P8106 Final Codes

Yuxuan Chen | Yuan Meng | Paula Wu

```r
bc_df = read.csv("./data/breast-cancer.csv", row.names = NULL) %>%
  dplyr::select(-c(1,33)) %>%
  janitor::clean_names() %>%
  mutate(diagnosis = factor(diagnosis, level = c("B", "M")))
unique(bc_df$diagnosis)
```

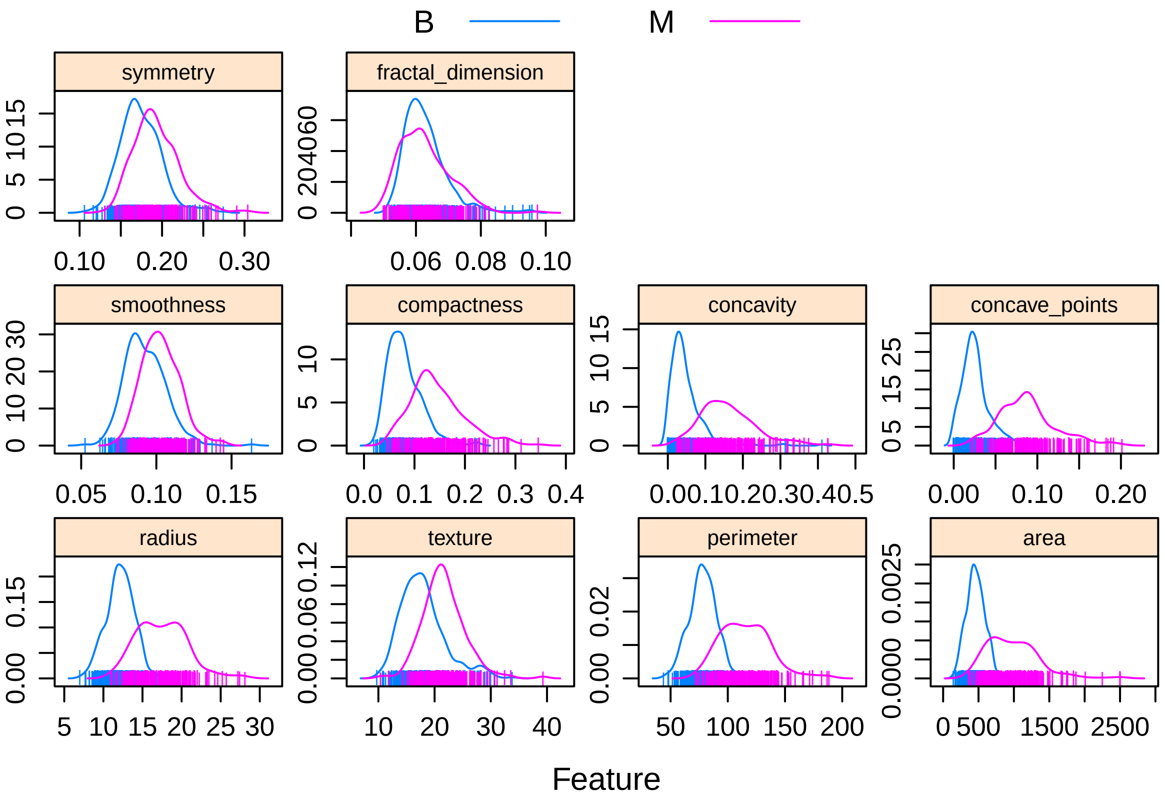
```
## [1] M B
## Levels: B M
```

```r
formula_all = parse(text = paste0("diagnosis ~ ", paste(colnames(bc_df[2:31]),collapse = " + ")))[[1]]
```

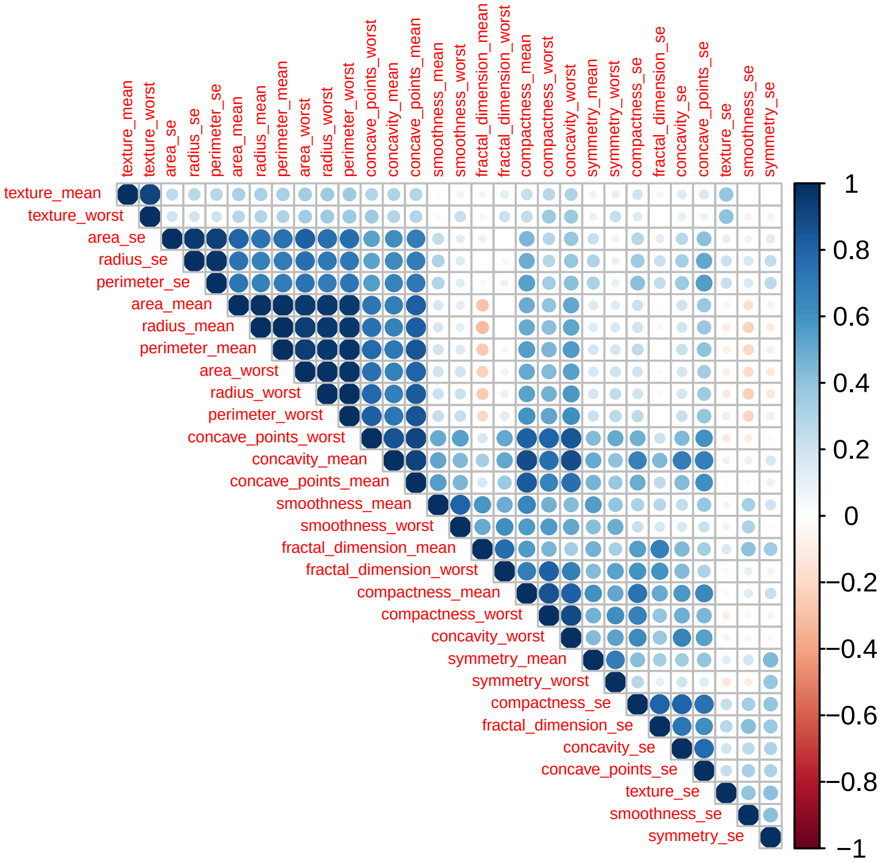
```r
# partitioning data
set.seed(31)
indexTrain <- createDataPartition(bc_df$diagnosis, p = 0.7, list = FALSE)
trainData = bc_df[indexTrain, ]
testData = bc_df[-indexTrain,]
x = model.matrix(diagnosis~., trainData)[,-1]
y = trainData$diagnosis
```

