Introduction to Machine Learning

Lab 1

Anton Persson, Emil Klasson Svensson, Mattias Karlsson, Rasmus Holm

2016-11-08

Contents

Assignment 1

In this assignment we have used the spambase data set that contains word frequencies for mails classified as either spam or non-spam. The data set contains 2740 observations. In order to test the k-nearest neighbor algorithm, we have separated the data set into two sets, training and test sets, of equal size. In the first couple of exercises we have set the threshold = 0.5 and the cosine distance to compute dissimilarities.

1.3

Here we used the training set to predict the test set using $k = 5$ and the resulting confusion matrix can be seen below.

```
#>           predicted
#> actual     non-spam spam
#>   non-spam      695  242
#>   spam          193  240
```

The misclassification rate is calculated as

$$\frac{242+193}{695+242+193+240} = \frac{87}{274} \approx 31.8\%.$$

And below is the result from predicting the training data.

```
#>           predicted
#> actual     non-spam spam
#>   non-spam      787  158
#>   spam          119  306
```

The misclassification rate is calculated as

$$\frac{158+119}{787+158+119+306} = \frac{277}{1370} \approx 20.2\%.$$

1.4

Below are the same results as previously but by setting $k = 1$. First is the result from predicting the test set and the other is from predicting the training set.

```
#>           predicted
#> actual     non-spam spam
#>   non-spam      639  298
#>   spam          178  255
```

The misclassification rate is calculated as

$$\frac{298+178}{639+298+178+255} = \frac{238}{685} \approx 34.7\%.$$

The performance have degraded compared to $k = 5$ which is not suprising since using $k = 1$ results in a more complex model since the number of effective parameters are $\frac{\#\text{ of training samples}}{k}$ and it overfits the training data.

```
#>           predicted
#> actual     non-spam spam
#>   non-spam      939    6
#>   spam            2  423
```

The misclassification rate is calculated as

$$\frac{6+2}{939+6+2+423} = \frac{4}{685} \approx 0.6\%.$$

As expected, the number of errors are very low but unexpectedly not 0 errors. This has probably to do with numerical precision, but we can at least see that using $k = 1$ entail in very low error rate compared to $k = 5$ on the training data.

1.5

Here we used the k-nearest neighbor algorithm from the kknn package with $k = 5$ and used the Euclidean distance instead of the cosine distance. Below is the result from classifying the test set.

```
#>           predicted
#> actual     non-spam spam
#>   non-spam      646  291
#>   spam          186  247
```

The misclassification rate is calculated as

$$\frac{291+186}{646+291+186+247} = \frac{477}{1370} \approx 34.8\%.$$

The misclassification rate is very similar to the one using $k = 1$ with cosine distance and a few percentage points above $k = 5$ with cosine distance. This indicates that the Euclidean distance is not a particular good dissimilarity measure in this case and the cosine distance should be favored which further support why the cosine similarity/distance is widely used in text mining.

1.6

In order to compare the performance of the classifiers and how it changes with the threshold, we have fixed $k = 5$ and classified the test data with both cosine and Euclidean distance. The threshold that have been used are $0.05, 0.1, \ldots, 0.95$ and figure 1 shows the performance comparison using ROC curves. As the false positive rate increases, the threshold increases.

