Machine Learning Lab 1

Farid Musayev, Kristina Levina, and Wuhao Wang

18 November, 2021

Contribution

All the group members have finished 3 assignments and helped each other with problems and reached a consensus on solutions to the problems. To be more specific:

Wuhao Wang is responsible for assignment 1 and its report.
Kristina Levina is responsible for assignment 2 and its report.
Farid Musayev is responsible for assignment 3 and its report.

Assignment1

Q1:

Load data and then split them into three part. And also rename the columns.

Q2:

1) digits 6 0 7 can be predicted very well both in training data and test data(over 98.8% correct rate).

2) digits 2 3 9 are also in good condition(over 95%).

3) digits 1 5, the performance not very good(over 90%).

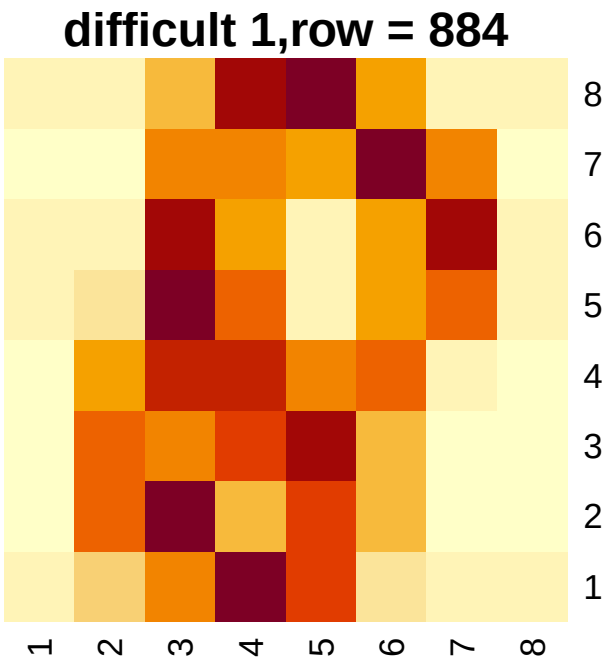
4) digits 4 8, not very good(lower than 90%).

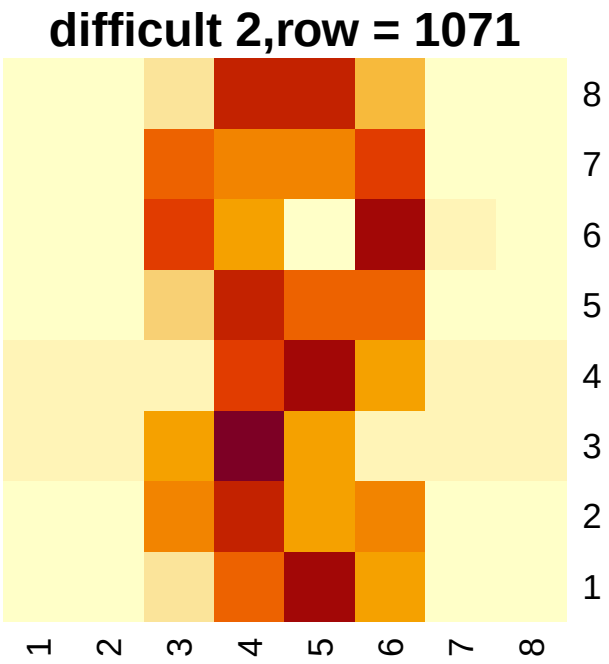
5) overall : average error rate is 0.045 in training data and 0.053 in test data. This model is not over fitting.

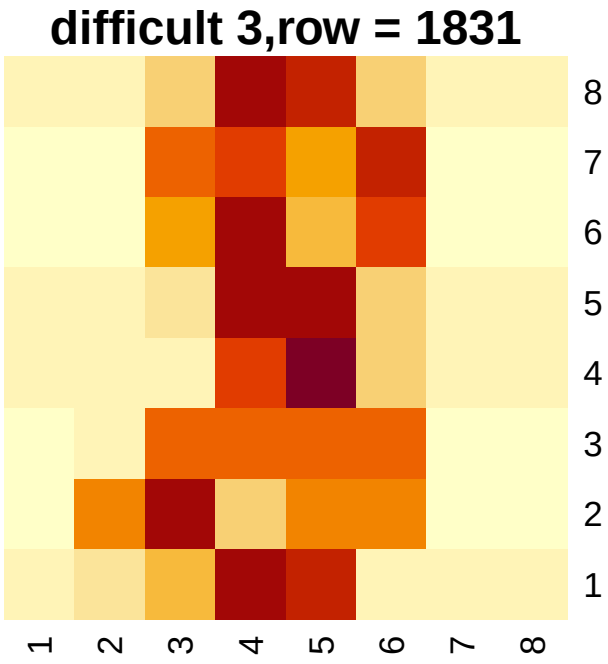
```
##                       0          1          2          3         4          5 6
## MisC rates 0.01282051 0.05813953 0.02970297 0.03603604 0.1376147 0.09708738 0
##                       7         8          9
## MisC rates 0.008928571 0.1025641 0.04494382
```

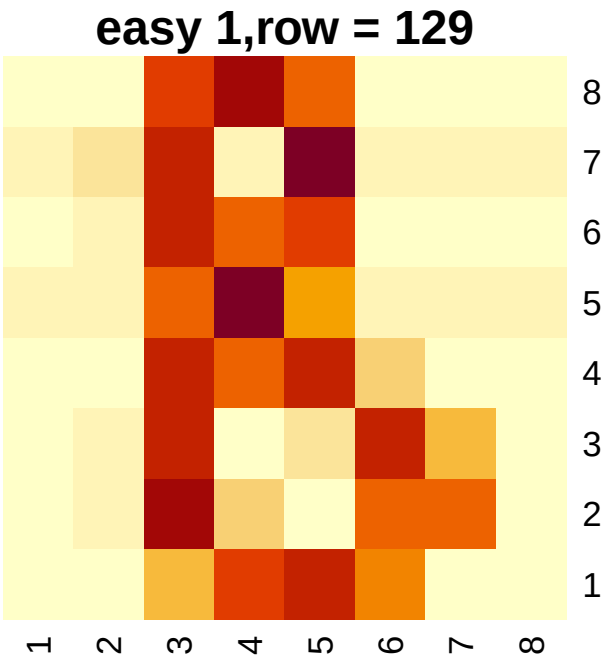
Q3:

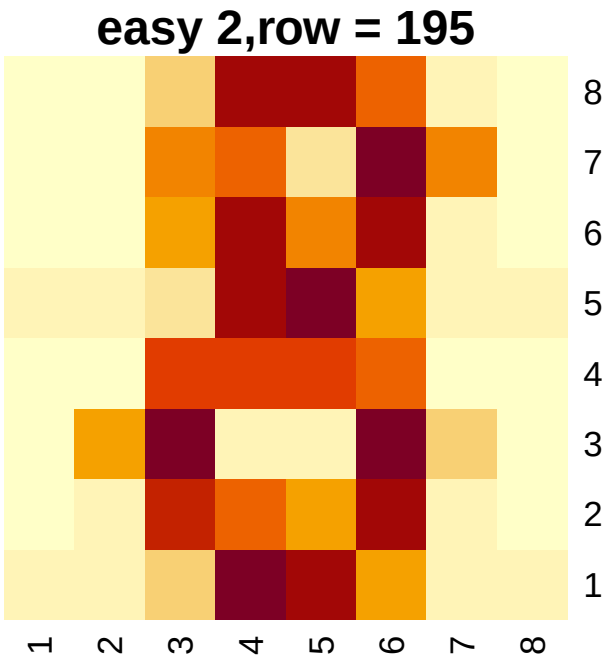
Using ***sort()*** function to find the correct prediction with highest probability and lowest probability. Then reshape them into 8*8 matrix. By viewing these observations in this way, is much easier to recognize.


difficult 1,row = 884
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8


difficult 2,row = 1071
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8


difficult 3,row = 1831
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8


easy 1,row = 129
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8



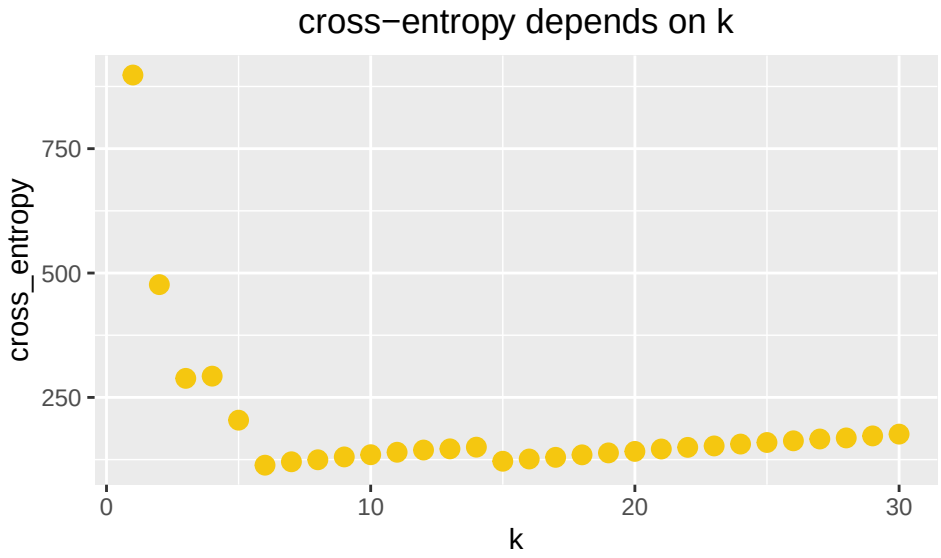
Q4:

Training data into a loop with k increasing from 1 to 30, the complexity grows when k increases. Overall, the error rate keeps increasing when k increases. For the valid data,the optimal k is k == 3, then error rate keeps increasing until k == 25. when k == 3, the error rate for training, valid and test data are around 0.012, 0.025 and 0.024.

###

Q5: Now, the best k is 6. for this problem, cross-entropy not only help us know more details about our

model.
For example, if an observation is '8' with prediction probability 0,0,0,0,0,0,0,1,0 another observation is also '8' with prediction probability 0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.2,0.1, although two observations will be predicted as '8', obviously the result of first observation is better than second one. In other words, the first prediction result is far more confirmed, and the ideal situation is : all the prediction shows similar result to first prediction result. That is why cross-entropy is better in this problem.



