HW2

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.

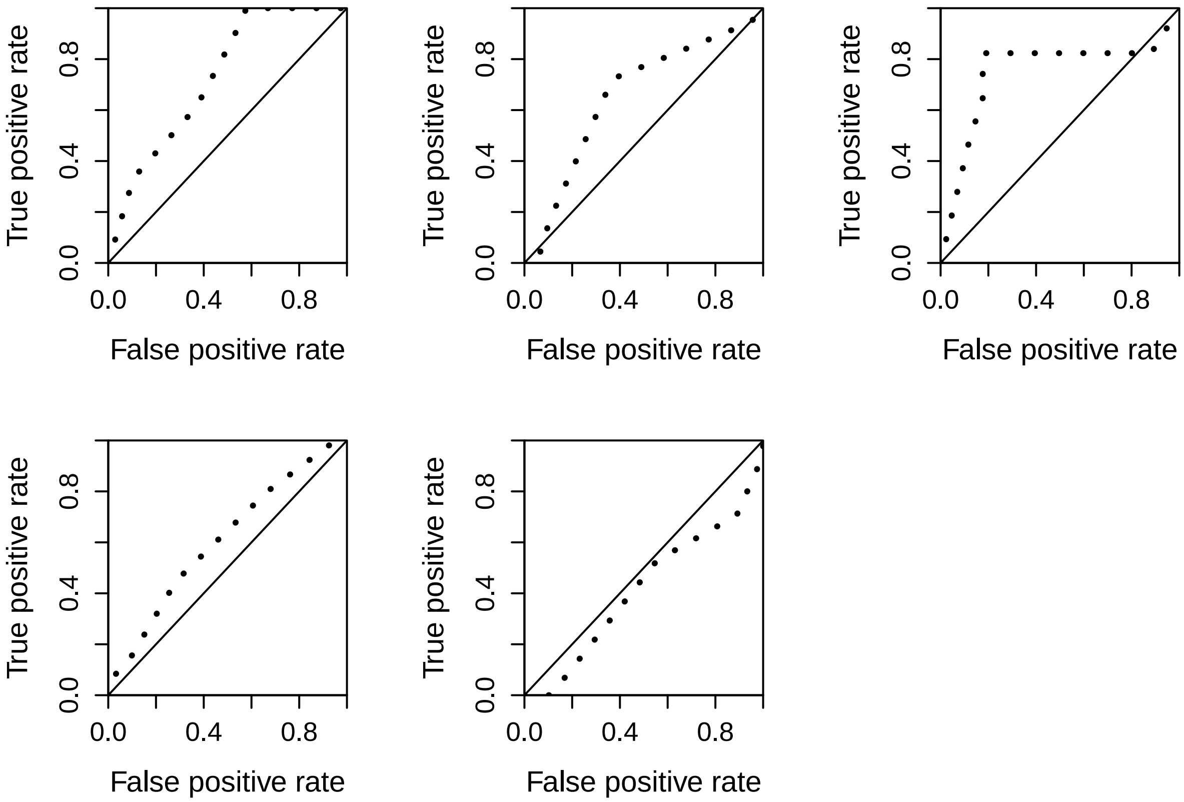
Answer

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

```
# Create the classifier
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)
}
```

```
# 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))
```



