90-10 Set

Machine Learning Goal: Classify based on Endo vs. No endo

Importing data

```
require(caret)  # contains most of the prediction functions we'll use
```

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
```

```
require(rpart)  # needed for the plotting
```

```
## Loading required package: rpart
```

```
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
require(e1071)
```

```
## Loading required package: e1071
```

```
# read in the core gene data
core_data <- read.csv("../../../data/genbank/06-mw-genes.csv")
core_class <- core_data[,4] # obtain endo classifications
core_exp <- core_data[,6:387] # expression data
core_genes <- cbind(core_class, core_exp)
colnames(core_genes)[1] <- "endo"

# read in training set v2
train2 <- read.csv("../../../data/machine-learning/mann-whitney/model-9010.csv")
train_class2 <- train2[,4]
train_exp2 <- train2[,6:387]
endo_train2 <- cbind(train_class2, train_exp2)
colnames(endo_train2)[1] <- "severity"

# read in test set v2
test2 <- read.csv("../../../data/machine-learning/mann-whitney/validation-9010.csv")
test_class2 <- test2[,4]
test_exp2 <- test2[,6:387]
endo_test2 <- cbind(test_class2, test_exp2)
colnames(endo_test2)[1] <- "severity"
```

kNN

```
# define tuning parameters
fitControl_cv4 <- trainControl(method = "cv", number = 4)
values.k <- data.frame(k = 1:82)

# run kNN model with 5-fold CV and various k's
set.seed(4747)
trainKNN2 <- train(severity ~ ., data = endo_train2,
                   method = "knn", trControl = fitControl_cv4,
                   tuneGrid = values.k)

# output results to find optimal k
trainKNN2$finalModel
```

```
## 11-nearest neighbor model
## Training set outcome distribution:
##
##     Endometriosis Non-Endometriosis
##                70                63
```

Optimal model is k = 44. The associated model-building accuracy with this is 0.7537.

We apply this model to the test set of 30 observations. Our test set accuracy is 70%.

```
# obtain accuracy on the OUTER test set with k = 51 model
set.seed(47)
predict_knn2 <- confusionMatrix(data = predict(trainKNN2, newdata = endo_test2),
                 reference = endo_test2$severity)
predict_knn2$table
```

```
##                     Reference
## Prediction           Endometriosis Non-Endometriosis
##   Endometriosis                  6                 2
##   Non-Endometriosis              1                 6
```

```
# print test accuracy
knn_accuracy2 <- predict_knn2$overall[1]
knn_accuracy2
```

```
## Accuracy
##      0.8
```

Random Forests

```r
# set tuning method and parameter
control <- trainControl(method = "oob")
mtry.tune <- data.frame(mtry = 1:100)

# run and optimize random forest with ntree = 250
# optimal mtry = 20
# OOB error = 24.58%
set.seed(4747)
rf.250_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 250, importance = TRUE)
rf.250_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 250, mtry = param$mtry, importance = TRUE)
##               Type of random forest: classification
##                     Number of trees: 250
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 32.33%
## Confusion matrix:
##                     Endometriosis Non-Endometriosis class.error
## Endometriosis                  52                18   0.2571429
## Non-Endometriosis              25                38   0.3968254
```

```r
# run with ntree = 300
# optimal mtry = 20
# OOB error = 25.42 %
set.seed(4747)
rf.300_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 300, importance = TRUE)
rf.300_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry, importance = TRUE)
##               Type of random forest: classification
##                     Number of trees: 300
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 30.83%
## Confusion matrix:
##                     Endometriosis Non-Endometriosis class.error
## Endometriosis                  53                17   0.2428571
## Non-Endometriosis              24                39   0.3809524
```

```r
# run with ntree = 400
# optimal mtry = 20
# OOB error = 25.42
set.seed(4747)
rf.400_2 <- train(severity ~ ., data = endo_train2, method = "rf",
```

