Laboratory 1 Report

Johan Lundin (johlu023) Greeshma Jeev Koothuparambil (greko370)
Daisuke Ronald Ssegwanyi Yamashita (ronss490)
Sangeeth Sankunny Menon (sansa237)

2023-12-20

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

Task 1-3

The confusion matrix for the train data is as follows:

```
confusionmat1
```

```
##    fit1
##       0   1   2   3   4   5   6   7   8   9
##   0 202   0   0   0   0   0   0   0   0   0
##   1   0 179  11   0   0   0   0   1   1   3
##   2   0   1 190   0   0   0   0   1   0   0
##   3   0   0   0 185   0   1   0   1   0   1
##   4   1   3   0   0 159   0   0   7   1   4
##   5   0   0   0   1   0 171   0   1   0   8
##   6   0   2   0   0   0   0 190   0   0   0
##   7   0   3   0   0   0   0   0 178   1   0
##   8   0  10   0   2   0   0   2   0 188   2
##   9   1   3   0   5   2   0   0   3   3 183
```

The misclassification error is 0.0450026

```
##   Class TotalNumber ErrorPredictions
## 0     0         202                0
## 1     1         195               16
## 2     2         192                2
## 3     3         188                3
## 4     4         175               16
## 5     5         181               10
## 6     6         192                2
## 7     7         182                4
## 8     8         204               16
## 9     9         200               17
```

From the table we can see that the maximum error by the model (using it own training data) happened in the case of 9. The predictions for number '0' turned out without any errors. The prediction quality on the training data looks good as 95.5% of the predictions are correct

The confusion matrix for the train data is as follows:

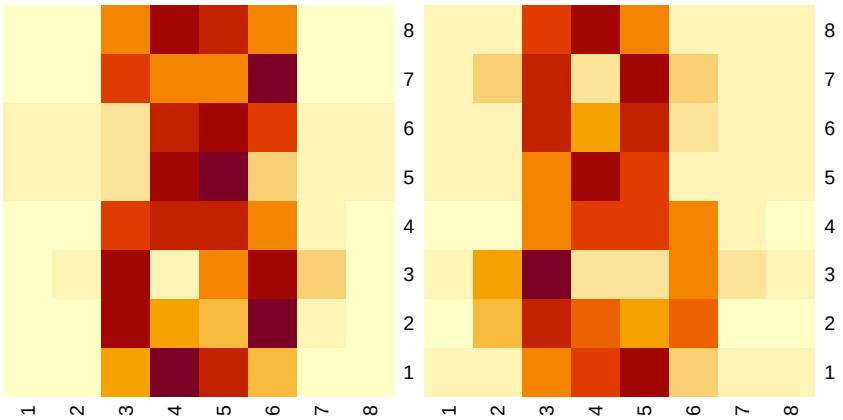
```
confusionmat
```

```
##    fit
##       0   1   2   3   4   5   6   7   8   9
##   0  77   0   0   0   1   0   0   0   0   0
##   1   0  81   2   0   0   0   0   0   0   3
##   2   0   0  98   0   0   0   0   0   3   0
##   3   0   0   0 107   0   2   0   0   1   1
##   4   0   0   0   0  94   0   2   6   2   5
##   5   0   1   1   0   0  93   2   1   0   5
##   6   0   0   0   0   0   0  90   0   0   0
##   7   0   0   0   1   0   0   0 111   0   0
##   8   0   7   0   1   0   0   0   0  70   0
##   9   0   1   1   1   0   0   0   1   0  85
```

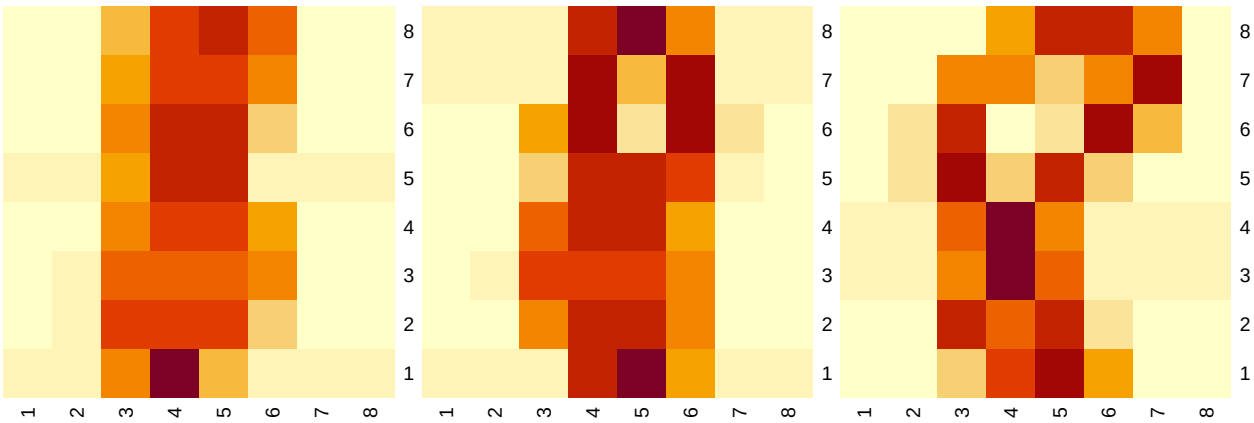
The misclassification error is 0.0532915

```
##   Class TotalNumber ErrorPredictions
## 0     0          78                1
## 1     1          86                5
## 2     2         101                3
## 3     3         111                4
## 4     4         109               15
## 5     5         103               10
## 6     6          90                0
## 7     7         112                1
## 8     8          78                8
## 9     9          89                4
```