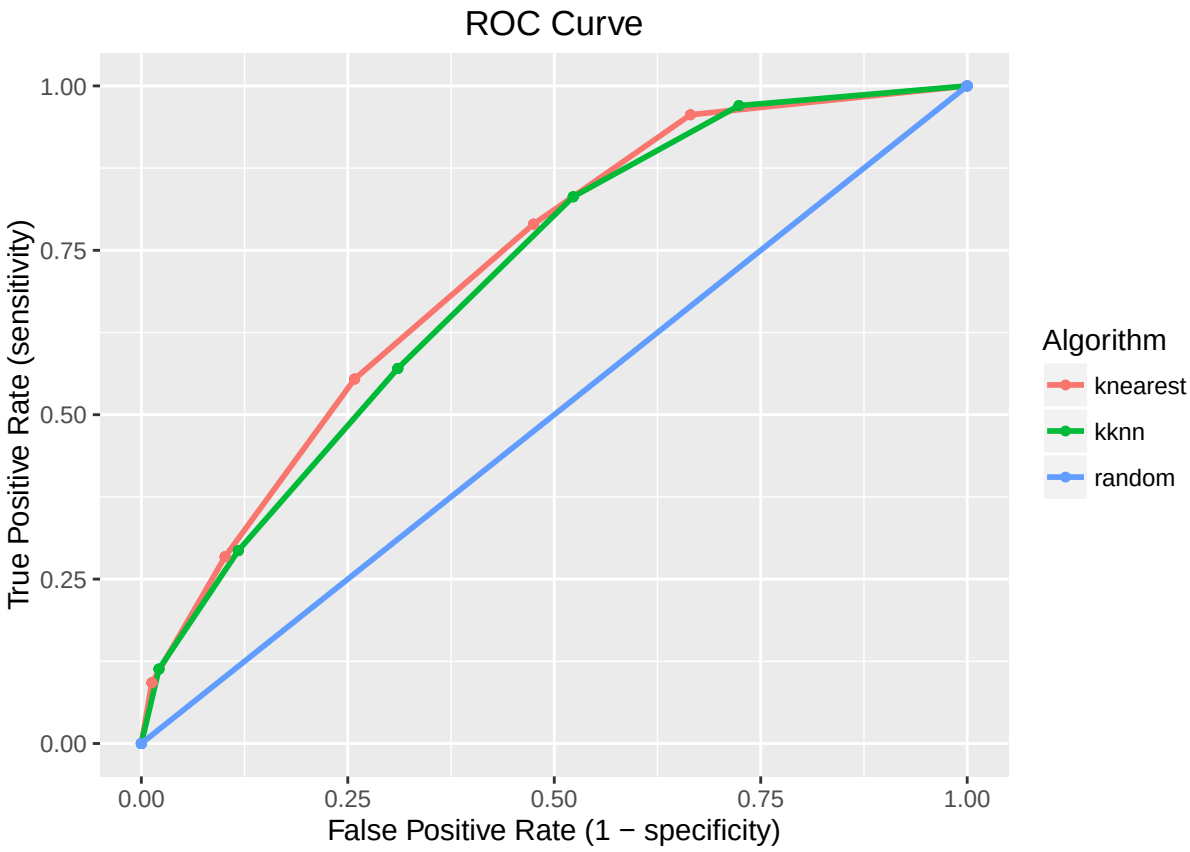


Figure 1: Receiver operating characteristic curves.

We can see that using cosine distance, i.e. the red curve, yield in general better performance than Euclidean distance, the green curve. The point on the red curve at around 0.25 false positive rate have a threshold $\in [0.45, 0.6]$ which contains 0.5. We would expect it to be the optimal threshold given that the errors, FPs and FNs, have equal cost and the ROC curves support that.