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

```r
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 = "|",
            strip=strip.custom(par.strip.text=list(cex=.7)),
            auto.key = list(columns = 2))
```



```
#predictor correlations
corrplot(cor(bc_df[,-1]),
         method = "circle",
         type = "upper",
         tl.cex = 0.5,
         order = "hclust")
```



Modeling:

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

Penalized Logistic Regression

```
glmnGrid = expand.grid(.alpha = seq(0,1,length = 21),
                      .lambda = exp(seq(-8,-1,length =50)))
set.seed(31)
glm_fit = train(x = x,
                y = y,
                method = "glmnet",
                tuneGrid = glmnGrid,
                metric = "ROC",
                trControl = ctrl)
glm_fit$bestTune
```

```
##     alpha      lambda
## 469  0.45 0.004389362
```

```
color_set = rainbow(25)
parameter_set = list(superpose.symbol = list(col = color_set),
                     superpose.line = list(col = color_set))
glm_plot = plot(glm_fit, par.settings = parameter_set, xTrans = function(x) log(x))
```

```
set.seed(31)

#vip(glm_fit)
#glm.pred.prob = predict(glm_model,
#                        type = "response")
#glm.pred = rep("0", length(glm.pred.prob))
#glm.pred[glm.pred.prob > 0.5] = "1"
#confusionMatrix(data = factor(glm.pred, levels = c("1", "0")),
#                reference = factor(train_data$diagnosis),
#                positive = "1")
#glm.pred.prob.test = predict(glm_fit, type = "response", newdata = testData)
#roc.glm.test = roc(testData$diagnosis, glm.pred.prob.test)
#plot(roc.glm.test, legacy.axes = TRUE, print.auc = TRUE)
```

Fit MARS

```
set.seed(31)
mars_grid = expand.grid(degree = 1:5,
                         nprune = 2:20)

mars_fit = train(x = x,
                  y = y,
                 method = "earth",
                 tuneGrid = mars_grid,
                 metric = "ROC",
                 trControl = ctrl)

mars_plot = ggplot(mars_fit, highlight = TRUE)
mars_fit$bestTune
```

```
##   nprune degree
## 7      8      1
```

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

```
##                     (Intercept)              h(1299-area_worst)
##                    1.682024e+01                   -3.260643e-02
## h(0.07911-concave_points_worst)            h(17.68-radius_mean)
##                    1.132215e+02                    2.046438e+00
##  h(0.07507-concave_points_mean)          h(35.64-texture_worst)
##                   -1.660836e+02                   -3.391145e-01
##        h(perimeter_worst-124.3)        h(perimeter_worst-98.27)
##                   -3.899007e-03                   -3.267962e-02
```

```
#Training RMSE
mars_train_se = mean(mars_fit$resample$RMSE)
mars_train_se
```

```
## [1] NA
```

```
#Testing RMSE
mars_test_predict = predict(mars_fit,
                        newdata = testData)
mars_test_se = RMSE(mars_test_predict, testData$diagnosis)
mars_test_se
```

```
## [1] NA
```

Fit KNN

```
set.seed(31)

knn_fit = train(x = x,
                y = y,
                method = "knn",
                preProcess = c("center", "scale"),
                tuneGrid = data.frame(k = seq(1,50,by=1)),
                trControl = ctrl)
knn_fit$bestTune
```

```
##     k
## 28 28
```

```
knn_plot = ggplot(knn_fit,xTrans = function(x)log(x), highlight = TRUE)

#Training RMSE
knn_train_se = mean(knn_fit$resample$RMSE)
knn_train_se
```

```
## [1] NA
```

```
#Testing RMSE
knn_test_predict = predict(knn_fit,
                        newdata = testData)
knn_test_se = RMSE(knn_test_predict, testData$diagnosis)
knn_test_se
```

```
## [1] NA
```

