Models_Part3

Yuxuan Chen

5/8/2022

```
## Warning: package 'ggplot2' was built under R version 4.1.2

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2
```

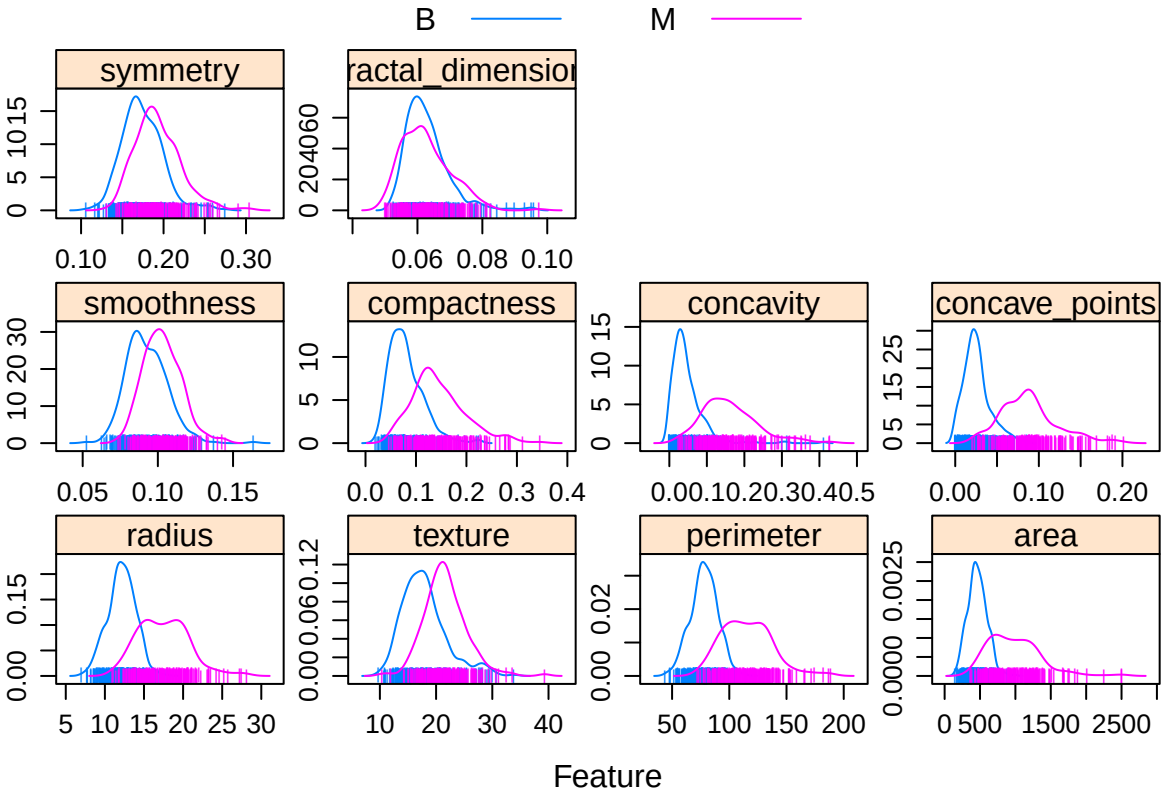
```
bc_df = read_csv("./data/breast-cancer.csv") %>%
  dplyr::select(-c(1, 33)) %>%
  janitor::clean_names() %>%
  # add extra row
  add_row(diagnosis = 'B', radius_mean = 7.76, texture_mean = 24.54,
          perimeter_mean = 47.92, area_mean = 181, smoothness_mean = 0.05263,
          compactness_mean = 0.04362, concavity_mean = 0,
          concave_points_mean = 0, symmetry_mean = 0.1587,
          fractal_dimension_mean = 0.05884, radius_se = 0.3857,
          texture_se = 1.428, perimeter_se = 2.548, area_se = 19.15,
          smoothness_se = 0.007189, compactness_se = 0.00466, concavity_se = 0,
          concave_points_se = 0, symmetry_se = 0.02676,
          fractal_dimension_se = 0.002783, radius_worst = 9.456,
          texture_worst = 30.37, perimeter_worst = 59.16, area_worst = 268.6,
          smoothness_worst = 0.08996, compactness_worst = 0.06444,
          concavity_worst = 0, concave_points_worst = 0,
          symmetry_worst = 0.2871, fractal_dimension_worst = 0.07039)
```

```
# partitioning data
set.seed(31)
indexTrain <- createDataPartition(bc_df$diagnosis, p = 0.7, list = FALSE)
trainData = bc_df[indexTrain, ]
testData = bc_df[-indexTrain,]
```

```
# very primitive EDA
bc_df_graph =
  bc_df %>%
  mutate(diagnosis = factor(recode(diagnosis, `1` = "M", `0` = "B"), level = c("B", "M")))

cancer_mean = bc_df_graph[, 2:11] %>% as_tibble()
```