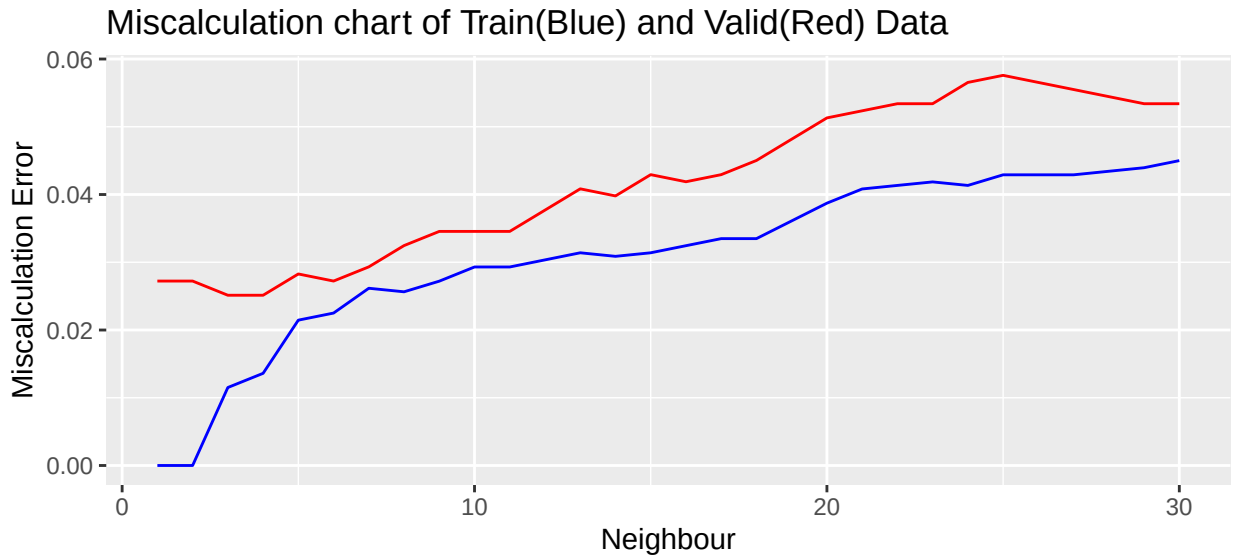
From the table we can see that the maximum error by the model (trained with training data) when using test data happened in the case of 4. The prediction for number '6' turned out without any errors. The prediction quality on the test data looks good as 95% of the predictions are correct.



The heat maps in 'easy' observations (above) exhibit the features of digit '8' and can be recognized as an eight, while the 'hard' observations (below) of 8 data is hardly recognizable an eight which explains the low probability value as well.



Task 4-5

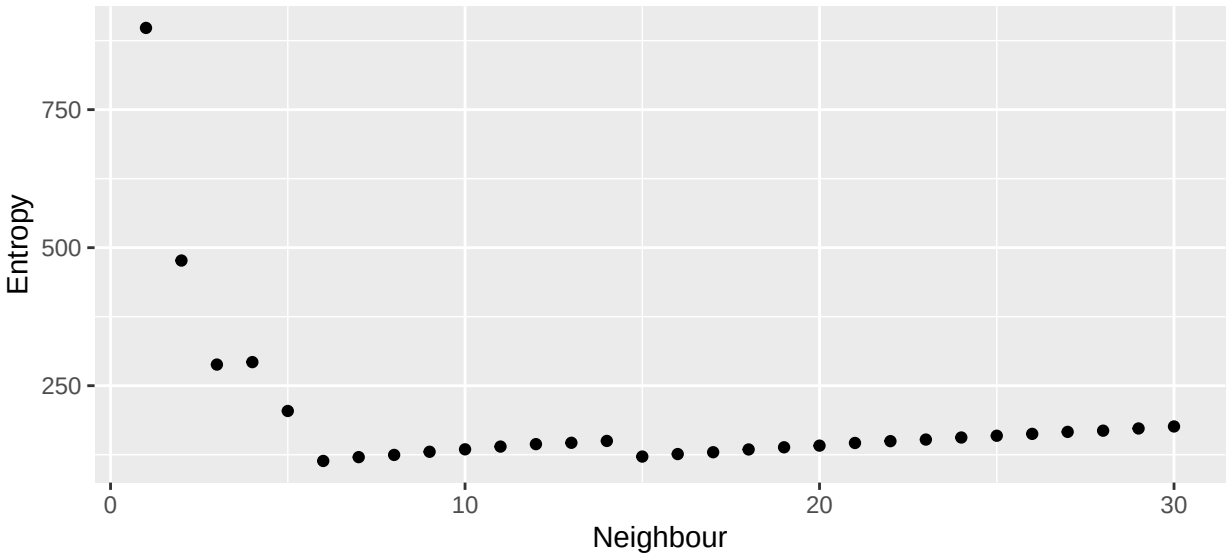


From the graph we can see that as the number of neighbours in the algorithm increases the model complexity increases and the misclassification error also considerably increase. For valid data it initially reduces to its optimal value and then increases further on.