LDA

```
# LDA
set.seed(31)
lda_fit = train(diagnosis ~. ,
                data = trainData,
                method = "lda",
                metric = "ROC",
                trControl = ctrl)
```

CART

```
# classification tree
set.seed(31)
rpart_fit = train(diagnosis ~., trainData,
                  method = "rpart",
                  tuneGrid = data.frame(cp = exp(seq(-20,-2, len = 50))),
                  trControl = ctrl,
                  metric = "ROC")
rpart_plot = ggplot(rpart_fit, highlight = TRUE)
rpart_fit$bestTune
```

```
##             cp
## 42 0.007163364
```

```r
rpart_pred = predict(rpart_fit, newdata = testData, type = "prob")[,2]
roc(testData$diagnosis, rpart_pred)
```

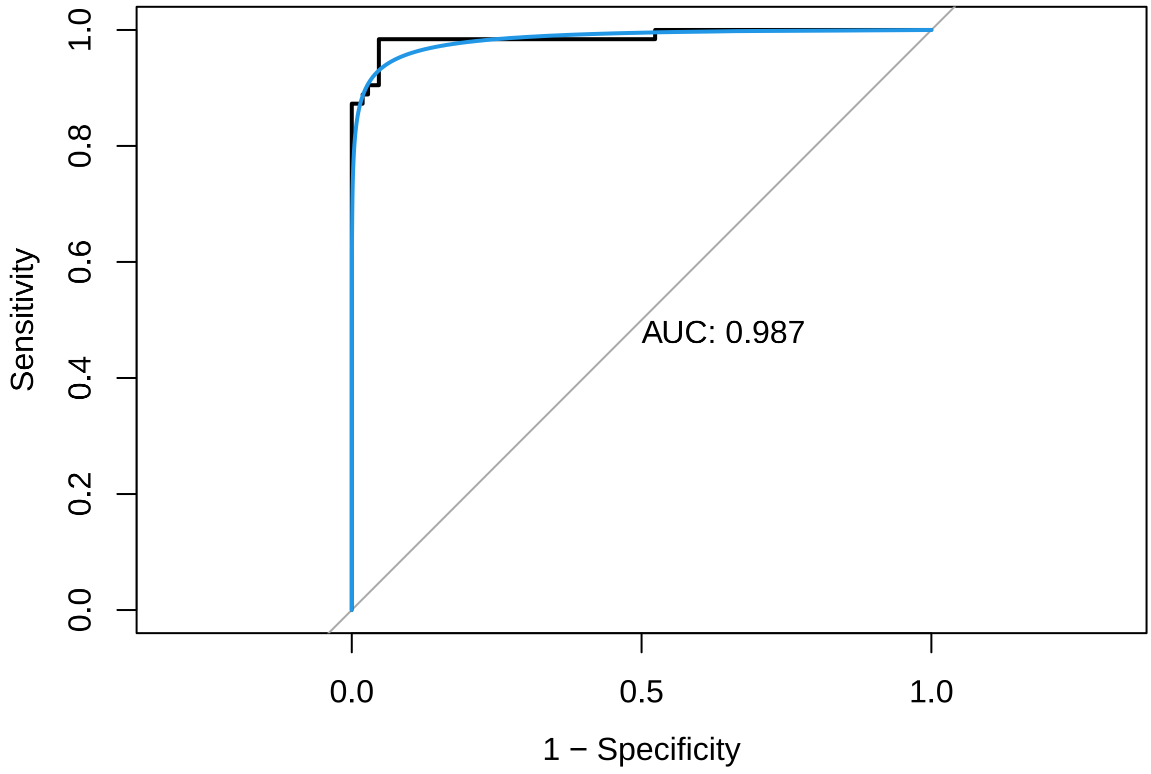
```
##
## Call:
## roc.default(response = testData$diagnosis, predictor = rpart_pred)
##
## Data: rpart_pred in 107 controls (testData$diagnosis B) < 63 cases (testData$diagnosis M).
## Area under the curve: 0.9456
```

random forest

```r
# random forest
set.seed(31)
rf_grid = expand.grid(mtry = 1:8,
                      splitrule = "gini",
                      min.node.size = seq(from = 2, to = 10, by = 2))
rf_fit = train(diagnosis ~., trainData,
               method = "ranger",
               tuneGrid = rf_grid,
               metric = "ROC",
               trControl = ctrl)
rf_plot = ggplot(rf_fit, highlight = TRUE)
rf_fit$bestTune
```

```
##   mtry splitrule min.node.size
## 1    1      gini             2
```

```r
rf_pred = predict(rf_fit, newdata = testData, type = "prob")[,1]
roc_rf = roc(testData$diagnosis, rf_pred)
plot(roc_rf, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc_rf), col = 4, add = TRUE)
```



AdaBoost

```
set.seed(31)
gbmA_grid = expand.grid(n.trees = c(2000,3000,4000,5000),
                        interaction.depth = 1:6,
                        shrinkage = c(0.0005,0.001,0.002),
                        n.minobsinnode = 1)

gbmA_fit = train(diagnosis ~.,
                 trainData,
                  tuneGrid = gbmA_grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)
gbmA_fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 64    5000                 4     0.002              1
```

```
gbm_plot = ggplot(gbmA_fit, highlight = TRUE)
```

