Laboration report in Machine Learning

Computer lab 1 block 1

732A99

Sigme Cinar
Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

11 November 2023

Contents

1 Assignment 1. Handwritten digit recognition with K-nearest neighbors.

The data in this task is from the file optdigits.csv. Data consists of 3822 handwritten digits from 0 to 9 and are stored as images of size 8x8.

1.1 1.1

Question: Import the data into R and divide it into training, validation and test sets (50%/25%/25%) by using the partitioning principle specified in the lecture slides.

Answer: The code used is presented as follows:

```
# Read in data
data <- read.csv("optdigits.csv")

# Renaming the response variable and changing it to a factor variable
data <- rename(data, y=X0.26)
data$y <- as.factor(data$y)

# Partitioning training data (50%)
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]

# Partitioning validation data (25%)
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

# Partitioning test data (25%)
id3=setdiff(id1,id2)
test=data[id3,]
```

1.2 1.2

Question: Use training data to fit 30-nearest neighbor classifier with function kknn() and kernel=”rectangular” from package kknn and estimate

- Confusion matrices for the training and test data (use table())
- Misclassification errors for the training and test data

Answer: The confusion matrix for the model trained on training data with k=30 and evaluated on training data is presented in table 1.

```r
# kknn on training data and evaluation on training data
model_kknn_train <- kknn(formula=y~., train=train, test=train, kernel="rectangular", k=30)
conf_mat_train <- table(train$y, model_kknn_train$fitted.values)
acc_train <- sum(diag(conf_mat_train)) / sum(conf_mat_train)
miss_train <- 1-acc_train

# kknn on training data and evaluation on test data
model_kknn_test <- kknn(formula=y~., train=train, test=test, kernel="rectangular", k=30)
conf_mat_test <- table(test$y, model_kknn_test$fitted.values)
acc_test <- sum(diag(conf_mat_test)) / sum(conf_mat_test)
miss_test <- 1-acc_test

# Rows are true values, columns are model prediction
kable(conf_mat_train, caption = "Confusion matrix for training data, model
      predictions by columns and true value by rows.")
```

Table 1: Confusion matrix for training data, model predictions by columns and true value by rows.

	0	1	2	3	4	5	6	7	8	9
0	177	0	0	0	1	0	0	0	0	0
1	0	174	9	0	0	0	1	0	1	3
2	0	0	170	0	0	0	0	1	2	0
3	0	0	0	197	0	2	0	1	0	0
4	0	1	0	0	166	0	2	6	2	2
5	0	0	0	0	0	183	1	2	0	11
6	0	0	0	0	0	0	200	0	0	0
7	0	1	0	1	0	1	0	192	0	0
8	0	10	0	1	0	0	2	0	190	2
9	0	3	0	4	2	0	0	2	4	181

```r
miss_train
```

```
## [1] 0.04238619
```

The misclassification error on training data from table 1 is around 4.24%. The two number with the highest number of wrong predictions are 8 and 9, with 15 wrong predictions each. The model also struggles to predict number 1 and 5 where both had 14 wrong predictions. Examining the most common misclassification, the number 5 was predicted as 9 a total of eleven times and the number 8 was predicted as 1 a total of ten times.

The easiest numbers to predict are 0 and 6. For 0 the model correctly predicted 177 out of 178 numbers and did not predict the number 0 for any other number. For the number 6, the model correctly predicted all 200 numbers, but it did however predict six other numbers as 6 incorrectly.

The confusion matrix for the model trained on training data with k=30 and evaluated on test data is presented in table 2.

```
kable(conf_mat_test, caption = "Confusion matrix for test data, model
      predictions by columns and true value by rows.")
```

Table 2: Confusion matrix for test data, model predictions by columns and true value by rows.

	0	1	2	3	4	5	6	7	8	9
0	97	0	0	0	0	0	1	0	0	0
1	0	91	3	0	0	0	0	0	0	3
2	0	0	93	1	0	0	0	0	1	0
3	0	0	0	95	0	0	0	2	1	0
4	1	0	0	0	89	0	1	5	1	3
5	0	1	0	1	0	79	1	0	0	5
6	0	0	0	0	0	0	94	0	0	0
7	0	2	0	0	0	1	0	91	1	0
8	0	3	0	1	0	0	1	0	86	0
9	0	0	0	4	0	0	0	2	1	94

```
miss_test
```

```
## [1] 0.04916318
```

The misclassification error on test data from table 2 is around 4.92%. The two number with the highest number of wrong predictions are 4 and 5. The number 4 had ten wrong predictions and the number 5 had eight wrong predictions.

