Introduction to Machine Learning

Lab 1

Rasmus Holm

2016-11-07

Contents

Assignment 1

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

1.3

Here I 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

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

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

The misclassification rate is calculated as

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

1.5

Here I have used 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.

1.6

```
#>  [1] 1.00000000 0.95612009 0.95612009 0.95612009 0.78983834 0.78983834
#>  [7] 0.78983834 0.78983834 0.55427252 0.55427252 0.55427252 0.55427252
#> [13] 0.28406467 0.28406467 0.28406467 0.28406467 0.09237875 0.09237875
#> [19] 0.09237875 0.09237875 0.00000000
#>  [1] 1.0000000 0.9699769 0.9699769 0.9699769 0.8314088 0.8314088 0.8314088
#>  [8] 0.8314088 0.5704388 0.5704388 0.5704388 0.5704388 0.2933025 0.2933025
#> [15] 0.2933025 0.2933025 0.1131640 0.1131640 0.1131640 0.1131640 0.0000000
```

In order to compare the performance of the classifiers and how it changes with the threshold, I 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 decreases.

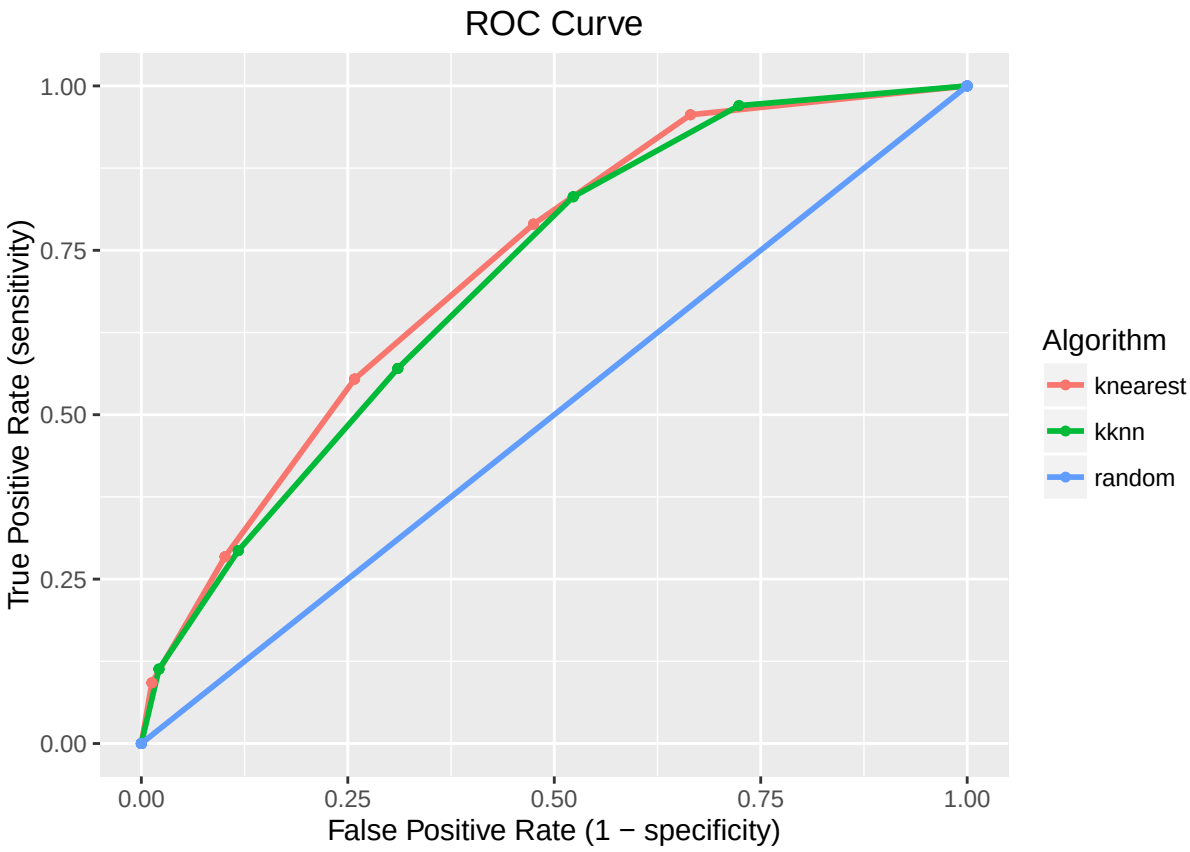

Figure 1: Receiver operating characteristic curves.