Appendix

```
library(ggplot2)
library(kknn)
library(reshape2)
# Assignment 1 KNN-method for hand-writing
################################
# loading data
data <- read.csv('optdigits.csv',header = FALSE)
#############################################################################
# pre-processing (change label into factor and rename columns)
data[,65] <- as.factor(data[,65])
for_rename <- 1:64
for_rename <- as.character(for_rename)
for_rename <- paste0('f',for_rename)
for_rename <- c(for_rename,'target')
colnames(data) <- for_rename
#############################################################################
# Question 1
#############################################################################
set.seed(12345)
n <- dim(data)[1]
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
set.seed(12345)
id1 <- setdiff(1:n, id)
id2 <- sample(id1, floor(n*0.25))
valid <- data[id2,]
set.seed(12345)
id3 <- setdiff(id1,id2)
test <- data[id3,]
```

```
rm(id,id1,id2,id3,n,for_rename)
################################################################################
# Question 2
################################################################################
# fit the model
kn <- train.kknn(target~.,data = train,ks = 30,kernel =  'rectangular')
# evaluate in the train data
pred_train <- predict(kn,train)
t_train <- table(train$target,pred_train)
# evaluate in the test data
pred_test <- predict(kn,test)
t_test <- table(test$target,pred_test)
# calculate the error rate
# for train data
error_rate_train <- 1-sum(diag(t_train))/sum(t_train) # 0.045
# for test data
error_rate_test <- 1-sum(diag(t_test))/sum(t_test) # 0.053

################################################################################
#Question 3
################################################################################
# train the model and fit the model to training data to get the probability.
wri_knn <- kknn(target~.,train = train,test = train,kernel = 'rectangular',k=30)
pre_knn <- predict(wri_knn)
# index_list_of_8 contains all the observations whose prediction and label are both 8.
index_list_of_8 <- which(train[,65]==8)[which(which(train[,65]==8)%in%which(pre_knn==8))]
# these two vector contains the probability of lowest and highest probability
lowest_prob <- sort(wri_knn[['prob']][index_list_of_8,9])[1:3]
highest_prob <- sort(wri_knn[['prob']][index_list_of_8,9],decreasing = TRUE)[1:2]
# find the most 2 difficult observations
lowest_pro_index <- index_list_of_8[which(wri_knn[['prob']][index_list_of_8,9]%in%lowest_prob)][1:3]
# find the 3 easiest observations
highest_pro_index <- index_list_of_8[which(wri_knn[['prob']][index_list_of_8,9]%in%highest_prob)][1:2]
# get the pixel map of good and bad
pixelMapsBad <- train[lowest_pro_index,-65]
pixelMapsGood <- train[highest_pro_index,-65]
# restore the graphic, using heatmap() to view.
figGoodEight1<- matrix(unlist(pixelMapsGood[1,], use.names = FALSE), ncol = 8, nrow = 8, byrow = TRUE)
figGoodEight2<- matrix(unlist(pixelMapsGood[2,], use.names = FALSE), ncol = 8, nrow = 8, byrow = TRUE)
figBadEight1 <- matrix(unlist(pixelMapsBad[1,], use.names = FALSE), ncol = 8, nrow = 8, byrow = TRUE)
figBadEight2 <- matrix(unlist(pixelMapsBad[2,], use.names = FALSE), ncol = 8, nrow = 8, byrow = TRUE)
figBadEight3 <- matrix(unlist(pixelMapsBad[3,], use.names = FALSE), ncol = 8, nrow = 8, byrow = TRUE)
# heatmap(figGoodEight1, Colv = NA, Rowv = NA, main = 'highest probability,row = 129')
# heatmap(figGoodEight2, Colv = NA, Rowv = NA, main = 'highest probability,row = 195')
# heatmap(figBadEight1, Colv = NA, Rowv = NA, main = 'lowest probability,row = 884')
# heatmap(figBadEight2, Colv = NA, Rowv = NA, main = 'lowest probability,row = 1071')
# heatmap(figBadEight3, Colv = NA, Rowv = NA, main = 'lowest probability,row = 1831')
################################################################################
#Question 4
################################################################################
kx <- 1:30
error_train <- c()
error_valid <- c()
```

```
for(i in 1:30)
{
  # model training
  kn <- train.kknn(target~.,data = train,ks = i,kernel = "rectangular")
  # predication
  predic_train <- predict(kn,train)
  predic_valid <- predict(kn,valid)
  # make confusion matrix
  t_train <- table(train$target,predic_train,dnn = c('true','predict'))
  t_valid <- table(valid$target,predic_valid,dnn = c('true','predict'))
  # compute error rate
  error_rate_train <- 1-sum(diag(t_train))/sum(t_train)
  error_rate_valid <- 1-sum(diag(t_valid))/sum(t_valid)
  # add error_rate into a vector for drawing graph
  error_train <- c(error_train,error_rate_train)
  error_valid <- c(error_valid,error_rate_valid)
  # remove redundant data to save memory
  rm(predic_train,predic_valid,error_rate_train,error_rate_valid)
}
# gathering data into a data.frame to apply ggplot2
df <- data.frame(k = 1:30,error_train=error_train,error_valid=error_valid)
# reshape data so that 2 plot lines can be plotted in a single graph
df1 <- melt(df,id.vars='k')
# plot
p1 <-ggplot(df1,aes(x=k,y=value))+
    geom_point(aes(color=variable))+
    ggtitle('error_rate : valid vs train')+
    theme(plot.title = ggplot2::element_text(hjust=0.5))
# from the graph the optimal k is 7
kbest <- train.kknn(target~.,data = train,kernel = 'rectangular',ks=7)
pred_test <- predict(kbest,newdata = test)
t <- table(test$target,pred_test)
error_rate_test <- 1-sum(diag(t))/sum(t)
# remove redundant data
rm(df,df1,pred_test,kbest)
################################################################################
#Question 5
################################################################################
ent_list <- c()
for(i in 1:30)
{
  kn <- kknn(target~.,train = train,test = valid,kernel = 'rectangular',k=i)
  ent <- 0
  for(j in 1:length(valid[,1]))
  {
     col_index <- as.numeric(as.character(valid[j,65]))+1
     ent <- ent - log(as.numeric(as.character((kn[['prob']][j,col_index]))))+1e-15)
  }
  ent_list <- c(ent_list,ent)
}
# make df to draw graph
df <- data.frame(k = 1:30,cross_entropy = ent_list)
p5entro <-ggplot(df,aes(x=k,y=cross_entropy))+
```

```
          geom_point(color=7,size = 3)+
          ggtitle('cross-entropy depends on k')+
          theme(plot.title = ggplot2::element_text(hjust=0.5))
#print(p5entro)
```

Assignment 2 in Machine Learning

Farid Musayev, Kristina Levina, and Wuhao Wang

17 November, 2021

Assignment 2

The data were scaled and divided into training and test data (60/40), as per the instructions. A linear regression model was computed from the training data without using an intercept. The estimated training and test MSEs are

The train MSE is 0.8731931

The test MSE is 0.9294911

In the model summary output, the most contributing parameters are marked by stars. The contribution of the terms is determined using the magnitude of p-values. The lower the p-values, the greater the contribution of the terms. In our case, the most contributing parameters (marked by three stars in the summary output) are Jitter(Abs), Shimmer:APQ5, Shimmer:APQ11, NHR, HNR, DFA, and PPE.

The obtained residual standard error: 0.9366 on 3509 degrees of freedom.

