HW2: KNN classifier

Benjamin Hillmann

CSCI 5461 Fall 2016

Question 1

Implement a kNN classifier, using the Pearson correlation coefficient as the similarity metric between cell lines' expression profiles. For each of the 5 drugs, you should create a classifier that takes a cell line expression profile as input and produces a score that predicts whether it is sensitive or resistant to the given drug. More specifically, your classifier for each drug should take a gene expression profile for a cell line of interest, compute the pairwise Pearson correlation coefficients between the test cell line and each of the training cell lines to find its k nearest neighbors. The prediction score assigned to the test cell line should then be calculated based on the drug sensitivity labels of its k nearest neighbors (e.g., the number or the fraction of drug-sensitive neighbors among the k total). Note that, for the classifier that you create for a given drug, the cell lines with missing measurements (NA values) should be excluded.

Answer

The KNN classifier for question 1 is implemented in the *R* folder in the script *bh_knn.R*.

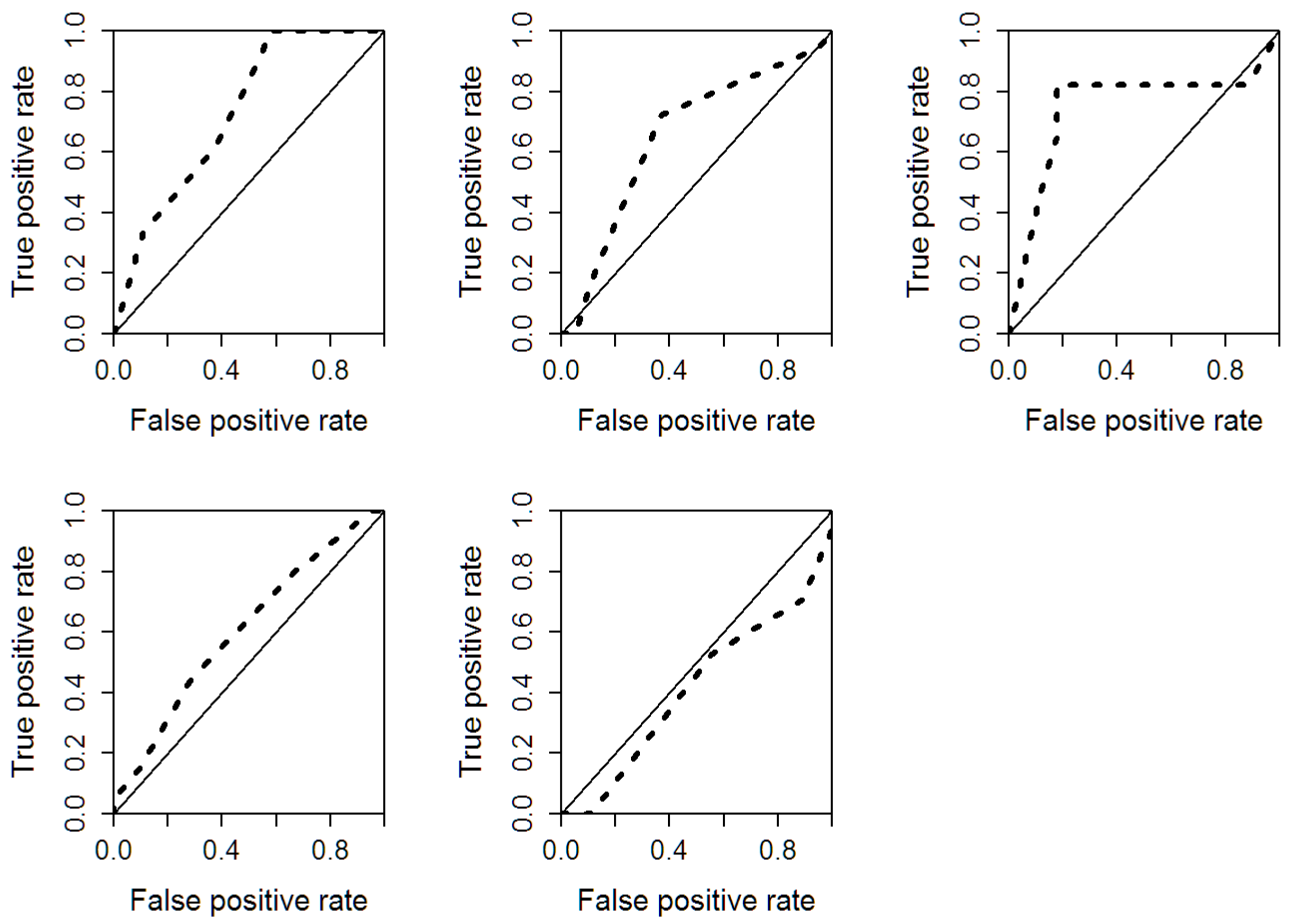
Question 2

kNN performance evaluation (30 points): Set k=5 for all of the evaluations in this problem. Apply leave-one-out cross-validation (LOOCV) to measure the performance of your classifier. For each of the 5 drugs, plot an ROC curve with sensitivity on the y-axis and (1-specificity) on the x-axis. Use