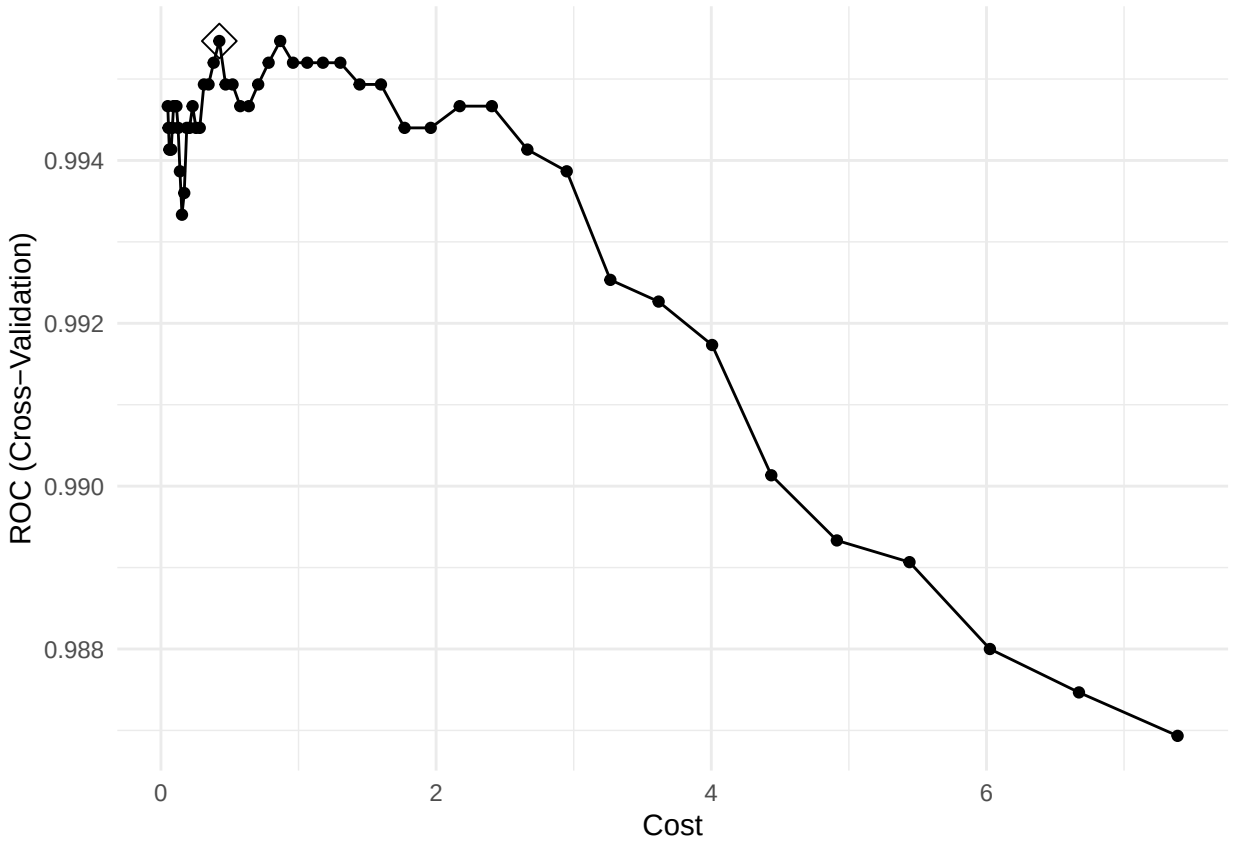
```
colnames(cancer_mean) = gsub("_mean", "", colnames(cancer_mean))
featurePlot(x = cancer_mean,
            y = bc_df_graph$diagnosis,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



SVM (linear and radial kernel)

a) Linear Kernel