According to the graph the optimal k seems to be two values which are 3 and 4. We choose 4 as the optimal k value. The misclassification error for the train, valid and test data on 4-Nearest neighbors are as follows:

- Misclassification error for Train Data : 0.0136054
- Misclassification error for Valid Data : 0.0251309
- Misclassification error for Test Data : 0.0250784

As the correct prediction rate on the data sets falls in the range of 97.5% to 98.7% the model can be seen as a good prediction model.



The cross entropy plot shows that cross entropy for Valid data is at the lowest for k= 6 which is 113.768. Cross entropy uses probability distribution for error calculation while MSE uses just misclassification values for the estimation of error. Evaluating a model based on the probability distribution seems efficient for multinomial distribution classes as it considers the confidence of prediction given out by the model.

Assignment 2: Linear regression and ridge regression.

Task 1-2

```
summary(model_train)
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0255 -0.7363 -0.1087  0.7333  2.1960
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.575e-15  1.583e-02   0.000 1.000000
## Jitter...      1.869e-01  1.496e-01   1.250 0.211496
## Jitter.Abs.   -1.696e-01  4.081e-02  -4.156 3.32e-05 ***
## Jitter.RAP    -5.270e+00  1.884e+01  -0.280 0.779688
## Jitter.PPQ5   -7.457e-02  8.778e-02  -0.850 0.395659
## Jitter.DDP     5.250e+00  1.884e+01   0.279 0.780541
## Shimmer        5.924e-01  2.060e-01   2.876 0.004055 **
## Shimmer.dB.   -1.727e-01  1.393e-01  -1.239 0.215380
## Shimmer.APQ3   3.207e+01  7.717e+01   0.416 0.677738
## Shimmer.APQ5  -3.875e-01  1.138e-01  -3.405 0.000669 ***
## Shimmer.APQ11  3.055e-01  6.124e-02   4.989 6.37e-07 ***
## Shimmer.DDA   -3.239e+01  7.717e+01  -0.420 0.674739
## NHR           -1.854e-01  4.557e-02  -4.068 4.85e-05 ***
## HNR           -2.385e-01  3.640e-02  -6.553 6.45e-11 ***
## RPDE           4.068e-03  2.267e-02   0.179 0.857576
## DFA           -2.803e-01  2.014e-02 -13.919  < 2e-16 ***
## PPE            2.265e-01  3.289e-02   6.886 6.75e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9396 on 3508 degrees of freedom
## Multiple R-squared:  0.1212, Adjusted R-squared:  0.1172
## F-statistic: 30.24 on 16 and 3508 DF,  p-value: < 2.2e-16
```

```
MSE_train_data
```

```
## [1] 0.8785431
```

```
MSE_test_data
```

```
## [1] 0.9354477
```

Thus, the MSE for the train data and test data is 0.88 and 0.94 respectively. The variables that contribute significantly to the model are Jitter.Abs, Shimmer.APQ5, Shimmer.APQ11, NHR, HNR, DFA, and PPE. This is because the these variables have either a significantly low p-value, or have a p-value equal to zero. They have a p-value less than the significance level.

Task 3

All the four functions are given in the appendix below.

Task 4

```
## Sample data..
results <- data.frame(
  degree_of_freedom = c(DF(1), DF(100), DF(1000)),
  Lambda = c(1, 100, 1000),
  train_data_MSE = c(MSE_train_data_1, MSE_train_data_100, MSE_train_data_1000),
  test_data_MSE = c(MSE_test_data_1, MSE_test_data_100, MSE_test_data_1000)
)
# Print the table.
knitr::kable(results, caption = "Sample data")
```

Table 1: Sample data

degree_of_freedom	Lambda	train_data_MSE	test_data_MSE
13.860736	1	0.8786272	0.9349975
9.924887	100	0.8844099	0.9323320
5.643925	1000	0.9211232	0.9539399

As the value of lambda increases, the degree of freedom decreases. This means the predicted results would be less and less accurate. Higher lambda values lead to more aggressive shrinkage to the coefficients towards zero, thus reducing the variance of the parameter estimates. The larger the lambda, the more coefficients are pushed towards zero, reducing their impact on the model's prediction. However, there is a trade-off while higher lambda values can reduce overfitting and improve generalization to some extent, excessively high values might lead to underfitting. This occurs when the model becomes too constrained, resulting in poor performance on both the training and test data. Therefore, the most appropriate penalty parameter among the selected ones is lambda equals 100.

```
MSE_train_data_1
```

```
## [1] 0.8786272
```

```
MSE_test_data_1
```

```
## [1] 0.9349975
```

```
MSE_train_data_100
```

```
## [1] 0.8844099
```

```
MSE_test_data_100
```

```
## [1] 0.932332
```

```
MSE_train_data_1000
```

```
## [1] 0.9211232
```

```
MSE_test_data_1000
```

