85 16.3
## Missouri          9.0     178       70 28.2
## New Jersey        7.4     159       89 18.8
## Oklahoma          6.6     151       68 20.0
## Oregon            4.9     159       67 29.3
## Rhode Island      3.4     174       87  8.3
## Tennessee        13.2     188       59 26.9
## Texas            12.7     201       80 25.5
## Virginia          8.5     156       63 20.7
## Washington        4.0     145       73 26.2
## Wyoming           6.8     161       60 15.6
```

The states in cluster 2 include: Arkansas, Colorado, Georgia, Massachusetts, Missouri, New Jersey, Oklahoma, Oregon, Rhode Island, Tennessee, Texas, Virginia, Washington, Wyoming.

Cluster 3

```
arrests[ind3.complete == 3,]
##                Murder Assault UrbanPop Rape
## Connecticut       3.3     110       77 11.1
## Hawaii            5.3      46       83 20.2
## Idaho             2.6     120       54 14.2
## Indiana           7.2     113       65 21.0
## Iowa              2.2      56       57 11.3
## Kansas            6.0     115       66 18.0
## Kentucky          9.7     109       52 16.3
## Maine             2.1      83       51  7.8
## Minnesota         2.7      72       66 14.9
## Montana           6.0     109       53 16.4
## Nebraska          4.3     102       62 16.5
## New Hampshire     2.1      57       56  9.5
## North Dakota      0.8      45       44  7.3
## Ohio              7.3     120       75 21.4
## Pennsylvania      6.3     106       72 14.9
## South Dakota      3.8      86       45 12.8
## Utah              3.2     120       80 22.9
## Vermont           2.2      48       32 11.2
## West Virginia     5.7      81       39  9.3
## Wisconsin         2.6      53       66 10.8
```

The states in cluster 3 include: Connecticut, Hawaii, Idaho, Indiana, Iowa, Kansas, Kentucky, Maine, Minnesota, Missouri, Nebraska, New Hampshire, North Dakota, Ohio, Pennsylvania, South Dakota, Utah, Vermont, West Virginia and Wisconsin.

(b) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

By default, the `scale()` function scales the data to have a mean of 0 and a standard deviation of 1.

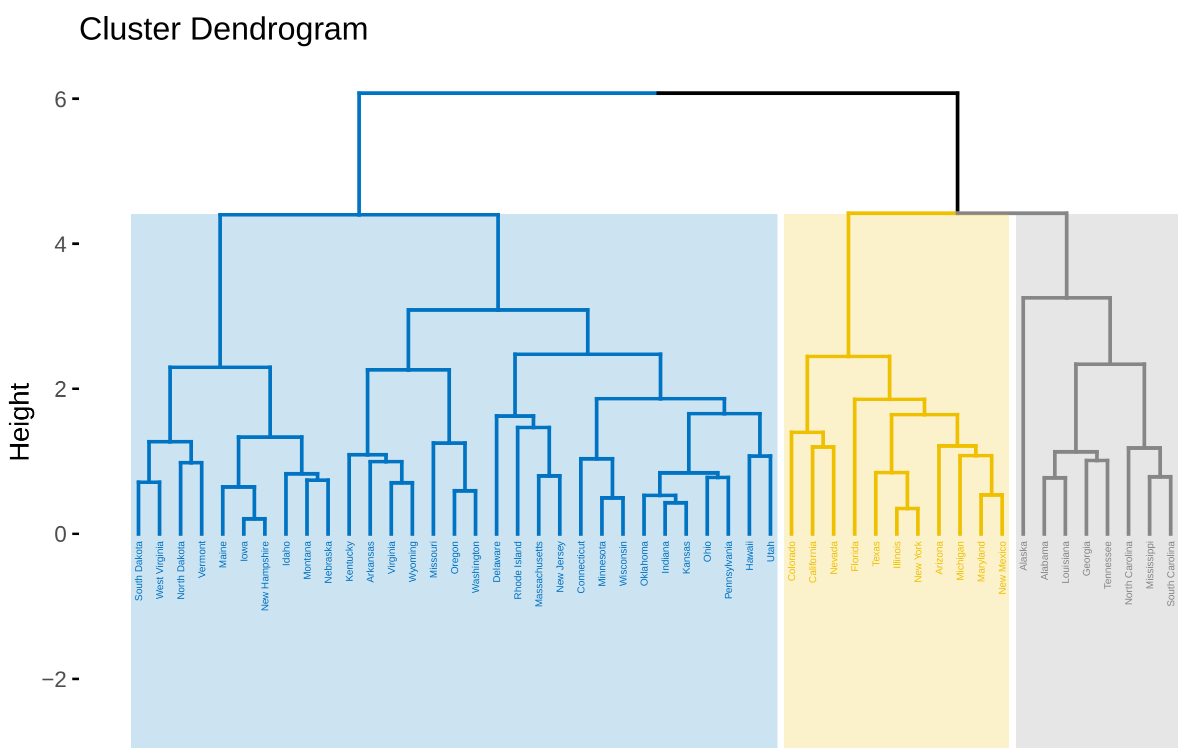
```r
# scale the variables
arrests_scaled <- scale(arrests)
```

Complete linkage and Euclidean distance is specified.