the predicted scores based on the number of drug-sensitive neighbors (among the k total) for each cell line to rank the predictions in order to draw the ROC curve. For each ROC curve, be sure to plot the performance expected by a random classifier. Answer the following questions: (a) Does your classifier work better than a random classifier? For which drugs? Refer to specific evidence from your analysis to justify your answer. (b) For the drug on which your approach appears to work the best, how good is the classification performance? Pick a point on the ROC curve, and report the relevant metrics (true positives, false positives).

Answer

```r
# Create the classifier, set K=5
classifier = function(train, test, labels) {bh_knn(train, test, labels, k=5)}

# For storing KNN sensitivity results array
results = array(NA, dim(labels))
for (i in 1:nrow(labels)) {
  # Get the labels for the drug
  cl = labels[i,]
  # Get all values
  cl.keep = !is.na(cl)
  # Remove samples without labeled data
  subset.gene = t(gene.exp[,cl.keep])
  # The actually dense array
  cl.kept = cl[cl.keep]
  # Run leave-one-out cross-validation
  results[i,][cl.keep] = bh_loocv(classifier, subset.gene, cl.kept)
}
```

```r
# Create vector to store AUC results
par(mfrow=c(2,3))
auc.results = 1:nrow(labels)
for (i in 1:nrow(labels)) {
  # Plot and return AUC
  auc.results[i] = bh_plot_auc(results[i,][!is.na(labels[i,])], labels[i,][!is.na(labels
[i,])])$auc
}
par(mfrow=c(1,1))
```