We can see that using cosine distance, i.e. the red curve, yield in general better performance than Euclidean distance, the green curve. The points which have the lowest false positive rate, around 0.25, in the red curve have a threshold ≥ 0.8 while the point with false positive rate around 0.4 in the green curve have a threshold $= 0.95$.

Assignment 2

In this assignment I have used the machine data set that contains information about the lifetime of machines. The data set contains 48 observations.

2.2

From the given formula the lifetime of the machines is an exponential distribution and figure 2 further supports that.

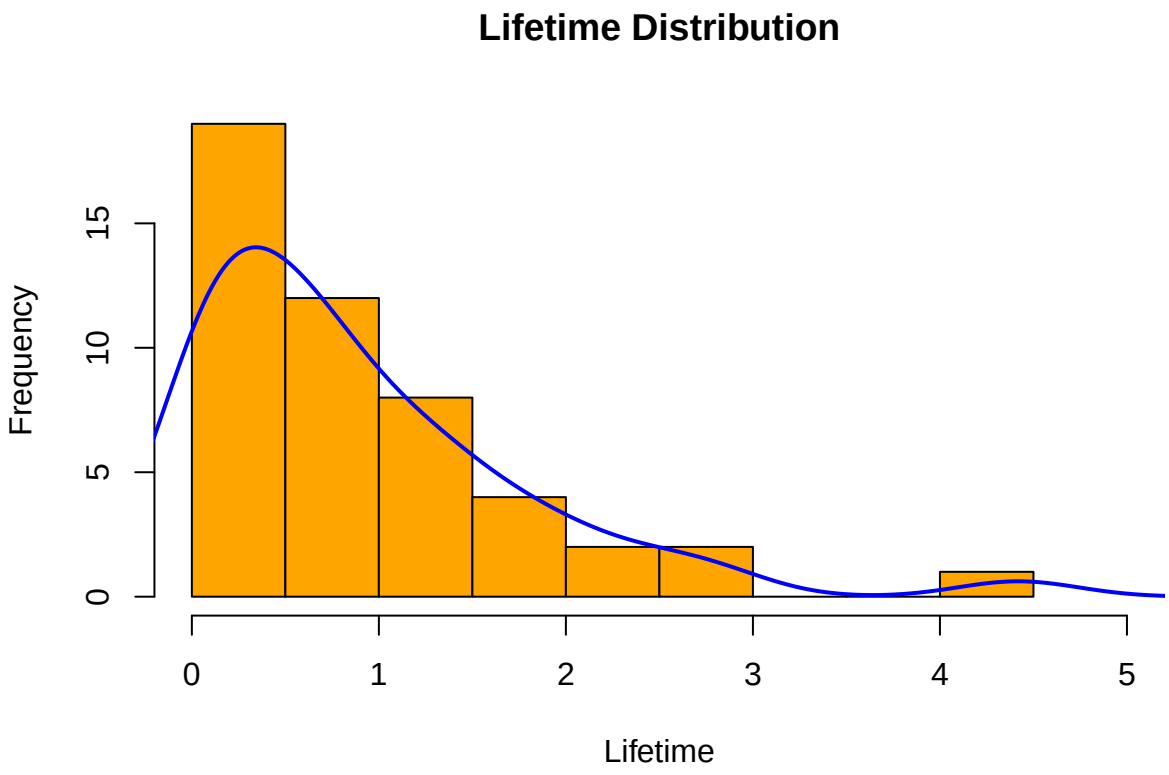

Figure 2: The distribution of the machine data set.

To compute the maximum likelihood estimate, the following formula is used

$$L(\theta) = f(x|\theta) = \prod_{i=1}^{m} f(x_i|\theta) \tag{1}$$

and it is equivalent to maximize the log-likelihood which is given by

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m} \log f(x_i|\theta). \tag{2}$$

Figure 3 shows the log-likelihood over various thetas and the maximum likelihood estimate is given by $\theta \in [1.0, 1.2]$.