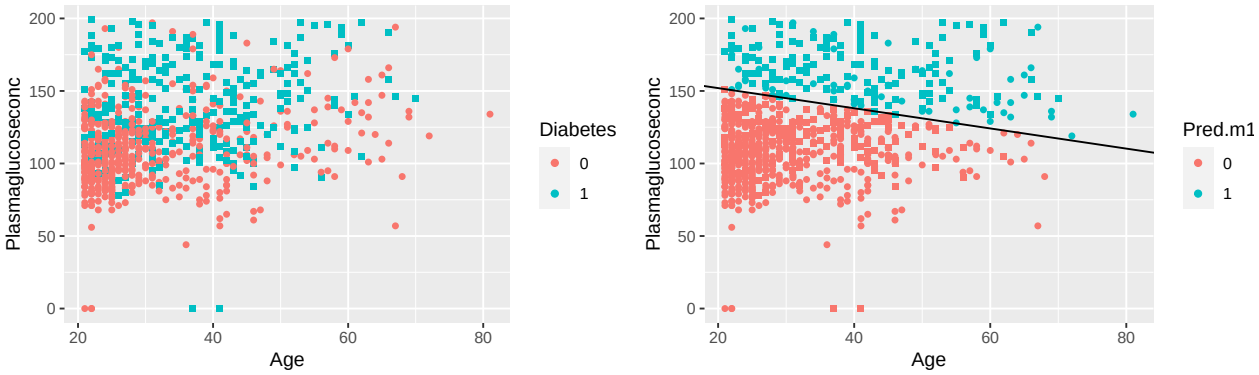
```
## [1] 0.9539399
```

Assignment 3: Logistic regression and basis function expansion.

Task 1-3

The scatter-plot below to the left show the age (x-axis) and Plasma Glucose Concentration (y-axis) with the patients with diabetes (1) given as blue and the patients not diagnosed with diabetes with red (0).

Given the scatter-plot below to the left it is difficult to discern a clear pattern, and it is difficult to classify diabetes, but what could be discerned is that a higher plasma glucose concentration seems to correlate with diabetes, especially for the lower and middle age groups, for the older ages the data is not as clear even though the higher plasma glucose concentration still seem to be indicate diabetes. However for the middle age groups it seems like a more modest plasma glucose level still results in diabetes (or the conclusion is that the younger age groups with medium levels will get it later).



Using the glm-function and 0.5 as the probability threshold gives the scatter-plot above to the right, all probabilities below 0.5 is set to 0 (and the model thus predicts those to not having diabetes) and all probabilities of 0.5 and above is set to 1 (i.e. predicted to have diabetes). The missclassifcation rate is 26.3% and the probabilistic equation is

$$P(x_1, x_2) = \frac{1}{1 + e^{-(b_0 + b_1 x_2 + b_2 x_2)}}$$

were the constants b_0, x_1 and x_2 was obtained from the logistic regression model function

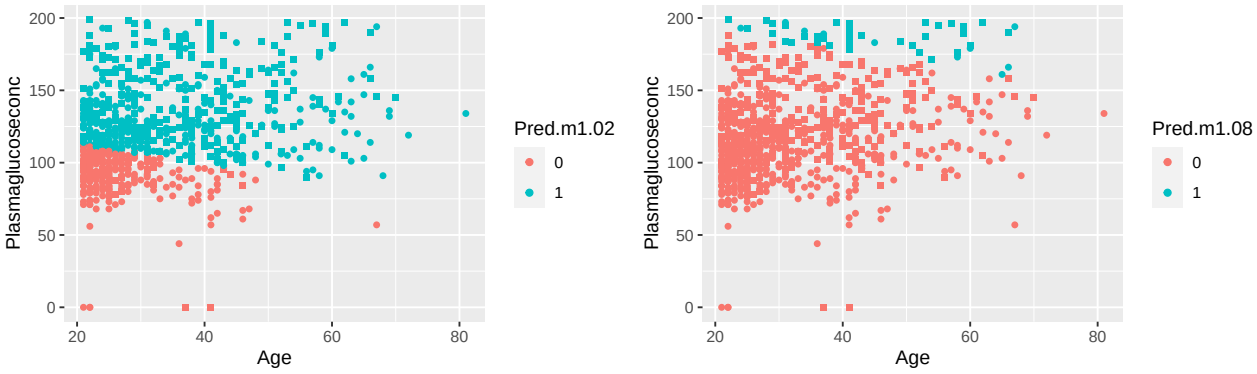
```
coefficients(m1)
```

```
##           (Intercept)              data$Age data$Plasmaglucoseconc
##           -5.91244906           0.02477835           0.03564404
```

However by just visually comparing with the first plot (with the actual diabetes cases), a linear decision boundary is quite a crude approximation, a more curved decision boundary might be a better fit, especially for the values in the lower middle ages (~40 years). The decision boundary in a logistic regression are linear, so the equation for the decision boundary is in the form $y = kx + m$ were k and m is giving by the slope and intercept $k = \frac{b_0}{-x_2}$ = -0.695 and $m = \frac{x_1}{-x_2}$ = 165.87