SVM (linear and radial kernel)

a) Linear Kernel

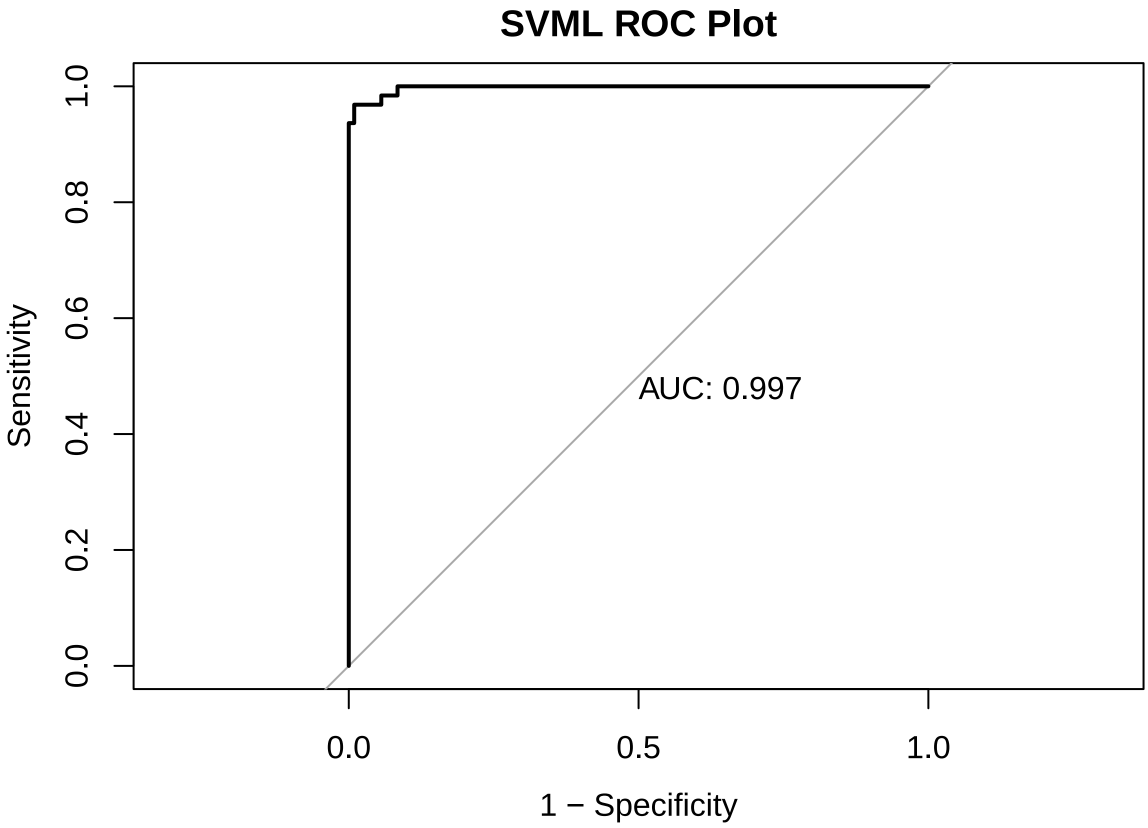
```
set.seed(31)
svml_fit <- train(diagnosis~.,
                  data = trainData,
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-3,2,len = 50))),
                  trControl = ctrl)
svml_plot = ggplot(svml_fit,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

svml_fit$bestTune
```

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

```
svml_fit$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(svml_fit)
train_error = mean(pred_svml_train != trainData$diagnosis)

## Linear Kernel Test Error Rate
pred_svml_test = predict(svml_fit, newdata = testData, type = "raw")
test_error = mean(pred_svml_test != testData$diagnosis)

# SVML roc:
pred_svml = predict(svml_fit, 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)
svmr_fit <- 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
```