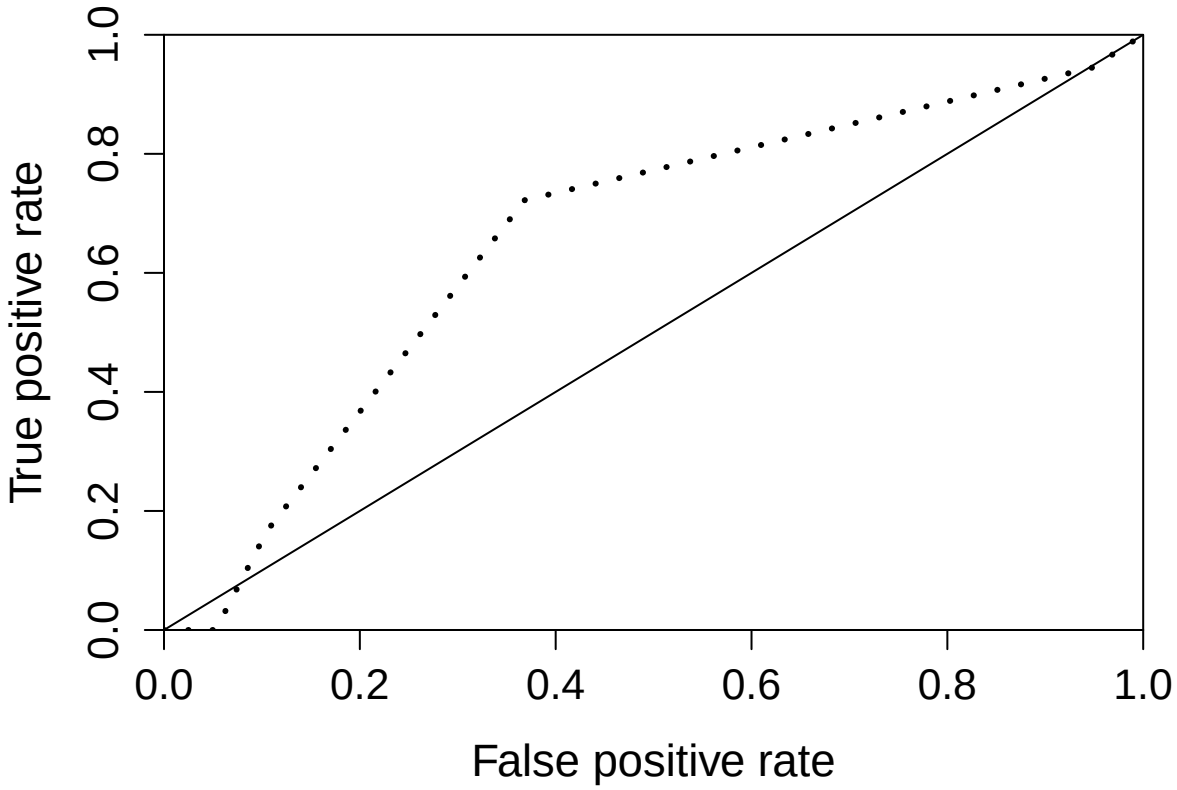

```r
# The results
auc.results
```

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

```r
mean(auc.results)
```

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

```r
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)
```