```
ctrl <- trainControl(method = "cv", classProbs = TRUE, summaryFunction = twoClassSummary)
set.seed(31)
model.svml <- train(diagnosis~.,
                    data = trainData,
                    method = "svmLinear2",
                    preProcess = c("center", "scale"),
                    tuneGrid = data.frame(cost = exp(seq(-3,2,len = 50))),
                    trControl = ctrl)
ggplot(model.svml, highlight = TRUE)
```




```
model.svml$bestTune
```

```
##         cost
## 22 0.4243728
```

```
model.svml$finalModel
```

```
##
## Call:
## svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
##     probability = classProbs)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.4243728
##
## Number of Support Vectors:  32
```

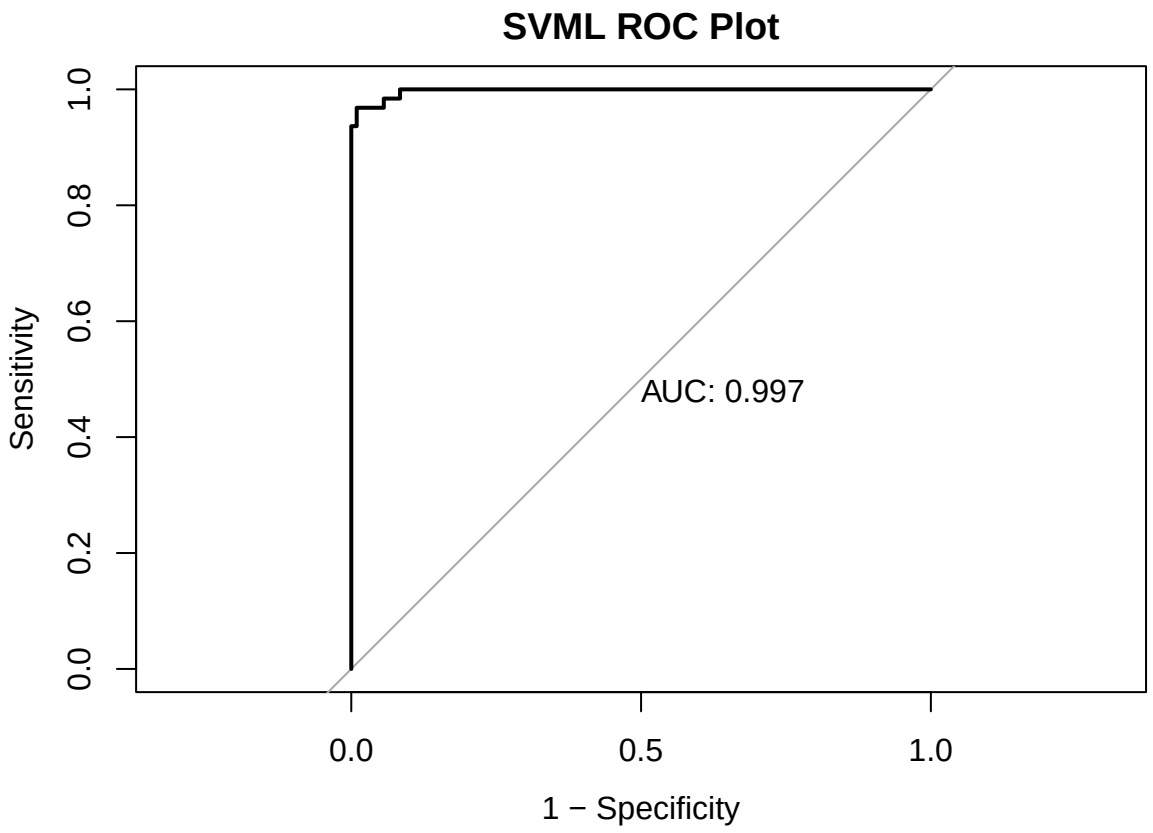
```
## Linear Kernel Training Error Rate
pred_svml_train = predict(model.svml)
train_error = mean(pred_svml_train != trainData$diagnosis)

