p8106_hw5_yg2625

Yue Gu

April 27, 2019

This problem involves the OJ data set which is part of the ISLR package. The data contains 1070 purchases where the customer either purchased Citrus Hill or Minute Maid Orange Juice. A number of characteristics of the customer and product are recorded. Use set.seed() for reproducibility. Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

load data

```
data(OJ)
oj_data = OJ %>%
  janitor::clean_names()


# create a training set containing 800 obs, and a test set containing the remaining obs
set.seed(1)
rowTrain = createDataPartition(y = oj_data$purchase,
                               p = 799/1070,
                               list = F)
train_data = oj_data[rowTrain, ]
test_data = oj_data[-rowTrain, ]
```

(a) Fit a support vector classifier (linear kernel) to the training data with Purchase as the response and the other variables as predictors. What are the training and test error rates?

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

set.seed(1)
# fit model
svml.fit <- train(purchase ~ .,
                  data = train_data,
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-5,-1,len=50))),
                  trControl = ctrl)
# model output
svml.fit$finalModel
```

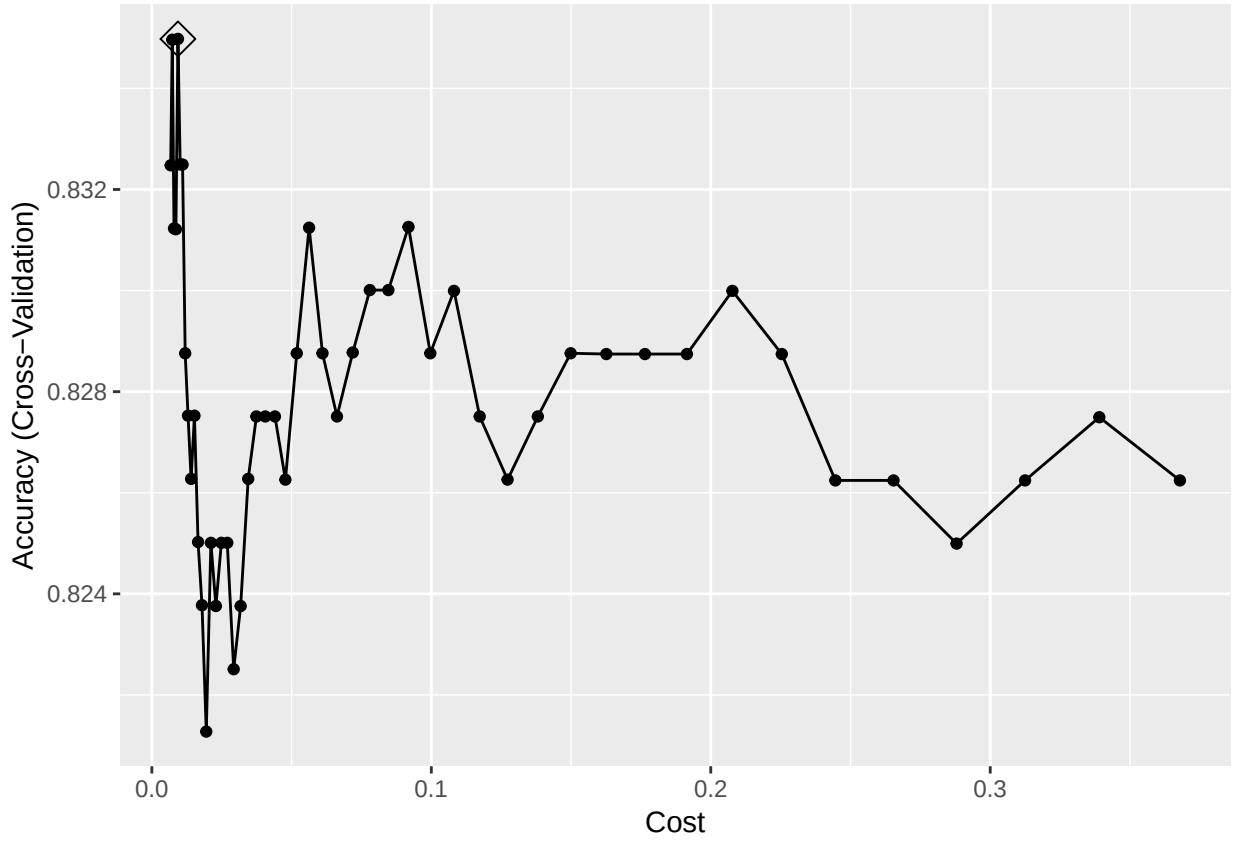
```
##
## Call:
## svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
##     probability = classProbs)
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.00933981
##       gamma:  0.05882353
##
## Number of Support Vectors:  444
```

```
# best tunning parameter
svml.fit$bestTune
```

```
##         cost
## 5 0.00933981
```

```
# Accuracy plot
ggplot(svml.fit, highlight = TRUE)
```




```
# training error rate
pred_train = predict(svml.fit)
mean(train_data$purchase != pred_train)
```

```
## [1] 0.16125
```

```
# test error rate
pred_test = predict(svml.fit, newdata = test_data, type = "raw")
mean(test_data$purchase != pred_test)
```

```
## [1] 0.1703704
```

The training error is 0.161, the test error is 0.170.

(b) Fit a support vector machine with a radial kernel to the training data. What are the training and test error rates?

