STAT30270 Statistical Machine Learning Assignment 3

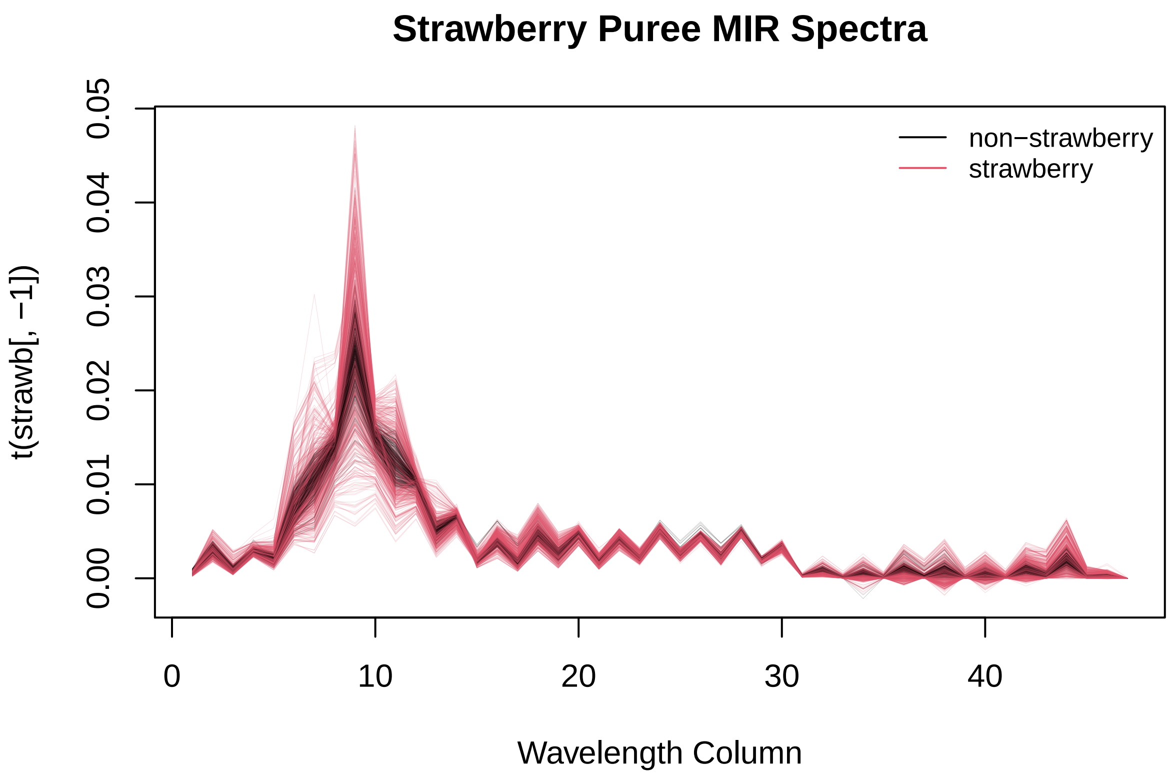
Fanahan McSweeney - 20203868

April 07, 2021

1. Compare random forests, boosting, and SVM classifiers in application to classification of the puree spectra. You need to employ an appropriate cross-validation procedure to train, validate, and test the models. In implementing SVM classifiers, you must tune some of the hyperparameters.

Firstly to visualise the MIR spectral data for the puree samples, I have loaded the spectral data and plotted it below, colouring each observation based on its class (i.e. black for non-strawberry samples, red for strawberry samples).

```r
# read in then MIR spectral data
strawb <- read.csv("Data/data_mir_strawberry.csv")

# plot all MIR spectra, colour observations based on class
matplot(t(strawb[,-1]), type = "l", lty=1, lwd=0.1,
        col = adjustcolor((strawb[,1]=="nonstrawberry")+1, 0.2),
        main = "Strawberry Puree MIR Spectra", xlab="Wavelength Column")
legend("topright", c("non-strawberry", "strawberry"), col = 1:2, lty = 1, bty="n", cex=0.8)
```



Next, I will reserve a random subset of the data (20%) to be used as a test set. This will be used later when evaluating the predictive performance of the best model when classifying new unseen observations.

I will then use k-fold cross-validation to perform validation on SVM, random forest and boosting classifiers. I have chosen to use 4 fold cross-validation, splitting the remaining 80% of the data into 4 folds each containing 20% of the total data. To further account for the variation in results due to random sampling of the data I will also replicate the cross-validation procedure above 10 times.