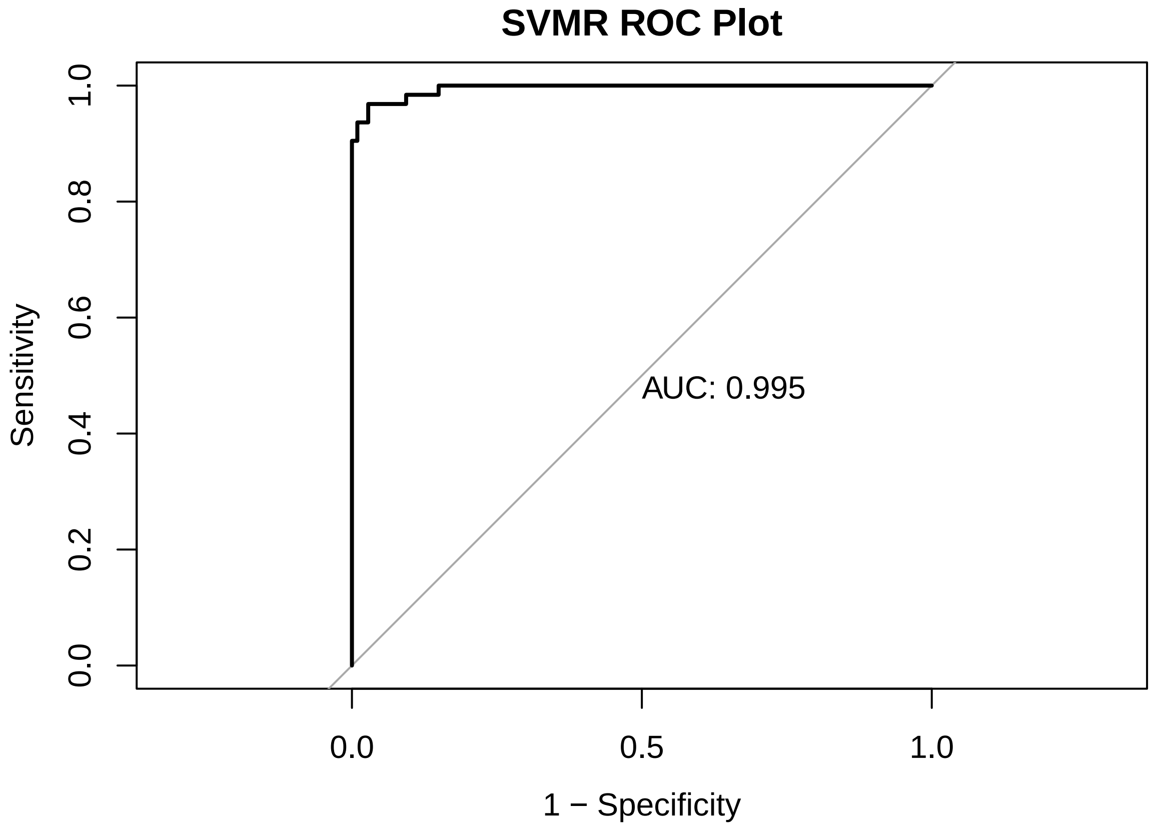
```
myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
svmr_plot = ggplot(svmr_fit, highlight = TRUE, par.settings = myPar)

svmr_fit$bestTune
```

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

```
svmr_fit$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.
```

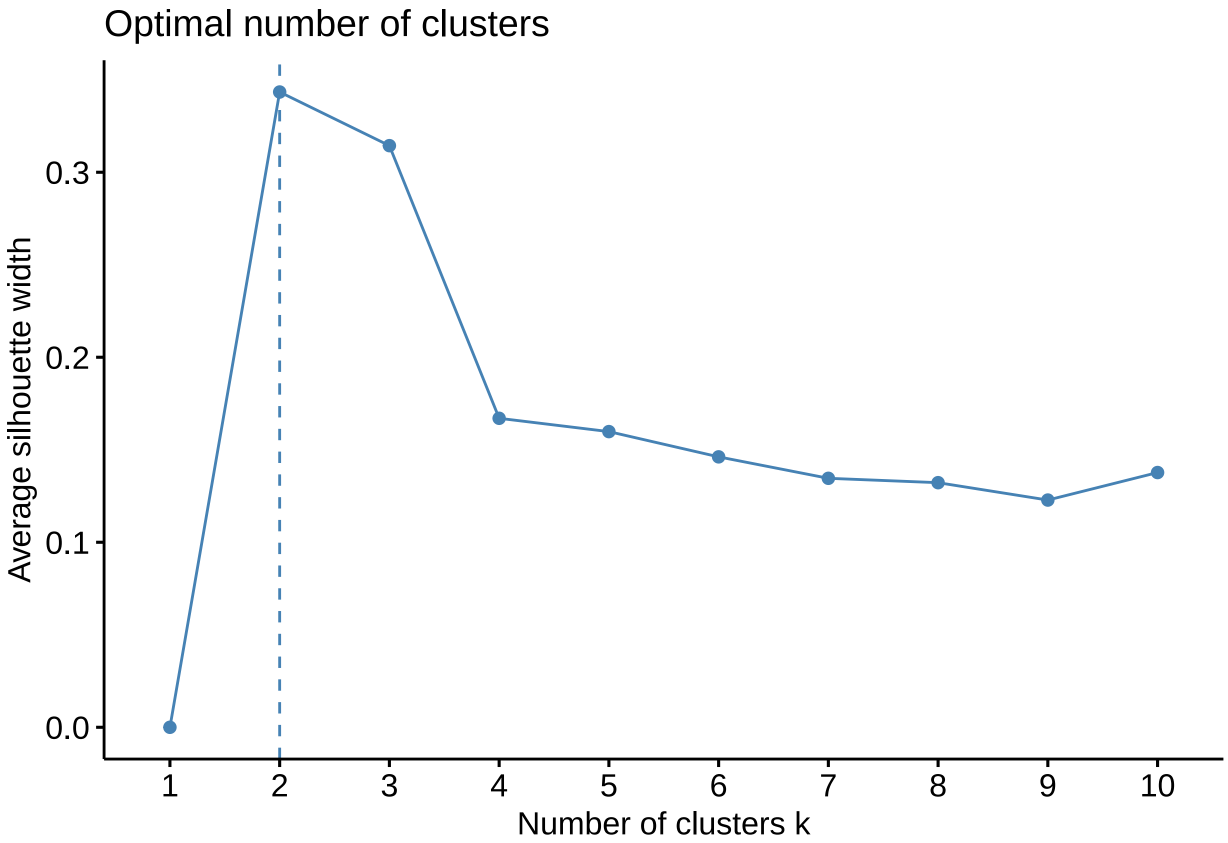
```
# Radial Kernel training error rate
pred_svmr_train = predict(svmr_fit)
train_svmr_error = mean(pred_svmr_train != trainData$diagnosis)

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

# SVMR roc:
pred_svmr = predict(svmr_fit, 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")
```