```
svmr.grid <- expand.grid(C = exp(seq(-5,2,len=20)),
                         sigma = exp(seq(-8,-1,len=10)))
set.seed(1)
# fit model
svmr.fit <- train(purchase ~ ., train_data,
                  method = "svmRadial",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl)
# model output
svmr.fit$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 0.387760103296325
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0163895537902136
##
## Number of Support Vectors : 455
##
## Objective Function Value : -150.3717
## Training error : 0.15625
```

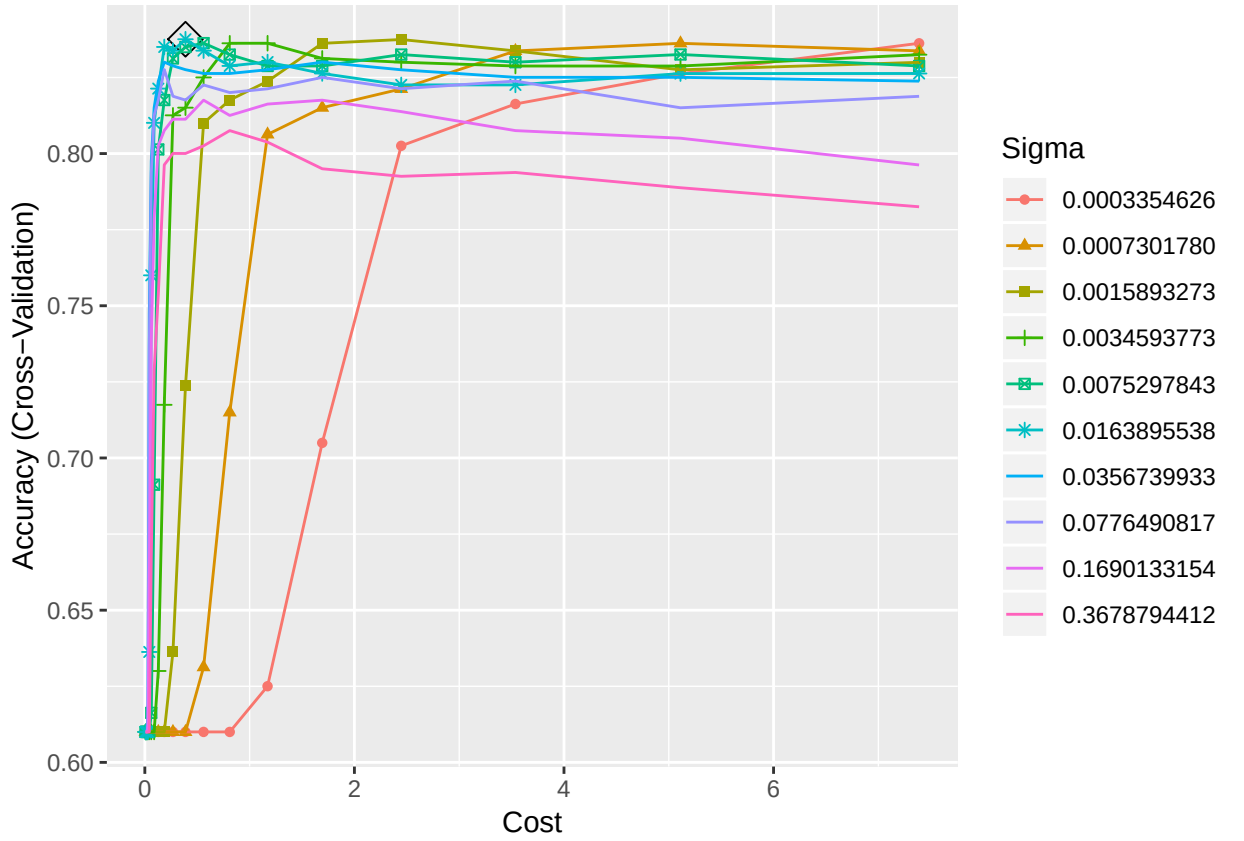
```
# best tunning parameter
svmr.fit$bestTune
```

```
##          sigma         C
## 116 0.01638955 0.3877601
```

```
# Accuracy plot
ggplot(svmr.fit, highlight = TRUE)
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have
## 10. Consider specifying shapes manually if you must have them.
```

```
## Warning: Removed 80 rows containing missing values (geom_point).
```




```
# trainning error rate
pred_train2 = predict(svmr.fit)
mean(train_data$purchase != pred_train2)
```