For the SVM classifier, I have chosen to use a Gaussian Radial Basis function kernel. During the cross-validation, I will test a range of values for the hyperparameters σ and cost C. Specifically, I will test values of 10, 100, 200, 300 and 400 for C, and values of 0.01, 0.02, 0.05, 0.1 and 0.15 for σ, to determine which combination of these values achieves the best validation performance.

For both the random forest and boosting classifiers, I will use the default parameter settings during cross-validation.

```r
# Load relevant libraries
library(randomForest)
library(adabag)
library(kernlab)
library(knitr)
library(kableExtra)

# create data frame of MIR spectra for all observations
x_tot <- strawb[,-1]
# get vector of class data for all observations, convert to numeric factor
# (1=strawberry, 2=nonstrawberry)
y_tot <- as.factor(strawb$class)
y_tot <- as.numeric(y_tot)
strawb$class <- as.factor(y_tot)

# set aside test data
N <- nrow(x_tot)
# set seed for reproducibility
set.seed(20203868)
# reserve 20% of the observations for testing
test <- sample(1:N, N*0.2)
# create x vector and y matrix for test set
x_test <- x_tot[test,]
y_test <- y_tot[test]

# select remaining data for training and validation
trainval <- setdiff(1:N, test)
x <- x_tot[trainval,]
y <- y_tot[trainval]
N_trainval <- nrow(x)

# function to compute classification accuracy
class_acc <- function(y, yhat) {
  tab <- table(y, yhat)
  return(sum(diag(tab))/sum(tab) )
}

# create grid of C and sigma values to test with SVM
C <- c(10, 100, 200, 300, 400)
sigma <- c(0.01, 0.02, 0.05, 0.1, 0.15)
```

```
grid <- expand.grid(C, sigma)
colnames(grid) <- c("C","sigma")

# total size of the grid for SVM
n_mod <- nrow(grid)

K <- 4 # set number of folds
R <- 10 # set number of replicates

out_svm <- vector("list", R) # store accuracy output (SVM)
out_rf <- vector("list", R) # store accuracy output (random forest)
out_boost <- vector("list", R) # store accuracy output (boosting)

tables_svm <- vector("list", R) # store class. accuracy tables (SVM)
tables_rf <- vector("list", R) # store class. accuracy tables (random forest)
tables_boost <- vector("list", R) # store class. accuracy tables (boosting)


# repeat the following steps R times (number of replicates)
for(r in 1:R) {
  acc_svm <- matrix(NA, K, n_mod) # accuracy of the classifiers in the K folds for SVM
  acc_rf <- matrix(NA, K) # accuracy of the classifiers in the K folds for random forest
  acc_boost <- matrix(NA, K) # accuracy of the classifiers in the K folds for boosting

  # create empty list of lists (dimentions=K x n_mod) to store class.acc. tables (SVM)
  tab_svm0 <- vector("list", n_mod)
  tab_svm <- vector("list", K)
  for(k in 1:K) tab_svm[[k]] <- tab_svm0
  # create empty list (length=K) to store class.acc. tables (random forest)
  tab_rf <- vector("list", K)
  # create empty list (length=K) to store class.acc. tables (boosting)
  tab_boost <- vector("list", K)

  # split training/validation data into K folds
  folds <- rep(1:K, ceiling(N_trainval/K))
  folds <- sample(folds) # random permute
  folds <- folds[1:N_trainval] # ensure we got N_train data points

  # repeat the following steps K times (for each of the K folds)
  for(k in 1:K) {
    # set current train data to all data in training/validation set minus the current fold 'k'
    train_fold <- which(folds!=k)
    # set current validation data to the current fold 'k'
    validation <- setdiff(1:N_trainval, train_fold)

    ## SVM models
    # loop through all combinations of C and sigma values
    for(j in 1:n_mod) {
      # fit SVM model to current training data, with current C and sigma value selection
      fit_svm <- ksvm(class~., data=strawb[train_fold,], type="C-svc", kernel="rbfdot",
                C=grid$C[j], kpar=list(sigma=grid$sigma[j]))
      # predict classes of current validation set
      pred_svm <- predict(fit_svm, newdata=x[validation,])
```

```r
      # calculate classification accuracy of the validation data
      acc_svm[k,j] <- class_acc(pred_svm, y[validation])
      # add confusion matrix table to list
      tab_svm[[k]][[j]] <- table(pred_svm, y[validation])

    }

    ## Random Forest Models
    # fit random forest model to current training data
    fit_rf <- randomForest(class ~ ., data = strawb[train_fold,])
    # predict classes of current validation set
    pred_rf <- predict(fit_rf, newdata=x[validation,], type="class")
    # calculate classification accuracy of the validation data
    acc_rf[k] <- class_acc(pred_rf, y[validation])
    # add confusion matrix table to list
    tab_rf[[k]] <- table(pred_rf, y[validation])


    ## Boosting Models
    # fit boosting model to current training data
    fit_boost <- boosting(class ~ ., data = strawb[train_fold,],
                          coeflearn = "Breiman", boos = FALSE)
    # predict classes of current validation set
    pred_boost <- predict(fit_boost, newdata=x[validation,])
    # calculate classification accuracy of the validation data
    acc_boost[k] <- class_acc(pred_boost$class, y[validation])
    # add confusion matrix table to list
    tab_boost[[k]] <- table(pred_boost$class, y[validation])

  }
  # add classification accuracy values for all models for current replicate to output lists
  out_svm[[r]] <- acc_svm
  out_rf[[r]] <- acc_rf
  out_boost[[r]] <- acc_boost

  # add lists of class.acc tables from current replicate to output lists
  tables_svm[[r]] <- tab_svm
  tables_rf[[r]] <- tab_rf
  tables_boost[[r]] <- tab_boost
}
```