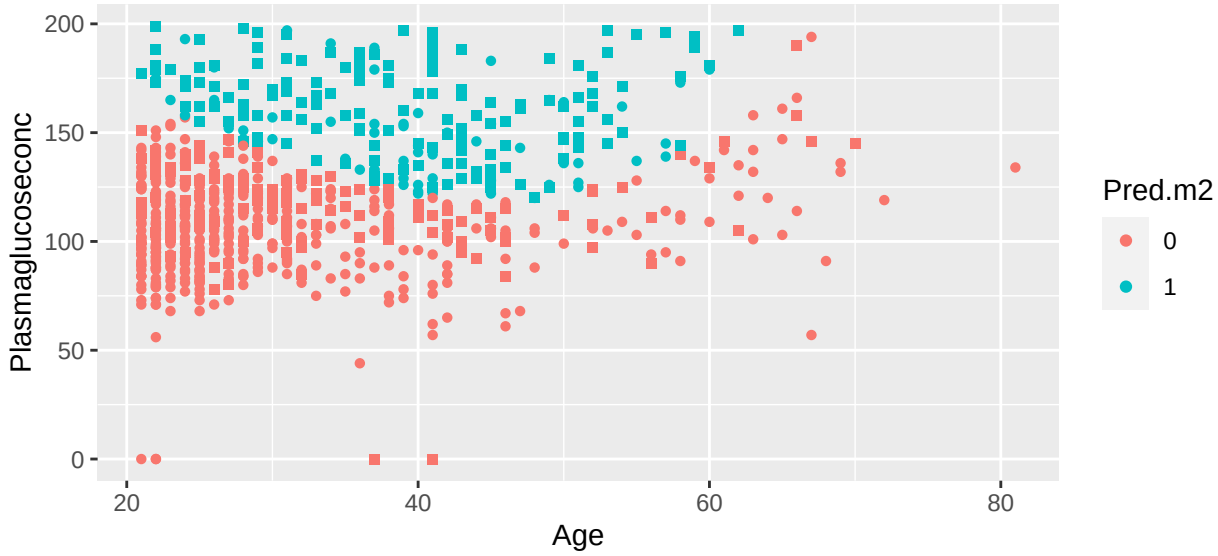
Task 4-5

Changing the probability threshold (r-value) to 0.2 and 0.8 gives the results shown in the plots below.



Changing the threshold value changes the decision boundary as expected, setting the decision threshold to 0.2 (left plot) means that all probabilities of 0.2 and above will be classified as diabetes, which of course gives predicts more diabetes and the opposite with a decision threshold of 0.8 (right plot). In example for screening purposes it might be better to have a lower threshold in order to not miss fewer potential diabetes patients.

By using the expansion trick has affected the decision boundary making it a polynomial (exponential) instead of a linear decision boundary, and more in line what a visual interpretation would give as a better fit for the lower middles ages. The missclassifcation rate is 24.1%, and thus slightly lower (than the previous 26.3%). However for the higher ages, everything gets predicted as diabetes.



APPENDIX

Statement of Contribution

Greeshma Jeev was responsible for the code for the first assignment.
Daisuke Ronald was responsible for the code for the second assignment.
Johan and Sangeeth was responsible for the code for the third assignment. The code submitted in this report was written by Johan with assistance and feedback from Sangeeth.

The assignments were briefly examined in the group and suggestions and clarifications was made, for example adding missing plots or values that were requested in the assignment.

Johan was responsible for creating the report (i.e. taking the code for each assignment, create the layout in rmarkdown and inserting plots and comments/analyses made by the person responsible) into the report.