Assignment 2

2.2

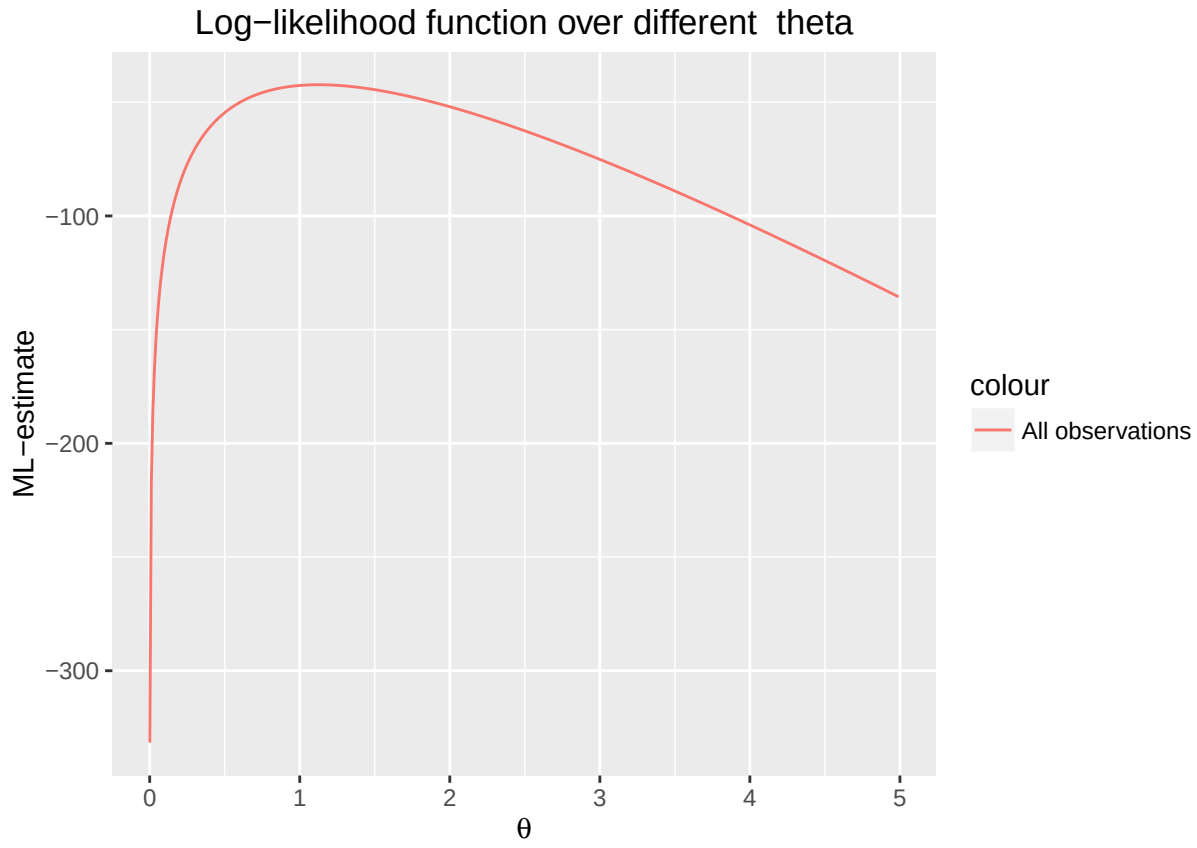
What is the distrubution type of x?

The distrubution given is a exponential distrubution.

Write a function that computes the log-likelihood for a given θ and data vector x

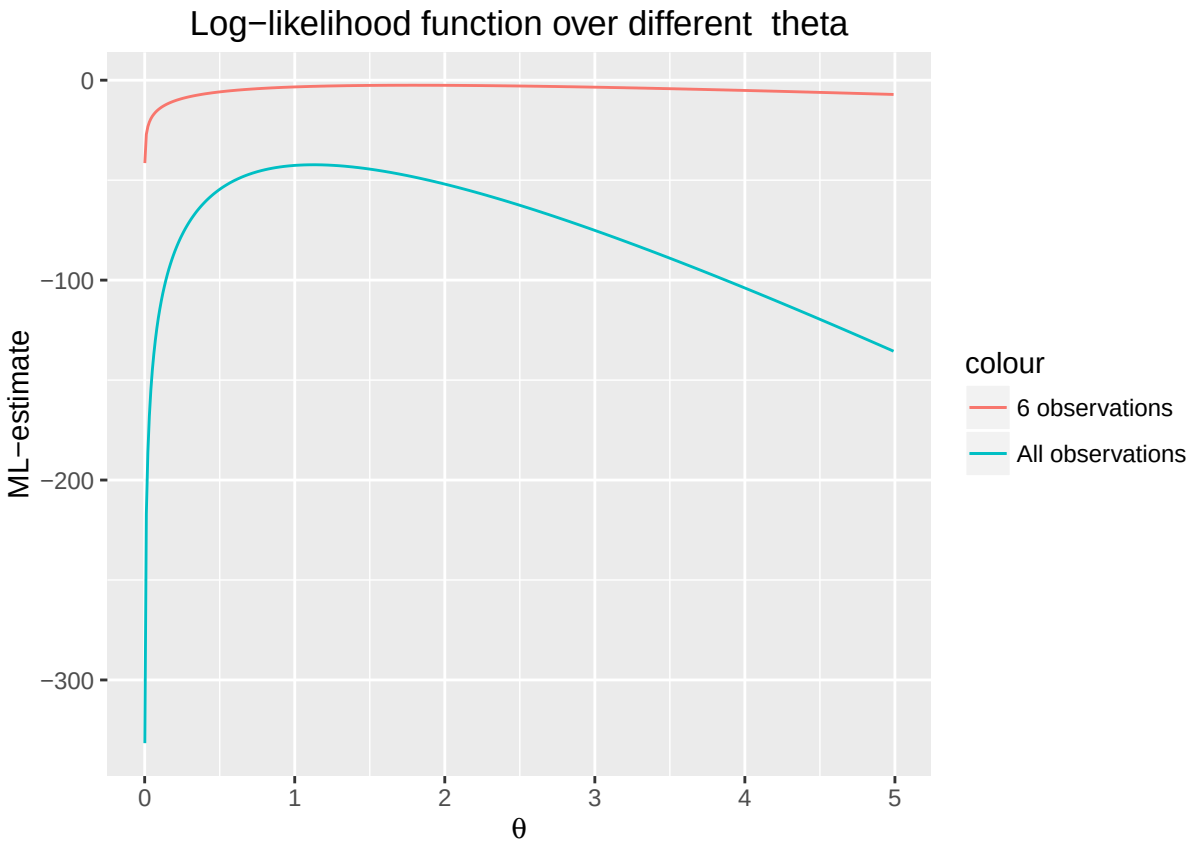
The log-likelihood was computed to $log\, p(x|\theta) = n\, log(\theta) - \theta \sum_{n=1}^{N} \mathrm{x_i}$