The table printed below shows the average classification accuracy obtained during cross-validation for each of the tested SVM classifiers. The cost value and σ value for the SVM model that achieved the best classification accuracy are also printed below.

```r
# get mean accuracy across all folds for all replicates (for all SVM models)
avg_fold_acc_svm <- t(sapply(out_svm, colMeans) )
# get mean overall accuracy across all replicates (for all SVM models)
avg_acc_svm <- colMeans(avg_fold_acc_svm)

# get mean accuracy across all folds for all replicates (for random forest model)
avg_fold_acc_rf <- t(sapply(out_rf, colMeans) )
# get mean overall accuracy across all replicates (for random forest model)
```

```
avg_acc_rf <- mean(avg_fold_acc_rf)

# get mean accuracy across all folds for all replicates (for boosting model)
avg_fold_acc_boost <- t(sapply(out_boost, colMeans) )
# get mean overall accuracy across all replicates (for boosting model)
avg_acc_boost <- mean(avg_fold_acc_boost)

# create table of C and sigma values, as well as validation classification accuracy values obtained
grid_acc <- cbind(grid, avg_acc_svm)
kable(grid_acc, align = 'c')
```

C	sigma	avg_acc_svm
10	0.01	0.9726840
100	0.01	0.9786543
200	0.01	0.9800535
300	0.01	0.9801810
400	0.01	0.9832306
10	0.02	0.9754791
100	0.02	0.9820859
200	0.02	0.9852624
300	0.02	0.9856418
400	0.02	0.9860225
10	0.05	0.9781454
100	0.05	0.9856398
200	0.05	0.9856398
300	0.05	0.9855129
400	0.05	0.9852585
10	0.10	0.9820846
100	0.10	0.9856385
200	0.10	0.9855116
300	0.10	0.9855116
400	0.10	0.9855116
10	0.15	0.9827204
100	0.15	0.9850047
200	0.15	0.9850047
300	0.15	0.9850047
400	0.15	0.9850047