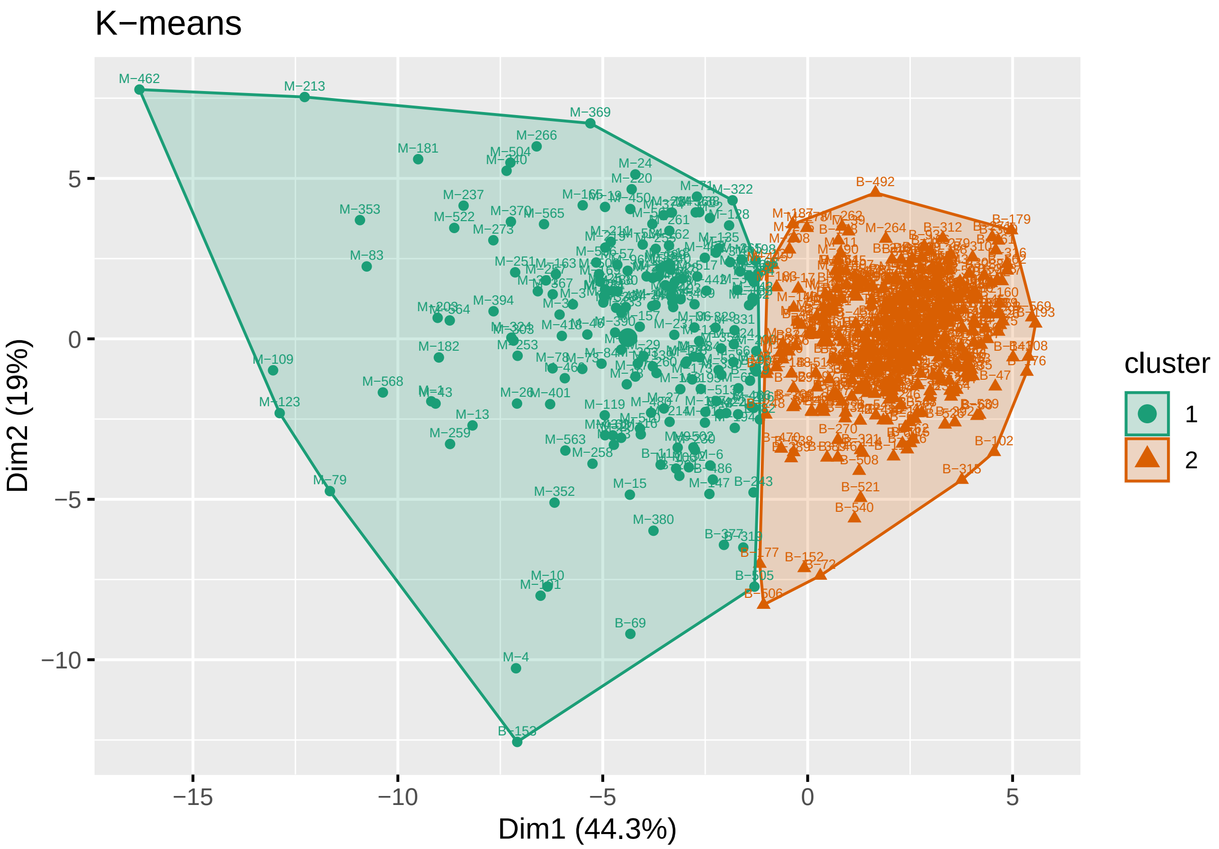
##Cluster Analysis

K-mean clustering

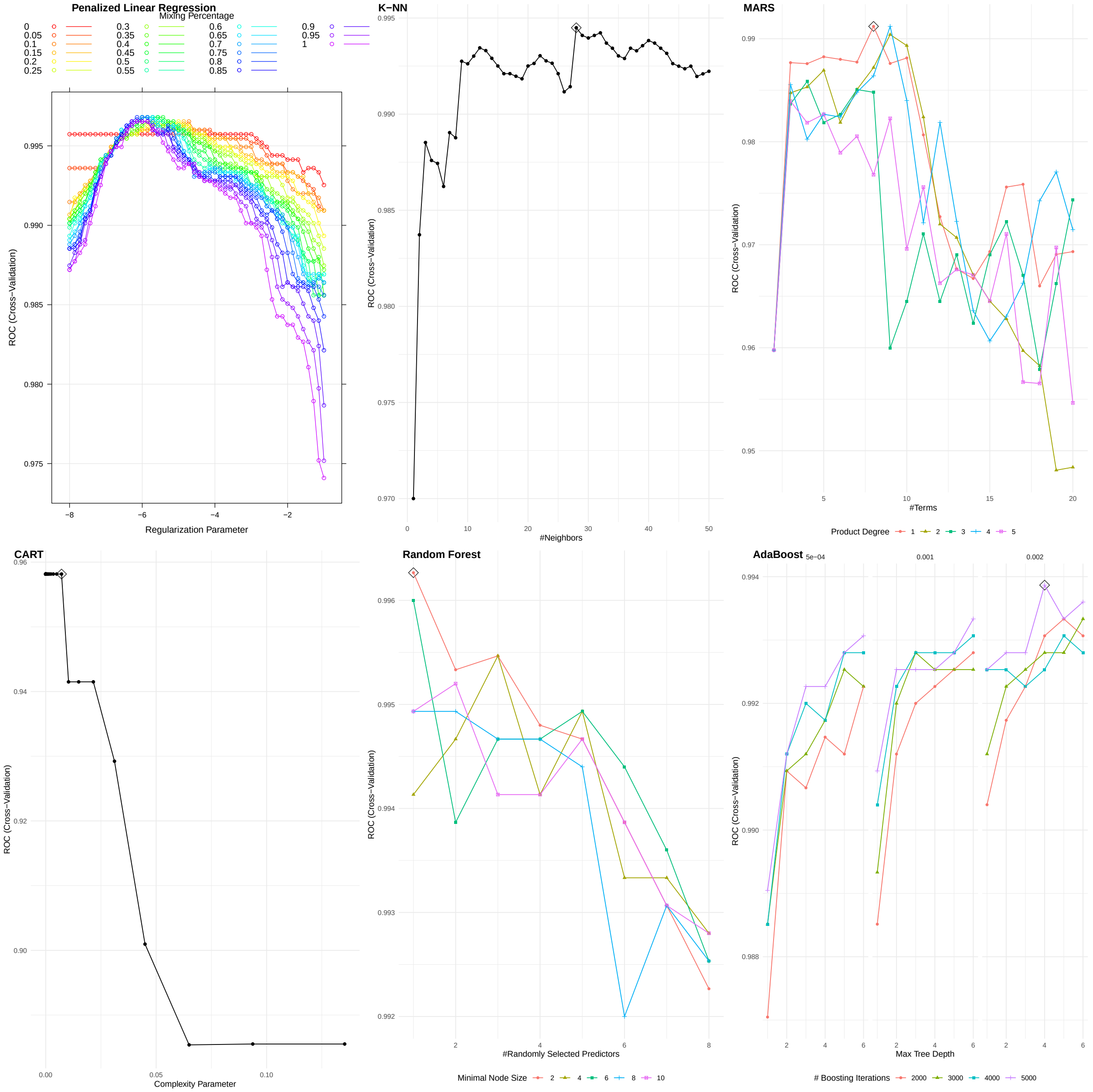
```
index = seq.int(nrow(bc_df))
class = paste0(bc_df$diagnosis,"-",index)

