HIV-1 protease cleavage site in-silico prediction

Julio M. Fernandez

28/10/2017

Contents

1. Algorithm k-NN

The k-NN algorithm is a classification method that intents to match, or classify, an unknown subject to a category or label based on its similarity to other related and well known elements. k-nn is a classification by similarity algorithm. The total numbers of elements to consider when classifying is given by k.

Comparison between unknown and known subjects is carried out by distance, where a set of numerical parameters from both subjects are compared using a series of methods.

Distance calculation methods can use the Euclidian or Hamming methods, among others.The Euclidean method uses the quadratic pythagorean equation to calculate the direct distance between two points from a set of coordinates. The Hamming distance is used when comparing binary numbers and it is based in the direct count of common binary digits among numbers.

Strengths	Weakness
* Simple and affective	* It is not a model base procedure
* Makes no assumptions about the distribution of the data	* Requires selection of k
* Fast training phase	* Slow classification phase
	* Requires data processing

2. Article reading and analysis

The related article can be found at:

https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu810

3. Orthogonal codification function

For this section, a set of three functions have been created. Each one of them takes care of a specific step in the codification process. Splitting the solution to this problem in three methods or functions allows us to make use of the *lapply* function without occurring into custom *for* loops.

This first function will create a binary number of 20 digits and switch to "1" the position indicated by the parameter *x*. The function will return a 20 digit strings of zeros and ones (19+1).

```
setActiveAminoAcidInBinaryMask <- function(x, n) {
    mask <- rep(0, n)
    mask[x] = 1
    seq <- gsub(", ", "", toString(mask))

    return(seq)
}
```

The next function will convert a given amino acid sequence into binary sequence by using an amino acid sequence pattern. The function returns a list of string binary sequences.

```
aminoAcidSequenceToBinary <- function(sequence, aminoAcidPattern) {
    sequenceVector <- strsplit(sequence, "")[[1]]
    n <- nchar(aminoAcidPattern)
    aminoAcidPatternVector <- strsplit(aminoAcidPattern, "")[[1]]

    aminoAcidPositionInPatternVector <- match(sequenceVector, aminoAcidPatternVector)
    bin <- lapply(aminoAcidPositionInPatternVector, setActiveAminoAcidInBinaryMask,
        n = n)

    return(bin)
}
```

This last function will convert a binary string into a "digit by digit" binary vector. It accepts an amino acid sequence and a pattern. It returns a vector with 20 elements, one per binary element in the sequence. This is our main method and entry point for amino acid to binary conversion.

```
toOrthogonalCode <- function(x, aminoAcidPattern) {
    codec <- aminoAcidSequenceToBinary(x, aminoAcidPattern)
    strCode <- paste(unlist(codec), collapse = "")
    y <- strsplit(strCode, "")[[1]]

    return(as.numeric(y))
}
```

4. k-NN cassification script

a) Data reading

We first load the data file into a *dataframe*, rename the labels and result column and turn the results into a more readable factor.

```
impensData <- read.csv("impensData.txt", header = FALSE)
impensData$V2 <- factor(impensData$V2, levels = c(-1, 1), labels = c("Uncleaved",
    "Cleaved"))
colnames(impensData) <- c("sequence", "result")
head(impensData)
```

```
##   sequence    result
## 1 AAAGKSGG Uncleaved
## 2 AAAVDAGM Uncleaved
## 3 AAGKSGGG Uncleaved
## 4 AALALEYG   Cleaved
## 5 AANDGPMP Uncleaved
## 6 AASAAAVD Uncleaved
```

b) Orthogonal codification

From the sequence column of our main *dataframe*, we generate a coded matrix based on the orthogonal sequence algorithm developed in the previous step.

This is a two steps procedure where we first get a list of binary sequences from the algorithm and then transform the sequence list into a matrix.

```
aminoAcidPattern <- "ARNDCQEGHILKMFPSTWYV"

impensDataBinaryList <- lapply(as.vector(impensData$sequence), toOrthogonalCode,
    aminoAcidPattern = aminoAcidPattern)
impensDatabinaryMatrix <- do.call(rbind, impensDataBinaryList)
```

We now merge the current data *dataframe* with the coded animo acid binary sequence (because of space constraints, a partial view of the final *dataframe* is displayed).

```
impensData <- data.frame(impensData, impensDatabinaryMatrix)
head(impensData[, 1:10])
```

```
##   sequence    result X1 X2 X3 X4 X5 X6 X7 X8
## 1 AAAGKSGG Uncleaved  1  0  0  0  0  0  0  0
## 2 AAAVDAGM Uncleaved  1  0  0  0  0  0  0  0
## 3 AAGKSGGG Uncleaved  1  0  0  0  0  0  0  0
## 4 AALALEYG   Cleaved  1  0  0  0  0  0  0  0
## 5 AANDGPMP Uncleaved  1  0  0  0  0  0  0  0
## 6 AASAAAVD Uncleaved  1  0  0  0  0  0  0  0
```

Let's now take a look at the result column proportions.

```
round(prop.table(table(impensData$result)) * 100, digits = 1)
```

```
##
## Uncleaved   Cleaved
##      84.3      15.7
```

Finally, we export the data into its own CSV file for further use.

```
write.csv(impensData, file = "orthoImpensData.csv")
```

c) Generate training and testing data from main data set.