Plot the curve showing the dependence of log-likelihood on θ where the entire data is used for fitting.



The maximum log-likelihood is estimated to about 1.1 according to the plot.

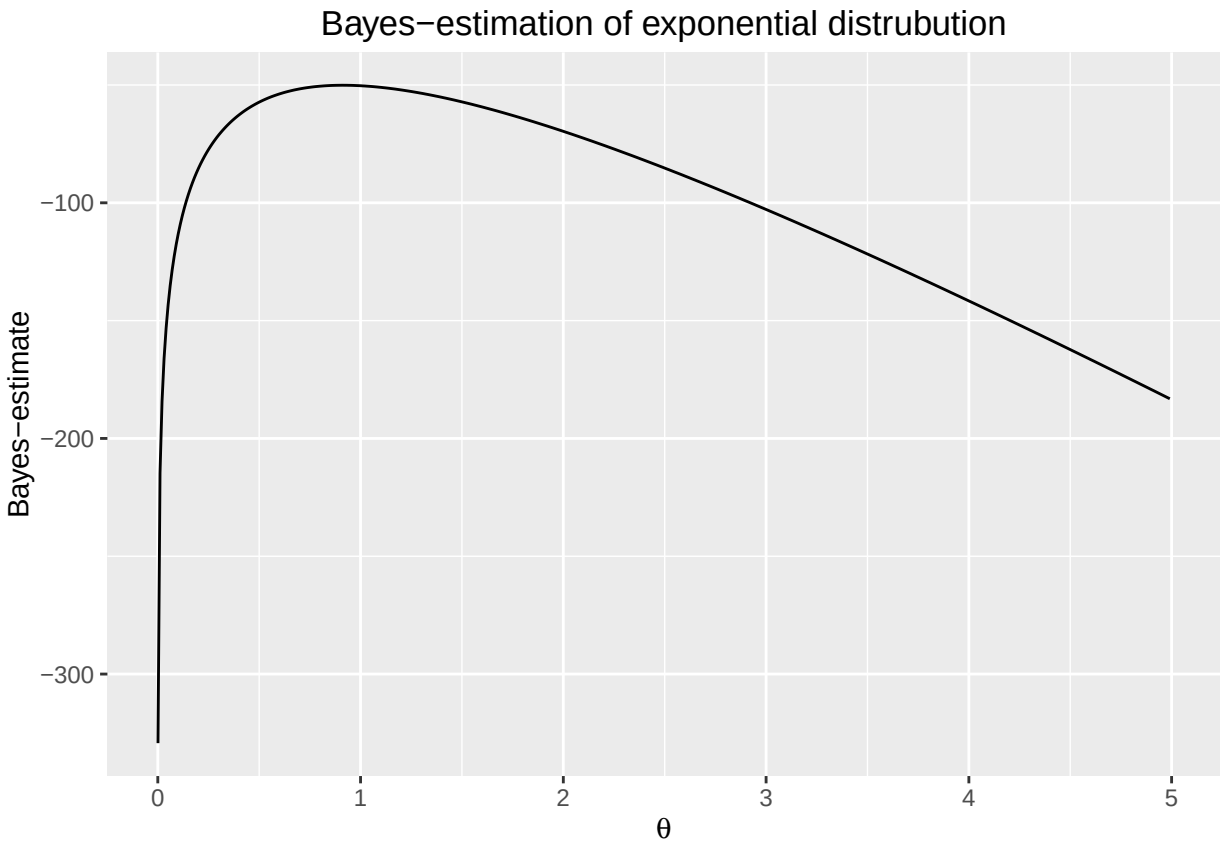
2.3



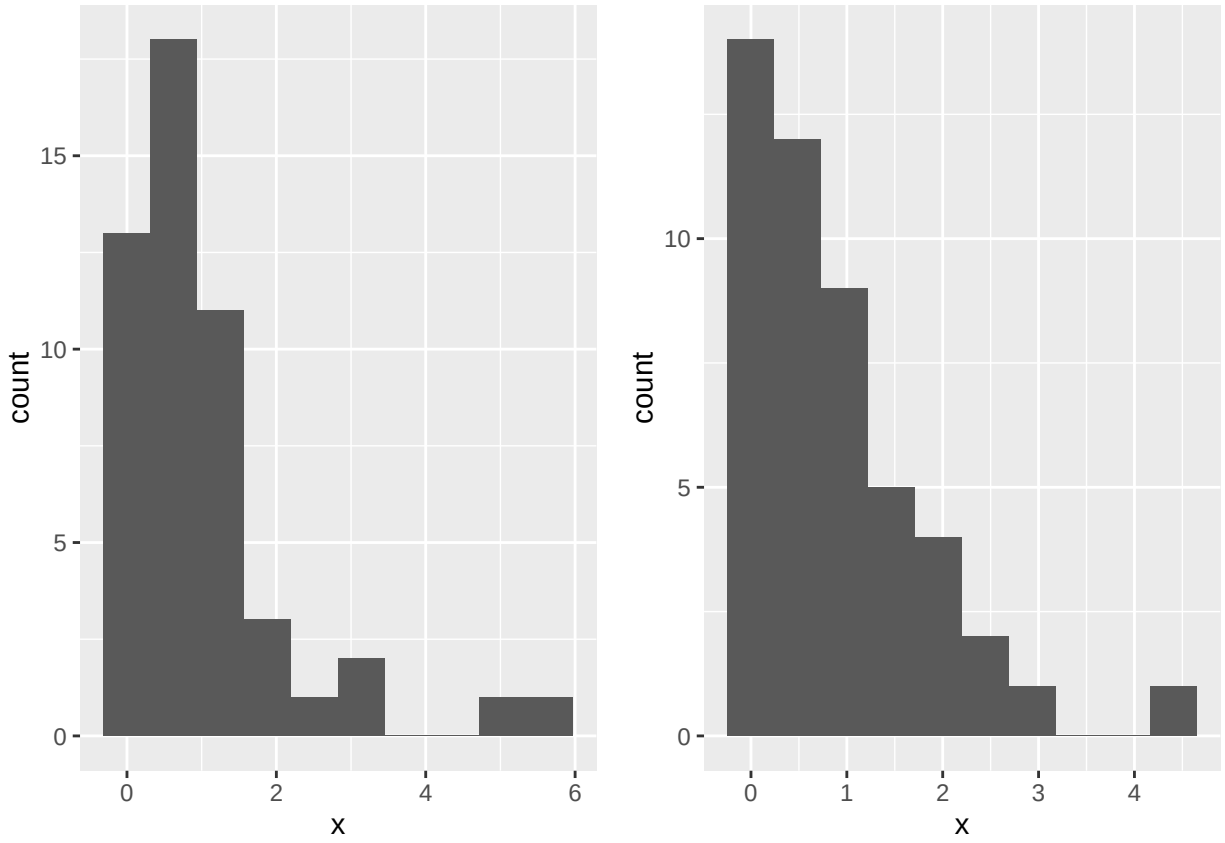
The light red line shows the likelihood function of different theta using six observations and the light blue line shows the estimate using all observations. Both lines increases at the starting values of theta and reaches their peaks at about the same time. The light red line has a higher intercept than the light blue curve, they differ with about 300 units.