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
```

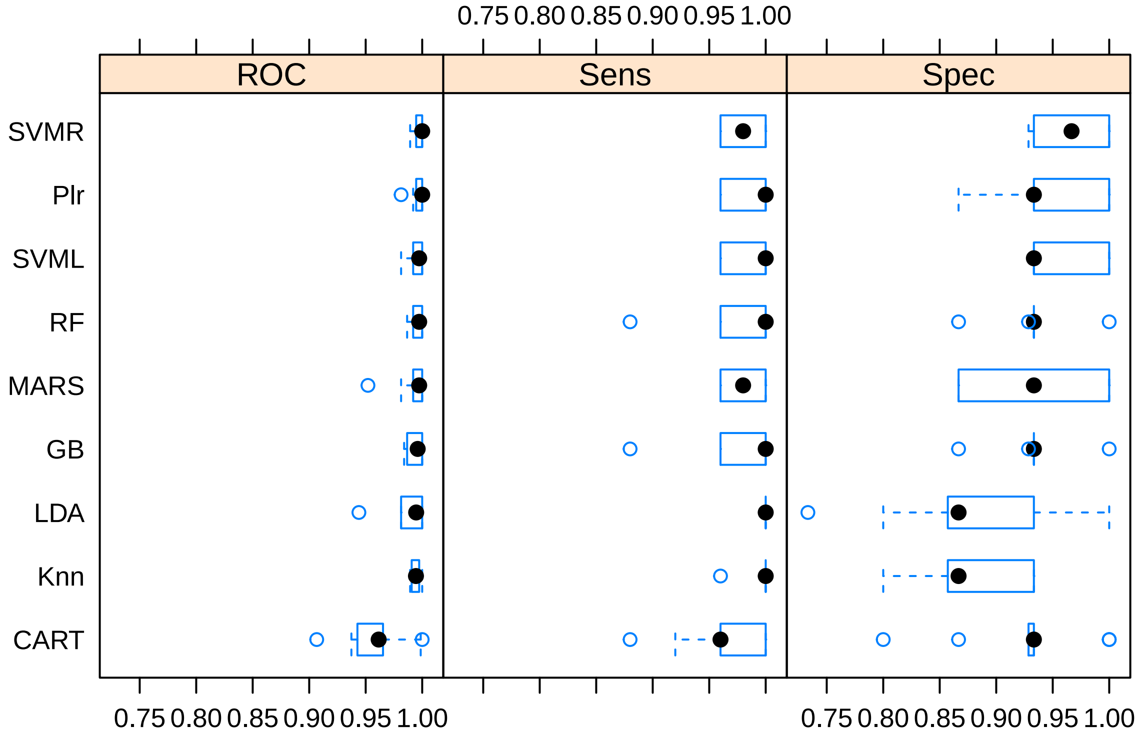


```
library(cowplot)
plot_grid(glm_plot, knn_plot, mars_plot, rpart_plot, rf_plot, gbm_plot, labels = c('Penalized Linear Reg
```