```
## [1] 0.15625
```

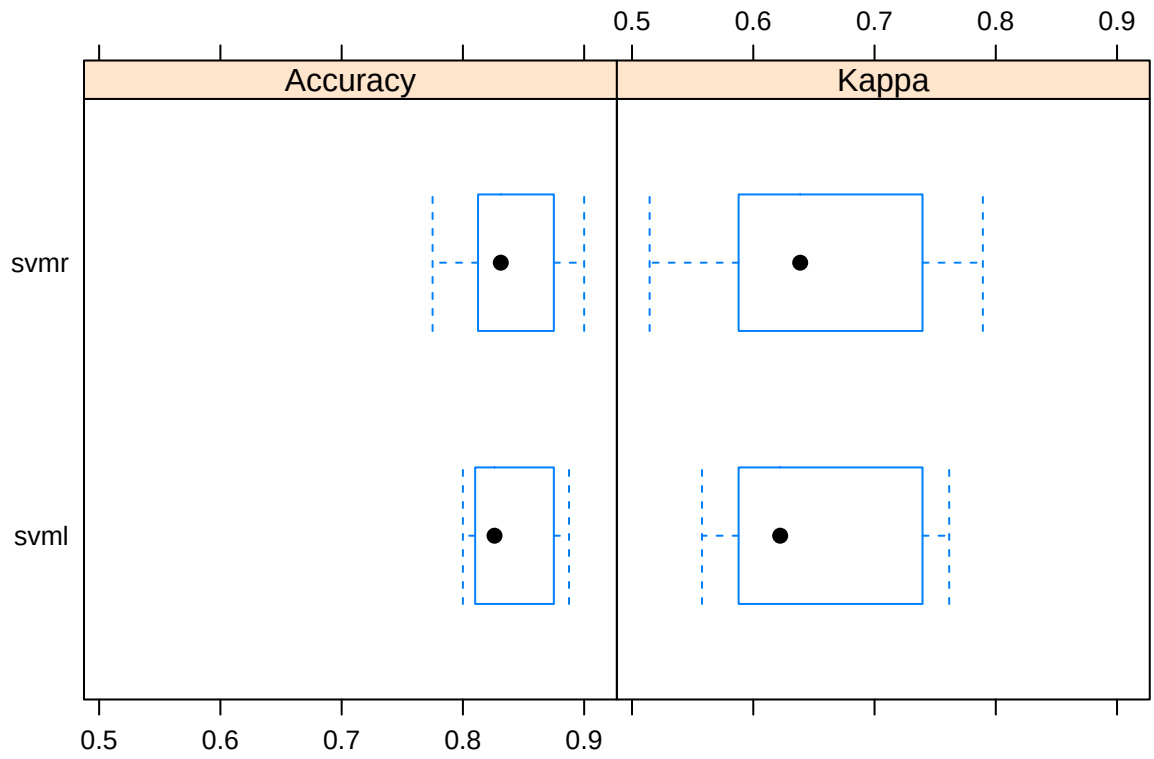
```
# test error rate
pred_test2 = predict(svmr.fit, newdata = test_data, type = "raw")
mean(test_data$purchase != pred_test2)
```

```
## [1] 0.1703704
```

The training error rate is 0.156 and the test error rate is 0.170.

(c) Which approach seems to give a better result on this data?

```
resamp <- resamples(list(svmr = svmr.fit, svml = svml.fit))
bwplot(resamp)
```




```
# test data performance
pred.svml <- predict(svml.fit, newdata = test_data)
pred.svmr <- predict(svmr.fit, newdata = test_data)

# linear kernel
confusionMatrix(data = pred.svml,
                reference = test_data$purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 146  27
##         MM  19  78
##
##                Accuracy : 0.8296
##                  95% CI : (0.7794, 0.8725)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : 5.295e-15
##
##                   Kappa : 0.6365
##  Mcnemar's Test P-Value : 0.302
##
##             Sensitivity : 0.8848
##             Specificity : 0.7429
##          Pos Pred Value : 0.8439
```

```
##          Neg Pred Value : 0.8041
##              Prevalence : 0.6111
##          Detection Rate : 0.5407
##    Detection Prevalence : 0.6407
##       Balanced Accuracy : 0.8139
##
##        'Positive' Class : CH
##
```

```
# radial kernel
confusionMatrix(data = pred.svmr,
                reference = test_data$purchase)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 146  27
##         MM  19  78
##
##                Accuracy : 0.8296
##                  95% CI : (0.7794, 0.8725)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : 5.295e-15
##
##                   Kappa : 0.6365
##  Mcnemar's Test P-Value : 0.302
##
##             Sensitivity : 0.8848
##             Specificity : 0.7429
##          Pos Pred Value : 0.8439
##          Neg Pred Value : 0.8041
##              Prevalence : 0.6111
##          Detection Rate : 0.5407
##    Detection Prevalence : 0.6407
##       Balanced Accuracy : 0.8139
##
##        'Positive' Class : CH
##
```

Based on the confusion matrix result, linear kernel and radial provides the same accracy and kappa on the test data. However, on the boxplot, svmr provides slightly larger kappa and accuracy compared to svml. Hence, radial kernel approach seems to give a better result on the data.