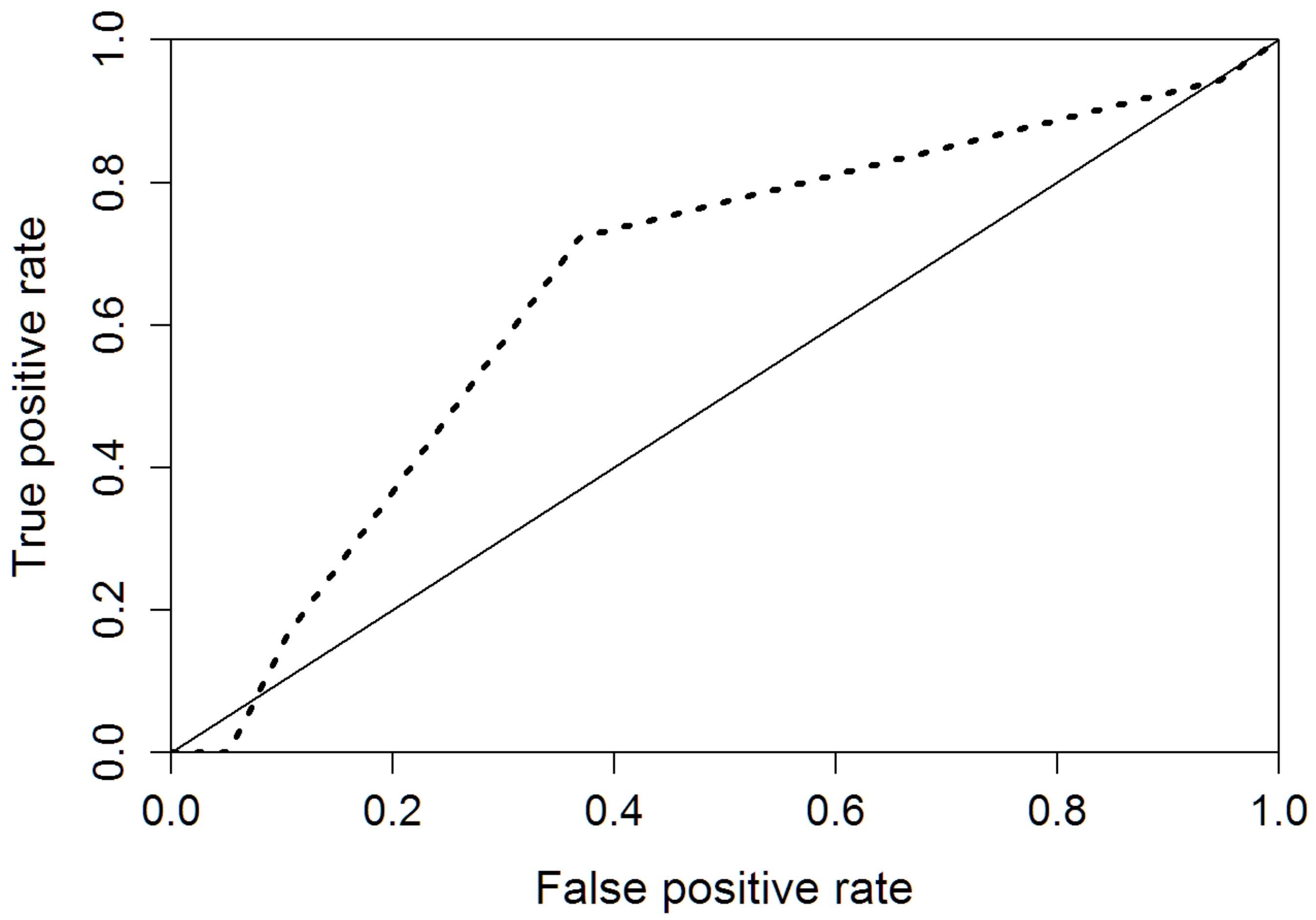
```
# The auc for each drug
# We beat random for every drug but the last one. That one is 4-HC(DNA alkylator). The fo
ur that beat random, (auc > .5) are Mebendazole(Tubulin), Methylglyoxol(Pyruvate), Disulf
iram(ALDH2)  and Everolimus(mTOR). This evidence but the area under the curve in the auc
plots beating greater than the line x=y.
auc.results
```

```
## [1] 0.7309942 0.6549708 0.7491349 0.6095238 0.4150327
```

```
# The average auc for the results
mean(auc.results)
```

```
## [1] 0.6319313
```

```
# How ROCR works, I calculate the TP and FP after the fact
# Will pick a point on this auc plot for Disulfiram(ALDH2).
auc.rates = bh_plot_auc(results[2,][!is.na(labels[2,])], labels[2,][!is.na(labels[2,])])
```



```
tp = sum(labels[2,][!is.na(labels[2,])] == 0)*auc.rates$tp[3]
fp = sum(labels[2,][!is.na(labels[2,])] == 1)*auc.rates$fp[3]
tp
```

```
## [1] 57
```

```
fp
```

```
## [1] 36
```

```
#Point for second plot
tp/(tp+fp)
```

```
## [1] 0.6129032
```

