P8106 HW5

Brian Jo Hsuan Lee

4/28/2022

Load packages

```
library(tidyverse)
library(caret)
library(ISLR)
library(factoextra)
```

Problem 1: Auto Classifier using Support Vector Machine

Load and tidy the auto data, and split it into training and testing sets

```
data = read_csv("auto.csv", col_types = "fdiidfff") %>%
  mutate(
    cylinders = fct_relevel(cylinders, "3", "4", "5", "6", "8"),
    origin = fct_relevel(origin, "1", "2", "3")
  )

set.seed(2022)
rowTrain = createDataPartition(y = data$mpg_cat,
                               p = 0.7,
                               list = FALSE)
```

Set 10 fold cross validation to optimize tuning parameters

```
ctrl = trainControl(method = "cv")
```

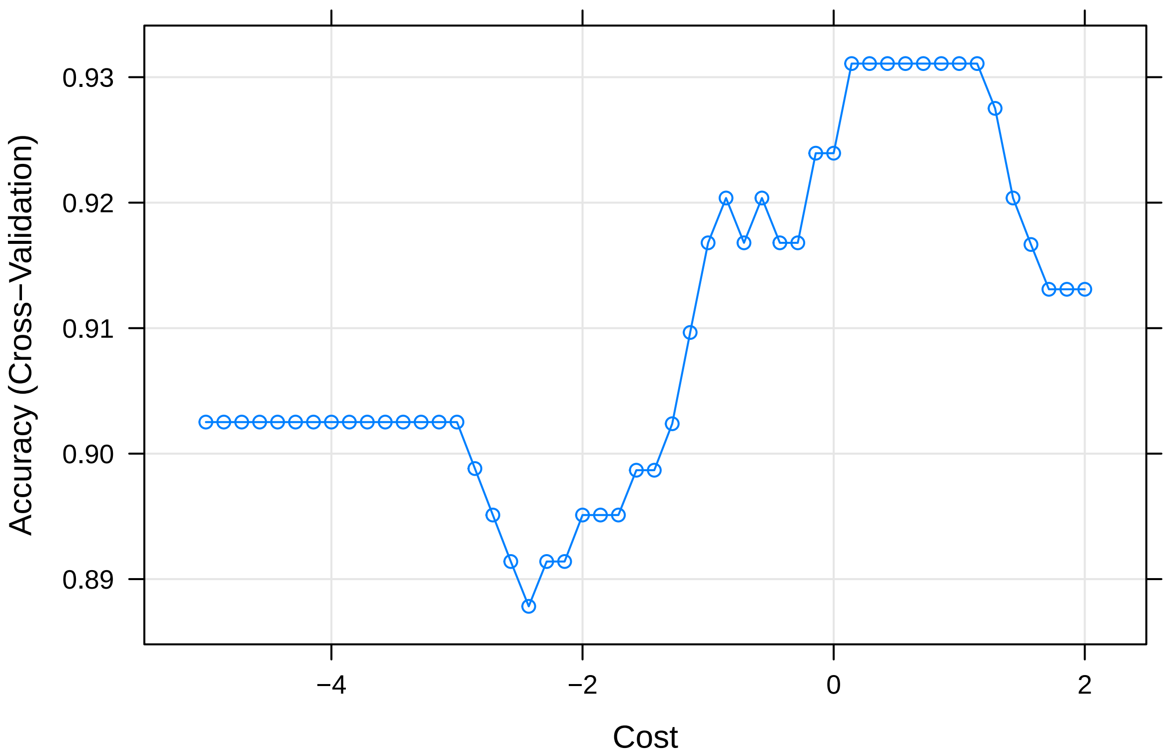
a) **Support Vector Classifier with Linear Kernel**

The following is a fit using a linear kernel implemented by the `kernlab` package

```
svm1.grid = data.frame(C = exp(seq(-5, 2, len = 50)))

# kernlab
set.seed(2022)
svml.fit = train(mpg_cat ~ . ,
                 data = data[rowTrain,],
                 method = "svmLinear",
                 # preProcess = c("center", "scale"),
                 tuneGrid = svm1.grid,
                 trControl = ctrl)

plot(svml.fit, highlight = TRUE, xTrans = log)
```