## Linear Kernel Test Error Rate
```

```
pred_svml_test = predict(model.svml, newdata = testData, type = "raw")
test_error = mean(pred_svml_test != testData$diagnosis)
```

```
# SVML roc:
pred_svml = predict(model.svml, newdata = testData, type = "prob")[,1]
roc_svml = roc(testData$diagnosis, pred_svml,
               levels = c("B", "M"))

plot.roc(roc_svml, legacy.axes = TRUE, print.auc = TRUE,main = "SVML ROC Plot")
```

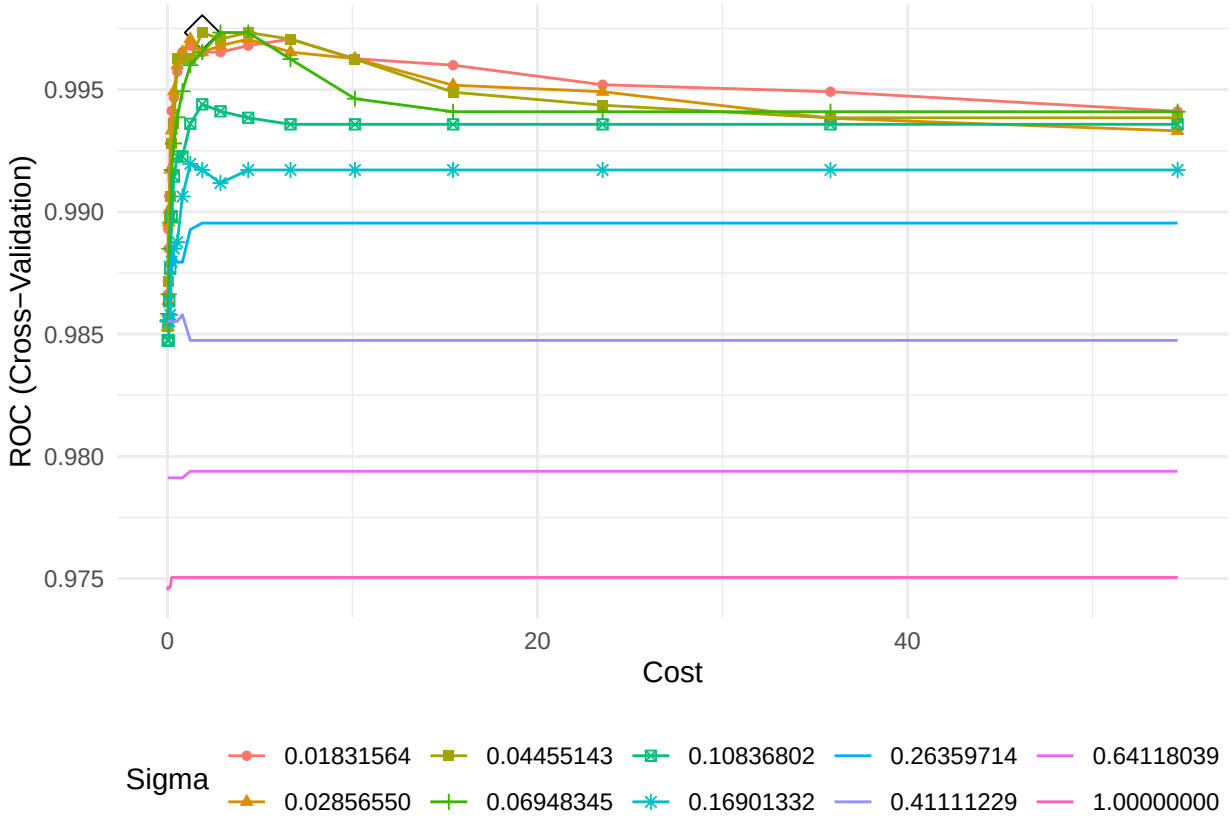


b) Radial Kernel