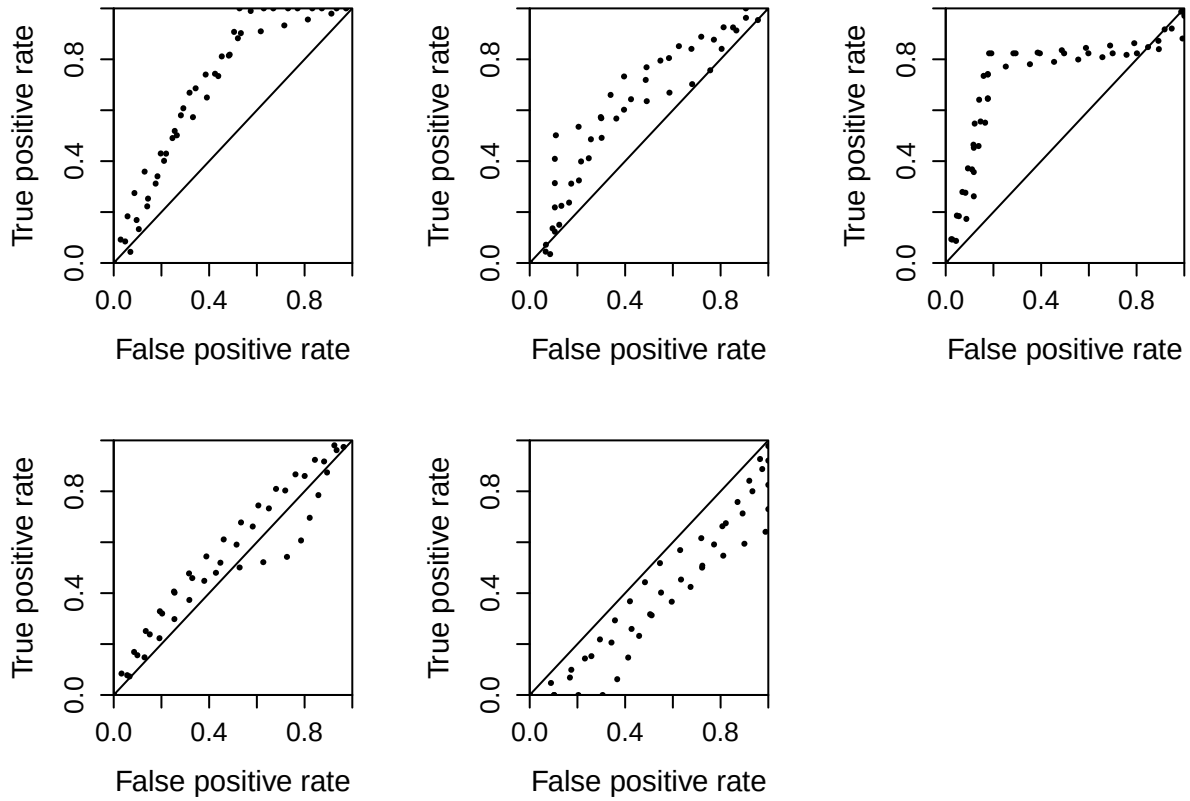
```
## [1] 0.6129032
```

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

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

```
# 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))
}
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)
```

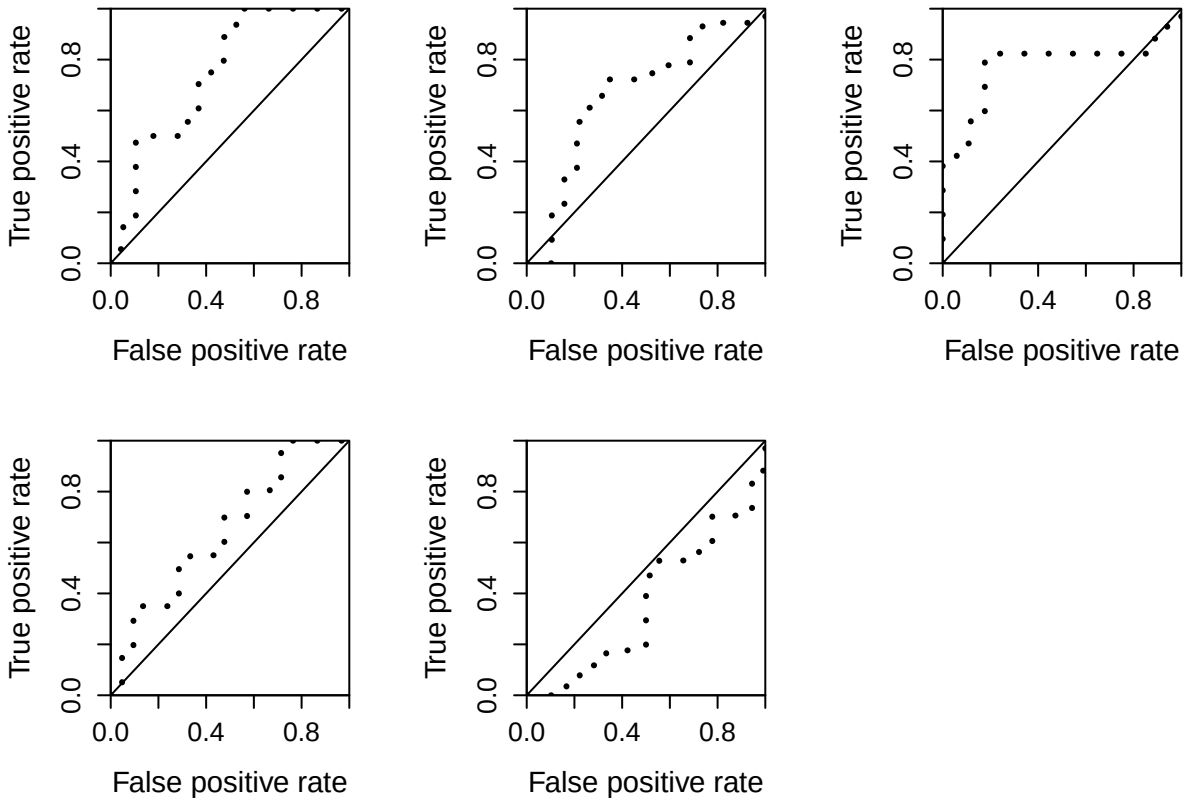
```
## [1] 0.5957828 0.6319313 0.5790263
```

```
# 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,])]
}

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
```