Next, the four function were constructed for optimisation (Loglikelihood, Ridge, RidgeOpt, and DF). These functions were constructed using the formula of loglikelihood, as per Eq. (3.20) from the course book (Andreas Lindholm (2021)):

$$ln(p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta})) = -\frac{n}{2}ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{\theta}^T\mathbf{x} - y_i)^2 \tag{1}$$

By using function RidgeOpt, optimal $\boldsymbol{\theta}$ parameters were calculated. The obtained parameters were then used to predict the motor_UPDRS values (target variable) for training and test data. Using different regularisation factors λ, the following results were obtained:

λ	MSE_test	MSE_tr	DF	σ
1	0.929	0.873	13.863	0.934
100	0.926	0.879	9.939	0.938
1000	0.948	0.916	5.643	0.957

As one can observe, the test value of the MSE is lowest for $\lambda = 100$. We can conclude that too large penalization does not contribute to better model. Optimum $\lambda = 100$. Regarding the degrees of freedom (Trevor Hastie (2017)), the larger the λ, the smaller the values of the diagonal elements of the hat matrix. The more degrees of freedom the model has, the better estimators should be obtained. However, we observe that for $\lambda = 1$, although the degrees of freedom are highest, the model performance is lower than when $\lambda = 100$. We can conclude that these two dependences (increase in degrees in freedom => better model and insufficient penalization => worse model) are balancing each other.

Appendix

```
################################################################################
# 1
################################################################################
data <- read.csv2("parkinsons.csv")

n <- dim(data)[1]

# Compute the number of columns
n_col <- length(unlist(strsplit(data[5,], ","))) #5 is just a random existing row

col_names <- strsplit(paste0("subject#,age,sex,test_time,motor_UPDRS,total_UPDRS,",
                      "Jitter(%),Jitter(Abs),Jitter:RAP,Jitter:PPQ5,Jitter:DDP,",
                      "Shimmer,Shimmer(dB),Shimmer:APQ3,Shimmer:APQ5,",
                      "Shimmer:APQ11,Shimmer:DDA,NHR,HNR,RPDE,DFA,PPE"), ",")

data <- lapply(data, function(i){
  i <- as.numeric(unlist(strsplit(i, ",")))
})

# Turn the obtained list into the matrix and to the data.frame.
data <- as.data.frame(matrix(unlist(data), nrow = n, ncol = n_col, byrow = TRUE))

names(data) <- unlist(col_names)

# Scale the data and use only the variables of interest in the upcoming analysis.
data <- scale(data[,c(5,7:22)])
n <- dim(data)[1]
data <- as.data.frame(data)

# Divide the data, as per the instructions
set.seed(12345)
id = sample(1:n, floor(n * 0.6))
train = data[id, ]
#print(dim(train)) #3525 entries

test = data[-id, ]

################################################################################
# 2
################################################################################

model <- lm(formula = motor_UPDRS ~ . - 1, data = train)

print(summary(model))
```

```
##
## Call:
## lm(formula = motor_UPDRS ~ . - 1, data = train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.0119 -0.7270 -0.1018  0.7384  2.1959
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## `Jitter(%)`       0.181065   0.144249   1.255 0.209481
## `Jitter(Abs)`    -0.169830   0.040851  -4.157 3.30e-05 ***
## `Jitter:RAP`     -5.098809  18.184783  -0.280 0.779196
## `Jitter:PPQ5`    -0.071777   0.084701  -0.847 0.396816
## `Jitter:DDP`      5.079056  18.188164   0.279 0.780069
## Shimmer           0.590992   0.205286   2.879 0.004015 **
## `Shimmer(dB)`    -0.172860   0.139380  -1.240 0.214983
## `Shimmer:APQ3`   32.213852  77.012847   0.418 0.675759
## `Shimmer:APQ5`   -0.386846   0.113713  -3.402 0.000677 ***
## `Shimmer:APQ11`   0.310256   0.062270   4.982 6.58e-07 ***
## `Shimmer:DDA`   -32.529915  77.012630  -0.422 0.672761
## NHR              -0.186755   0.045741  -4.083 4.55e-05 ***
## HNR              -0.239777   0.036565  -6.558 6.27e-11 ***
## RPDE              0.003958   0.022611   0.175 0.861052
## DFA              -0.277038   0.019888 -13.930  < 2e-16 ***
## PPE               0.229006   0.033264   6.885 6.84e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9366 on 3509 degrees of freedom
## Multiple R-squared:  0.1212, Adjusted R-squared:  0.1172
## F-statistic: 30.25 on 16 and 3509 DF,  p-value: < 2.2e-16
```

```
pred_tr <- predict(object = model)
pred_test <- predict(object = model, newdata = test)

MSE_tr <- mean((train$motor_UPDRS - pred_tr)^2)
MSE_test <- mean((test$motor_UPDRS - pred_test)^2)

cat("The train MSE is ", MSE_tr, sep = "", "\n")
```

```
## The train MSE is 0.8731931
```

```
cat("The test MSE is ", MSE_test, sep = "", "\n")
```

```
## The test MSE is 0.9294911
```

```
################################################################################
# 3
################################################################################

# a

Loglikelihood <- function(par){ #par[1] is sigma and par[2:17] is theta vector
  X <- t(as.matrix(train[,-1]))
  n <- dim(train)[1]
  theta <- t(par[2:17])
```

```r
  result <- -n*log(2 * pi * par[1]^2)/2 -
    sum((theta %*% X - train$motor_UPDRS)^2)/(2 * par[1]^2)
  return(result)
}

Ridge <- function(par, lambda){
  result <- -Loglikelihood(par) + lambda * sum(par[2:17]^2)
  return(result)
}

RidgeOpt <- function(lambda){
  initial_sigma <- 0.9366
  initial_theta <- coef(model)
  result <- optim(par = c(initial_sigma, initial_theta), fn = Ridge,
                  lambda = lambda, method = "BFGS")
  return(result$par)
}

DF <- function(lambda){
  X <- as.matrix(train[,-1])
  # trace of the hat matrix
  result <- sum(diag(X %*% solve(t(X) %*% X + lambda * diag(16)) %*% t(X)))
  return(result)
}

##############################################################################
# 4
##############################################################################

lambda <- c(1, 100, 1000)
MSE_tr <- c()
MSE_test <- c()
DoF <- c()
sigma <- c()
for (l in lambda) {
  theta_lambda <- RidgeOpt(l)[2:17]
  X_tr <- t(as.matrix(train[,-1]))
  X_test <- t(as.matrix(test[,-1]))
  theta <- t(theta_lambda)
  pred_tr <- theta %*% X_tr
  pred_test <- theta %*% X_test
  MSE_tr <- c(MSE_tr, mean((train$motor_UPDRS - pred_tr)^2))
  MSE_test <- c(MSE_test, mean((test$motor_UPDRS - pred_test)^2))
  DoF <- c(DoF, DF(l))
  sigma <- c(sigma, RidgeOpt(l)[1])
}
df <- cbind.data.frame(lambda = lambda, MSE_test = MSE_test, MSE_tr = MSE_tr,
                       DoF = DoF, sigma = sigma)
print(round(df, digits = 3))
```

```
##   lambda MSE_test MSE_tr    DoF sigma
## 1      1    0.929  0.873 13.863 0.934
## 2    100    0.926  0.879  9.939 0.938
```

```
## 3   1000    0.948  0.916  5.643 0.957
```