```
svmr.grid <- expand.grid(C = exp(seq(-4,4,len=20)),
                         sigma = exp(seq(-4,0,len=10)))
# tunes over both cost and sigma
set.seed(31)
model.svmr <- train(diagnosis ~ . ,
                    data = trainData,
                    method = "svmRadialSigma",
                    preProcess = c("center", "scale"),
                    tuneGrid = svmr.grid,
                    trControl = ctrl)
```

```
## maximum number of iterations reached 0.0001344337 0.0001316066maximum number of iterations reached 0
```

```
myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
ggplot(model.svmr, highlight = TRUE, par.settings = myPar)
```




```
model.svmr$bestTune
```

```
##          sigma        C
## 113 0.04455143 1.880578
```

```
model.svmr$finalModel
```

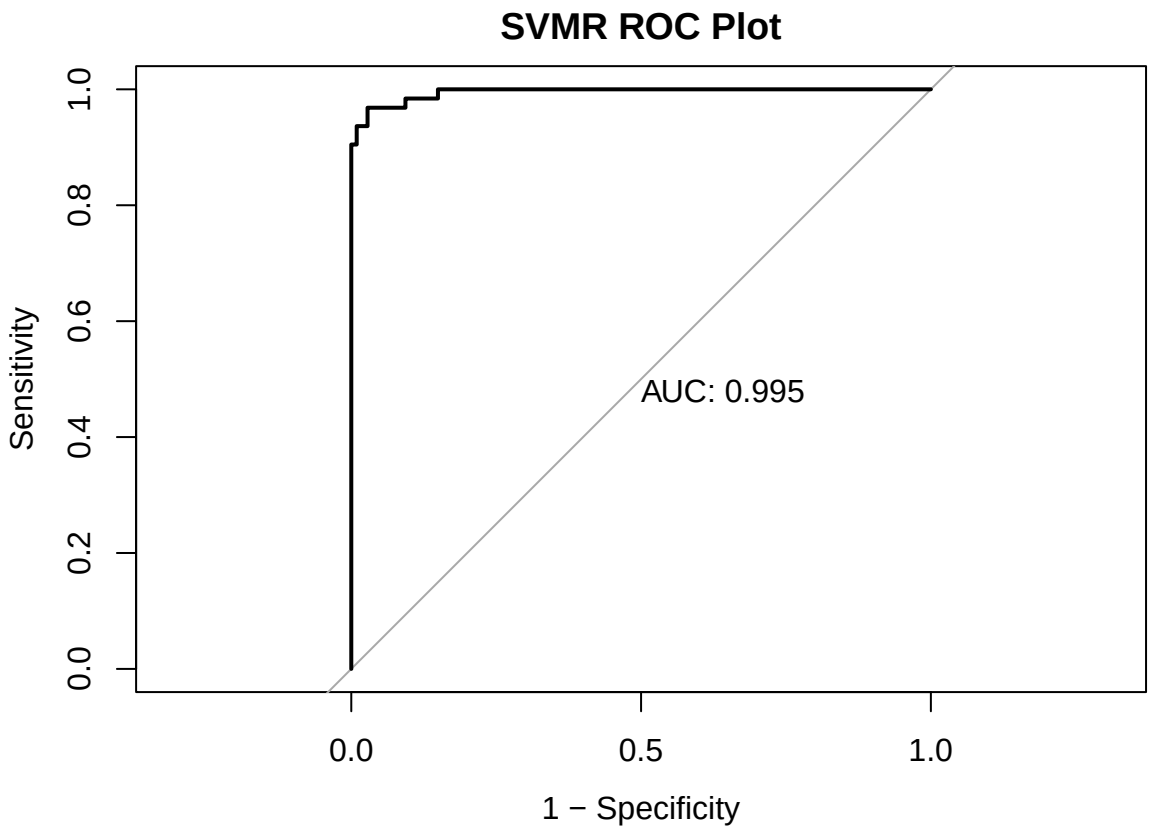
```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.88057756929153
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0445514262444897
##
## Number of Support Vectors : 98
##
## Objective Function Value : -55.3619
## Training error : 0.007519
## Probability model included.
```