The optimum cost is 1.154

```
svml.fit$bestTune
```

```
##           C
## 37 1.153565
```

b) **Support Vector Classifier with Radial Kernel**

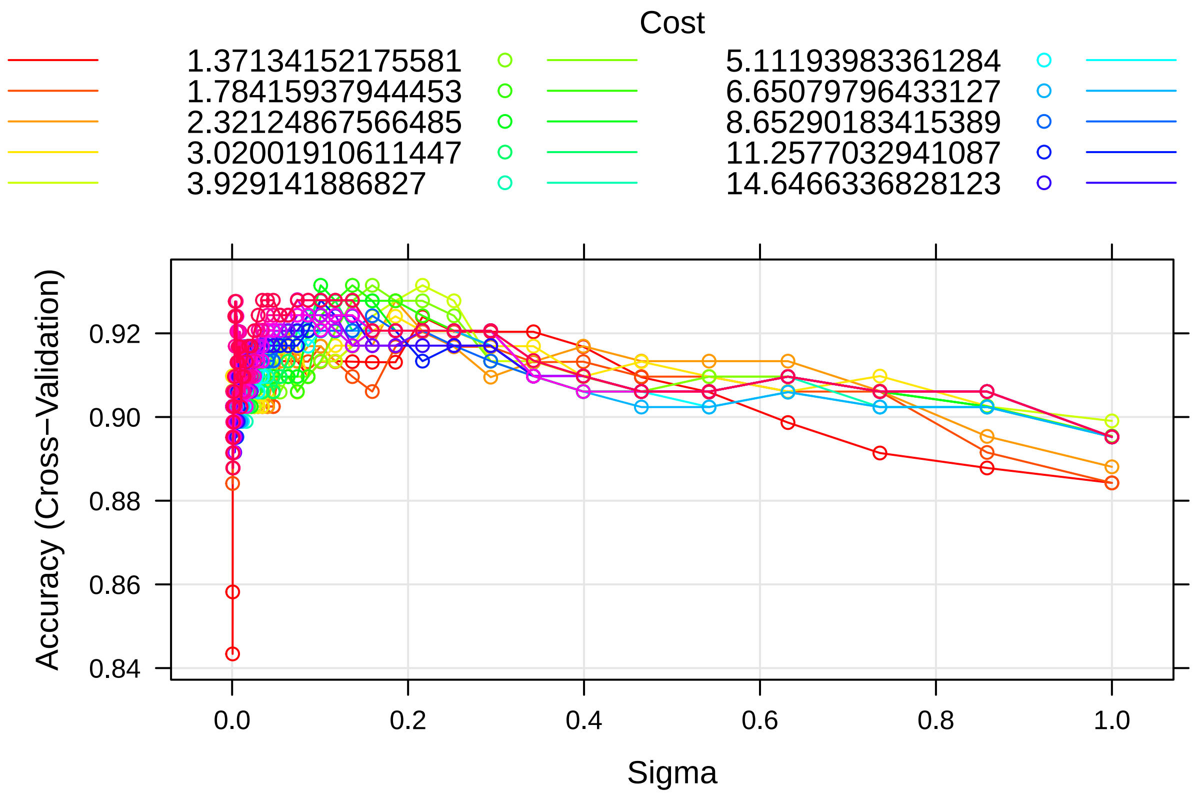
The following is a fit using a radial kernel, tuning over both cost and sigma

```
svmr.grid = expand.grid(C = exp(seq(-1, 4, len = 20)),
                        sigma = exp(seq(-7.5, 0, len = 50)))

set.seed(2022)
svmr.fit = train(mpg_cat ~ . ,
                 data = data,
                 subset = rowTrain,
                 method = "svmRadialSigma",
                 tuneGrid = svmr.grid,
                 trControl = ctrl)

myCol = rainbow(20)
myPar = list(superpose.symbol = list(col = myCol),
             superpose.line = list(col = myCol))

plot(svmr.fit, highlight = TRUE, par.settings = myPar)
```



The optimum cost and sigma are 1.054 and 0.216, respectively.

```
svmr.fit$bestTune
```

```
##         sigma        C
## 240 0.2164031 1.054041
```