The easiest numbers to predict are 0 and 6. For 0 the model correctly predicted 97 out of 98 number and the misclassification was as number 6.
For the number 6, the model correctly predicted all 94 numbers, but it did however predict four times for other numbers as 6 incorrectly.

The overall prediction quality from the model is good, especially for the numbers 0 and 6.

1.3 1.3

Question: Find any 2 cases of digit "8" in the training data which were easiest to classify and 3 cases that were hardest to classify (i.e. having highest and lowest probabilities of the correct class). Reshape features for each of these cases as matrix 8x8 and visualize the corresponding digits (by using e.g. heatmap() function with parameters Colv=NA and Rowv=NA) and comment on whether these cases seem to be hard or easy to recognize visually.

Answer: The code used to find the 2 digits that are hardest to classify were found with the code as follows

```
y <- train$y
fit_y <- model_kknn_train$fitted.values
# probabilities given from number 0 to 9, index 9 = number 8.
prob_8 <- model_kknn_train$prob[, 9]
```

```r
# Data frame consisting of true value of y, model prediction and the models
# probability that the number is 8.
data_8 <- data.frame(y = y, fit_y = fit_y, prob = prob_8)
data_8$observation_id <- rownames(data_8)

# Only observations with the label 8 is kept.
data_8 <- data_8[data_8$y == "8", ]
head(arrange(data_8, prob), 2)
```

```
##   y fit_y      prob observation_id
## 1 8     6 0.1000000           1624
## 2 8     1 0.1666667           1663
```

From the output, observation 1624 and 1663 were hardest to classify as 8 from the model. The three observations that were easiest to identify as 8 were found with the code as follows

```r
tail(arrange(data_8, prob), 3)
```

```
##     y fit_y prob observation_id
## 203 8     8    1           1810
## 204 8     8    1           1811
## 205 8     8    1           1864
```

From the output observation 1810, 1811, and 1864 were three observations that were easiest to identify as 8 with a probability from the model as 100% (in total there were 49 observations that had 100% probability).

A function that reshapes each observation to a 8x8 cases and then visualizing the result in a heatmap was done with the code as follows

```r
# Change colour palette to black and white
colfunc <- colorRampPalette(c("white", "black"))

plot_8 <- function(index){
  title <- paste0("Obs: ", index)
  # Reshapes the observations to a 8x8
  plot <- as.matrix(train[index, -65]) # Remove response variable
  plot <- matrix(plot, nrow=8, byrow=TRUE)
  heatmap(plot, col=colfunc(16), Colv=NA, Rowv=NA, main=title, margins=c(2,2))
}
```

The heatmaps for observations 1624 and 1663 are presented in figure 1.

```r
plot_8(1624)
plot_8(1663)
```