```r
# Radial Kernel training error rate
pred_svmr_train = predict(model.svmr)
train_svmr_error = mean(pred_svmr_train != trainData$diagnosis)

# Radial Kernel test error rate
pred_svmr_test = predict(model.svmr, newdata = testData, type = "raw")
test_svmr_error = mean(pred_svmr_test != testData$diagnosis)
```

```r
# SVMR roc:
pred_svmr = predict(model.svmr, newdata = testData, type = "prob")[,1]
roc_svmr = roc(testData$diagnosis, pred_svmr,
               levels = c("B", "M"))

plot.roc(roc_svmr, legacy.axes = TRUE, print.auc = TRUE,main = "SVMR ROC Plot")
```



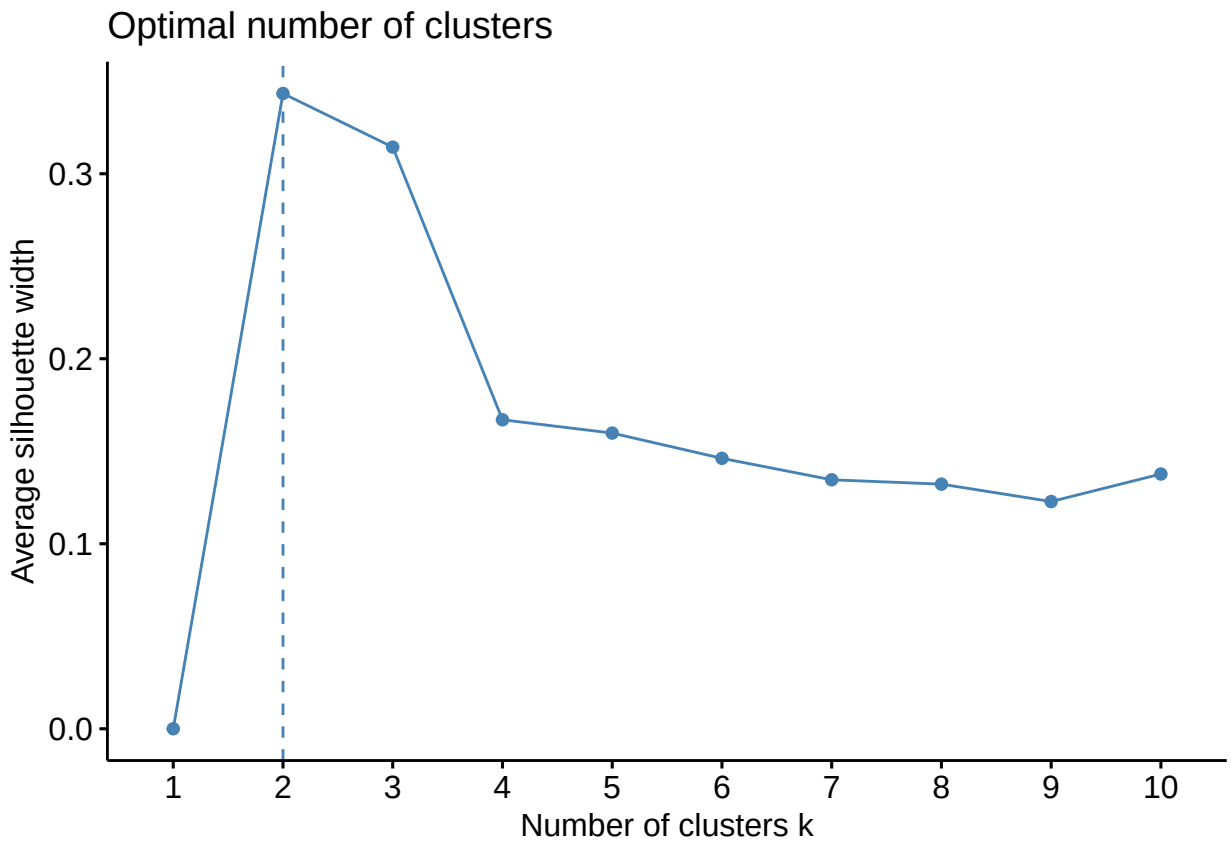
#Cluster Analysis

K-mean clustering

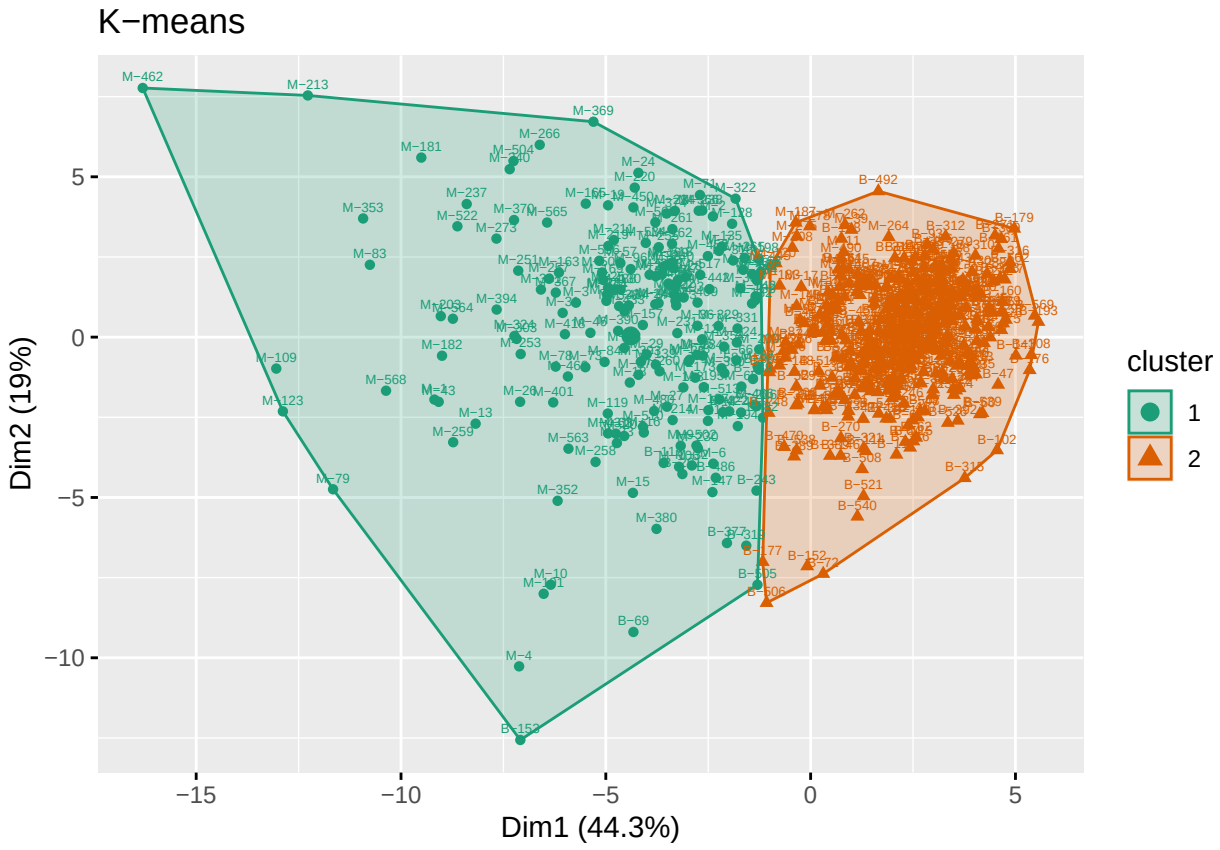
```r
index = seq.int(nrow(bc_df))
class = paste0(bc_df$diagnosis,"-",index)

bc_df_scale = bc_df[,2:31] %>% as.data.frame()
rownames(bc_df_scale) = class
bc_df_scale = bc_df_scale %>% scale()
```