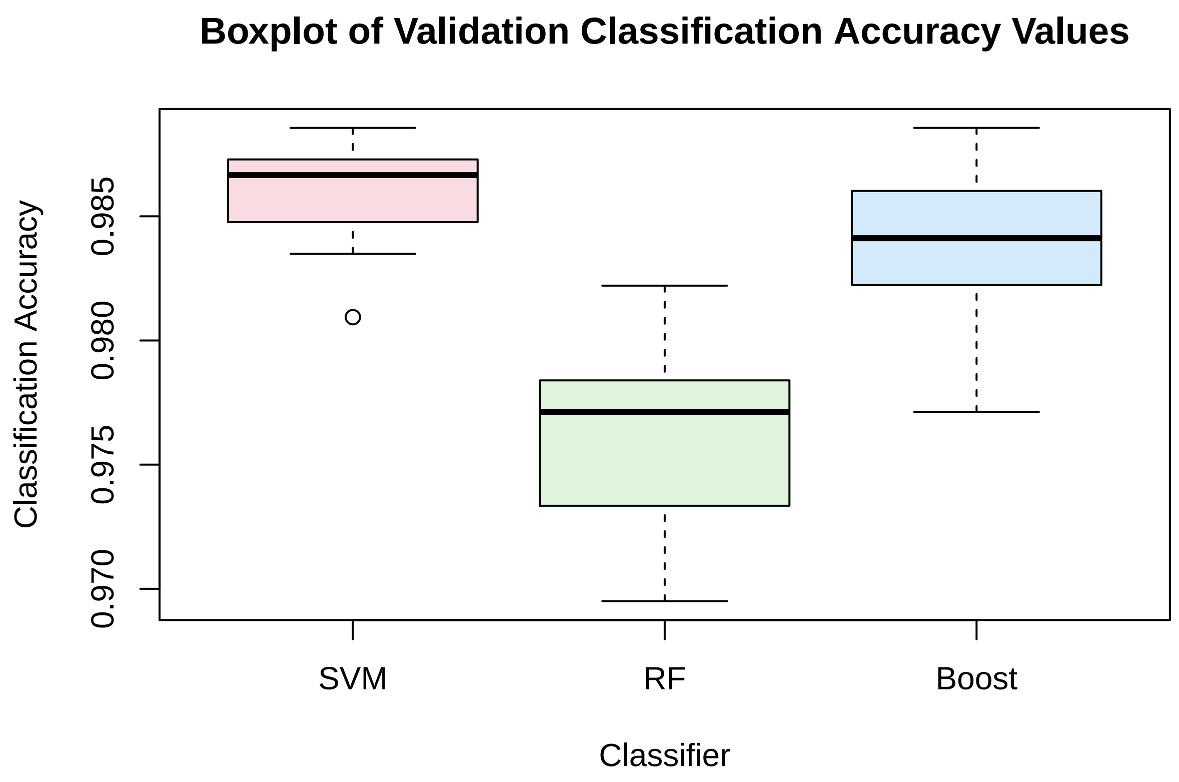
```
# get index of SVM model with best classification accuracy
best <- which.max(grid_acc$avg_acc_svm)

# get C and sigma values for the best SVM model
bestC <- grid_acc[best,1]
bestSigma <- grid_acc[best,2]
cat("\nThe optimum SVM model had a C value of", bestC,
    "and a sigma value of", bestSigma, "\n")
```

```
##
## The optimum SVM model had a C value of 400 and a sigma value of 0.02
```

Below, the boxplot shows the average classification accuracy obtained for the optimum SVM classifier, the random forest classifier and the boosting classifier for each replicate during cross-validation.

While all 3 models achieved very high classification accuracy, the optimum SVM model (C=400 and σ=0.02) performed slightly better than the other models. The mean classification accuracy over all replicates is also printed for each of the models below.

```r
# Plot boxplot of classification accuracy
boxplot(avg_fold_acc_svm[,best], avg_fold_acc_rf, avg_fold_acc_boost, col=adjustcolor(c(2:4), 0.2),
        names=c("SVM", "RF", "Boost"), ylab="Classification Accuracy", xlab="Classifier",
        main="Boxplot of Validation Classification Accuracy Values")
```



```r
# Plot mean validation classification accuracy values for the best SVM, random forest
# and boosting models
cat("\nAverage Validation Classification Accuracy (SVM):", avg_acc_svm[best],
    "\nAverage Validation Classification Accuracy (Random Forest):", avg_acc_rf,
    "\nAverage Validation Classification Accuracy (Boosting):", avg_acc_boost, "\n")
```

```
##
## Average Validation Classification Accuracy (SVM): 0.9860225
## Average Validation Classification Accuracy (Random Forest): 0.9763675
## Average Validation Classification Accuracy (Boosting): 0.9839901
```

2. Evaluate and discuss the predictive performance of the best model at classifying new observations.

The optimum SVM classifier (C=400 and σ=0.02) was used to predicted the classes of the observations in the test set that was set aside earlier. The classification accuracy achieved and a confusion matrix comparing the actual classes to the classes predicted by the SVM model are printed below.

```r
# fit SVM model to full set of training/validation data using the obtained
# optimum C and sigma values
fit_final <- ksvm(class~., data=strawb[trainval,], type="C-svc", kernel="rbfdot",
            C=bestC, kpar=list(sigma=bestSigma))

# predict classes of the test set
pred_test <- predict(fit_final, newdata=x_test)
cat("\nTest Set Classification Accuracy (SVM):", class_acc(pred_test, y_test), "\n")
```

```
##
## Test Set Classification Accuracy (SVM): 0.9744898
```

```r
# create table comparing predicted classes to actual classes for test data
tab <- table(pred_test, y_test)
colnames(tab) <- c("strawberry (actual)", "non-strawberry (actual)")
rownames(tab) <- c("strawberry (predicted)", "non-strawberry (predicted)")
add_header_above(kable(tab, align = 'c'),
                 c(" "=1, "Test Data Classification Accuracy Confusion Matrix (SVM)"=2))
```

	Test Data Classification Accuracy Confusion Matrix (SVM)	
	strawberry (actual)	non-strawberry (actual)
strawberry (predicted)	129	3
non-strawberry (predicted)	2	62

Clearly, the model was very effective at classifying the "new" observations in the test data set, achieving a classification accuracy of 0.974. This is only slightly lower than the average classification accuracy achieved by the model during cross-validation earlier. Therefore, it doesn't appear that overfitting has occurred.