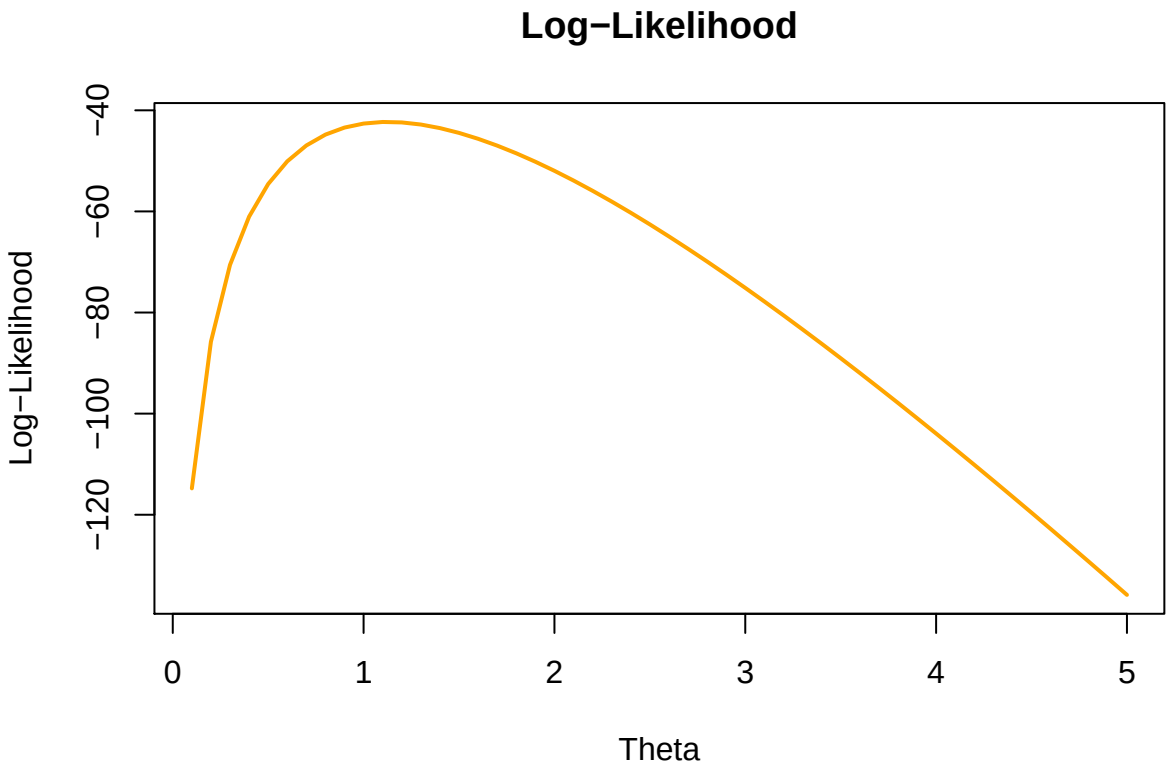

Figure 3: The log-likelihoods over various thetas using the complete data set.

2.3

Figure 4 shows the log-likelihoods when using the complete data set but also when using only 6 observations. Figure 5 shows the log-likelihoods using 6 observations on it own and the maximum $\theta \in [1.5, 2]$ which is not the same as found using the complete data set. However, using the complete data set is more reliable because we do not lose any available information which cannot be said when picking just a handful of observations. We should therefore be more confident in the estimate using the complete data set, i.e. $\theta \in [1.0, 1.2]$.