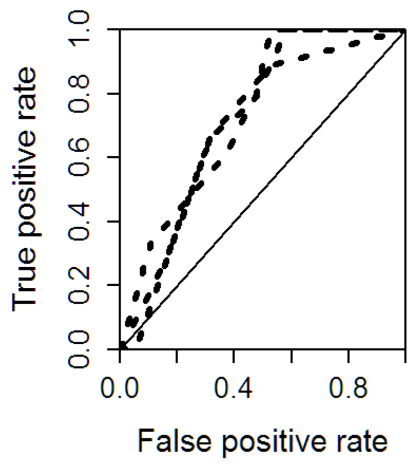
```
fp/(tp+fp)
```

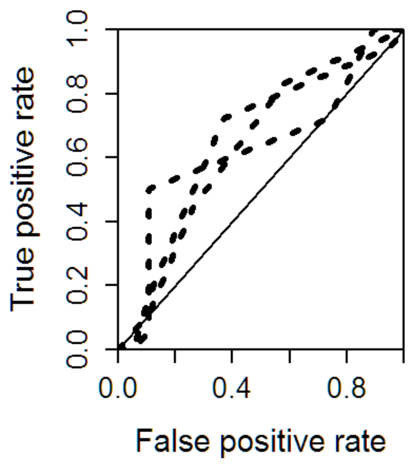
```
## [1] 0.3870968
```

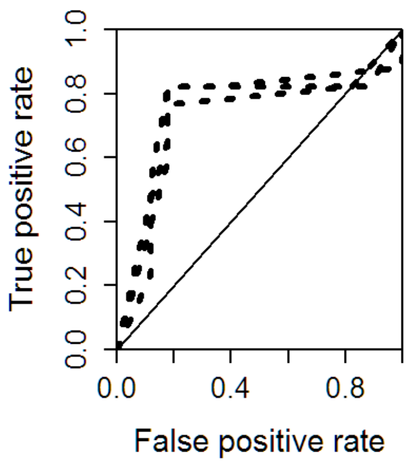
Question 3

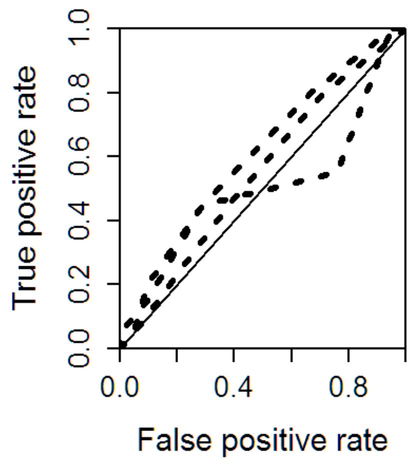
Exploration of parameters affecting kNN performance (30 points): For this problem, we will explore how the performance is influenced by the key parameter in the kNN classifier, k, and the scoring function. (a) Rerun your classification results with k=3,5,7. Again, sort the predicted scores for the cell lines based on the number of their nearest neighbors that are drug-sensitive (similar to Problem 2). For each drug, create a single figure, but plot the ROC curves for all values of k on the same curve, and report the area under the curve (AUC) for each ROC curve. Discuss the results of your analysis. Does the choice of k affect the performance of the classifier? (b) Set k=5. Instead of scoring each cell line with the number of its nearest neighbors that are drug-sensitive, use the sign weighted score. Sort the predictions of drug-sensitivity by this score. Plot new ROC curves for each drug, and report the area under the curve for the new ROC curves, comparing the approach from Problem 2 with this approach.

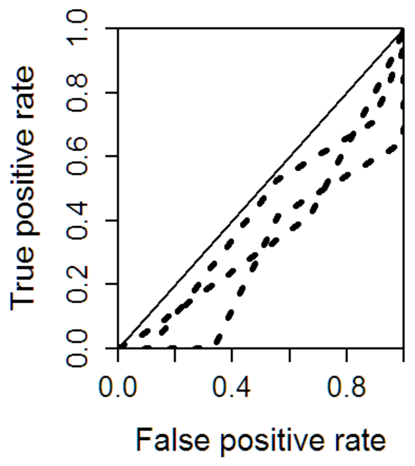
```r
# Set up the AUC results array
k.aucs = array(NA, c(nrow(labels),3))
par(mfrow=c(2,3))
for (i in 1:nrow(labels)) {
  # Get the labels for the drug
  cl = labels[i,]
  # Get all values
  cl.keep = !is.na(cl)
  # Remove samples without labeled data
  subset.gene = t(gene.exp[,cl.keep])
  # The actually dense array
  cl.kept = cl[cl.keep]
  k.auc.results = array(NA, c(length(cl.kept), 3))
  ks = c(3,5,7)
  for (j in 1:length(ks)) {
    k.temp = ks[j]
    classifier.temp = function(train, test, labels) {
      bh_knn(train, test, labels, k=k.temp)
    }
    # Run leave-one-out cross-validation
    k.auc.results[,j] = bh_loocv(classifier.temp, subset.gene, cl.kept)
  }
  k.aucs[i,] = bh_plot_auc(k.auc.results, replicate(3, cl.kept))$auc
}
par(mfrow=c(1,1))
```











```
# Rows are drugs
# Columnes are c(3,5,7)
k.aucs
```