Code for Assignment 1.

```
library(kknn)
library(ggplot2)
library(gridExtra)
#1.1
digitsdf <- read.csv("optdigits.csv", header = F)
colnames(digitsdf)[65] <- "Digit"

digitsdf[,65] <- as.factor(digitsdf[,65])

set.seed(123)
n=dim(digitsdf)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.50))
train=digitsdf[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=digitsdf[id2,]
id3=setdiff(id1,id2)
test=digitsdf[id3,]

#1.2
#training on training data and testing on test data
digits.kknn <- kknn(Digit~., train, test, k = 30,
                    kernel = "rectangular")
fit <- fitted(digits.kknn)
confusionmat <- table(test$Digit, fit)
tempcm <- confusionmat
diag(tempcm) <- 0
errortotal <- sum(tempcm)
miscal <- errortotal/ sum(confusionmat)

# both training and testing on training data
digits.kknn1 <- kknn(Digit~., train, train, k = 30,
                     kernel = "rectangular")
fit1 <- fitted(digits.kknn1)
```

```
confusionmat1 <- table(train$Digit, fit1)
tempcm1 <- confusionmat1
diag(tempcm1) <- 0
errortotal1 <- sum(tempcm1)
miscal1 <- errortotal1/ sum(confusionmat1)

## Analysis:

#Train -Test model
pred <- rowSums(confusionmat)
mispred <- rowSums(tempcm)
predanalysis <- data.frame(Class = c(0:9), TotalNumber = pred, ErrorPredictions =
                             mispred)



# Train-Train model:
pred1 <- rowSums(confusionmat1)
mispred1 <- rowSums(tempcm1)
predanalysis1 <- data.frame(Class = c(0:9), TotalNumber = pred1, ErrorPredictions =
                              mispred1)
```

```
#1.3

probability <- digits.kknn1$prob

train1 <- train
train1$pro <- probability[,9]

#drawing out all 8 class data
data8 <- train1[which(train1$Digit==8),]
row.names(data8) <- 1:nrow(data8)
ordpro <- order(data8$pro,decreasing = F)
data8 <- data8[order(data8$pro,decreasing = F),]


hard8 <- data8[c(1:3),]
easy8 <- data8[c((nrow(data8)-1):nrow(data8)),]
heatmapmat <- function(rowvals){
  matval <- as.data.frame(split(rowvals,1:8))
  colnames(matval) <- c(1:8)
  return(matval)
}
firsteasy8 <- heatmapmat(as.numeric(easy8[1,-c(65,66)]))
firsteasy8mat <- as.matrix(firsteasy8)
# heatmap(firsteasy8mat, Colv = NA, Rowv = NA)

secondeasy8 <- heatmapmat(as.numeric(easy8[2,-c(65,66)]))
secondeasy8mat <- as.matrix(secondeasy8)
# heatmap(secondeasy8mat, Colv = NA, Rowv = NA)

firsthard8 <- heatmapmat(as.numeric(hard8[1,-c(65,66)]))
firsthard8mat <- as.matrix(firsthard8)
```

```r
# heatmap(firsthard8mat, Colv = NA, Rowv = NA)

secondhard8 <- heatmapmat(as.numeric(hard8[2,-c(65,66)]))
secondhard8mat <- as.matrix(secondhard8)
# heatmap(secondhard8mat, Colv = NA, Rowv = NA)

thirdhard8 <- heatmapmat(as.numeric(hard8[3,-c(65,66)]))
thirdhard8mat <- as.matrix(thirdhard8)
# heatmap(thirdhard8mat, Colv = NA, Rowv = NA)


#1.4

miscalfun <- function(j, x){
  digits.kknn <- kknn(Digit~., train, test = x, k = j,
                      kernel = "rectangular")
  fit <- fitted(digits.kknn)
  confusionmat <- table(x$Digit, fit)
  tempcm <- confusionmat
  diag(tempcm) <- 0
  errortotal <- sum(tempcm)
  miscal <- errortotal/ sum(confusionmat)
  return(miscal)
}

miscalarraytrain <- c()
miscalarrayvalid <- c()
for (i in 1:30) {
  miscalarraytrain[i] <-miscalfun(i, train)
  miscalarrayvalid[i] <-miscalfun(i, valid)
}

miscaldf <- data.frame(Neighbour = c(1:30), ErrorTrain = miscalarraytrain,

                       ErrorValid = miscalarrayvalid)

miscalcplot = ggplot(miscaldf) +
  geom_line( aes(Neighbour, ErrorTrain), colour = 'blue') +

  geom_line( aes(Neighbour, ErrorValid), colour = 'red')+

  labs(title = "Miscalculation chart of Train(Blue) and Valid(Red) Data",
       y = "Miscalculation Error")


digitstest.kknn <- kknn(Digit~., train, test, k = 4,
                    kernel = "rectangular")
fit <- fitted(digitstest.kknn)
confusionmat <- table(test$Digit, fit)
tempcm <- confusionmat
diag(tempcm) <- 0
errortotal <- sum(tempcm)
miscaltest <- errortotal/ sum(confusionmat)
```

```
#1.5

crossentropyfun <- function(j){
  digits.kknn <- kknn(Digit~., train, valid, k = j,
                      kernel = "rectangular")
  probability <- as.data.frame(digits.kknn$prob)
  probability$Class <- valid$Digit

  fit <- fitted(digits.kknn)
  probability$fit <- fit

  probability$probval <-NA
  for (i in 1:nrow(probability)) {

      probability$probval[i] <- probability[i,(as.numeric(probability[i,11]))]

  }
  probability$logprobval <- log(probability$probval+ 1e-15)

  CEP <-  - sum(probability$logprobval[!is.na(probability$logprobval)])

  return(CEP)

}

CEParray <- c()
for (i in 1:30) {
  CEParray[i] <-crossentropyfun(i)
}
CEPdf <- data.frame(Neighbour = c(1:30), Entropy = CEParray)
ceplot = ggplot(CEPdf,mapping = aes(x= Neighbour , y =Entropy))+ geom_point()
```