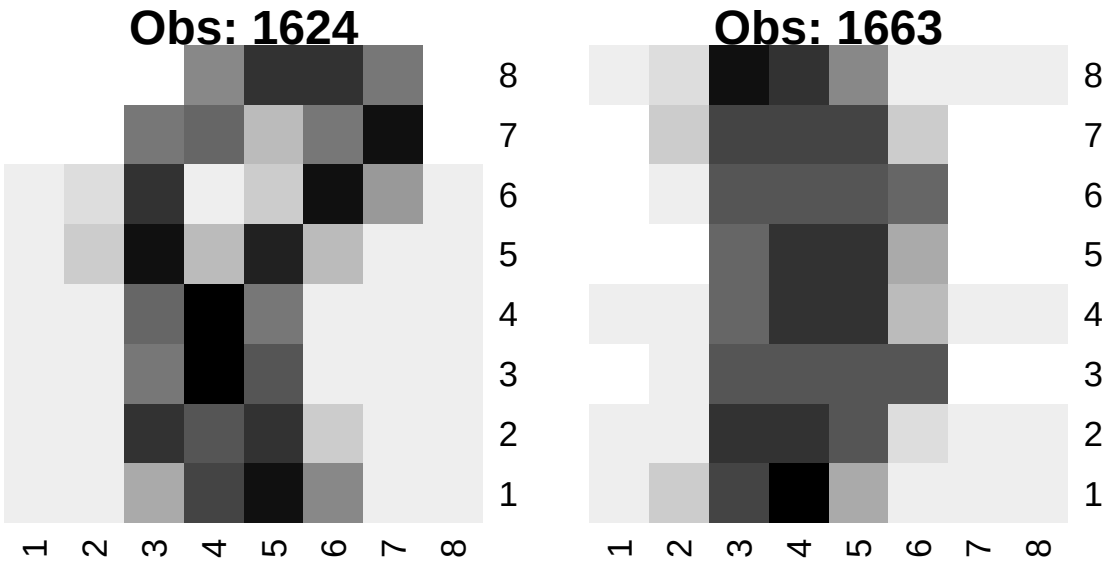

Figure 1: Heatmap for two observations that were hard to classify: 1624 and 1663.

In figure 1, it is hard to visually recognize what number the observations 1624 and 1663 are.

The heatmaps for observations 1810, 1811, and 1864 are presented in figure 2.

```
plot_8(1810)
plot_8(1811)
plot_8(1864)
```