Resources

Andreas Lindholm, et al. 2021. *Machine Learning. A First Course for Engineers and Scientists.* This material will be published by Cambridge University Press. This pre-publication version is free to view; download for personal use only.

Trevor Hastie, Jerome Friedma, Robert Tibshirani. 2017. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction.* Springer Series in Statistics.

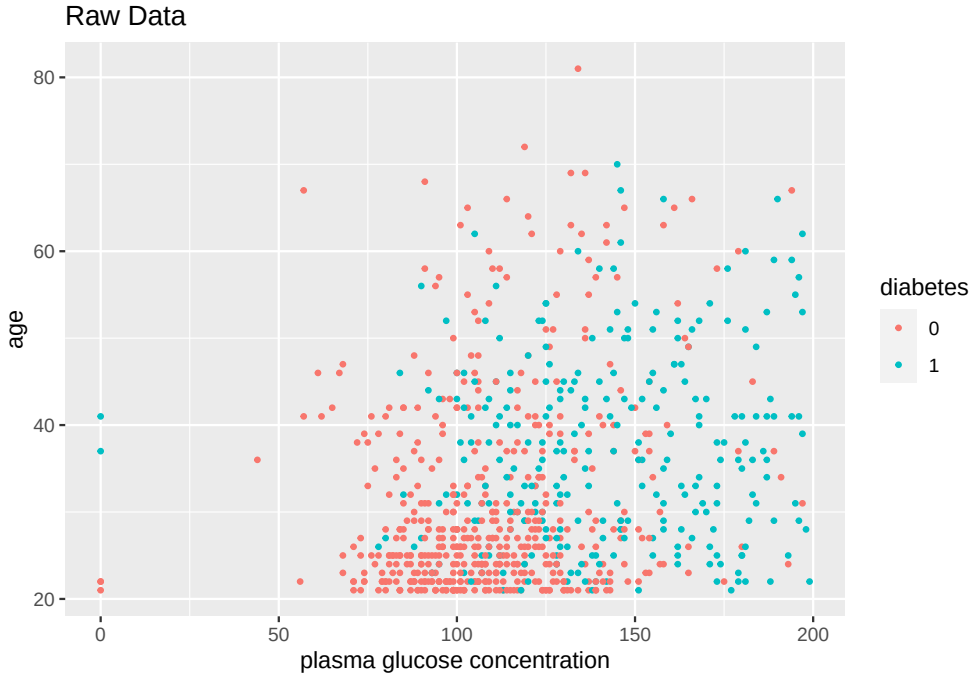
732A99 Machine Learning | Assignment 3

Group 8

18 November, 2021

Q1

Although elements of clustering exist in the bottom middle part of the plot, it is hard to draw a distinct decision boundary. The problem is that this classification problem is both *imbalanced* and *asymmetric*. *Imbalance* is caused by unequal distribution of *positive* and *negative* responses in the data frame. From the responses, it can be calculated that 65% of results from diabetes has "YES" (or 1 for positive) status while only 35% has "NO" (or 0 negative) status. In addition, this is a medical diagnosis classification problem meaning that *false negatives* are considered to be more severe than *false positives*. This makes the problem *asymetric*. Results of applied logistic regression must be analyzed from confusion matrix, recall, precision and F-scores.



Q2

Estimated parameters from the model are provided below:
Intercept, θ_0 = -5.912
Plasma Glucose Concentration, θ_1 = 0.036
Age, θ_2 = 0.025

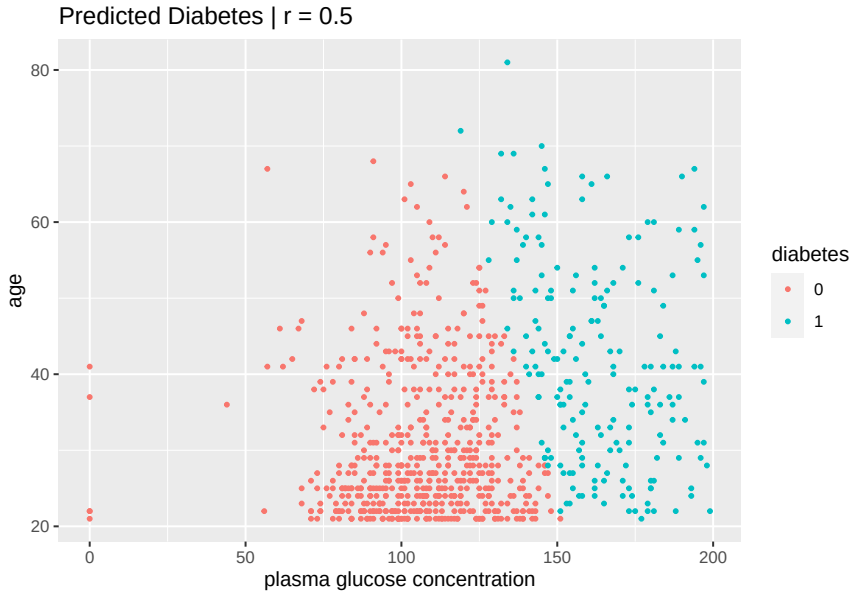
Probabilistic equation for **logistic regression** is:

$$p(y = m|x, \theta) = \frac{1}{1 + exp(-(\theta_0 + x_1\theta_1 + x_2\theta_2))}$$

p(y) - probability of y = m
m - [0,1] where 0 stands for *"no"* and 1 stands for *"yes"* status of diabete
x_1 - Plasma Glucose Concentration
x_2 - Age
θ_i - estimated parameter for $i = 0, 1, 2$