```r
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 400, importance = TRUE)
rf.400_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 400, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 56
##
##         OOB estimate of  error rate: 32.33%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 52                18   0.2571429
## Non-Endometriosis             25                38   0.3968254
```

```r
# run with ntree = 500
# optimal mtry = 13
# OOB error = 27.12 %
set.seed(4747)
rf.500_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 500, importance = TRUE)
rf.500_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 500, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 31.58%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 54                16   0.2285714
## Non-Endometriosis             26                37   0.4126984
```

```r
# run with ntree = 550
# optimal mtry = 26
# OOB error = 25.42 %
set.seed(4747)
rf.550_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 550, importance = TRUE)
rf.550_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 550, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 550
## No. of variables tried at each split: 11
##
```

```
##         OOB estimate of  error rate: 30.08%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 54                16   0.2285714
## Non-Endometriosis             24                39   0.3809524
```

```
# run with ntree = 600
# optimal mtry = 20
# OOB error = 25.42 %
set.seed(4747)
rf.600_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 600, importance = TRUE)
rf.600_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 600, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 600
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 30.08%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 53                17   0.2428571
## Non-Endometriosis             23                40   0.3650794
```

```
# run with ntree = 700
# optimal mtry = 31
# OOB error = 25.42 %
set.seed(4747)
rf.700_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 700, importance = TRUE)
rf.700_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 700, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 700
## No. of variables tried at each split: 98
##
##         OOB estimate of  error rate: 33.08%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 50                20   0.2857143
## Non-Endometriosis             24                39   0.3809524
```

```
# run with ntree = 800
# optimal mtry = 7
# OOB error = 27.97 %
set.seed(4747)
rf.800_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
```

```r
                 tuneGrid = mtry.tune, ntree = 800, importance = TRUE)
rf.800_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 800, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 800
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 33.08%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                53                17   0.2428571
## Non-Endometriosis            27                36   0.4285714
```

```r
# run with ntree = 900
# optimal mtry = 39
# OOB error = 26.27 %
set.seed(4747)
rf.900_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 900, importance = TRUE)
rf.900_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 900, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 900
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 31.58%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                53                17   0.2428571
## Non-Endometriosis            25                38   0.3968254
```

```r
# run with ntree = 1000
# optimal mtry = 44
# OOB error = 26.27 %
set.seed(4747)
rf.1000_2 <- train(severity ~ ., data = endo_train2, method = "rf",
                 trControl = control, na.action = na.roughfix,
                 tuneGrid = mtry.tune, ntree = 1000, importance=TRUE)
rf.1000_2$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 1000, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 30.83%
```

```
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 53                17   0.2428571
## Non-Endometriosis             24                39   0.3809524
```

We found that a Random Forests model with ntree = 250 and mtry = 20 works optimally. The associated model building error with those parameters is 0.7542.

```r
# build the final model with mtry = 20 and ntree = 250
rf.final2 <- rf.250_2
```

To confirm that this optimization worked, I will apply the same parameters using the RandomForests() function rather than the `caret` package. We obtain a similar OOB error rate/accuracy.

```r
# try using Random Forest function
set.seed(4747)
rf.function2 <- randomForest(severity ~ ., data = endo_train2, mtry = 20, ntree = 250, importance = TRU
rf.function2
```

```
##
## Call:
##  randomForest(formula = severity ~ ., data = endo_train2, mtry = 20,      ntree = 250, importance = 
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 20
##
##         OOB estimate of  error rate: 36.84%
## Confusion matrix:
##                    Endometriosis Non-Endometriosis class.error
## Endometriosis                 48                22   0.3142857
## Non-Endometriosis             27                36   0.4285714
```

Apply this to our test data to obtain the test accuracy. We obtain an accuracy of 76.67%.