In order to generate our training and testing data, we opted to use the *createDataPartition* function from the *caret* package. This function ensures that both data sets have a balanced number of positive and negative outcomes. We will also create a label variable per set with the outcome.

```
p <- 0.67

set.seed(123)
in_train <- createDataPartition(impensData$result, p = p, list = FALSE)

impensData_train <- impensData[in_train, ]
impensData_train <- impensData_train[-c(1, 2)]

impensData_test <- impensData[-in_train, ]
impensData_test <- impensData_test[-c(1, 2)]

impensData_train_labels <- impensData[in_train, 2]
```

```
impensData_test_labels <- impensData[-in_train, 2]

head(impensData_train[, 1:10])
```

```
##    X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## 1   1  0  0  0  0  0  0  0  0   0
## 5   1  0  0  0  0  0  0  0  0   0
## 6   1  0  0  0  0  0  0  0  0   0
## 8   1  0  0  0  0  0  0  0  0   0
## 9   1  0  0  0  0  0  0  0  0   0
## 12  1  0  0  0  0  0  0  0  0   0
```

d) k-NN prediction test.

The following function applies the k-NN algorithm to a set of testing and training data and a series of k values. It returns a *dataframe* with the prediction results per value of k.

```
process_data_knn <- function(data_train, data_test, data_train_labels,
    k) {
    knn_results <- list()
    for (k_index in k) {
        knn_results[[length(knn_results) + 1]] <- knn(train = data_train,
            test = data_test, cl = data_train_labels, k = k_index,
            prob = TRUE)
    }
    predictions <- data.frame(knn_results)
    colnames(predictions) <- k

    return(predictions)
}
```

We now process our data for our set of k values.

```
k <- c(3, 5, 7, 11)

impensData_predictions <- process_data_knn(impensData_train, impensData_test,
    impensData_train_labels, k)
head(impensData_predictions)
```

```
##           3         5         7        11
## 1 Uncleaved Uncleaved Uncleaved Uncleaved
## 2 Uncleaved Uncleaved Uncleaved Uncleaved
## 3 Uncleaved Uncleaved Uncleaved Uncleaved
## 4 Uncleaved Uncleaved Uncleaved Uncleaved
## 5 Uncleaved Uncleaved Uncleaved Uncleaved
## 6 Uncleaved Uncleaved Uncleaved Uncleaved
```

e) k-NN test analysis

This first function generates a *CrossTable* report per prediction item stored in the prediction *dataset*.

```
crossTableResults <- function(predictions, test_labels) {
    for (i in names(predictions)) {
        cat("---------------------- K =", i, "----------------------\n\n")
```

```r
        CrossTable(x = test_labels, y = predictions[[i]], prop.chisq = FALSE)
        cat("\n\n\n")
    }
}
```

The following function generates a confusion matrix report per prediction item stored in the prediction *dataset*.

```r
confusionSummary <- function(predictions, labels, positive) {
    for (i in names(predictions)) {
        cat("---------------------- K =", i, "----------------------\n\n")
        print(confusionMatrix(predictions[[i]], labels, positive = positive))
        cat("\n\n\n")
    }
}
```

This function will generate a comparison table with estimators obtained from the confusion table.

```r
knnPerformanceSummary <- function(predictions, labels, positive) {
    summary <- data.frame()

    for (i in names(predictions)) {
        x <- confusionMatrix(predictions[[i]], labels, positive = positive)

        prob <- attr(predictions[[i]], "prob")
        pred <- prediction(predictions = prob, labels)
        perf.auc <- performance(pred, measure = "auc")

        tp = x[["table"]][2, 2]
        tn = x[["table"]][1, 1]
        fp = x[["table"]][2, 1]
        fn = x[["table"]][1, 2]

        kappa = round(x[["overall"]]["Kappa"], 3)
        accurary = round((tp + tn)/(tp + tn + fp + fn), 3)
        error = round(1 - accurary, 3)
        sensitivity = round(tp/(tp + fn), 3)
        specificity = round(tn/(tn + fp), 3)
        precision = round(tp/(tp + fp), 3)
        recall = round(tp/(tp + fn), 3)
        f = round((2 * precision * recall)/(recall + precision), 3)
        auc = round(unlist(perf.auc@y.values), 3)

        d <- data.frame(i, tp, tn, fp, fn, kappa, accurary, error,
            sensitivity, specificity, precision, recall, f, auc)
        summary <- rbind(summary, d)
    }

    rownames(summary) <- names(predictions)
    colnames(summary) <- c("k", "TP", "TN", "FP", "FN", "kappa", "Accurary",
        "Error", "Sensitivity", "Specificity", "Precision", "Recall",
        "F", "AUC")

    return(summary)
}
```