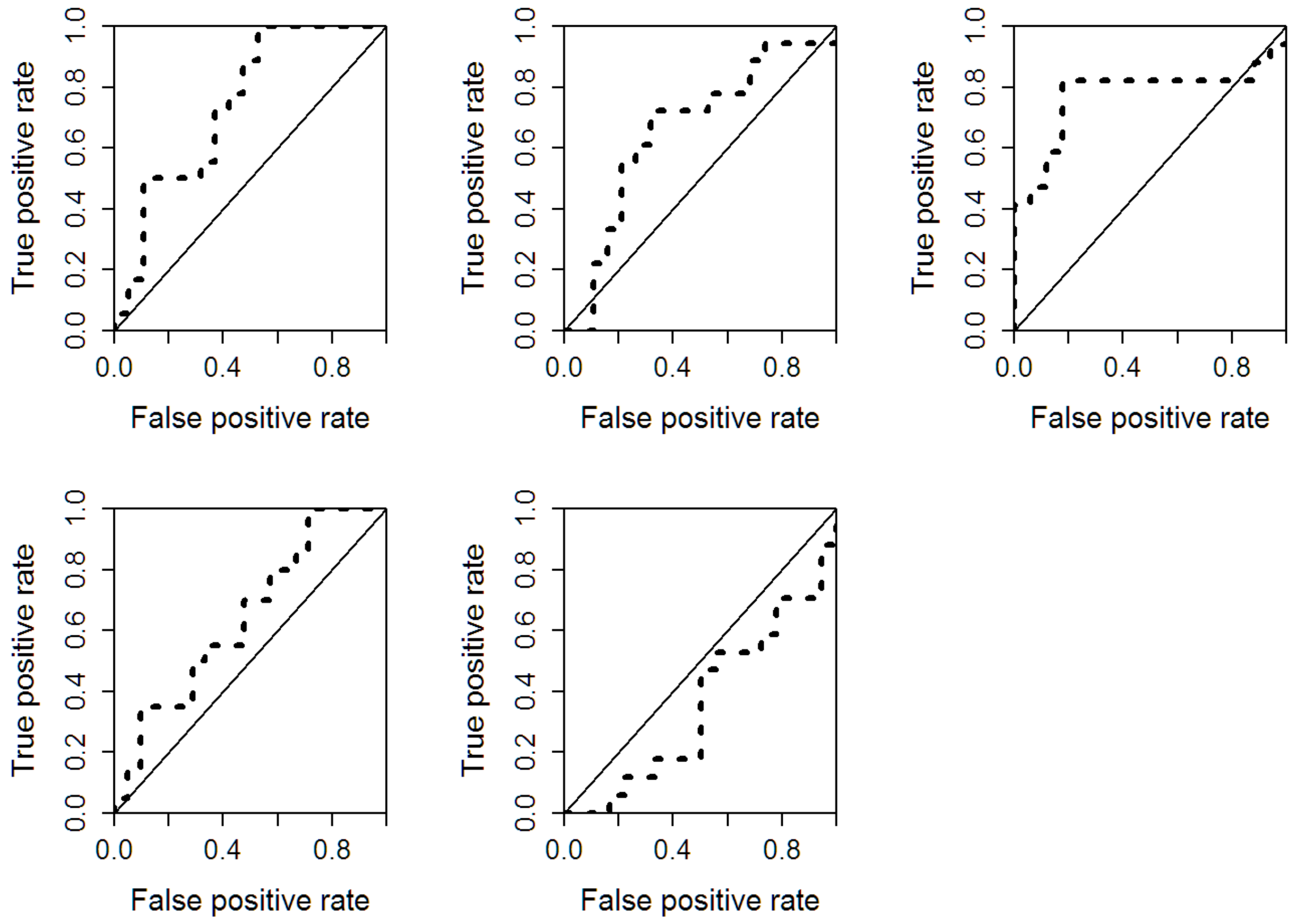
```
##           [,1]      [,2]      [,3]
## [1,] 0.6988304 0.7309942 0.7309942
## [2,] 0.6374269 0.6549708 0.6286550
## [3,] 0.7266436 0.7491349 0.7508651
## [4,] 0.5500000 0.6095238 0.5035714
## [5,] 0.3660131 0.4150327 0.2810458
```

```
apply(k.aucs, 2, mean)
```

```
## [1] 0.5957828 0.6319313 0.5790263
```

```
# It appears that changing K does give the classifer slightly different results. The samp
le size might be too small to really tell though. K=5 is the best for this scoring metri
c.
```

```r
# The sign classifier (uses a different package)
classifier.sign = function(train, test, labels) {
  bh_knn_sign(train, test, labels, k=5)
}

par(mfrow=c(2,3))
# The results array
results.sign = array(NA, dim(labels))
for (i in 1:nrow(labels)) {
  cl = labels[i,]
  cl.keep = !is.na(cl)
  subset.gene = t(gene.exp[,cl.keep])
  cl.kept = cl[cl.keep]
  results.sign[i,][cl.keep] = bh_loocv(classifier.sign, subset.gene, cl.kept)
}

auc.results.sign = 1:nrow(labels)
for (i in 1:nrow(labels)) {
  auc.results.sign[i] = bh_plot_auc(results.sign[i,][!is.na(labels[i,])], labels[i,][!is.
na(labels[i,])])$auc
}

par(mfrow=c(1,1))
```



```
auc.results.sign
```

```
## [1] 0.7456140 0.6608187 0.7750865 0.6476190 0.3594771
```