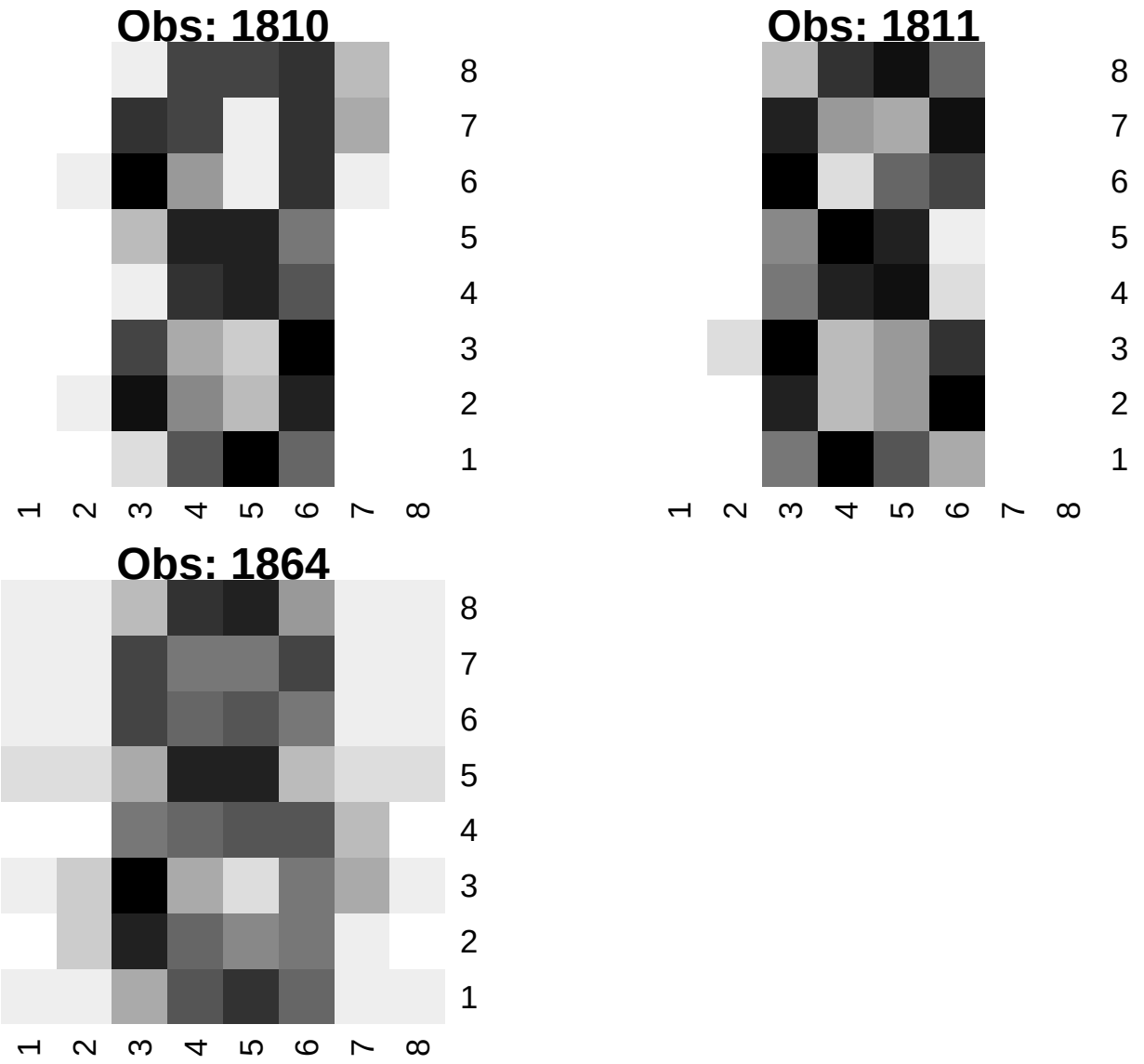

Figure 2: Heatmap for three observations that were easy to classify: 1810, 1811, and 1864.

In figure 2, is is easy to visually recognize that observations 1810, 1811, and 1864 are of the number 8.

1.4 1.4

Question: Fit a K-nearest neighbor classifiers to the training data for different values of $K = 1, 2, ..., 30$ and plot the dependence of the training and validation misclassification errors on the value of K (in the same plot). How does the model complexity change when K increases and how does it affect the training and validation errors? Report the optimal k according to this plot. Finally, estimate the test error for the model having the optimal K, compare it with the training and validation errors and make necessary conclusions about the model quality.

Answer: The code to create K-nearest neighbor clasifiers for different k and the misclassification error on training and validation is as follows

```
fit_kknn <- function(k){
  model_kknn_train <- kknn(formula=y~., train=train, test=train, kernel="rectangular", k=k)
  # Confusion matrix for train data
  conf_mat_train <- table(model_kknn_train$fitted.values, train$y)
  acc_train <- sum(diag(conf_mat_train)) / sum(conf_mat_train)
  # Missclassification for training data
  miss_train <- 1-acc_train

  model_kknn_valid <- kknn(formula=y~., train=train, test=valid, kernel="rectangular", k=k)
  # Confusion matrix for validation data
  conf_mat_valid <- table(model_kknn_valid$fitted.values, valid$y)
  acc_valid <- sum(diag(conf_mat_valid)) / sum(conf_mat_valid)
  # Missclassification for validation data
  miss_valid <- 1-acc_valid

  result <- c(miss_train, miss_valid)
  return(result)
}

# Missclassification for k=1,...,30 for training and validation data
result <- data.frame(train = 0, valid = 0)
for(i in 1:30){
  model <- fit_kknn(i)
  result[i,1] <- model[1]
  result[i,2] <- model[2]
}
result$index <- 1:30
```

The plot showing the dependence of misclassification error for training and validation data is presented in figure 3.

```
ggplot(result, aes(x=index)) +
  geom_line(aes(y=train, colour="train")) +
  geom_point(aes(y=train, colour="train")) +
  geom_line(aes(y=valid, colour="valid")) +
  geom_point(aes(y=valid, colour="valid")) +
  scale_color_manual(name = "Data",
```

```
                    values = c("train" = "steelblue", "valid" = "indianred")) +
  scale_x_continuous(breaks = c(seq(from=0, to=30, by=5))) +
  scale_y_continuous(limits = c(0, 0.06)) +
  theme_bw() +
  labs(x = "k",
       y = "Missclassification rate")
```