The difference is that the estimate using all observations dies down at a much faster rate than the one using six observations and displays a much clearer max-value than the one with six observations. This shows a large dependence of data that the Maximum Likelihood estimators use.

2.4


Bayes−estimation of exponential distrubution
Bayes−estimate
−100
−200
−300
0
1
2
3
4
5
θ

2.5



The distribution over lifetime of machines have similar properties as the sampled observations from the exponential distribution with the ML-estimate of theta. One observation departs in the machine data set, it has a value above 4 wich is a considerably bigger then the rest.

Appendix

Code for Assignment 1

```
library(kknn)
library(ggplot2)
library(reshape)

data <- read.csv("../data/spambase.csv", sep=",", header=TRUE)

n <- nrow(data)
set.seed(12345)
id <- sample(1:n, floor(n * 0.5))

train <- data[id,]
test <- data[-id,]

train_labels <- factor(train[, ncol(train)], levels=c(0, 1), labels=c("non-spam", "spam"))
test_labels <- factor(test[, ncol(test)], levels=c(0, 1), labels=c("non-spam", "spam"))

single_threshold <- 0.5

knearest <- function(data, k, newdata, threshold=0.5) {
    stopifnot(k > 0)

    train_labels <- data[, ncol(data)]
    train <- as.matrix(data[, -ncol(data)])
    train <- train / sqrt(rowSums(train^2))

    test_labels <- newdata[, ncol(newdata)]
    test <- as.matrix(newdata[, -ncol(newdata)])
    test <- test / sqrt(rowSums(test^2))

    cosine_sim <- train %*% t(test)
    cosine_dis <- 1 - cosine_sim

    ordering <- as.matrix(t(apply(cosine_dis, 2, order))[, 1:k])

    predicted <- as.matrix(apply(ordering, 1, function(x) {
        mean(train_labels[x])
    }))

    predicted
}

predicted_result <- function(predicted, actual, threshold) {
    predicted <- as.numeric(predicted > threshold)
    predicted <- factor(predicted, levels=c(0, 1), labels=c("non-spam", "spam"))
    table(actual, predicted)
}
predicted <- knearest(train, 5, test)
predicted_result(predicted, test_labels, single_threshold)
predicted <- knearest(train, 5, train)
```

```
predicted_result(predicted, train_labels, single_threshold)
predicted <- knearest(train, 1, test)
predicted_result(predicted, test_labels, single_threshold)
predicted <- knearest(train, 1, train)
predicted_result(predicted, train_labels, single_threshold)
kknn.fit <- kknn(Spam ~ ., train=train, test=test, distance=2, k=5, kernel="rectangular")
predicted <- fitted(kknn.fit)
predicted_result(predicted, test_labels, single_threshold)
threshold <- seq(0.05, 0.95, by=0.05)

predicted_knearest <- knearest(train, 5, test)

kknn.fit <- kknn(Spam ~ ., train=train, test=test, distance=2, k=5, kernel="rectangular")
predicted_kknn <- fitted(kknn.fit)

knearest_sensitivity <- rep(0, length(threshold))
knearest_specificity <- rep(0, length(threshold))

kknn_sensitivity <- rep(0, length(threshold))
kknn_specificity <- rep(0, length(threshold))

sensitivity <- function(confusion_matrix) {
    confusion_matrix[4] / (confusion_matrix[2] + confusion_matrix[4])
}

specificity <- function(confusion_matrix) {
    confusion_matrix[1] / (confusion_matrix[1] + confusion_matrix[3])
}

for (i in 1:length(threshold)) {
    knearest_prediction <- predicted_result(predicted_knearest, test_labels, threshold[i])
    knearest_sensitivity[i] <- sensitivity(knearest_prediction)
    knearest_specificity[i] <- specificity(knearest_prediction)

    kknn_prediction <- predicted_result(predicted_kknn, test_labels, threshold[i])
    kknn_sensitivity[i] <- sensitivity(kknn_prediction)
    kknn_specificity[i] <- specificity(kknn_prediction)
}

knearest_x <- c(1, 1 - knearest_specificity, 0)
knearest_y <- c(1, knearest_sensitivity, 0)

kknn_x <- c(1, 1 - kknn_specificity, 0)
kknn_y <- c(1, kknn_sensitivity, 0)

knearest_data <- data.frame(x=knearest_x, y=knearest_y,
                            label=rep("knearest", length(knearest_x)))

kknn_data <- data.frame(x=kknn_x, y=kknn_y,
                        label=rep("kknn", length(kknn_x)))

reference_line <- data.frame(x=as.numeric(seq(0, 1, 0.05) > 0.5),
                             y=as.numeric(seq(0, 1, 0.05) > 0.5),
```

```
                              label=rep("random", length(knearest_x)))

complete_data <- melt(rbind(knearest_data, kknn_data, reference_line), id=c("x", "y"))
names(complete_data)[4] <- "Algorithm"
ggplot(data=complete_data) + ggtitle("ROC Curve") +
    xlab("False Positive Rate (1 - specificity)") +
    ylab("True Positive Rate (sensitivity)") +
    geom_line(aes(x=x, y=y, color=Algorithm), size=1) +
    geom_point(aes(x=x, y=y, color=Algorithm), size=1.25)
```