Substituting:

$$p(y) = \frac{1}{1 + exp(-(-5.912 + 0.036x_1 + 0.025x_2))}$$



Training misclassification rate = 0.263
This is a high value meaning that more than a quarter of the data has been misclassified. However, for both *imbalanced* and *asymetric* scenarios *misclassification error* is a poor indicator of the model performance. Thus, to make a better assessment of the model, it is important to calculate and analyze *confusion matrix, recall, precision and F-score* parameters for $r = 0.5$:

```
## Confusion Matrix | r = 0.5
```

```
##           real
## predicted   0   1
##         0 436 138
##         1  64 130
```

```
## misclassification_rate =  0.2630208
```

```
## F_score =  0.5134281
```

```
## recall =  0.4850746
```

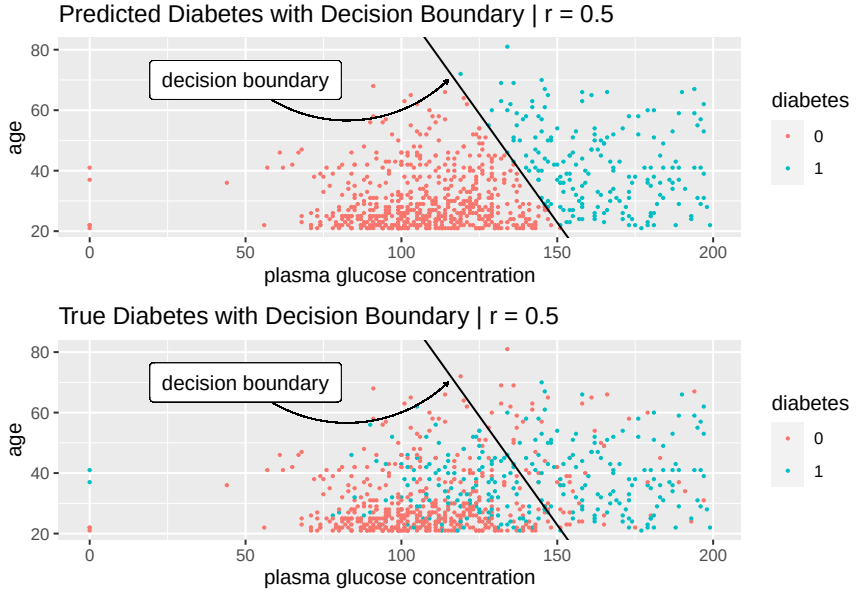
```
## precision =  0.6701031
```

Confusion matrix is a very clear example that model performs poorly on *false negatives*, in other words, only (roughly speaking) half of the people with "YES" diabete status have been successfully predicted by the model. Since the model is *asymetric*, *false negatives* deserve more importance, thus, to estimate F-score, $beta = 2$ is considered. Nevertheless, recall and F-score values are still around 0.5. It makes the model with $r = 0.5$ threshold unreliable in this situation.

Q3

To derive equation of decision boundary, consider:
$x_2 = x_1 k + b$, $k = slope$ and $b = intercept$
$p(y) = 0.5$ when $r = 0.5$
As mentioned earlier, for the logistic regression:
$p(y) = \frac{1}{1+exp(-(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2))} = 0.5$
$exp(-(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2)) = 1$ and $\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 = 0$
For *intercept* b, substitute $x_2 = x_1 k + b$, $x_1 = 0$, $x_0 = 1$ into above equation to get:
$\theta_0 + \theta_2 b = 0$

Since this is an equation of line, *intercept* $b = -\frac{\theta_0}{\theta_2}$ and *slope* $k = -\frac{\theta_1}{\theta_2}$. Note that, this is only valid for two-dimensional problem (two features x_1 and x_2). As it will be seen later, the line is no more linear when more than two features are used for the logistic regression.



The decision boundary line separates data quite good. However, it behaves good for *predicted responses* not *real responses* ! As it can be seen above, for *real responses* decision boundary performs poorly.

Q4.

For $r = 0.2$ and $r = 0.8$, b *intercept* in the equation of decision boundary will change. However, k *slope* value stays the same since the problem is still two-dimensional (only two features are included into the model) and line remains linear. The slope could have changed, if the model has been run with other two features since parameters would be different in that case.
For $r = 0.2$, $b = \frac{-ln(4)-\theta_0}{\theta_2}$ and for $r = 0.8$, $b = \frac{ln4-\theta_0}{\theta_2}$

Below are a set of parameters to analyze the model performance:

```
##
##
## Confusion Matrix | r = 0.2

##           real
## predicted   0   1
##         0 238  24
##         1 262 244

## misclassification_rate =  0.3723958

## F_score =  0.7731305

## recall =  0.9104478

## precision =  0.4822134

##
##
## Confusion Matrix | r = 0.8

##           real
## predicted   0   1
##         0 490 232
##         1  10  36

## misclassification_rate =  0.3151042

## F_score =  0.1610018
```

```
## recall =  0.1343284
```

```
## precision =  0.7826087
```

It is obvious that $r = 0.8$ is the worst case scenario where the model has failed to correctly predict 87% of patients with "YES" diabete status. This is also supported by a very low F-score. It should be mentioned that in scenarios where *false negatives* are prioritized over *false positives*, recall is given a greater priority than precision. For this case, recall also has a very low value (must be close to 1).
For $r = 0.2$, situation is much better for *false negatives*, in other words, 91% of patients with "YES" diabete status has been predicted correctly. Model also has a relatively high F-score and recall. The problem with this scenario is that the model performs poorly for patients with "NO" diabete status or *true negatives*. Here, looking at the confusion matrix, it can be said that roughly half of the patients with "NO" diabete status has been predicted incorrectly. As it has been mentioned earlier, since this is a medical experiment involving patient diagnosis, one must aim for minimizing amount of *false negatives*. Thus, the model with $r = 0.2$ can be considered viable. Despite having a higher misclassification rate, the model with threshold $r = 0.2$ performs better that the model with $r = 0.5$.