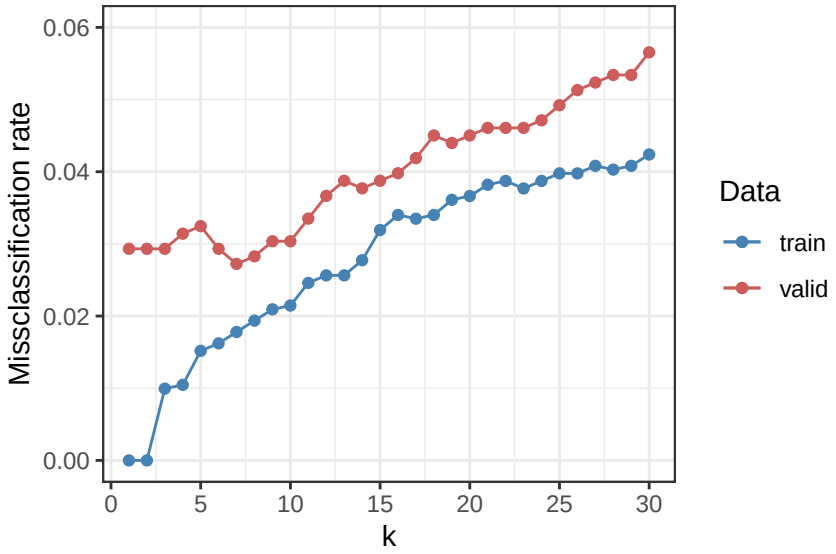

Figure 3: Misclassification errors for training and validation data for k=1,. . . ,30 on model trained on training data.

In figure 3, the model complexity is highest when k = 1 and decreases with larger k. With k = 1, the prediction in the model is by the observation in training data closest to the value we want to predict and when k is equal to the number of observations, the model will always predict the same value (except situations with ties). The training error increases when the model complexity decreases, this is because the model will be less overfitted on the training data with larger k and after some k, the model will be underfitted. Validation error also generally increased when the model complexity decreased, however at some k, the model will be between overfitted and underfitted which will give lowest validation error. This happened at k=7, which is considered to be the optimal k.

```
which(result$valid == min(result$valid))
```

```
## [1] 7
```

With k=7, the error for the test data is calculated. The errors for training, validation, and test data are compared in table 3.

```
model_test_7 <- kknn(formula = y~., train = train, test = test, kernel = "rectangular", k=7)
```

```
conf_mat_test <- table(model_test_7$fitted.values, test$y)
acc_test <- sum(diag(conf_mat_test)) / sum(conf_mat_test)
miss_test <- 1-acc_test

table_data <- cbind(training=result[7, 1], validtion=result[7, 2], test=miss_test)
kable(table_data, digits=3, caption="Misclassification error k=7 for different data.")
```

Table 3: Misclassification error k=7 for different data.

training	validtion	test
0.018	0.027	0.039

In table 3, the error for training is the lowest, followed by validation, and then test. Since the error for training is decently lower the model is a bit overfitted on training data. The difference between validation and test can be interpreted that for k=7 the error is smallest for the validation data, but it might not be the best since there is a bias when picking k from validation data.

1.5 1.5

Question: Fit K-nearest neighbor classifiers to the training data for different values of $K = 1, 2, ..., 30$, compute the error for the validation data as cross-entropy (when computing log of probabilities add a small constant within log, e.g. 1e-15, to avoid numerical problems) and plot the dependence of the validation error on the value of K. What is the optimal K value here? Assuming that response has multinomial distribution, why might the cross-entropy be a more suitable choice of the error function than the misclassification error for this problem?

Answer: The code used to compute cross-entropy for validation data

```
cross_entropy <- function(k){
  model_kknn_valid <-
    kknn(formula = y~.,
         train = train,
         test = valid,
         kernel = "rectangular",
         k=k)

  y <- as.integer(valid$y)

  prob <- c()
  for(i in 1:length(y)){
    prob[i] <- model_kknn_valid$prob[i, y[i]]
  }

  value <- -sum(log(prob + 1e-15))
  return(value)
}
```

```
result <- c()
for(i in 1:30){
  model <- cross_entropy(i)
  result[i] <- model
}
```

The cross-entropy for different k is presented in figure 4.

```
plot_data <- data.frame(index=1:30, result)
ggplot(plot_data, aes(x=index, y=result)) +
  geom_point(color="forestgreen") +
  geom_line(color="forestgreen") +
  scale_x_continuous(breaks = c(seq(from=0, to=30, by=5))) +
  theme_bw() +
  labs(x="K",
       y="Cross-entropy")
```