Code for Assignment 2.

```
### Assignment 2: Linear regression and ridge regression


# Part 1

parkhinson <- read.csv("parkinsons.csv", header = TRUE)
library(caret)
parkhinson <- parkhinson[ , c(7:22, 5)]



# Split the data into train data and test data
n <- nrow(parkhinson)
```

```r
set.seed(12345)
id <- sample(1:n, floor(n*0.6))
train_data <- parkhinson[id,]
id1 <- setdiff(1:n, id)
set.seed(12345)
id2 <- sample(id1, floor(n*0.4))
test_data <- parkhinson[id2,]

# Scale the parkhinson data.
scaler <- preProcess(train_data)
train_data <- predict(scaler,train_data)
test_data <- predict(scaler,test_data)

# Part 2

## Compute a linear regression model from train data and estimate
## train and test MSE.

## Fit a linear regression model on the training data,

model_train <- lm(motor_UPDRS ~. , data = train_data)
summary(model_train)

## Estimate the train MSE and test MSE.

# Prediction for train and test data.
fit_train_data <- predict(model_train, train_data[, -17])
fit_test_data <- predict(model_train, test_data[, -17])

train_data$fit <- fit_train_data
test_data$fit <- fit_test_data

# Compute the MSE for train and test data.
MSE_train_data <- mean((train_data$motor_UPDRS - fit_train_data)^2)
MSE_test_data <- mean((test_data$motor_UPDRS - fit_test_data)^2)



# Part 3

## a)

# Implement 4 functions using basic R commands
# (no external packages)

## Loglikelihood function(computes the loglikelihood function
## logP(T|theta, sigma) for the model and training data.
loglikelihood <- function(theta, sigma) {
  n <- nrow(train_data)
  mat_train <- as.matrix(train_data[,-c(17,18)])
  thetabytrain <- mat_train %*% theta
  log_likelihood <- (-n/2 * log(2 * pi) - n/2*log(sigma^2)) - (1/(2 * sigma^2)) *
    (sum((train_data[ , 17] - thetabytrain)^2))
```

```r
  return(log_likelihood)
}

theta<- as.matrix(model_train$coefficients[-1], nrow = length(model_train$coefficients[-1]))
sigma<- summary(model_train)$sigma
loglikelihood(theta, sigma)

## Ridge function.

Ridge <- function(theta, sigma, lambda) {

  ridge_penalty <- lambda * sum(theta^2)
  log_likelihood <- loglikelihood(theta, sigma)
  ridge_minus_log_likelihood <- -log_likelihood + ridge_penalty
  return(ridge_minus_log_likelihood)
}

## RidgeOpt function(computes the optimal theta and sigma for given lambda).

lambda <-1
RidgeOpt <- function(lambda) {

  optimisefun <- function(x) {
    t <- x[1:16]
    s <- x[17]
    return(Ridge(t, s, lambda))
  }

  optimal <- optim(rep(1, 17), fn = optimisefun, method = "BFGS")
  return(optimal)
}

coef_ridge <- RidgeOpt(lambda)

## DF function(computes the degrees of freedom of the Ridge model)

DF <- function(lambda){
    X <- as.matrix(train_data[ , -c(17,18)])
    I <- diag(16)
    df <- sum(diag(X %*% solve((t(X) %*% X + lambda * I)) %*% t(X)))

  return(df)
}

freedom1 <- DF(1)
freedom100 <- DF(100)
freedom1000 <- DF(1000)

### Question 4

# When lambda = 100,

theta_100 <- RidgeOpt(100)
```

```
y_hat_train <- as.matrix(train_data[ , -c(17, 18)]) %*%
  as.matrix(theta_100$par[-17], nrow=length(theta_100$par[-17]))

# Calculate the MSE for train data.
MSE_train_data_100 <- 1/length(y_hat_train)*sum((train_data[ , 17] - y_hat_train)^2)

# Calculate the MSE for test data.

y_hat_test <- as.matrix(test_data[ , -c(17, 18)]) %*%
  as.matrix(theta_100$par[-17], nrow=length(theta_100$par[-17]))

MSE_test_data_100 <- 1/length(y_hat_test)*sum((test_data[ , 17] - y_hat_test)^2)

# When lambda = 1,

theta_1 <- RidgeOpt(1)

y_hat_train <- as.matrix(train_data[ , -c(17, 18)]) %*%
  as.matrix(theta_1$par[-17], nrow=length(theta_1$par[-17]))

# Calculate the MSE for train data when lambda = 1.
MSE_train_data_1 <- 1/length(y_hat_train)*sum((train_data[ , 17] - y_hat_train)^2)

# Calculate the MSE for test data when lambda = 1.

y_hat_test <- as.matrix(test_data[ , -c(17, 18)]) %*%
  as.matrix(theta_1$par[-17], nrow=length(theta_1$par[-17]))
MSE_test_data_1 <- 1/length(y_hat_test)*sum((test_data[ , 17] - y_hat_test)^2)

# When lambda = 1000,

theta_1000 <- RidgeOpt(1000)

y_hat_train <- as.matrix(train_data[ , -c(17, 18)]) %*%
  as.matrix(theta_1000$par[-17], nrow=length(theta_1000$par[-17]))

# Calculate the MSE for train data lambda = 1000.
MSE_train_data_1000 <- 1/length(y_hat_train)*sum((train_data[ , 17] - y_hat_train)^2)

# Calculate the MSE for test data lambda = 1000.

y_hat_test <- as.matrix(test_data[ , -c(17, 18)]) %*%
  as.matrix(theta_1000$par[-17], nrow=length(theta_1000$par[-17]))
MSE_test_data_1000 <- 1/length(y_hat_test)*sum((test_data[ , 17] - y_hat_test)^2)
```

```
## Sample data..
results <- data.frame(
  degree_of_freedom = c(DF(1), DF(100), DF(1000)),
  Lambda = c(1, 100, 1000),
  train_data_MSE = c(MSE_train_data_1, MSE_train_data_100, MSE_train_data_1000),
  test_data_MSE = c(MSE_test_data_1, MSE_test_data_100, MSE_test_data_1000)
)
# Print the table.
```

```r
knitr::kable(results, caption = "Sample data")
```