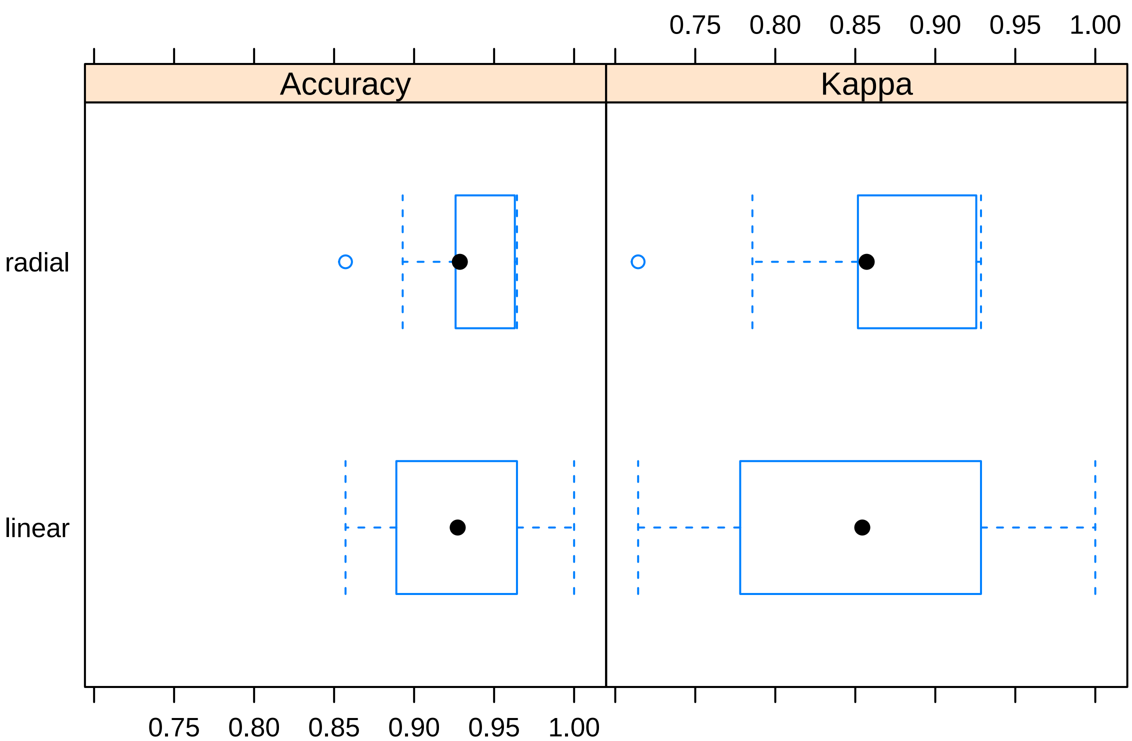
Training accuracy and Cohen's Kappa coefficient comparison between the 2 SVMs (linear vs radial kernals). The higher the better.

```
resamp = resamples(list(linear = svml.fit, radial = svmr.fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: linear, radial
## Number of resamples: 10
##
## Accuracy
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## linear 0.8571429 0.8898810 0.9272487 0.9310847 0.9642857 1.0000000    0
## radial 0.8571429 0.9265873 0.9285714 0.9314815 0.9629630 0.9642857    0
##
## Kappa
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## linear 0.7142857 0.7799902 0.8543956 0.8622098 0.9285714 1.0000000    0
## radial 0.7142857 0.8530220 0.8571429 0.8628915 0.9256198 0.9285714    0
```

```
bwplot(resamp)
```



The two SVMs have extremely similar mean and median training accuracies and κ coefficients. The classifier using radial kernel edged out the linear kernel variant ever so slightly with a mean accuracy of 0.9312 and a mean κ coefficient of 0.8624. Yet, the two are largely interchangeable in terms of classifying the training data at hand, and the decision should be based on other factors such as model interpretability.