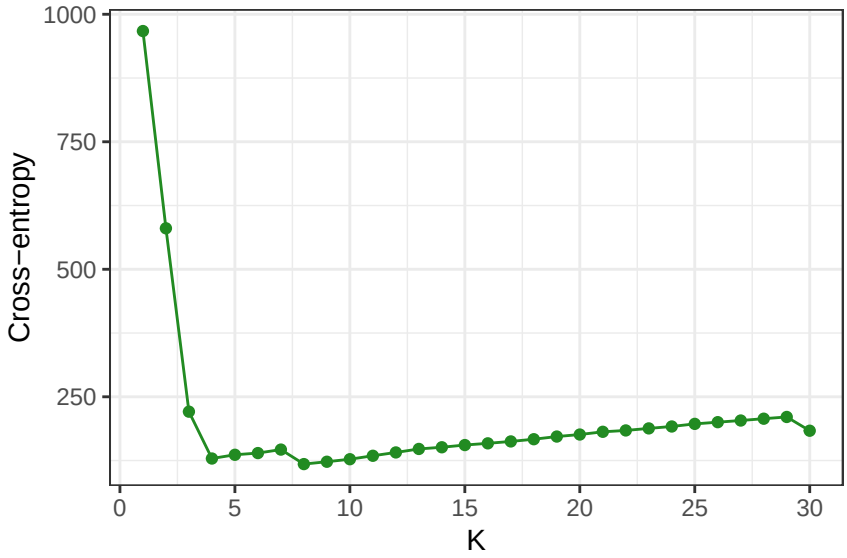

Figure 4: Cross-entropy error for validation data for different values of k for kknn models.

```
which(min(result) == result)
```

```
## [1] 8
```

From figure 4, the model with k=8 has the lowest value for cross-entropy and is considered to be the best k.

If the response has multinomial distribution, the maximum likelihood estimation is:

$$L(Y_i = C_1, Y_i = C_2, ..., Y_i = C_m|\theta) = \prod_{i=1}^{N} p_\theta(Y_i = C_m) \tag{1}$$

where Y_i is observation i, N is number of observations, and C_m is class m.

The log-likelihood of equation 1 is:

$$logL(Y_i = C_1, Y_i = C_2, ..., Y_i = C_m) = \sum_{i=1}^{N} logp_\theta(Y_i = C_m) \tag{2}$$

2 Statement of Contribution

We worked on the assignment individually for the computer labs (to be more efficient when asking questions), Duc on task 1, Sigme on task 2, and William on task 3. We later solved all assignment individually and compared and discussed our solutions before dividing the task of writing the laboration report.

2.1 Question 1

Text written by Duc.

2.2 Question 2

Text written by Sigme.

2.3 Question 3

Text written by William.