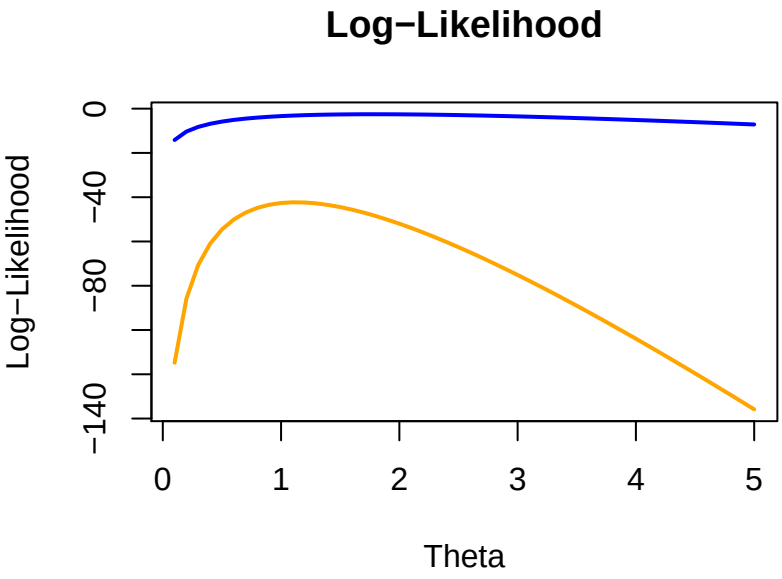

Figure 4: The log-likelihoods over various thetas. Orange line uses the complete data set while the blue line uses only 6 data points.

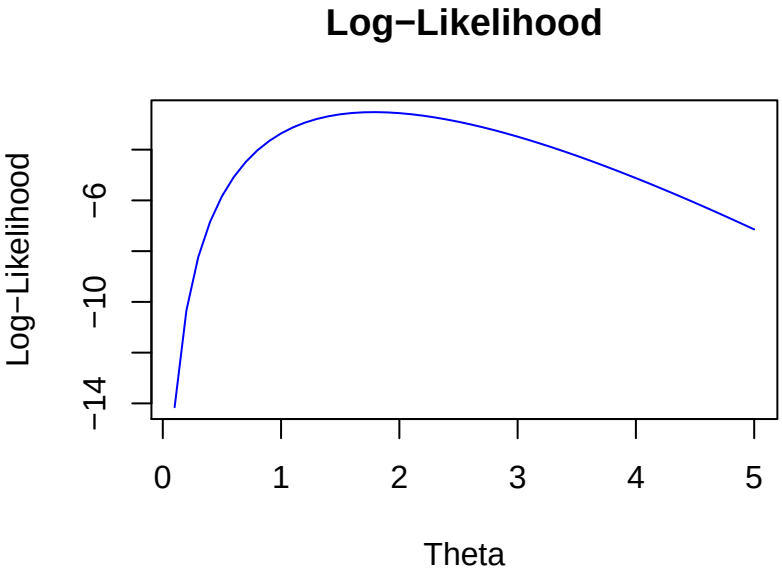

Figure 5: The log-likelihoods over various thetas using only 6 observations.

2.4

Here I have computed the log-posterior using all the observations and figure 6 shows the result for various θ. The optimal $\theta \in [0.8, 1.0]$ which is close to the ones found previously.