I - CrossTable

```r
crossTableResults(impensData_predictions, impensData_test_labels)
```

```
## ---------------------- K = 3 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  312
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       262 |         1 |       263 |
##              |     0.996 |     0.004 |     0.843 |
##              |     0.870 |     0.091 |           |
##              |     0.840 |     0.003 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |        39 |        10 |        49 |
##              |     0.796 |     0.204 |     0.157 |
##              |     0.130 |     0.909 |           |
##              |     0.125 |     0.032 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       301 |        11 |       312 |
##              |     0.965 |     0.035 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## ---------------------- K = 5 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  312
```

```
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       260 |         3 |       263 |
##              |     0.989 |     0.011 |     0.843 |
##              |     0.861 |     0.300 |           |
##              |     0.833 |     0.010 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |        42 |         7 |        49 |
##              |     0.857 |     0.143 |     0.157 |
##              |     0.139 |     0.700 |           |
##              |     0.135 |     0.022 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       302 |        10 |       312 |
##              |     0.968 |     0.032 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## ---------------------- K = 7 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  312
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       262 |         1 |       263 |
##              |     0.996 |     0.004 |     0.843 |
##              |     0.856 |     0.167 |           |
##              |     0.840 |     0.003 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |        44 |         5 |        49 |
##              |     0.898 |     0.102 |     0.157 |
##              |     0.144 |     0.833 |           |
##              |     0.141 |     0.016 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       306 |         6 |       312 |
##              |     0.981 |     0.019 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## ---------------------- K = 11 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  312
##
##
##               | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       262 |         1 |       263 |
##              |     0.996 |     0.004 |     0.843 |
##              |     0.853 |     0.200 |           |
##              |     0.840 |     0.003 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |        45 |         4 |        49 |
##              |     0.918 |     0.082 |     0.157 |
##              |     0.147 |     0.800 |           |
##              |     0.144 |     0.013 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       307 |         5 |       312 |
##              |     0.984 |     0.016 |           |
## -------------|-----------|-----------|-----------|
##
##
```

II - Confusion table

```
confusionSummary(impensData_predictions, impensData_test_labels, "Cleaved")
```

```
## ---------------------- K = 3 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       262      39
##   Cleaved           1      10
##
```

```
##                Accuracy : 0.8718
##                  95% CI : (0.8295, 0.9068)
##     No Information Rate : 0.8429
##     P-Value [Acc > NIR] : 0.09049
##
##                   Kappa : 0.2926
##  Mcnemar's Test P-Value : 4.909e-09
##
##             Sensitivity : 0.20408
##             Specificity : 0.99620
##          Pos Pred Value : 0.90909
##          Neg Pred Value : 0.87043
##              Prevalence : 0.15705
##          Detection Rate : 0.03205
##    Detection Prevalence : 0.03526
##       Balanced Accuracy : 0.60014
##
##        'Positive' Class : Cleaved
##
##
##
##
## --------------------- K = 5 ---------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       260      42
##   Cleaved           3       7
##
##                Accuracy : 0.8558
##                  95% CI : (0.8118, 0.8928)
##     No Information Rate : 0.8429
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1944
##  Mcnemar's Test P-Value : 1.473e-08
##
##             Sensitivity : 0.14286
##             Specificity : 0.98859
##          Pos Pred Value : 0.70000
##          Neg Pred Value : 0.86093
##              Prevalence : 0.15705
##          Detection Rate : 0.02244
##    Detection Prevalence : 0.03205
##       Balanced Accuracy : 0.56573
##
##        'Positive' Class : Cleaved
##
##
##
##
## --------------------- K = 7 ---------------------
```

```
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       262      44
##   Cleaved           1       5
##
##                Accuracy : 0.8558
##                  95% CI : (0.8118, 0.8928)
##     No Information Rate : 0.8429
##     P-Value [Acc > NIR] : 0.2974
##
##                   Kappa : 0.1528
##  Mcnemar's Test P-Value : 3.825e-10
##
##             Sensitivity : 0.10204
##             Specificity : 0.99620
##          Pos Pred Value : 0.83333
##          Neg Pred Value : 0.85621
##              Prevalence : 0.15705
##          Detection Rate : 0.01603
##    Detection Prevalence : 0.01923
##       Balanced Accuracy : 0.54912
##
##        'Positive' Class : Cleaved
##
##
##
##
## ---------------------- K = 11 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       262      45
##   Cleaved           1       4
##
##                Accuracy : 0.8526
##                  95% CI : (0.8083, 0.89)
##     No Information Rate : 0.8429
##     P-Value [Acc > NIR] : 0.3543
##
##                   Kappa : 0.1226
##  Mcnemar's Test P-Value : 2.298e-10
##
##             Sensitivity : 0.08163
##             Specificity : 0.99620
##          Pos Pred Value : 0.80000
##          Neg Pred Value : 0.85342
##              Prevalence : 0.15705
##          Detection Rate : 0.01282
##    Detection Prevalence : 0.01603
##       Balanced Accuracy : 0.53892
##
##        'Positive' Class : Cleaved
##
```