Q5.

After application of basis function expansion, **training misclassification error** decreases to 0.245. As in the previous scenarios, a set of the model performance parameters must be analyzed:

```
##
##
## Confusion Matrix for BFE | r = 0.5
```

```
##           real
## predicted   0   1
##         0 433 121
##         1  67 147
```

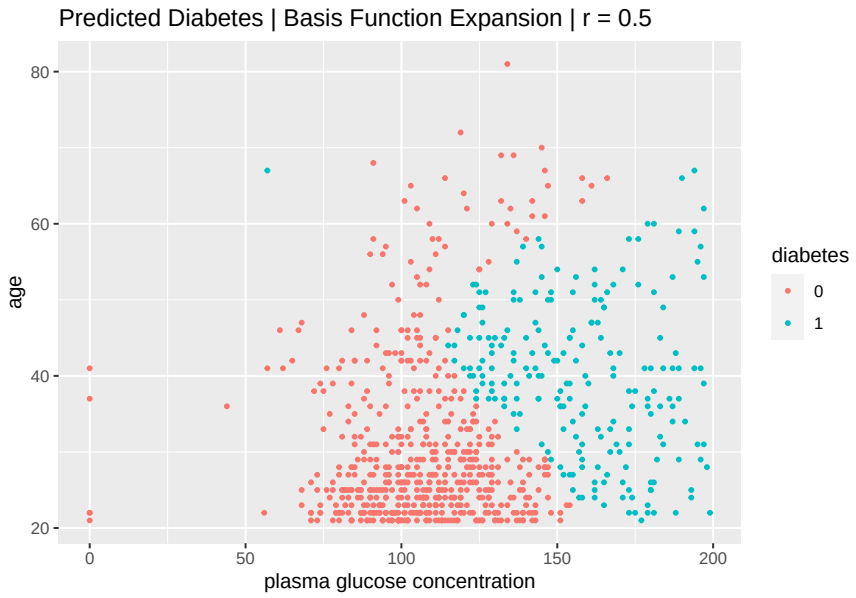
```
## misclassification_rate =  0.2447917
```

```
## F_score =  0.5715397
```

```
## recall =  0.5485075
```

```
## precision =  0.6869159
```

Here, threshold is the same $r = 0.5$, however, the number of features has been increased to 7. The model performance is very similar to the one with only two features (discussed in Q2). Slight improvements in F-score, recall and, as a result, in decreasing number of *false negatives* can be seen. However, the results are hardly feasible for the medical diagnosis experiment.



The model is no longer two-dimensional and, thus, the decision boundary is hard to define from two features. However, since x_1 and x_2 still have a larger impact in comparison with other features (much lower p-values), the decision boundary can be approximated by plotting only these two features as in the previous scenarios. If drawn, a decision boundary will have something close to the shape of an inverted hyperbola.

To sum up the analysis of all previous models, the model with $r = 0.2$ threshold appears to be the most appropriate for the given experiment. The important point is that only training data has been used for the analysis of the logistic regression model. To generalize beyond the training data, the model has to demonstrate decent performance on the test data.

