APSTA-GE 2011. Homework #3

January 2025: Yael Beshaw

```r
# likely useful libraries and initial seed set.
library(caret)
library(cluster)
library(NbClust)
library(klaR)
library(ggplot2)
library(GGally)
library(e1071)
library(knitr)
library(LiblineaR)
set.seed(2011)
```

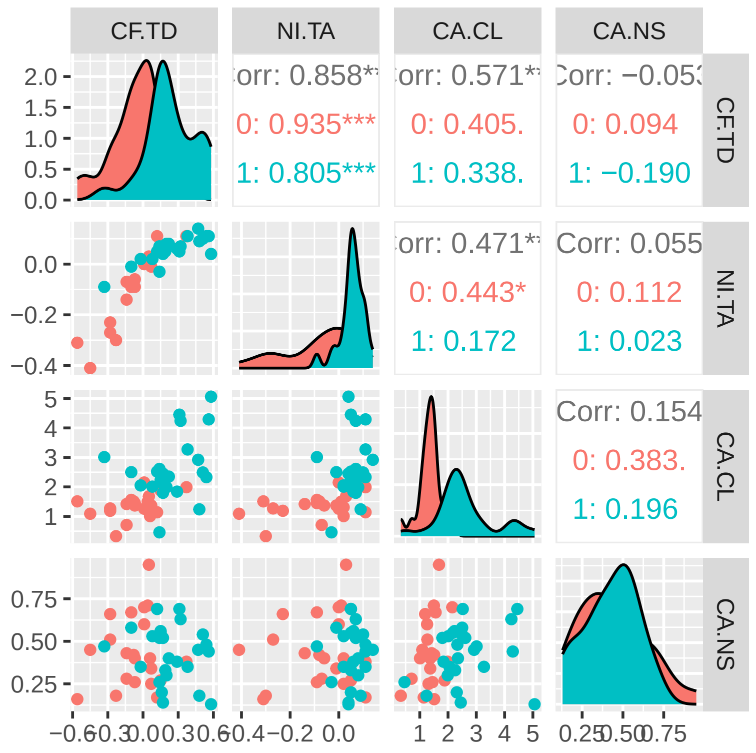
The dataset you will use for this classification problem is bankruptcy.csv, taken from Johnson and Wichern (chapter 11).

Annual financial data are collected for bankrupt firms approximately 2 years prior to their bankruptcy and for financially sound firms at about the same time. The data file contains five variables:

Population: bankruptcy status (0 = bankrupt; 1 = not bankrupt)
CF.TD: (cash flow)/(total debt)
NI.TA: (net income)/(total assets)
CA.CL: (current assets)/(current liabilities)
CA.NS: (current assets)/net sales)

The problem is to classify these firms based on these 4 features. We provide R code to read in the data, set the target outcome to be a factor, and visualize the multivariate relationships:

```r
dta <- data.frame(read.csv("/Users/yaelbeshaw/R Scripts and Projects/NYU-APSTA-GE-2011/bankruptcy.csv"))
dta$Population <- factor(dta$Population)
ggpairs(data = dta, columns = 2:5, mapping = ggplot2::aes(colour = Population))
```



Q1:

Based on this plot, how many points to expect to be difficult to classify [worth 1pt]

Based on this plot, at least 10 data points are expected to be difficult to classify. Taking a look at the correlations between the variables, the most difficult to classify seem to be between CF.TD and NI.TA as they have the highest correlation. There then seems to be a little more seperation as we move down the rows, and the plots of CA.NS, allow us to clearly guess how many points are overlapping. I was able to count 8 to 10 data points, indicating that at the very least 10 data points are expected to be difficult but there is a possibility that it could be more than 10.

Q2:

Use the NaiveBayes classifier (klaR package), no LaPlace correction; do not use kernel option. Fit to the entire dataset (no cross-validation).
Make predictions; compute and show the accuracy and confusion matrix [1 pt each; 3 pts total]

```r
# NaiveBayes Classifer
classsifer <- NaiveBayes(Population ~ ., data = dta, fL = 0,
    usekernel = FALSE)

# Predictions
preds <- predict(classsifer)

# Confusion Matrix
cM <- confusionMatrix(preds$class, dta$Population)

cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  3 24
##
##                Accuracy : 0.913
##                  95% CI : (0.7921, 0.9758)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 5.991e-08
##
##                   Kappa : 0.8234
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8571
##             Specificity : 0.9600
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4565
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 0
##
```

Accuracy of the NaiveBayes classifier in relation to the dataset is 91.3%.