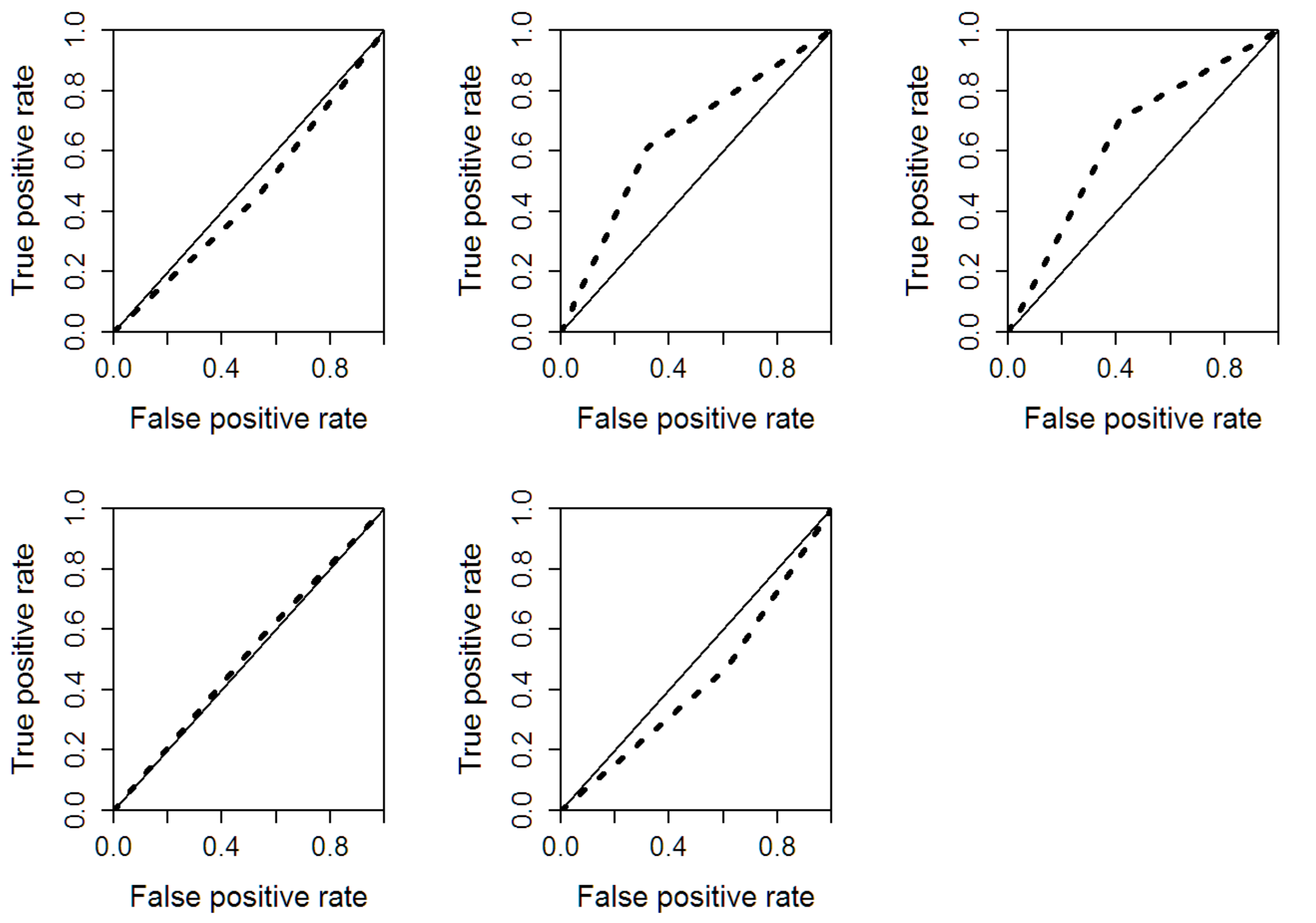
```
# Average the results
mean(auc.results.sign)
```

```
## [1] 0.6377231
```

```
# The advanced scoring metric seems to slightly outclass the weighted voting scheme. It p
robably has to do with the smoothing the sign function provides. The sample size is small
again, and we don't have enough power to really say.
```