```
set.seed(31)
fviz_nbclust(bc_df_scale,
             FUNcluster = kmeans,
             method = "silhouette")
```



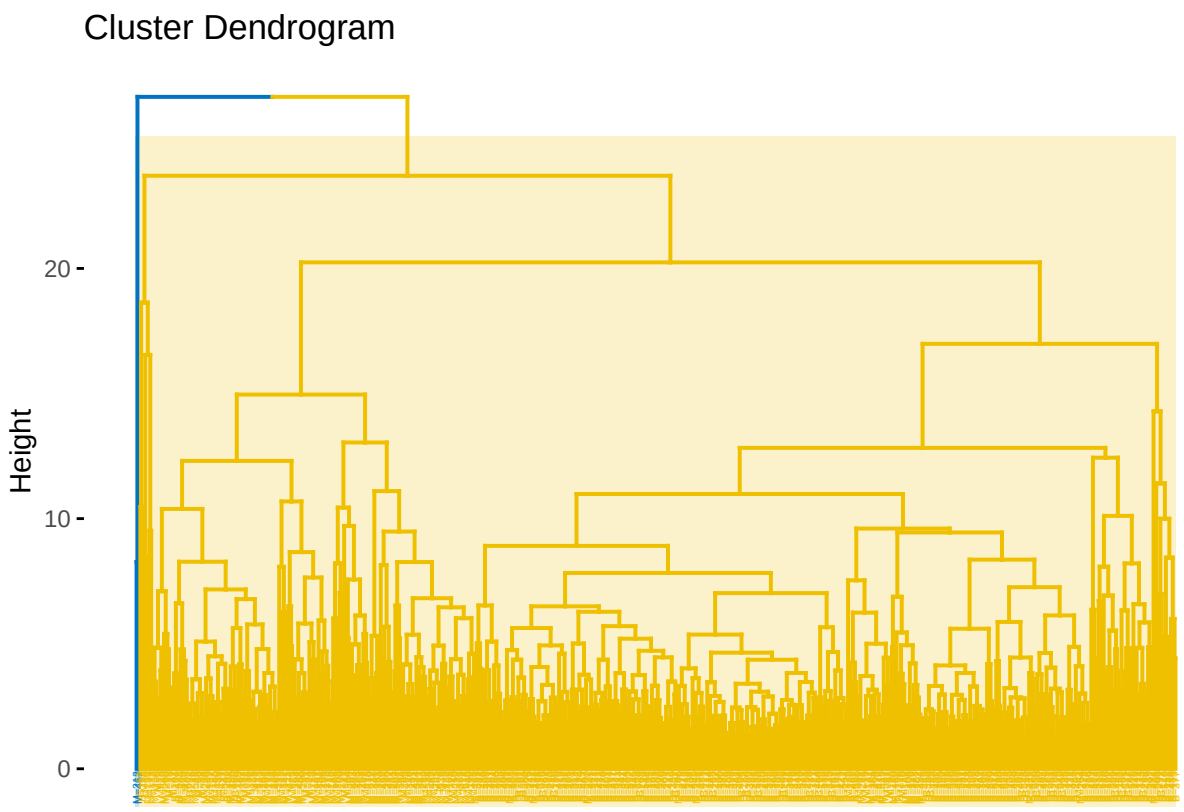

```
km <- kmeans(bc_df_scale, centers = 2, nstart = 20)
km_vis <- fviz_cluster(list(data = bc_df_scale, cluster = km$cluster),
                       ellipse.type = "convex",
                       geom = c("point","text"),
                       labelsize = 5,
                       palette = "Dark2") + labs(title = "K-means")
km_vis
```



Hierarchical clustering using complete linkage and Euclidean distance

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

fviz_dend(hc.complete.scaled, k = 2,
          cex = 0.3,
          palette = "jco",
          color_labels_by_k = TRUE,
          rect = TRUE, rect_fill = TRUE, rect_border = "jco",
          labels_track_height = 1)
```




```
bc.complete.scaled <- cutree(hc.complete.scaled, 2)
```