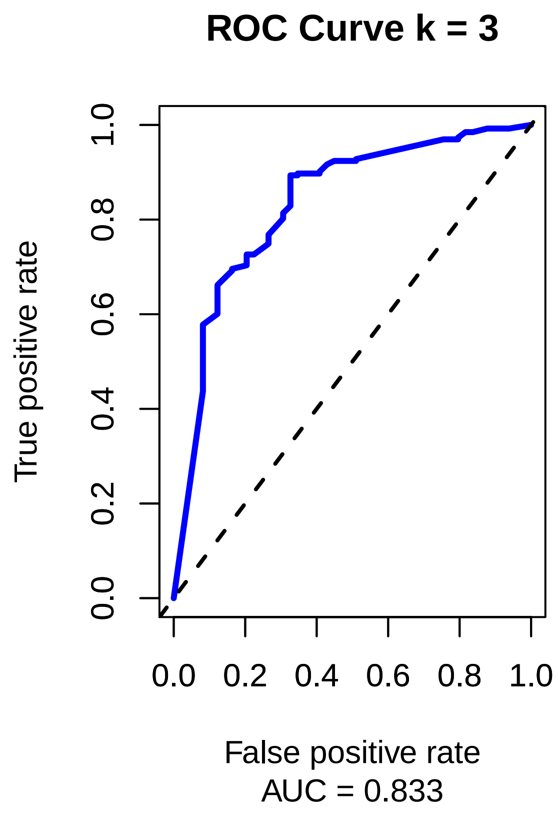
III - ROC Analisys

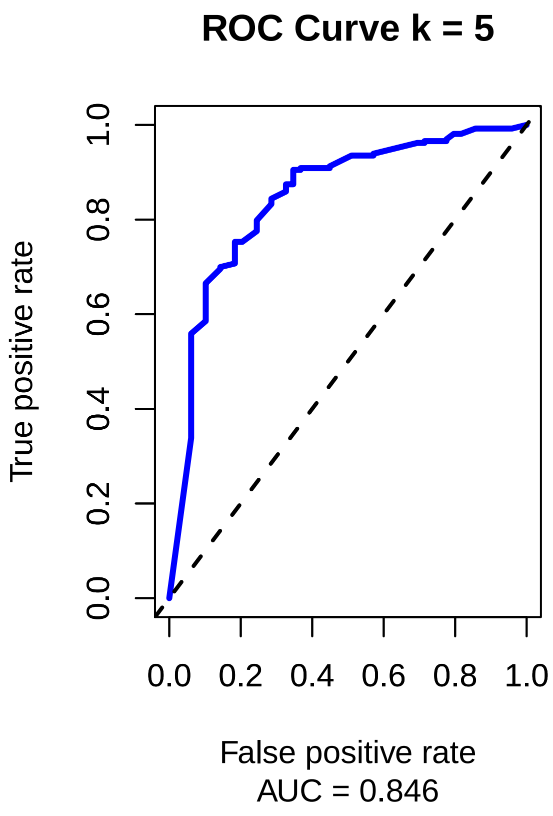
The following function will generate a ROC chart report per prediction.

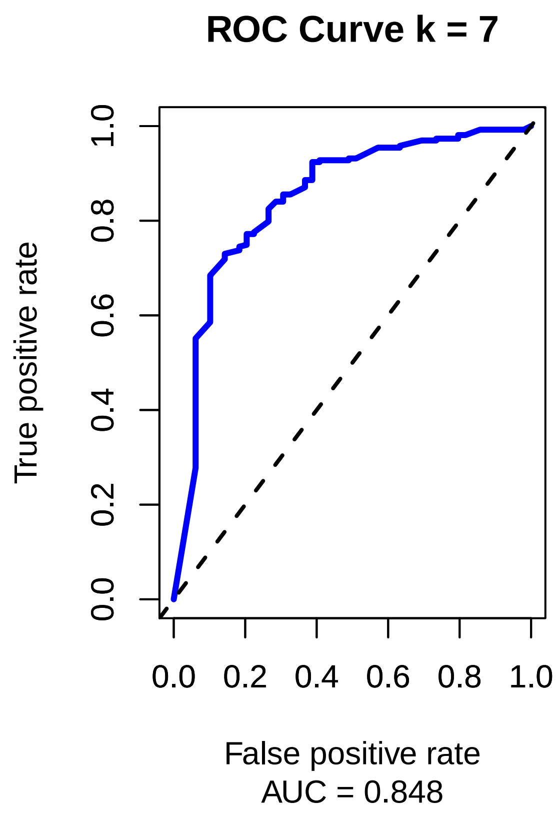
```
ROCAnalisys <- function(predictions, labels) {
    par(mfrow = c(1, 2))
    for (i in names(predictions)) {
        prob <- attr(predictions[[i]], "prob")
        pred <- prediction(predictions = prob, labels)
        perf.auc <- performance(pred, measure = "auc")
        auc = round(unlist(perf.auc@y.values), 3)

        title = sprintf("ROC Curve k = %s", i)
        subtitle = sprintf("AUC = %s", auc)

        pred <- prediction(predictions = prob, labels)
        perf <- performance(pred, measure = "tpr", x.measure = "fpr")
        plot(perf, main = title, sub = subtitle, col = "blue", lwd = 3)
        axis(side = 1, at = seq(0, 1, by = 0.2))
        axis(side = 2, at = seq(0, 1, by = 0.2))
        abline(a = 0, b = 1, lwd = 2, lty = 2)
        cat("\n\n")
    }
}
```

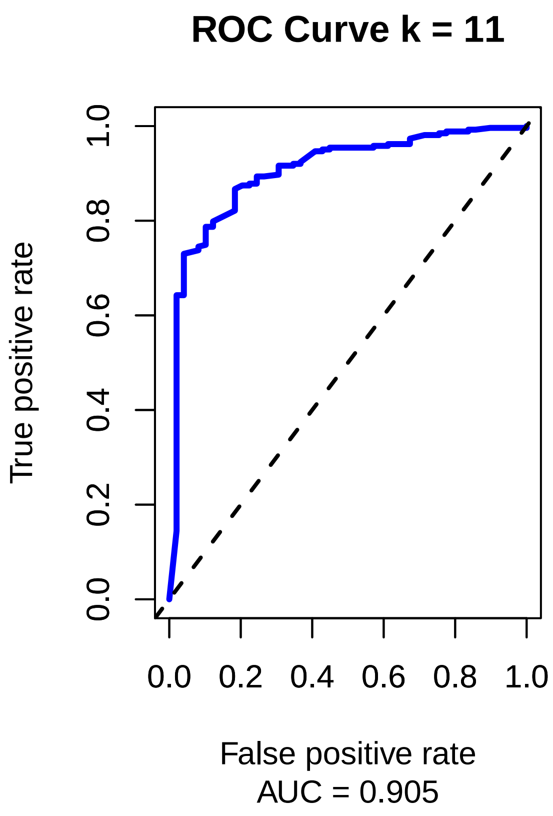
We now execute the ROC chart report.

```
ROCAnalisys(impensData_predictions, impensData_test_labels)
```