Testing data performance

```r
pred.svml = predict(svml.fit, newdata = data[-rowTrain,])
pred.svmr = predict(svmr.fit, newdata = data[-rowTrain,])

confusionMatrix(data = pred.svml,
                reference = data$mpg_cat[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   51    2
##       high   7   56
##
##                Accuracy : 0.9224
##                  95% CI : (0.8578, 0.9639)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8448
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.8793
##             Specificity : 0.9655
##          Pos Pred Value : 0.9623
##          Neg Pred Value : 0.8889
##              Prevalence : 0.5000
##          Detection Rate : 0.4397
##    Detection Prevalence : 0.4569
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : low
##
```

```r
confusionMatrix(data = pred.svmr,
                reference = data$mpg_cat[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   52    3
##       high   6   55
##
##                Accuracy : 0.9224
##                  95% CI : (0.8578, 0.9639)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8448
##
##  Mcnemar's Test P-Value : 0.505
##
```

```
##             Sensitivity : 0.8966
##             Specificity : 0.9483
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.9016
##              Prevalence : 0.5000
##          Detection Rate : 0.4483
##    Detection Prevalence : 0.4741
##       Balanced Accuracy : 0.9224
##
##        'Positive' Class : low
##
```

The two SVMs also have extremely similar testing accuracies, both at 0.9224, confirming the expectation that they are virtually comparable.

Problem 2: US State Classifier using Hierarchical Clustering

Load the US Arrest dataset

```r
data2 = USArrests
```

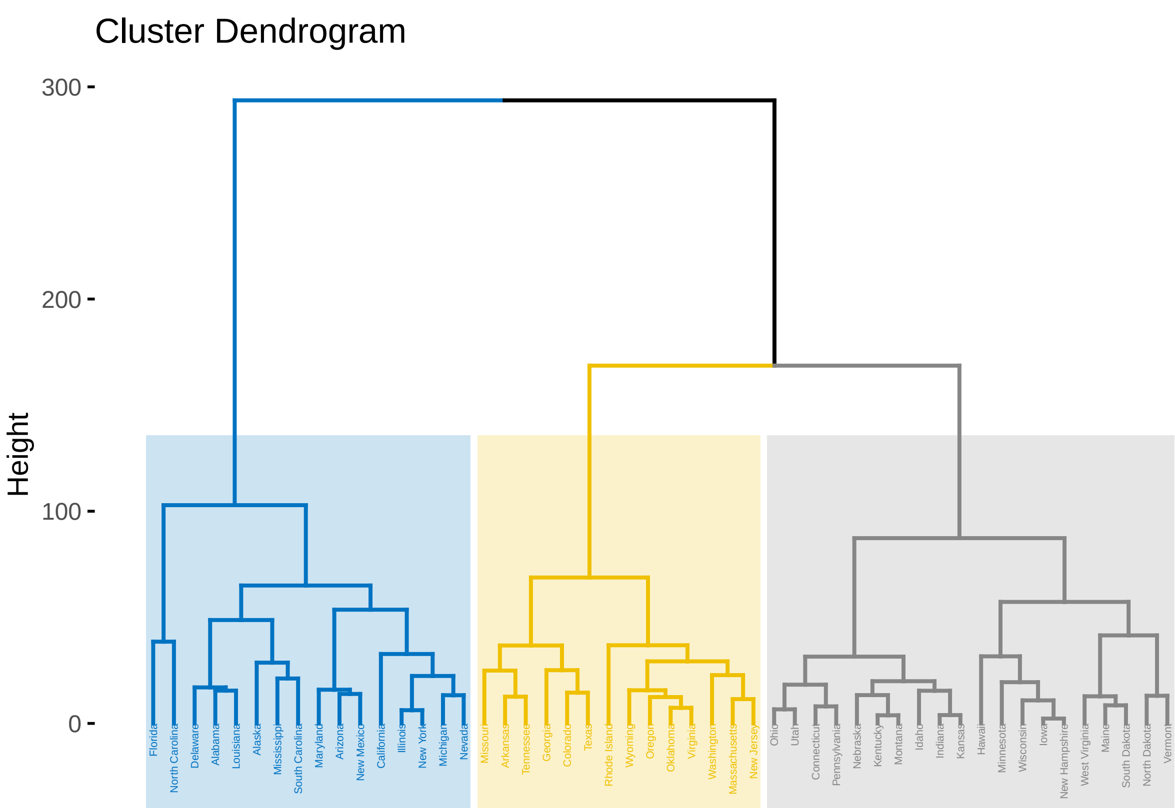
a) **Use hierarchical clustering with complete linkage and Euclidean distance for state clustering.**