Extra Credit 1

```r
library(randomForest)

k = 10
par(mfrow=c(2,3))
auc.results.rf = 1:nrow(labels)

# The results array
for (i in 1:nrow(labels)) {
  cl = labels[i,]
  cl.keep = !is.na(cl)
  subset.gene = gene.exp[,cl.keep]
  cl.kept = cl[cl.keep]
  data <- as.data.frame(prcomp(as.data.frame(subset.gene), center=T, scale=T)$rotation[,
1:10])
  colnames(data) = paste("x", colnames(data), sep="")
  rownames(data) <- paste("x", rownames(data), sep="")
  data$response <- as.factor(cl.kept)


  ids <- sample(1:k, nrow(data), replace = TRUE)
  list <- 1:k

  # prediction and testset data frames that we add to with each iteration over
  # the folds

  prediction <- data.frame(Predicted=c())
  testsetCopy <- data.frame(Actual=c())

  for (j in 1:k){
    # remove rows with id i from dataframe to create training set
    # select rows with id i to create test set
    trainingset <- subset(data, ids %in% list[-j])
    testset <- subset(data, ids %in% c(j))

    # run a random forest model
    mymodel <- randomForest(response ~ ., data = trainingset, ntree = 100)

    # remove response column 1
    temp <- as.data.frame(predict(mymodel, testset[,!(colnames(testset) == 'response')]))
    # append this iteration's predictions to the end of the prediction data frame
    prediction <- rbind(prediction, temp)

    # append this iteration's test set to the test set copy data frame
    testsetCopy <- rbind(testsetCopy, as.data.frame(testset$response))
  }

  # add predictions and actual Sepal Length values
  result <- cbind(prediction, testsetCopy[,1])
  names(result) <- c("Predicted", "Actual")
  result$Difference <- abs(result$Actual == result$Predicted)# add predictions and actual
Sepal Length values
  result <- cbind(prediction, testsetCopy[,1])
  names(result) <- c("Predicted", "Actual")
  result$Difference <- abs(result$Actual == result$Predicted)
  auc.results.rf[i] = bh_plot_auc(as.numeric(result$Predicted), as.numeric(result$Actua
l))$auc
}
par(mfrow=c(1,1))
```



```
auc.results.rf
```

```
## [1] 0.4590643 0.6476608 0.6470588 0.5142857 0.4297386
```

```
# Average the results
# Looks like KNN beats RF
mean(auc.results.rf)
```

```
## [1] 0.5395616
```