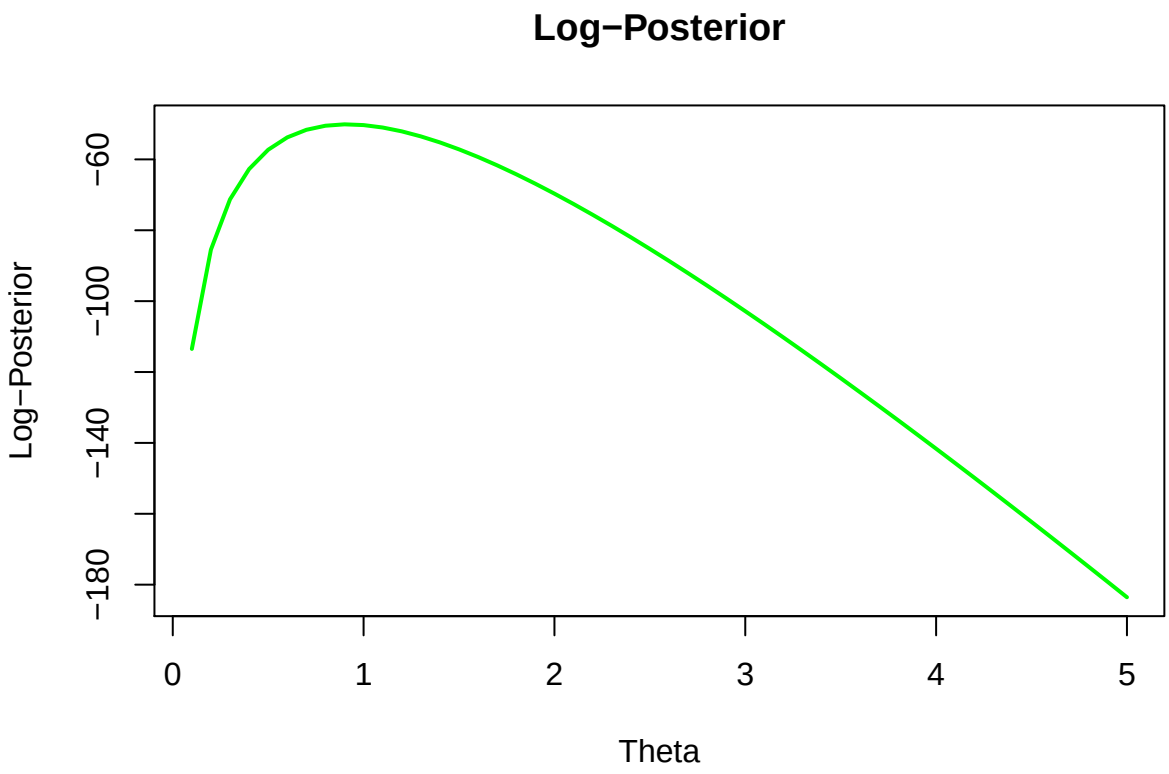

Figure 6: The log-posterior oover various thetas using all observations.

2.5

I used the optimal $\theta = 1.1$ found by the maximum log-likelihood and generated a new data set consisting of 50 observations. Figure 7 shows the machine data set and the generated data as histograms and we can see both have similar distributions as expected.