Present the 3 cluster dendrogram

```r
# compute the 3 clusters of states
hc_complete = hclust(dist(data2), method = "complete")

# display
fviz_dend(hc_complete, k = 3,
          cex = 0.3,
          palette = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE, rect_fill = TRUE, rect_border = "jco",
          labels_track_height = 40)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



List the state belong in each cluster

```r
# compute the 3 clusters of states (basically the above dendrogram in another format)
state_clusters = cutree(hc_complete, 3)

# record the states in each cluster
cl1 = row.names(data2[state_clusters == 1,])
```

```r
cl2 = row.names(data2[state_clusters == 2,])
cl3 = row.names(data2[state_clusters == 3,])

# create a table to display cluster information
table_height = max(length(cl1), length(cl2), length(cl3))
cluster_table = data.frame(matrix(ncol = 3, nrow = table_height)) %>%
  mutate(across(c(`X1`:`X3`), ~replace_na(.x, " ")))
colnames(cluster_table) = c("Cluster 1", "Cluster 2", "Cluster 3")
cluster_table[1:length(cl1), 1] = cl1
cluster_table[1:length(cl2), 2] = cl2
cluster_table[1:length(cl3), 3] = cl3


# display
knitr::kable(cluster_table, "simple")
```

Cluster 1	Cluster 2	Cluster 3
Alabama	Arkansas	Connecticut
Alaska	Colorado	Hawaii
Arizona	Georgia	Idaho
California	Massachusetts	Indiana
Delaware	Missouri	Iowa
Florida	New Jersey	Kansas
Illinois	Oklahoma	Kentucky
Louisiana	Oregon	Maine
Maryland	Rhode Island	Minnesota
Michigan	Tennessee	Montana
Mississippi	Texas	Nebraska
Nevada	Virginia	New Hampshire
New Mexico	Washington	North Dakota
New York	Wyoming	Ohio
North Carolina		Pennsylvania
South Carolina		South Dakota
		Utah
		Vermont
		West Virginia
		Wisconsin

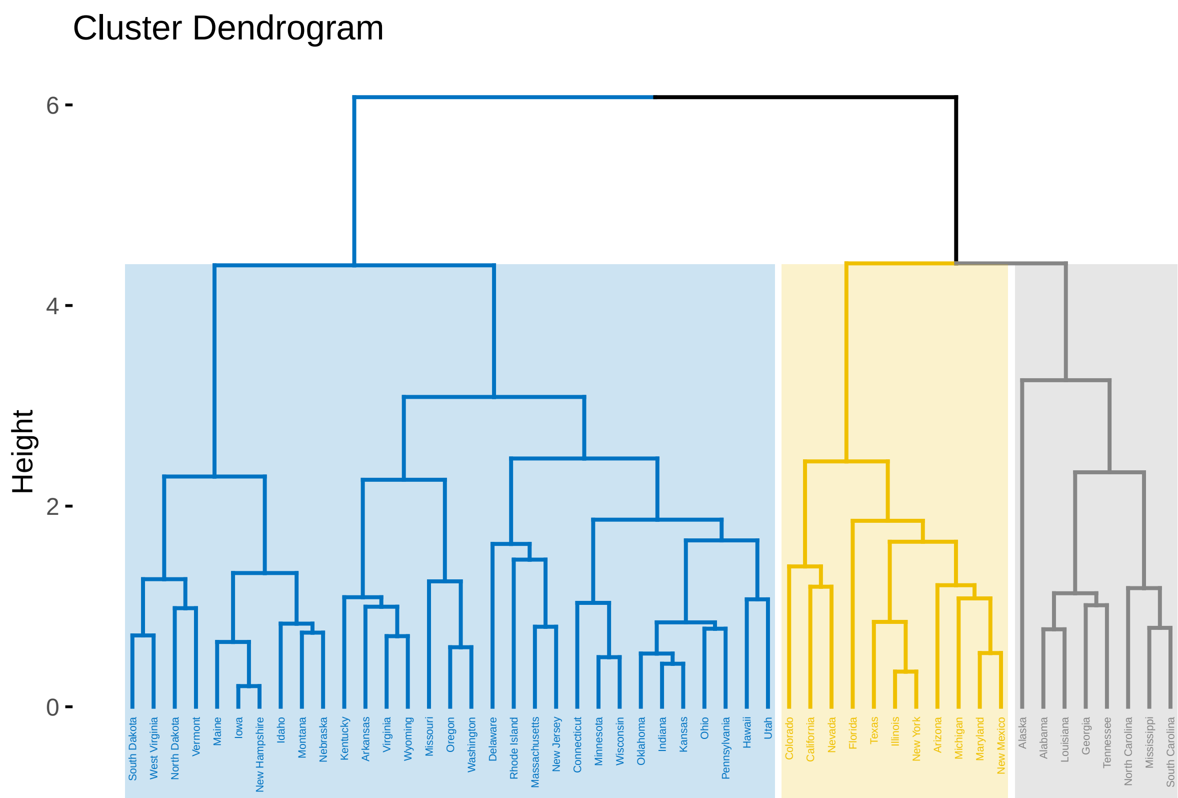
b) **Standardize all the covariates for state prediction, and repeat the hierarchical clustering**