IV - Comparison table

```r
impensData_test_summary <- knnPerformanceSummary(impensData_predictions,
    impensData_test_labels, "Cleaved")
impensData_test_summary
```

```
##     k TP  TN FP FN kappa Accurary Error Sensitivity Specificity Precision
## 3   3 10 262  1 39 0.293    0.872 0.128       0.204       0.996     0.909
## 5   5  7 260  3 42 0.194    0.856 0.144       0.143       0.989     0.700
## 7   7  5 262  1 44 0.153    0.856 0.144       0.102       0.996     0.833
## 11 11  4 262  1 45 0.123    0.853 0.147       0.082       0.996     0.800
##    Recall     F   AUC
## 3   0.204 0.333 0.833
## 5   0.143 0.237 0.846
## 7   0.102 0.182 0.848
## 11  0.082 0.149 0.905
```

V - Conclusion

Initially we can appreciate that there is a high rate of true positives for k=3. These true positive predictions decrease as k increases. Furthermore, the rate of false negatives also increases with k while the remaining two factors (true negatives and false positives) stay fixed for the most. This trend is also supported by the **accuracy** level.

Moving forward in the comparison table, we can appreciate that the **kappa statistic** is very low in all cases. This is an indicator that the accuracy is not as trustful as expected since most of the predictions do not match the provided estimations. This could be a consequence of the high false negative results obtained throughout the procedure.

This high false negative rate leads to a low **sensitivity** for all the values of k while the **specificity** remains high. This is again a sign of missleading uncleavege predictions. The model fails when predicting negative results while doing a better job with positive predictions.

The precision and recall rates tell us more of the same story. The high **precision** recalls for a great positive predictive capacity of the model while the low **recall** rate reflects that some of the positive predictions are not being properly identified as such. The low **F-value** obtained for all values of k clearly indicates an overall low performance rate for the model.

On the other hand, the **ROC** charts and the **AUC** values are all within acceptable parameters for all values of k. This is an indicative of how good the model is when predicting positive cases. It is important to notice that, unlike the values cases discussed above, this rate seems to increase along with k.

To conclude, we can say that at first sight a k=3 is an optimum value of k for this prediction based on the raw prediction values (low rate of false negative and false positives altogether). Nevertheless, a value of k=11 leads to a better AUC rate. Furthermore, we can say that, perhaps, the methodology used to solve this model may have lead to too many false negatives. Nevertheless, and because of the hight values of AUC, we can assert that the model is fairly acceptable for protein cleavage prediction.

5. k-NN classificator function (II)

a) Data reading

We first read the new data from the *shillingData* file.

```r
schillingData <- read.csv("schillingData.txt", header = FALSE)
schillingData$V2 <- factor(schillingData$V2, levels = c(-1, 1), labels = c("Uncleaved",
    "Cleaved"))
```

```
colnames(schillingData) <- c("sequence", "result")
head(schillingData)
```

```
##   sequence    result
## 1 AAAAAPAK Uncleaved
## 2 AAAAPAKV Uncleaved
## 3 AAAELGAR Uncleaved
## 4 AAAPAKVE Uncleaved
## 5 AAAPVAAA Uncleaved
## 6 AAAPVVPQ Uncleaved
```

b) Orthogonal codification

From the sequence column of our main *dataframe*, we generate a coded matrix based on the orthogonal sequence algorithm developed above.

This is a two steps procedure where we first get a list of binary sequences from the algorithm and then transform the sequence list into a matrix

```
schillingData_binaryList <- lapply(as.vector(schillingData$sequence),
    toOrthogonalCode, aminoAcidPattern = aminoAcidPattern)
schillingData_binaryMatrix <- do.call(rbind, schillingData_binaryList)
```

We now merge the current *dataframe* with our completed coded animo acid binary sequence.

```
schillingData <- data.frame(schillingData, schillingData_binaryMatrix)
head(schillingData[, 1:10])
```

```
##   sequence    result X1 X2 X3 X4 X5 X6 X7 X8
## 1 AAAAAPAK Uncleaved  1  0  0  0  0  0  0  0
## 2 AAAAPAKV Uncleaved  1  0  0  0  0  0  0  0
## 3 AAAELGAR Uncleaved  1  0  0  0  0  0  0  0
## 4 AAAPAKVE Uncleaved  1  0  0  0  0  0  0  0
## 5 AAAPVAAA Uncleaved  1  0  0  0  0  0  0  0
## 6 AAAPVVPQ Uncleaved  1  0  0  0  0  0  0  0
```