```r
hc.complete_scaled <- hclust(dist(arrests_scaled), method = "complete")
```

The function `fviz_dend()` is applied to visualize the dendrogram.

```r
set.seed(1)

fviz_dend(hc.complete_scaled, k = 3, # 3 clusters
          cex = 0.3,
          palette = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE, rect_fill = TRUE, rect_border = "jco",
          labels_track_height = 2.5)
```




```r
# cut the dendrogram at a height that results in three distinct clusters
ind3.complete_scaled <- cutree(hc.complete_scaled, 3)
```

Cluster 1

```r
arrests[ind3.complete_scaled == 1,]
##                Murder Assault UrbanPop Rape
## Alabama          13.2     236       58 21.2
## Alaska           10.0     263       48 44.5
## Georgia          17.4     211       60 25.8
## Louisiana        15.4     249       66 22.2
## Mississippi      16.1     259       44 17.1
## North Carolina   13.0     337       45 16.1
## South Carolina   14.4     279       48 22.5
## Tennessee        13.2     188       59 26.9
```

The states in cluster 1 include: Alabama, Alaska, Georgia, Louisiana, Mississippi, North Carolina, South Carolina, and Tennessee.

Cluster 2

```r
arrests[ind3.complete_scaled == 2,]
##            Murder Assault UrbanPop Rape
## Arizona       8.1     294       80 31.0
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
## Florida      15.4     335       80 31.9
## Illinois     10.4     249       83 24.0
## Maryland     11.3     300       67 27.8
## Michigan     12.1     255       74 35.1
## Nevada       12.2     252       81 46.0
## New Mexico   11.4     285       70 32.1
## New York     11.1     254       86 26.1
## Texas        12.7     201       80 25.5
```

The states in cluster 2 include: Arizona, California, Colorado, Florida, Illinois, Maryland, Michigan, Nevada, New Mexico, New York, and Texas.

Cluster 3

```r
arrests[ind3.complete_scaled == 3,]
##               Murder Assault UrbanPop Rape
## Arkansas         8.8     190       50 19.5
## Connecticut      3.3     110       77 11.1
## Delaware         5.9     238       72 15.8
## Hawaii           5.3      46       83 20.2
## Idaho            2.6     120       54 14.2
## Indiana          7.2     113       65 21.0
## Iowa             2.2      56       57 11.3
## Kansas           6.0     115       66 18.0
## Kentucky         9.7     109       52 16.3
## Maine            2.1      83       51  7.8
## Massachusetts    4.4     149       85 16.3
## Minnesota        2.7      72       66 14.9
## Missouri         9.0     178       70 28.2
## Montana          6.0     109       53 16.4
## Nebraska         4.3     102       62 16.5
## New Hampshire    2.1      57       56  9.5
## New Jersey       7.4     159       89 18.8
```

```
## North Dakota     0.8      45       44  7.3
## Ohio             7.3     120       75 21.4
## Oklahoma         6.6     151       68 20.0
## Oregon           4.9     159       67 29.3
## Pennsylvania     6.3     106       72 14.9
## Rhode Island     3.4     174       87  8.3
## South Dakota     3.8      86       45 12.8
## Utah             3.2     120       80 22.9
## Vermont          2.2      48       32 11.2
## Virginia         8.5     156       63 20.7
## Washington       4.0     145       73 26.2
## West Virginia    5.7      81       39  9.3
## Wisconsin        2.6      53       66 10.8
## Wyoming          6.8     161       60 15.6
```

The states in cluster 3 include: Arkansas, Connecticut, Delaware, Hawaii, Idaho, Indiana, Iowa, Kansas, Kentucky, Maine, Minnesota, Missouri, Nebraska, New Hampshire, New Jersey, Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South Dakota, Utah, Vermont, Virginia, Washington, West Virginia, Wisconsin, and Wyoming.

Does scaling the variables change the clustering results?

Scaling the variables changed the clustering results. Cluster 3 for the scaled dataset is much larger than in the non-scaled dataset, and all clusters contain different states. This can be due to a change in the distances between the observations. Clustering algorithms typically use distance measures to group similar observations together.

It's possible that scaling the variables may lead to more meaningful clusters, especially in scenarios where the variables are measured on different scales or units. In this dataset, the `UrbanPop` variable (percent of the population in each state living in urban areas) is at a different scale than the `Assault`, `Murder`, and `Rape` variables (number of arrests per 100,000 residents for each of the three crimes). If the variables are measured on different scales, then the clustering algorithm may give more weight to the variable with the larger scale. This can result in misleading or biased clustering results. This may be the reason why scaling the variables changed the clustering results. By standardizing the variables, we put them on a common scale, which can help to avoid these problems.

Should the variables be scaled before the inter-observation dissimilarities are computed?

Variables should be scaled before inter-observation dissimilarities are computed, especially if the variables are measured on different scales or units. Scaling helps to ensure that each variable contributes equally to the distances between the observations. Scaling the variables before computing inter-observation dissimilarities can help to ensure that the clustering analysis is not biased by differences in the scales of the variables. However, if the variables are already on the same scale, then scaling may not be necessary or required.