Appendix

```
# Assignment 3

 library(ggplot2)
 library(gridExtra)

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

colnames(df)<-c("not_pregnant", "plasma_glucose_concentration",
                "diastolic_blood_pressure", "triceps_skinfold_thickness",
                "2h_serum_insulin", "body_mass_index",
                "diabetes_pedigree_function", "age", "diabetes")

# 1._____________________________________

df$diabetes<-as.factor(df$diabetes)

pl<-ggplot(df)+
  geom_point(aes(x=plasma_glucose_concentration, y=age, color=diabetes),size=0.8)+
  ggtitle("Raw Data")+xlab("plasma glucose concentration")+ylab("age")
```

```r
#plot(pl)

# 2._____________________________________

# run logistic regression model

logreg<-glm(formula = diabetes ~ plasma_glucose_concentration + age,
                    data = df,
                    family = "binomial")

# use your model to predict responses for the whole data frame

predicted_diabetes<-predict(logreg, newdata=df, type='response')

r<-0.5   # define your threshold to classify probabilities

r05_predicted_diabetes<-as.factor(ifelse(predicted_diabetes >=r, 1, 0))

# calculate misclassification error

misclass_error<-mean(r05_predicted_diabetes!=df$diabetes)

cat("Misclassification error = ", misclass_error,"\n")
```

```
## Misclassification error =  0.2630208
```

```r
# attach your predicted values to original data frame

new_df<-cbind(df, r05_predicted_diabetes)

# 3._____________________________________________

tetta = coefficients(logreg)
intercept = -tetta[1]/tetta[3]
slope = -tetta[2]/tetta[3]

pl1<-ggplot(data=new_df, aes(x=plasma_glucose_concentration, y=age, color = r05_predicted_diabetes))+
  geom_point(size=0.4)+
  geom_abline(aes(slope = slope, intercept=intercept), color = 'black')+
  geom_curve(x = 50, y = 70, xend =115 , yend =70,
             arrow = arrow(length = unit(0.02, "npc"), type='closed'),
             size=0.3,
             curvature = 0.4,
             color='black',
             ncp = 40)+
  geom_label(label="decision boundary",
             label.size = 0.1,
             label.padding = unit(0.45, "lines"),
             x=50,
             y=70,
             color='black')+
  ggtitle("Predicted Diabetes with Decision Boundary | r = 0.5 ")+
  labs(color = "diabetes")+xlab("plasma glucose concentration")
```

```r
#plot(pl1)

# 4.______________________________________________

# r = 0.2

r=0.2

r02_predicted_diabetes<-as.factor(ifelse(predicted_diabetes > r, 1, 0))

new_df<-cbind(new_df, r02_predicted_diabetes)

tetta = coefficients(logreg)
intercept_02 = (-log(4)-tetta[1])/tetta[3]
slope = -tetta[2]/tetta[3]

pl2<-ggplot(data=new_df, aes(x=plasma_glucose_concentration, y=age, color = r02_predicted_diabetes))+
  geom_point(size=0.4)+
  geom_abline(aes(slope = slope, intercept=intercept_02), color = 'black')+
  geom_curve(x = 30, y = 70, xend =80 , yend =60,
             arrow = arrow(length = unit(0.02, "npc"),type='closed'),
             size=0.3,
             curvature = 0.4,
             color='black',
             ncp=40)+
  geom_label(label="decision boundary",
             label.size = 0.1,
             label.padding = unit(0.45, "lines"),
             x=30,
             y=70,
             color='black')+
  ggtitle("Predicted Diabetes with Decision Boundary", subtitle = "r = 0.2")+
  labs(color = "diabetes")+xlab("plasma glucose concentration")


#plot(pl2)

# r = 0.8

r=0.8

r08_predicted_diabetes<-as.factor(ifelse(predicted_diabetes > r, 1, 0))

new_df<-cbind(new_df, r08_predicted_diabetes)

tetta = coefficients(logreg)
intercept_08 = (log(4)-tetta[1])/tetta[3]
slope = -tetta[2]/tetta[3]

pl3<-ggplot(data=new_df, aes(x=plasma_glucose_concentration, y=age, color = r08_predicted_diabetes))+
  geom_point(size=0.4)+
  geom_abline(aes(slope = slope, intercept=intercept_08), color = 'black')+
  geom_curve(x = 80, y = 70, xend =150 , yend =70,
```

```
              arrow = arrow(length = unit(0.02, "npc"),type='closed'),
              size=0.3,
              curvature = 0.4,
              color='black',
              ncp=40)+
  geom_label(label="decision boundary",
              label.size = 0.1,
              label.padding = unit(0.45, "lines"),
              x=80,
              y=70,
              color='black')+
  ggtitle(label=NULL, subtitle = "r = 0.8")+
  labs(color = "diabetes")

#plot(pl3)

# 5._____________________________________________

# Basis function expansion

z1<-(df$plasma_glucose_concentration)^4
z2<-(df$plasma_glucose_concentration)^3*(df$age)
z3<-(df$plasma_glucose_concentration)^2*(df$age)^2
z4<-(df$plasma_glucose_concentration)*(df$age)^3
z5<-(df$age)^4

z_df<-cbind(df,z1,z2,z3,z4,z5)

z_logreg<-glm(formula = diabetes ~
              plasma_glucose_concentration + age + z1 + z2 + z3 + z4 + z5,
              data = z_df,
              family = "binomial")

z_predicted_diabetes<-predict(z_logreg, newdata=z_df, type='response')

r=0.5

z_predicted_diabetes<-ifelse(z_predicted_diabetes > r, 1, 0)

z_predicted_diabetes<-as.factor(z_predicted_diabetes)
z_misclass_error<-mean(z_predicted_diabetes!=z_df$diabetes)

cat("BFE Misclassification error = ", z_misclass_error)
```

```
## BFE Misclassification error =  0.2447917
```

```
new_z_df<-cbind(z_df, z_predicted_diabetes)

pl4<-ggplot(new_z_df)+
  geom_point(aes(x=plasma_glucose_concentration, y=age, color=z_predicted_diabetes),size=0.8)+
  ggtitle("Predicted Diabetes | Basis Function Expansion | r = 0.5")+
  labs(color = "diabetes")+xlab("plasma glucose concentration")
```

```
#plot(pl4)

#grid.arrange(grobs=list(pl, pl1, pl2, pl3, pl4))


# Final Model Assessment_____________________________________

# 0 -  "NO"  | status of diabete
# 1 - "YES" | status of diabete

beta=2

# confusion matrix, recall, precision and F-score for r = 0.5

c1<-table(new_df$r05_predicted_diabetes, new_df$diabetes,dnn=c("predicted", "real"))
misclass_rate<-mean(new_df$r05_predicted_diabetes!=df$diabetes)
recall=c1[4]/(c1[4]+c1[3])
precision=c1[4]/(c1[4]+c1[2])
f_score=(1+beta^2)*precision*recall/(beta^2*precision+recall)

# confusion matrix, recall, precision and F-score for r = 0.2

c2<-table(new_df$r02_predicted_diabetes, new_df$diabetes, dnn=c("predicted", "real"))
misclass_rate<-mean(new_df$r02_predicted_diabetes!=df$diabetes)
recall=c2[4]/(c2[4]+c2[3])
precision=c2[4]/(c2[4]+c2[2])
f_score=(1+beta^2)*precision*recall/(beta^2*precision+recall)


# confusion matrix, recall, precision and F-score for r = 0.8

c3<-table(new_df$r08_predicted_diabetes, new_df$diabetes, dnn=c("predicted", "real"))
misclass_rate<-mean(new_df$r08_predicted_diabetes!=df$diabetes)
recall=c3[4]/(c3[4]+c3[3])
precision=c3[4]/(c3[4]+c3[2])
f_score=(1+beta^2)*precision*recall/(beta^2*precision+recall)


# confusion matrix, recall, precision and F-score for BFE | r = 0.5

c4<-table(new_z_df$z_predicted_diabetes, new_z_df$diabetes, dnn=c("predicted", "real"))
misclass_rate<-mean(new_z_df$z_predicted_diabetes!=new_z_df$diabetes)
recall=c4[4]/(c4[4]+c4[3])
precision=c4[4]/(c4[4]+c4[2])
f_score=(1+beta^2)*precision*recall/(beta^2*precision+recall)
```