```r
set.seed(47)
predict_rf2 <- confusionMatrix(data = predict(rf.final2, newdata = endo_test2),
                reference = endo_test2$severity)
predict_rf2$table
```

```
##                      Reference
## Prediction            Endometriosis Non-Endometriosis
##   Endometriosis                   6                 1
##   Non-Endometriosis               1                 7
```

```r
# print test accuracy
rf_accuracy2 <- predict_rf2$overall[1]
rf_accuracy2
```

```
##  Accuracy
## 0.8666667
```

SVM

We use linear SVM because the relationship between the Endo vs no Endo is that one is two times the other (a linear separation). We have to tune the cost: 'C' parameter, which tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Single SVM on entire data

```r
set.seed(47)

# create vector for costs
cost <- c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)

# fit linear support vector classifier
svmL_2 <- train(severity ~ ., data = endo_train2, method="svmLinear",
                trControl = trainControl(method = "cv", number = 4),
                tuneGrid = expand.grid(C = cost),
                preProcess = c("center", "scale"))

# output the results
svmL_2
```

```
## Support Vector Machines with Linear Kernel
##
## 133 samples
## 382 predictors
##   2 classes: 'Endometriosis', 'Non-Endometriosis'
##
## Pre-processing: centered (382), scaled (382)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 99, 99, 100, 101
## Resampling results across tuning parameters:
##
##   C      Accuracy   Kappa
##   1e-03  0.6772644  0.3447931
##   1e-02  0.8047432  0.6077321
##   1e-01  0.7971674  0.5930600
##   1e+00  0.7971674  0.5930600
##   5e+00  0.7971674  0.5930600
##   1e+01  0.7971674  0.5930600
##   1e+02  0.7971674  0.5930600
##   1e+03  0.7971674  0.5930600
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```r
svmL_2$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
```

```
## SV type: C-svc  (classification)
##  parameter : cost C = 0.01
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 90
##
## Objective Function Value : -0.4371
## Training error : 0.045113
```

Optimal cost parameter of 0.1. The associated model-building error is 0.7448.

```
predict_svm2 <- confusionMatrix(data = predict(svmL_2, newdata = endo_test2),
                reference = endo_test2$severity)
predict_svm2$table
```

```
##                     Reference
## Prediction           Endometriosis Non-Endometriosis
##   Endometriosis                  6                 0
##   Non-Endometriosis              1                 8
```

```
# print test accuracy
svm_accuracy2 <- predict_svm2$overall[1]
svm_accuracy2
```

```
##  Accuracy
## 0.9333333
```

```
# ANOTHER WAY
# predict the test data
predict_SVM2 <- predict(svmL_2, endo_test2)
confusion_matrix2 <- table(endo_test2$severity, predict_SVM2)

# calculate test error rate
linearError2 <-  (confusion_matrix2[1,2] + confusion_matrix2[2,1]) / sum(confusion_matrix2)
linearError2
```

```
## [1] 0.06666667
```

We obtain an optimal SVM model with cost, C = 0.01. When applied to the test set, we obtain a test accuracy of 83.33%.

SVM on 10-fold test validation

Now, we use Edie's code to 10-fold CV on test data.

```
cost <- c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)

ten_fold <- sapply(1:100, function(t) {
  random_ix <- sample(x=c(1:148), size=round(148*0.8), replace=FALSE)
  random_ix_2 <- setdiff(c(1:148), random_ix)

  train_core <- core_genes[random_ix,]
  test_core <- core_genes[-random_ix,]

  trainSVM <- train(endo ~ ., data = train_core, method = "svmLinear",
                 trControl = trainControl(method = "none"),
```

```
                 tuneGrid = expand.grid(C = 0.10),
                 preProcess = c("center", "scale"))

  test.predict <- predict(trainSVM, test_core)

  confusion_matrix <- table(test_core$endo, test.predict)

  linearError <-  (confusion_matrix[1,2] + confusion_matrix[2,1]) / sum(confusion_matrix)
  linearError
})

mean(ten_fold)
```

```
## [1] 0.239
```