Let's take a look at the result column proportions.

```
round(prop.table(table(schillingData$result)) * 100, digits = 1)
```

```
##
## Uncleaved   Cleaved
##      86.7      13.3
```

Finally, we export the data into its own CSV file for further use.

```
write.csv(schillingData, file = "orthoSchillingData.csv")
```

c)k-NN prediction test

As with the previous exercise, we process the training schilling data by extracting the data parameters and the outcome or labels in different variables.

```
schillingData_train <- schillingData[-c(1, 2)]
schillingData_train_labels <- schillingData[, 2]
head(schillingData_train[, 1:10])
```

```
##   X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## 1  1  0  0  0  0  0  0  0  0   0
## 2  1  0  0  0  0  0  0  0  0   0
## 3  1  0  0  0  0  0  0  0  0   0
## 4  1  0  0  0  0  0  0  0  0   0
## 5  1  0  0  0  0  0  0  0  0   0
## 6  1  0  0  0  0  0  0  0  0   0
```

In this step we create our testing data set by using the full content of the *impensData* data package.

```
impensData_full_test <- impensData[-c(1, 2)]
impensData_full_test_labels <- impensData[, 2]
head(impensData_full_test[, 1:10])
```

```
##   X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## 1  1  0  0  0  0  0  0  0  0   0
## 2  1  0  0  0  0  0  0  0  0   0
## 3  1  0  0  0  0  0  0  0  0   0
## 4  1  0  0  0  0  0  0  0  0   0
## 5  1  0  0  0  0  0  0  0  0   0
## 6  1  0  0  0  0  0  0  0  0   0
```

Finally, we calculate our predictions based on the new training data.

```
impensData_full_predictions <- process_data_knn(schillingData_train,
    impensData_full_test, schillingData_train_labels, k)
head(impensData_full_predictions)
```

```
##           3         5         7        11
## 1 Uncleaved Uncleaved Uncleaved Uncleaved
## 2 Uncleaved Uncleaved Uncleaved Uncleaved
## 3 Uncleaved Uncleaved Uncleaved Uncleaved
## 4 Uncleaved Uncleaved Uncleaved Uncleaved
## 5 Uncleaved Uncleaved Uncleaved Uncleaved
## 6 Uncleaved Uncleaved Uncleaved Uncleaved
```

d) k-NN test analysis

I - CrossTable

```
crossTableResults(impensData_full_predictions, impensData_full_test_labels)
```

```
## ---------------------- K = 3 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  947
```

```
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       773 |        25 |       798 |
##              |     0.969 |     0.031 |     0.843 |
##              |     0.864 |     0.481 |           |
##              |     0.816 |     0.026 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |       122 |        27 |       149 |
##              |     0.819 |     0.181 |     0.157 |
##              |     0.136 |     0.519 |           |
##              |     0.129 |     0.029 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       895 |        52 |       947 |
##              |     0.945 |     0.055 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## ---------------------- K = 5 ----------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  947
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       787 |        11 |       798 |
##              |     0.986 |     0.014 |     0.843 |
##              |     0.864 |     0.306 |           |
##              |     0.831 |     0.012 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |       124 |        25 |       149 |
##              |     0.832 |     0.168 |     0.157 |
##              |     0.136 |     0.694 |           |
##              |     0.131 |     0.026 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       911 |        36 |       947 |
##              |     0.962 |     0.038 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## --------------------- K = 7 ---------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  947
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       790 |         8 |       798 |
##              |     0.990 |     0.010 |     0.843 |
##              |     0.858 |     0.308 |           |
##              |     0.834 |     0.008 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |       131 |        18 |       149 |
##              |     0.879 |     0.121 |     0.157 |
##              |     0.142 |     0.692 |           |
##              |     0.138 |     0.019 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       921 |        26 |       947 |
##              |     0.973 |     0.027 |           |
## -------------|-----------|-----------|-----------|
##
##
##
##
##
## --------------------- K = 11 ---------------------
##
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  947
##
##
##              | predictions[[i]]
##  test_labels | Uncleaved |   Cleaved | Row Total |
## -------------|-----------|-----------|-----------|
##    Uncleaved |       793 |         5 |       798 |
##              |     0.994 |     0.006 |     0.843 |
##              |     0.851 |     0.333 |           |
##              |     0.837 |     0.005 |           |
## -------------|-----------|-----------|-----------|
##      Cleaved |       139 |        10 |       149 |
##              |     0.933 |     0.067 |     0.157 |
##              |     0.149 |     0.667 |           |
##              |     0.147 |     0.011 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       932 |        15 |       947 |
##              |     0.984 |     0.016 |           |
## -------------|-----------|-----------|-----------|
##
##
```

II - Consusion table

```
confusionSummary(impensData_full_predictions, impensData_full_test_labels,
    "Cleaved")
```