Code for Assignment 2

```
machines <- data.frame(read.csv("../data/machines.csv"))
machines <- as.vector(unlist(machines$Length))
explog <-function(theta,data){
  return(length(data)*log(theta,base = exp(1)) - (theta * sum(data)))
}
theta <- seq(0.001, 5, by=0.01) #Generates a sequence of different
thetaML <- sapply( X = theta, FUN = explog, data = machines)
#apply the different thetas to the log-likelihood with the machine data-set.
plotData <- data.frame(x=theta,y=thetaML) #Put the data in a frame.

library(ggplot2)
#Plotting

p <- ggplot() + geom_line(data=plotData,aes(x=x,y=y,col="All observations")) +
                labs(title=paste("Log-likelihood function over different  theta"),
       x=expression(theta),y="ML-estimate")
plot(p)
thetaML3 <- sapply( X = theta, FUN = explog, data = machines[1:6])
#Using only the 6 first observations of machines
plotData$y2 <- thetaML3

p + geom_line(data = plotData, aes(x=x,y=y2,col="6 observations"))

postBayes <- function(data,theta){
  lambda <- 10
  return(length(data) * log(theta) - theta * sum(data) +
           log(lambda) - (lambda*theta))

}

thetaBayes <- sapply(theta,FUN = postBayes,data = machines)

plotDataBayes <- data.frame(x=theta,y=thetaBayes)
B <- ggplot() + geom_line(data=plotDataBayes,aes(x=x,y=y)) +
  labs(title="Bayes-estimation of exponential distrubution",
       x=expression(theta),y="Bayes-estimate")
plot(B)
library(gridExtra)
```

```r
#extract the exact theta for the  ML-estimate
maxTheta <- theta[thetaML == max(thetaML)]
# Generate 50 random variables from the exponential distrubution with
# the ML-estimate extracted
set.seed(12345)
randomExpData<- data.frame(x=rexp(50,rate = maxTheta))

#Plotting
ab<- ggplot(data = randomExpData,aes(x=x)) + geom_histogram(bins = 10)
abc<- ggplot(data =data.frame(x=machines),aes(x=x)) + geom_histogram(bins = 10)

grid.arrange( ab, abc, ncol=2)
```