```
# standardize (transform all within-variable variance to 1) the data
data2_std =
  USArrests %>%
  mutate_all(~(scale(.) %>% as.vector))

# compute and display the 3 clusters of states
hc_complete_std = hclust(dist(data2_std), method = "complete")

fviz_dend(hc_complete_std, k = 3,
          cex = 0.3,
          palette = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE, rect_fill = TRUE, rect_border = "jco",
          labels_track_height = 0.5)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```




```
state_clusters_std = cutree(hc_complete_std, 3)
cl1_std = row.names(data2[state_clusters_std == 1,])
cl2_std = row.names(data2[state_clusters_std == 2,])
cl3_std = row.names(data2[state_clusters_std == 3,])

table_height_std = max(length(cl1_std), length(cl2_std), length(cl3_std))
cluster_table_std = data.frame(matrix(ncol = 3, nrow = table_height_std)) %>%
  mutate(across(c(`X1`:`X3`), ~replace_na(.x, " ")))
colnames(cluster_table_std) = c("Cluster 1", "Cluster 2", "Cluster 3")
```

```r
cluster_table_std[1:length(cl1_std), 1] = cl1_std
cluster_table_std[1:length(cl2_std), 2] = cl2_std
cluster_table_std[1:length(cl3_std), 3] = cl3_std

knitr::kable(cluster_table_std, "simple")
```

Cluster 1	Cluster 2	Cluster 3
Alabama	Arizona	Arkansas
Alaska	California	Connecticut
Georgia	Colorado	Delaware
Louisiana	Florida	Hawaii
Mississippi	Illinois	Idaho
North Carolina	Maryland	Indiana
South Carolina	Michigan	Iowa
Tennessee	Nevada	Kansas
	New Mexico	Kentucky
	New York	Maine
	Texas	Massachusetts
		Minnesota
		Missouri
		Montana
		Nebraska
		New Hampshire
		New Jersey
		North Dakota
		Ohio
		Oklahoma
		Oregon
		Pennsylvania
		Rhode Island
		South Dakota
		Utah
		Vermont
		Virginia
		Washington
		West Virginia
		Wisconsin
		Wyoming

c) **Compare and interpret the non-standardized and standardized data**

The 2 hierarchical clusters are different, because the varying spread of each predictor resulted in different weights on the clustering computation. Features with larger variance had more influence on the eventual grouping. Therefore, data standardization is beneficial for accurate clustering if no predictors should be preferred over others. Murder rate, assault count, proportion of population in urban areas and rape rate are given no order of significance when it comes to state grouping. As such, their different data units (a mix of percentage and count data) should be corrected by standardization before calculating the inter-observation dissimilarities to achieve a more fitting result.