3 Appendix

The code used in this laboration report are summarised in the code as follows:

```
library(ggplot2)
library(kknn)
library(dplyr)
library(knitr)
knitr::opts_chunk$set(
  echo = TRUE,
  fig.width = 4.5,
  fig.height = 3)
# Read in data
data <- read.csv("optdigits.csv")

# Renaming the response variable and changing it to a factor variable
data <- rename(data, y=X0.26)
data$y <- as.factor(data$y)

# Partitioning training data (50%)
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]

# Partitioning validation data (25%)
id1=setdiff(1:n, id)
```

```r
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

# Partitioning test data (25%)
id3=setdiff(id1,id2)
test=data[id3,]
# kknn on training data and evaluation on training data
model_kknn_train <- kknn(formula=y~., train=train, test=train, kernel="rectangular", k=30)
conf_mat_train <- table(train$y, model_kknn_train$fitted.values)
acc_train <- sum(diag(conf_mat_train)) / sum(conf_mat_train)
miss_train <- 1-acc_train

# kknn on training data and evaluation on test data
model_kknn_test <- kknn(formula=y~., train=train, test=test, kernel="rectangular", k=30)
conf_mat_test <- table(test$y, model_kknn_test$fitted.values)
acc_test <- sum(diag(conf_mat_test)) / sum(conf_mat_test)
miss_test <- 1-acc_test

# Rows are true values, columns are model prediction
kable(conf_mat_train, caption = "Confusion matrix for training data, model
      predictions by columns and true value by rows.")
miss_train
kable(conf_mat_test, caption = "Confusion matrix for test data, model
      predictions by columns and true value by rows.")
miss_test
y <- train$y
fit_y <- model_kknn_train$fitted.values
# probabilities given from number 0 to 9, index 9 = number 8.
prob_8 <- model_kknn_train$prob[, 9]

# Data frame consisting of true value of y, model prediction and the models
# probability that the number is 8.
data_8 <- data.frame(y = y, fit_y = fit_y, prob = prob_8)
data_8$observation_id <- rownames(data_8)

# Only observations with the label 8 is kept.
data_8 <- data_8[data_8$y == "8", ]
head(arrange(data_8, prob), 2)
tail(arrange(data_8, prob), 3)
# Change colour palette to black and white
colfunc <- colorRampPalette(c("white", "black"))

plot_8 <- function(index){
  title <- paste0("Obs: ", index)
  # Reshapes the observations to a 8x8
  plot <- as.matrix(train[index, -65]) # Remove response variable
  plot <- matrix(plot, nrow=8, byrow=TRUE)
  heatmap(plot, col=colfunc(16), Colv=NA, Rowv=NA, main=title, margins=c(2,2))
```

```
}
plot_8(1624)
plot_8(1663)
plot_8(1810)
plot_8(1811)
plot_8(1864)
fit_kknn <- function(k){
  model_kknn_train <- kknn(formula=y~., train=train, test=train, kernel="rectangular", k=k)
  # Confusion matrix for train data
  conf_mat_train <- table(model_kknn_train$fitted.values, train$y)
  acc_train <- sum(diag(conf_mat_train)) / sum(conf_mat_train)
  # Missclassification for training data
  miss_train <- 1-acc_train

  model_kknn_valid <- kknn(formula=y~., train=train, test=valid, kernel="rectangular", k=k)
  # Confusion matrix for validation data
  conf_mat_valid <- table(model_kknn_valid$fitted.values, valid$y)
  acc_valid <- sum(diag(conf_mat_valid)) / sum(conf_mat_valid)
  # Missclassification for validation data
  miss_valid <- 1-acc_valid

  result <- c(miss_train, miss_valid)
  return(result)
}

# Missclassification for k=1,...,30 for training and validation data
result <- data.frame(train = 0, valid = 0)
for(i in 1:30){
  model <- fit_kknn(i)
  result[i,1] <- model[1]
  result[i,2] <- model[2]
}
result$index <- 1:30
ggplot(result, aes(x=index)) +
  geom_line(aes(y=train, colour="train")) +
  geom_point(aes(y=train, colour="train")) +
  geom_line(aes(y=valid, colour="valid")) +
  geom_point(aes(y=valid, colour="valid")) +
  scale_color_manual(name = "Data",
                     values = c("train" = "steelblue", "valid" = "indianred")) +
  scale_x_continuous(breaks = c(seq(from=0, to=30, by=5))) +
  scale_y_continuous(limits = c(0, 0.06)) +
  theme_bw() +
  labs(x = "k",
       y = "Missclassification rate")
which(result$valid == min(result$valid))
model_test_7 <- kknn(formula = y~., train = train, test = test, kernel = "rectangular", k=7)

conf_mat_test <- table(model_test_7$fitted.values, test$y)
```

```r
acc_test <- sum(diag(conf_mat_test)) / sum(conf_mat_test)
miss_test <- 1-acc_test

table_data <- cbind(training=result[7, 1], validtion=result[7, 2], test=miss_test)
kable(table_data, digits=3, caption="Misclassification error k=7 for different data.")
cross_entropy <- function(k){
  model_kknn_valid <-
    kknn(formula = y~.,
         train = train,
         test = valid,
         kernel = "rectangular",
         k=k)

  y <- as.integer(valid$y)

  prob <- c()
  for(i in 1:length(y)){
    prob[i] <- model_kknn_valid$prob[i, y[i]]
  }

  value <- -sum(log(prob + 1e-15))
  return(value)
}

result <- c()
for(i in 1:30){
  model <- cross_entropy(i)
  result[i] <- model
}
plot_data <- data.frame(index=1:30, result)
ggplot(plot_data, aes(x=index, y=result)) +
  geom_point(color="forestgreen") +
  geom_line(color="forestgreen") +
  scale_x_continuous(breaks = c(seq(from=0, to=30, by=5))) +
  theme_bw() +
  labs(x="K",
       y="Cross-entropy")
which(min(result) == result)
```