3. For real-world application of the model, a false positive (i.e. a non-strawberry sample classified as strawberry) is considered worst than a false negative (i.e. a strawberry sample classified as non-strawberry). Re-evaluate and discuss the predictive performance of the best model at classifying new observations in light of this information.

Below, I have recreated the confusion matrix obtained for the test data shown above, but this time I have divided each column (corresponding to the actual classes) by the total number of observations in that column. This gives a table showing the true positive rate (top-left), false positive rate (top-right), false negative rate (bottom-left) and true negative rate (bottom-right).

```r
# Print confusion matrix for test data class. accuracy in percentages
# (percentages calculated per column)
tab_pc <- round(t(t(tab)/colSums(tab))*100, 3)
add_header_above(kable(tab_pc, align = 'c'),
                 c(" "=1, "Test Data Classification Accuracy Confusion Matrix (SVM)\n- % of actual obse
```

	Test Data Classification Accuracy Confusion Matrix (SVM) - % of actual observations	
	strawberry (actual)	non-strawberry (actual)
strawberry (predicted)	98.473	4.615
non-strawberry (predicted)	1.527	95.385

We can see that in this case the rate of false positives (4.615%) is greater than the rate of false negatives (1.527%). While this is not ideal, this false positive rate is still relatively low. Ideally, we would run further tests on more new observations, or redo the process of separating test and training/validation data multiple times, and in each case evaluate the predictive performance of the model, but this would be very time consuming to carry out (with respect to this assignment). We can however look at the average false positive rate and false negative rate achieved during cross-validation without adding significant computation time to our code.

Below, I have printed a confusion matrix for the optimum SVM model. This table was obtained by calculating the average confusion matrix for the SVM model across each validation step during the cross-validation carried out earlier ("K" folds by "R" replicates). I have also recreated this table with each column divided by the total number of observations in that column, giving the equivalent true/false positive/negative rates.

```r
# SVM avg. validation accuracy cross-tab (best model)
#
# create 2x2 matrix of zeros
tab_sum_svm <- matrix(0,2,2)
# loop through all SVM runs for the best SVM model (R replicates by K folds)
for (r in 1:R) {
  for (k in 1:K) {
    # add current table to table sum
    tab_sum_svm <- tab_sum_svm + tables_svm[[r]][[k]][[best]]
  }
}

# Divide table of all class.acc. tables summed together, divide by K and R
tab1 <- tab_sum_svm/K/R
# rename row and columns names of table and print it
colnames(tab1) <- c("strawberry (actual)", "non-strawberry (actual)")
rownames(tab1) <- c("strawberry (predicted)", "non-strawberry (predicted)")
add_header_above(kable(tab1, align = 'c'), c(" "=1, "Validation Data Classification Accuracy Avg. Confu
```

	Validation Data Classification Accuracy Avg. Confusion Matrix (SVM)	
	strawberry (actual)	non-strawberry (actual)
strawberry (predicted)	123.55	1.05
non-strawberry (predicted)	1.70	70.45

```r
# Print confusion matrix for test data class. accuracy in percentages
# (percentages calculated per column)
tab2 <- round(t(t(tab1)/colSums(tab1))*100, 3)
add_header_above(kable(tab2, align = 'c'), c(" "=1, "Validation Data Classification Accuracy Avg. Confu
```

	Validation Data Classification Accuracy Avg. Confusion Matrix (SVM) - % of actual observations	
	strawberry (actual)	non-strawberry (actual)
strawberry (predicted)	98.643	1.469
non-strawberry (predicted)	1.357	98.531

Again, the average rate of false positives during cross-validation (1.469%) is greater than the average rate of false negatives (1.357%), although it is lower than the false positive rate observed when predicting the classes for the test data.

Considering the results above, I would still conclude that the SVM model used above is a relatively good predictor of new unseen data. However, it appears to result in a higher false positive rate than false negative rate. While this false positive rate appears to be relatively low in general, it appears to be significantly large in comparison to the false negative rate which is not desired.