Q3:

Repeat the classification using support vector machines (svm in e1071). Use a radial kernel. Make predictions; report the accuracy and the confusion matrix [1 pt. each; 3 pts total].

```r
# SVM Classifier
svm_classifer <- svm(Population ~ ., data = dta, kernel = "radial")

# Predictions
svm_predictions <- predict(svm_classifer)

# Confusion Matrix
confusionMatrix(svm_predictions, dta$Population)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  3 24
##
##                Accuracy : 0.913
##                  95% CI : (0.7921, 0.9758)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 5.991e-08
```

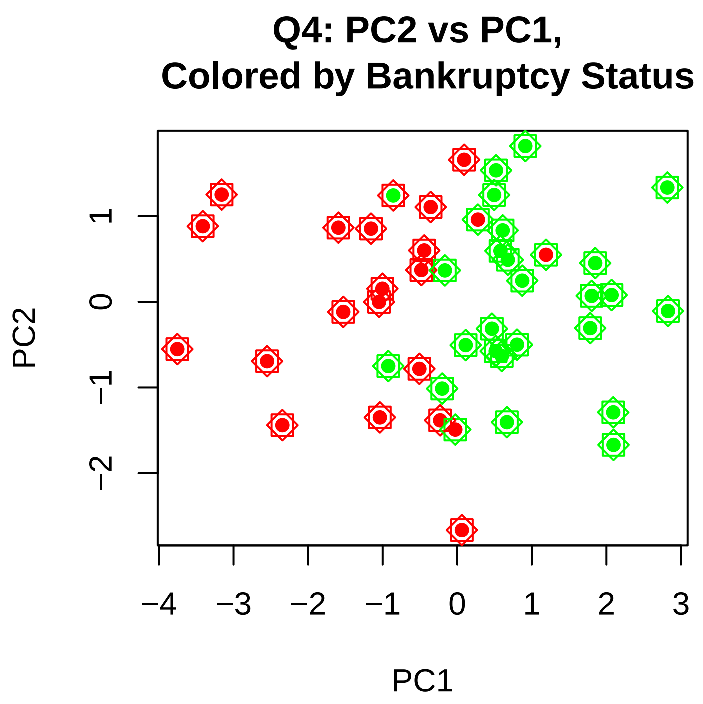
```
##
##                   Kappa : 0.8234
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8571
##             Specificity : 0.9600
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4565
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 0
##
```

Accuracy of the SVM classifier in relation to the dataset is also 91.3%.

Q4:

Compute the principal component scores, and use the first 2 PCs to plot (visualize) the data, using red and green colors to indicate the 0/1 Population variable. Superimpose the prediction (color) for the NB and SVM models, using a different symbol for each to reveal the points of confusion. [1 pt. each; 3 pts total] In R, I recommend pch 16, 0, 5, respectively, for raw, nb.pred, svm.pred, and cex=1.5 for the latter two.

```r
pc.dta <- princomp(dta[, 2:5], cor = TRUE)$scores

# Raw Plot
plot(pc.dta[, 1:2], col = ifelse(dta$Population == 0, "red",
    "green"), pch = 16, xlab = "PC1", ylab = "PC2", main = "Q4: PC2 vs PC1, \n Colored by Bankruptcy Sta

# Superimpose Predictions from Naive Bayes

## When pred$class = 0 across PC1 and PC2
points(pc.dta[preds$class == 0, 1], pc.dta[preds$class == 0,
    2], col = "red", pch = 0, cex = 1.5)

## When pred$class = 1 across PC1 and PC2
points(pc.dta[preds$class == 1, 1], pc.dta[preds$class == 1,
    2], col = "green", pch = 0, cex = 1.5)

# Superimpose Predictions from SVM

## When svm_predictions = 0 across PC1 and PC2
points(pc.dta[svm_predictions == 0, 1], pc.dta[svm_predictions ==
    0, 2], col = "red", pch = 5, cex = 1.5)

## When svm_predictions = 1 across PC1 and PC2
points(pc.dta[svm_predictions == 1, 1], pc.dta[svm_predictions ==
    1, 2], col = "green", pch = 5, cex = 1.5)
```



Q5 :