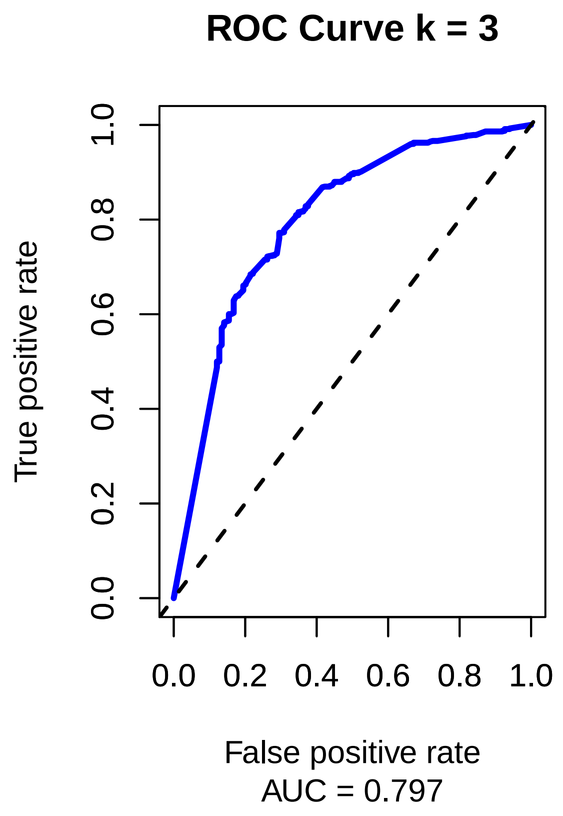
```
## ---------------------- K = 3 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       773     122
##   Cleaved          25      27
##
##                Accuracy : 0.8448
##                  95% CI : (0.8201, 0.8673)
##     No Information Rate : 0.8427
##     P-Value [Acc > NIR] : 0.4507
##
##                   Kappa : 0.2038
##  Mcnemar's Test P-Value : 2.415e-15
##
##             Sensitivity : 0.18121
##             Specificity : 0.96867
##          Pos Pred Value : 0.51923
##          Neg Pred Value : 0.86369
##              Prevalence : 0.15734
##          Detection Rate : 0.02851
##    Detection Prevalence : 0.05491
##       Balanced Accuracy : 0.57494
```

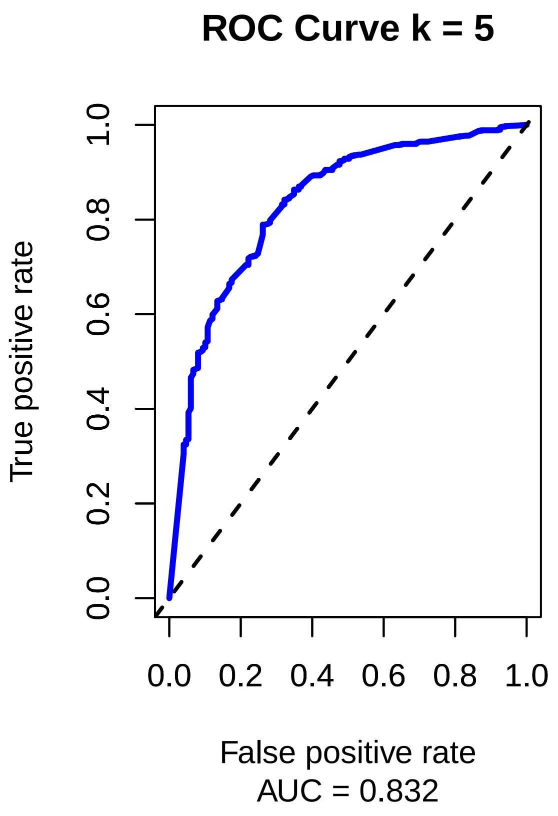
```
##
##        'Positive' Class : Cleaved
##
##
##
##
## ---------------------- K = 5 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       787     124
##   Cleaved          11      25
##
##                Accuracy : 0.8574
##                  95% CI : (0.8335, 0.8791)
##     No Information Rate : 0.8427
##     P-Value [Acc > NIR] : 0.1132
##
##                   Kappa : 0.2227
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.16779
##             Specificity : 0.98622
##          Pos Pred Value : 0.69444
##          Neg Pred Value : 0.86389
##              Prevalence : 0.15734
##          Detection Rate : 0.02640
##    Detection Prevalence : 0.03801
##       Balanced Accuracy : 0.57700
##
##        'Positive' Class : Cleaved
##
##
##
##
## ---------------------- K = 7 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       790     131
##   Cleaved           8      18
##
##                Accuracy : 0.8532
##                  95% CI : (0.8291, 0.8752)
##     No Information Rate : 0.8427
##     P-Value [Acc > NIR] : 0.1991
##
##                   Kappa : 0.1668
##  Mcnemar's Test P-Value : <2e-16
##
```