Code for Assignment 3.

```r
# INTRODUCTION TO MACHINE LEARNING A732A68 LIU
# LAB 1 / ASSIGNMENT 3 / LOGISTIC REGRESSION AND BASIS FUNCTION EXPANSION
# PIMA INDIANS

# Clear the workspace
rm(list = ls())

# Reading data file pima-indians-diabetes.csv. A header with names for each
# column is not included in the csv-file.

data = read.csv("pima-indians-diabetes.csv", header = FALSE)

# Since there are no headers, I've named them manually with the information
# given in the assignment.

colnames(data)[1] = "Pregnant"          # Number of times pregnant
colnames(data)[2] = "Plasmaglucoseconc" # Plasma glucose concentration
colnames(data)[3] = "Bloodpressure"     # Diastolic blood pressure
colnames(data)[4] = "Skinthick"         # Triceps skinfold thickness
colnames(data)[5] = "Seruminsul"        # Serum insulin
colnames(data)[6] = "BMI"               # Body Mass Index
colnames(data)[7] = "Diabetfunc"        # Diabetes pedigree function
colnames(data)[8] = "Age"               # Age (years)
colnames(data)[9] = "Diabetes"          # Diabetes (0 = no , 1 = yes)


# TASK 1
# Using package ggplot2 to plot age (x) vs Plasma glucose concentration (y) color
# coded from dark blue to yellow according to the value of the diabetes pedigree
# function. Also all subjects with a diabets diagnosis is plotted with a square,
# the rest with a circle.

library(ggplot2)
plot1 = ggplot(data, aes(Age, Plasmaglucoseconc))+
  geom_point(aes(color = as.factor(Diabetes)), shape=ifelse(data$Diabetes=="1", 15, 16) ) +
  theme(legend.position = "right") + labs(color="Diabetes")

# TASK 2
# Train a logistic regression model and make predictions for _all_ observations,
# so I'm not dividing the data into training and test.

# Using glm to "compute logistic regression" as used in Tutorial 1, probability
# statistics and machine learning in exercise 9.
# https://www.ida.liu.se/~732A99/info/tutorials/tutorial1.html
m1=glm(data$Diabetes~data$Age+data$Plasmaglucoseconc,  family = "binomial")

# Using m1 to give the probability of having diabetes. If the probability is above
```

```r
# r then it will be classified as having diabetes.

# Setting the probability threshold
r = 0.5

Prob.m1 = predict(m1, type="response")
Pred.m1 = ifelse(Prob.m1>r, "1", "0")

# Probabilistic equation
# The equation should look something like this
# Prob.m1 = 1 / (1+exp(- (b0+b1x2+b2x2)))
# Where b0, b1, b2 is given by
coefficients(m1)

# Miss-classification rate (code from the slides Lecture 1C, page 31).
missclass=function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

mc.m1 = missclass(data$Diabetes, Pred.m1)

# Scatter plot of the same kind as above, but showing the predicted values of
# diabetes as a color (red predicted as diabetes (1) and green as no diabetes (0)

plot2 = ggplot(data, aes(Age, Plasmaglucoseconc)) +
  geom_point(aes(color = Pred.m1), shape=ifelse(data$Diabetes=="1", 15, 16) )

# TASK 3
# Use the model in previous task to report
# a) equation of decision boundary
# b) curve showing this boundary to the scatter plot in step 2

# "The decision boundaries in logistic regression are linear" as stated in
# Lecture 1D. Taking the coefficients from the model (glm/m1) to get the slope (k)
# and the intercept value.

slop = coef(m1)[2]/(-coef(m1)[3])
intc = coef(m1)[1]/(-coef(m1)[3])


plot3 = plot2 + geom_abline(intercept = intc, slope = slop)

#TASK4
# Plot for two different probablity thresholds
r = 0.2
Pred.m1.02 = ifelse(Prob.m1>r, "1", "0")
r = 0.8
Pred.m1.08 = ifelse(Prob.m1>r, "1", "0")

# Load library to create side-by-side-plot
library(gridExtra)
```

```r
plot2.02 = ggplot(data, aes(Age, Plasmaglucoseconc)) +
  geom_point(aes(color = Pred.m1.02), shape=ifelse(data$Diabetes=="1", 15, 16) )

plot2.08 = ggplot(data, aes(Age, Plasmaglucoseconc)) +
  geom_point(aes(color = Pred.m1.08), shape=ifelse(data$Diabetes=="1", 15, 16) )


# TASK 5
# Performing a basis function expansion trick (as given by the lab instructions),
# I set x1 as the Age and x2 as the Plasmaglucoseconc(entration)

x1 = data$Age
x2 = data$Plasmaglucoseconc

# Then z1 = x1^4 , z2 = x1^3*x2 , z3 = x1^2 * x2^2, z4 = x1 * x2^2, z5= x2^4 and
# add them to the data set (as given by lab instruction)

data$z1 = x1^4
data$z2 = x1^3*x2
data$z3 = x1^2 * x2^2
data$z4 = x1 * x2^2
data$z5 = x2^4

# Doing the same thing as in Task 2 but with the new features.

m2=glm(data$Diabetes~data$Age+data$Plasmaglucoseconc+data$z1+data$z2+data$z3+data$z4,
       family = "binomial")

r = 0.5
Prob.m2 = predict(m2, type="response")
Pred.m2 = ifelse(Prob.m2>r, "1", "0")


mc.m2 = missclass(data$Diabetes, Pred.m2)

plot4 = ggplot(data, aes(Age, Plasmaglucoseconc)) +
  geom_point(aes(color = Pred.m2), shape=ifelse(data$Diabetes=="1", 15, 16) )
```