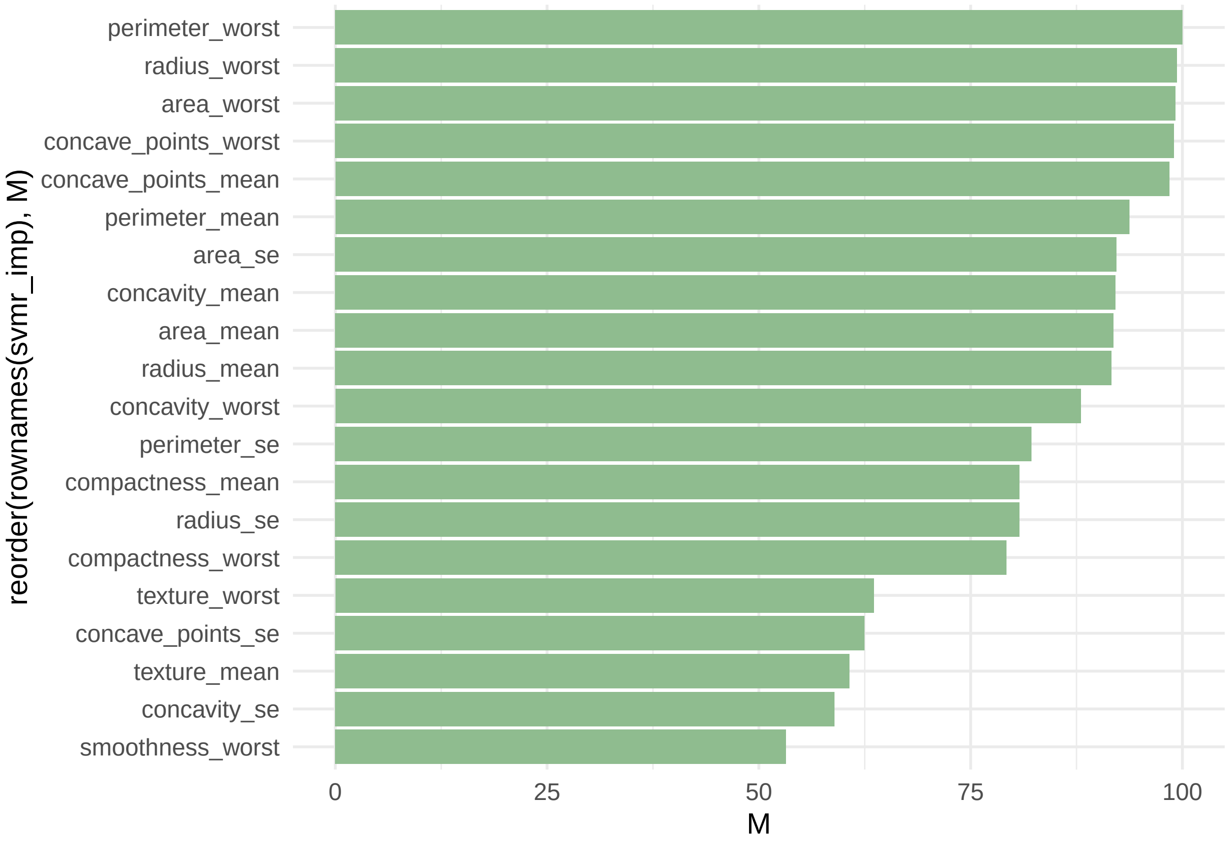
resampling results

```
resamp = resamples(list(Plr = glm_fit,
                        MARS = mars_fit,
                        Knn = knn_fit,
                        LDA = lda_fit,
                        CART = rpart_fit,
                        RF = rf_fit,
                        GB = gbmA_fit,
                        SVML = svml_fit,
                        SVMR = svmr_fit))
bwplot(resamp)
```



variable importance

```
svmr_imp = varImp(svmr_fit)$importance %>%
  arrange(desc(M)) %>%
  top_n(n = 20)
ggplot(svmr_imp, aes(x = reorder(rownames(svmr_imp),M), y = M, fill = M)) +
  geom_bar(stat="identity", position="dodge", fill = "darkseagreen") +
  coord_flip()
```


perimeter_worst
radius_worst
area_worst
concave_points_worst
concave_points_mean
perimeter_mean
area_se
concavity_mean
area_mean
radius_mean
concavity_worst
perimeter_se
compactness_mean
radius_se
compactness_worst
texture_worst
concave_points_se
texture_mean
concavity_se
smoothness_worst
reorder(rownames(svmr_imp), M)
0
25
50
75
100
M

test AUC