Use 10-fold cross-validation (CV) and the caret package to repeat 20 times, 90/10 CV runs of NB, reporting the out-of-sample accuracy (be sure to report which value is the accuracy), as well as the s.d. of the accuracy, use predictions to produce a confusion matrix from the optimal model returned by caret. [2 pts each; 6 pts total]

```
x <- model.matrix(~-1 + CF.TD + NI.TA + CA.CL + CA.NS, data = dta)
y <- dta$Population

CV_NB <- train(x, y, method = "nb", trControl = trainControl(method = "repeatedcv",
    number = 10, repeats = 20))

# Out of Sample Accuracy
print(CV_NB$results$Accuracy[1])
```

```
## [1] 0.8724167
```

```
# Out of Sample Accuracy, SD
print(CV_NB$results$AccuracySD[1])
```

```
## [1] 0.1536418
```

```
# Use Predictions to Create Confusion Matrix The Optimal
# Model is fl=0, usekernal= FALSE, adjust=1
print(CV_NB$bestTune)
```

```
##   fL usekernel adjust
## 1  0     FALSE      1
```

```r
opt_model.NB <- NaiveBayes(Population ~ ., data = dta, fL = 0,
    usekernel = FALSE, adjust = 1)
# Predictions
opt_predict.NB <- predict(opt_model.NB)

# Confusion Matrix
confusionMatrix(opt_predict.NB$class, dta$Population)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  3 24
##
##                Accuracy : 0.913
##                  95% CI : (0.7921, 0.9758)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 5.991e-08
##
##                   Kappa : 0.8234
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8571
##             Specificity : 0.9600
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4565
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 0
##
```

The out of sample accuracy is 86.66% (usekernal=FALSE), the standard deviation of this accuracy is 0.1492. As shown by (CV_NB$bestTune), the best model is one where fl=0, usekernal= FALSE, and adjust=1. Thus, I recreated the model, created predictions, and produced a confusion matrix with our variable of interest and the predictions of the new model.

Q6:

Use 10-fold cross-validation (CV) and the caret package to repeat 20 times, 90/10 CV runs of SVM, reporting the out-of-sample accuracy (be sure to report which value is the accuracy), as well as the s.d. of the accuracy, use predictions to produce a confusion matrix from the optimal model returned by caret. [2 pts each; 6 pts total]

```r
x <- model.matrix(~-1 + CF.TD + NI.TA + CA.CL + CA.NS, data = dta)
y <- dta$Population

CV_SVM <- train(x, y, method = "svmRadial", trControl = trainControl(method = "repeatedcv",
    number = 10, repeats = 20))
```

```
# Out of Sample Accuracy
print(CV_SVM$results$Accuracy[3])
```

```
## [1] 0.8795
```

```
# Out of Sample Accuracy, SD
print(CV_SVM$results$AccuracySD[3])
```

```
## [1] 0.1416453
```

```
# Use Predictions to Create Confusion Matrix The Optimal
# Model is sigma=0.3840564, C=1
print(CV_SVM$bestTune)
```

```
##       sigma C
## 3 0.3902214 1
```

```
opt_model.SVM <- svm(Population ~ ., data = dta, kernel = "radial",
    sigma = 0.3840564, C = 1)
# Predictions
opt_predict.SVM <- predict(opt_model.SVM)

# Confusion Matrix
confusionMatrix(opt_predict.SVM, dta$Population)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  3 24
##
##                Accuracy : 0.913
##                  95% CI : (0.7921, 0.9758)
##     No Information Rate : 0.5435
##     P-Value [Acc > NIR] : 5.991e-08
##
##                   Kappa : 0.8234
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8571
##             Specificity : 0.9600
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8889
##              Prevalence : 0.4565
##          Detection Rate : 0.3913
##    Detection Prevalence : 0.4130
##       Balanced Accuracy : 0.9086
##
##        'Positive' Class : 0
##
```

The out of sample accuracy is 87.95% (C=1), the standard deviation of this accuracy is 0.1416. As shown by (CV_SVM$bestTune), the best model is one where sigma = 0.3840564 and C = 1. Thus, I recreated the model, created predictions, and produced a confusion matrix with our variable of interest and the predictions of the new model.

Q7:

Compare the cross-validation results of the two algorithms (nb v. svm) – if the results differ explain why it might be so. For the SVM results, compare also the cross-validation-based classification (from the predict function) to the previous, non-cross-validated one (also using predict), and explain what we are doing differently (between the two approaches) that is improving the a posteriori classification (from predict function). Report the features and outcome for any points that are now correctly classified and were not previously (if you find any). [2 pts each comparison; 1 pt. for TAexplanation; 5 pts total].

There was a slight difference in the cross validation results of the two algorithms. For NB, we see that we have an out-of-sample accuracy of 86.66% with a standard deviation of 0.1492. With the SVM model we have an out-of-sample accuarcy of 87.95% with a standard deviation of 0.1416. We see that the SVM model is more accurate and it may be due to its methodology of finding the optimal parameters.

What we are doing differently between the two SVM approaches is now identifying C and sigma such that the model is the most optimal. Because our cross-validated model utilized C=1, which is the highest, we are decreasing possibilities for mis-classification. We also specify sigma = 0.3840564, allowing us to avoid overfitting. We did not specify these parameters with the non-cross validated model which would give way to improvement in the posteriori classification.

However, there is no difference between the accuracy we see in the confusion matrix, which is 91.3% for both versions. Indicating that while the cross-validated model does well out of sample, there is no difference in its predictive capabilities for in-sample.