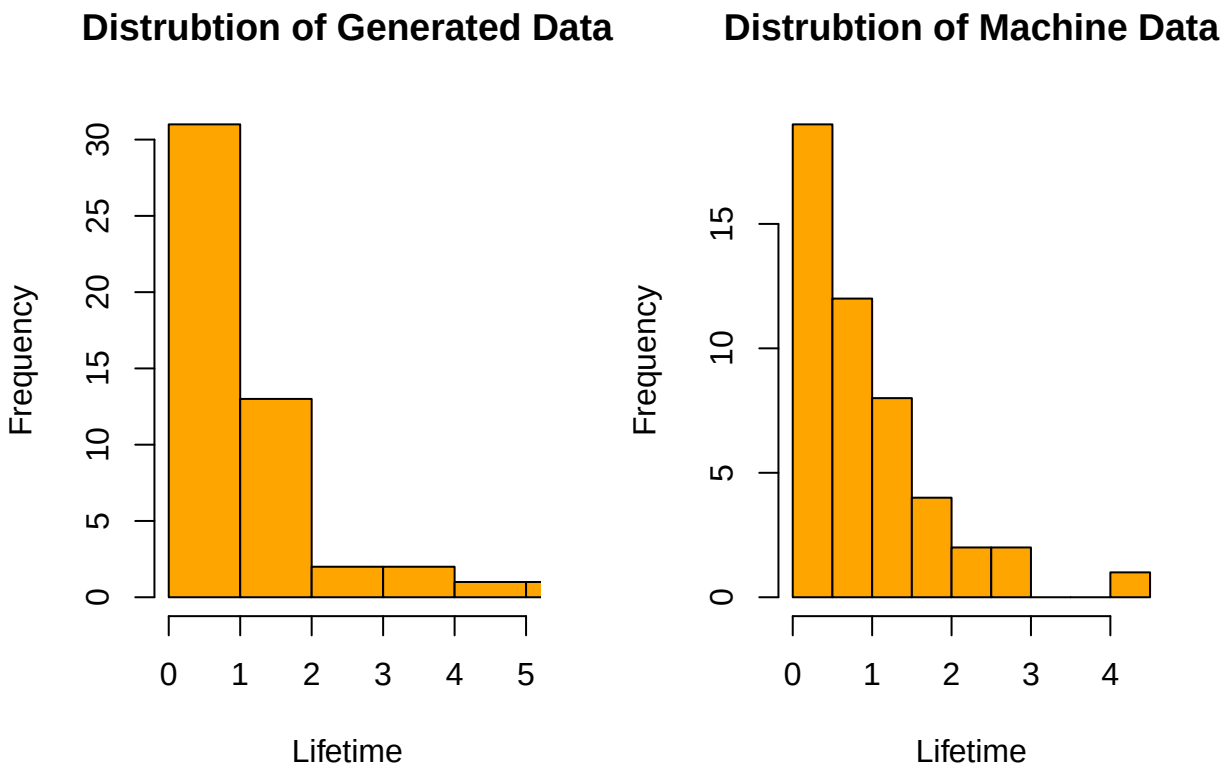

Figure 7: Left: Histogram of the generated data. Right: Histogram of the machine data.

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)

knearest_y

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

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() + ggtitle("ROC Curve") +
    xlab("False Positive Rate (1 - specificity)") +
    ylab("True Positive Rate (sensitivity)") +
    geom_line(data=complete_data, aes(x=x, y=y, color=Algorithm), size=1) +
    geom_point(data=complete_data, aes(x=x, y=y, color=Algorithm), size=1.25) +
    scale_x_continuous(limits = c(0, 1)) + scale_y_continuous(limits=c(0, 1)) +
    theme(plot.title=element_text(hjust=0.5))
```

Code for Assignment 2

```
length_histogram <- hist(data$Length, plot=FALSE)
multiplier <- length_histogram$counts / length_histogram$density
multiplier <- max(multiplier[which(!is.nan(multiplier))])
length_density <- density(data$Length)
length_density$y <- length_density$y * multiplier

log_likelihood <- function(x, theta) {
    log(theta * exp(-theta * x))
}

thetas <- seq(0.1, 5, by=0.1)

log_likelihoods <- sapply(thetas, function(x) {
    sum(log_likelihood(x=data$Length, theta=x))
})

best_theta <- thetas[which.max(log_likelihoods)]
plot(length_histogram, col="orange", main="Lifetime Distribution",
     xlab="Lifetime", ylab="Frequency", xlim=c(0, 5))
lines(length_density, col="blue", lwd=2)
plot(thetas, log_likelihoods, main="Log-Likelihood", col="orange",
     xlab="Theta", ylab="Log-Likelihood", type="l", lwd=2)
log_likelihoods_6 <- sapply(thetas, function(x) {
    sum(log_likelihood(x=data$Length[1:6], theta=x))
})

ylim <- c(min(min(log_likelihoods), min(log_likelihoods_6)),
          max(max(log_likelihoods), max(log_likelihoods_6)))
plot(thetas, log_likelihoods, col="orange",
     main="Log-Likelihood", xlab="Theta", ylab="Log-Likelihood",
     type="l", ylim=ylim, lwd=2)
lines(thetas, log_likelihoods_6, col="blue", lwd=2)
plot(thetas, log_likelihoods_6, type="l", main="Log-Likelihood",
     xlab="Theta", ylab="Log-Likelihood", col="blue")
```

```r
prior <- function(theta, lambda=10) {
    lambda * exp(-lambda * theta)
}

log_posteriors <- sapply(1:length(thetas), function(i) {
    log_likelihoods[i] + log(prior(thetas[i]))
})
plot(thetas, log_posteriors, col="green",
     main="Log-Posterior", xlab="Theta", ylab="Log-Posterior",
     type="l", lwd=2)
set.seed(12345)
new_data <- rexp(50, best_theta)
par(mfrow=c(1, 2))
hist(new_data, main="Distrubtion of Generated Data",
     xlab="Lifetime", ylab="Frequency", xlim=c(0, 5), col="orange")
hist(data$Length, main="Distrubtion of Machine Data",
     xlab="Lifetime", ylab="Frequency", col="orange")
```