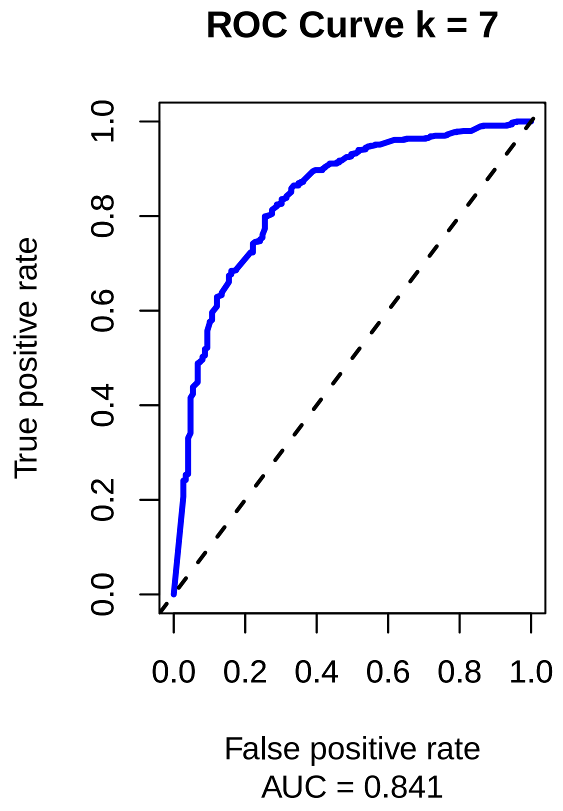
```
##             Sensitivity : 0.12081
##             Specificity : 0.98997
##          Pos Pred Value : 0.69231
##          Neg Pred Value : 0.85776
##              Prevalence : 0.15734
##          Detection Rate : 0.01901
##    Detection Prevalence : 0.02746
##       Balanced Accuracy : 0.55539
##
##        'Positive' Class : Cleaved
##
##
##
##
## ---------------------- K = 11 ----------------------
##
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Uncleaved Cleaved
##   Uncleaved       793     139
##   Cleaved           5      10
##
##                Accuracy : 0.8479
##                  95% CI : (0.8235, 0.8702)
##     No Information Rate : 0.8427
##     P-Value [Acc > NIR] : 0.3472
##
##                   Kappa : 0.0959
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.06711
##             Specificity : 0.99373
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.85086
##              Prevalence : 0.15734
##          Detection Rate : 0.01056
##    Detection Prevalence : 0.01584
##       Balanced Accuracy : 0.53042
##
##        'Positive' Class : Cleaved
##
```

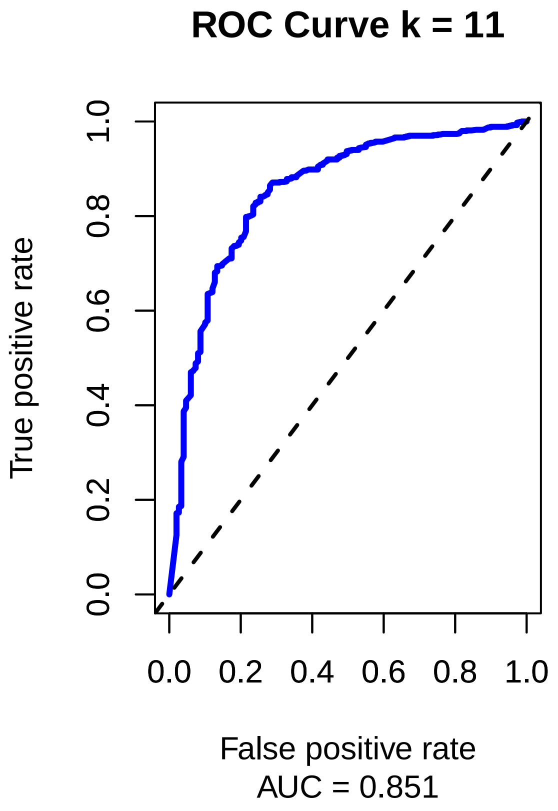
III - ROC Analisys

```
ROCAnalisys(impensData_full_predictions, impensData_full_test_labels)
```









IV - Comparison table

```
impensData_full_summary <- knnPerformanceSummary(impensData_full_predictions,
    impensData_full_test_labels, "Cleaved")
impensData_full_summary
```

```
##      k TP  TN FP  FN kappa Accurary Error Sensitivity Specificity Precision
## 3    3 27 773 25 122 0.204    0.845 0.155       0.181       0.969     0.519
## 5    5 25 787 11 124 0.223    0.857 0.143       0.168       0.986     0.694
## 7    7 18 790  8 131 0.167    0.853 0.147       0.121       0.990     0.692
## 11  11 10 793  5 139 0.096    0.848 0.152       0.067       0.994     0.667
##     Recall     F   AUC
## 3    0.181 0.268 0.797
## 5    0.168 0.271 0.832
## 7    0.121 0.206 0.841
## 11   0.067 0.122 0.851
```

V - Conclusion

For this study the number of false positives increases exponentially. We can say that the levels of false positives matches the positive predictions for low values of k and discards 50% of the true positives for high levels of k. This is very different than what we saw in the previous exercise with a reduced training data set.

Nevertheless, and because of the larger and increased number of true negatives, we are still able to keep a high **accuracy** level. It almost seems like that we have moved from a model with a great true positive prediction to an even better true negative prediction rate. Still, the **kappa statistic** remains within the same range (although a bit lower though) than before, showing there could be some misleading results in our predictions.

The **sensitivity** and **specificity** remains within the same parameters that before, although lower in both cases. There are also too many false negatives in the predictions.

The **precision** has fallen dramatically with this new training data set. This is justified by the large values of false positives obtained in relation with the true positives. More of the same happened with the **recall** value since all those false negatives were meant to be somewhere else. Once again, the **F-value** rate describes a very low performance of the prediction model.

On the bright side, the **ROC** charts and **AUC** values are within acceptable parameters describing a decent predictive model when it comes to positive predictions. Nevertheless, the predictive performance is lower that before.

I would say that, unless the previous exercise, this model works better under larger values of k (k=11 has the best AUC rate) but the overall performance and prediction rate has gone down. On the other hand, and by looking only at the change in false predictions, we can say that a value of k=7 keeps a perfect balance within the increase in the false negatives and the decrease in false positives

References

Lantz, Brett. 2015. Machine Learning with R. Packt Publishing Ltd.