

# Machine Learning Lab1

Lepeng Zhang, Xuan Wang, Priyarani Patil

2023-11-17

The group report was made based on the discussion after all of us had finished all three assignments.

Assignment 1 was mainly contributed by Lepeng Zhang.

Assignment 2 was mainly contributed by Xuan Wang.

Assignment 3 was mainly contributed by Priyarani Patil.

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

### Q1

See appendix.

### Q2

## Confusion matrix for the training data:

```
##          train_predict
## train_true  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
```

## Confusion matrix for the test data:

```
##          test_predict
## test_true  0  1  2  3  4  5  6  7  8  9
##          0 82  0  0  0  1  0  1  0  0  0
##          1  0 90  2  0  0  0  0  0  0  3
##          2  0  1 92  0  0  0  0  1  1  1
##          3  0  0  0 85  0  2  0  3  1  1
##          4  0  1  0  0 89  0  1  6  0  5
##          5  0  1  0  1  0 97  1  1  0  7
##          6  0  0  0  0  0  0 97  0  0  0
##          7  0  1  0  1  0  0  0 99  0  0
##          8  0  7  0  0  0  0  0  0 84  0
```

```

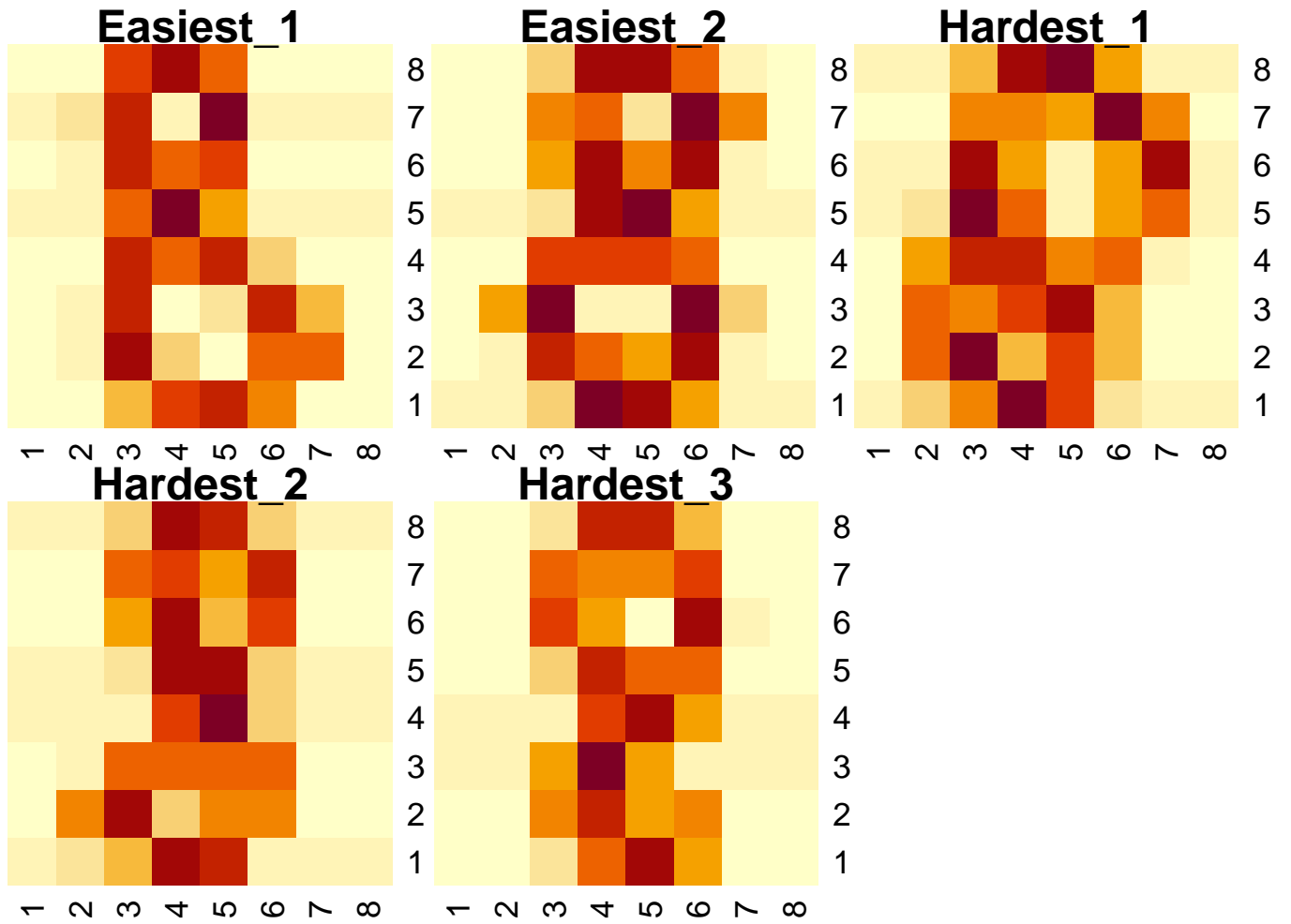
##          9  0  2  0  0  0  0  0  2  1 86
## Misclassification error for the training data: 0.045
## Misclassification error for the test data: 0.05852
## Misclassification error for each digit in the test data:
##      digit error_rate
## 5         4 0.12745098
## 6         5 0.10185185
## 9         8 0.07692308
## 4         3 0.07608696
## 10        9 0.05494505
## 2         1 0.05263158
## 3         2 0.04166667
## 1         0 0.02380952
## 8         7 0.01980198
## 7         6 0.00000000

```

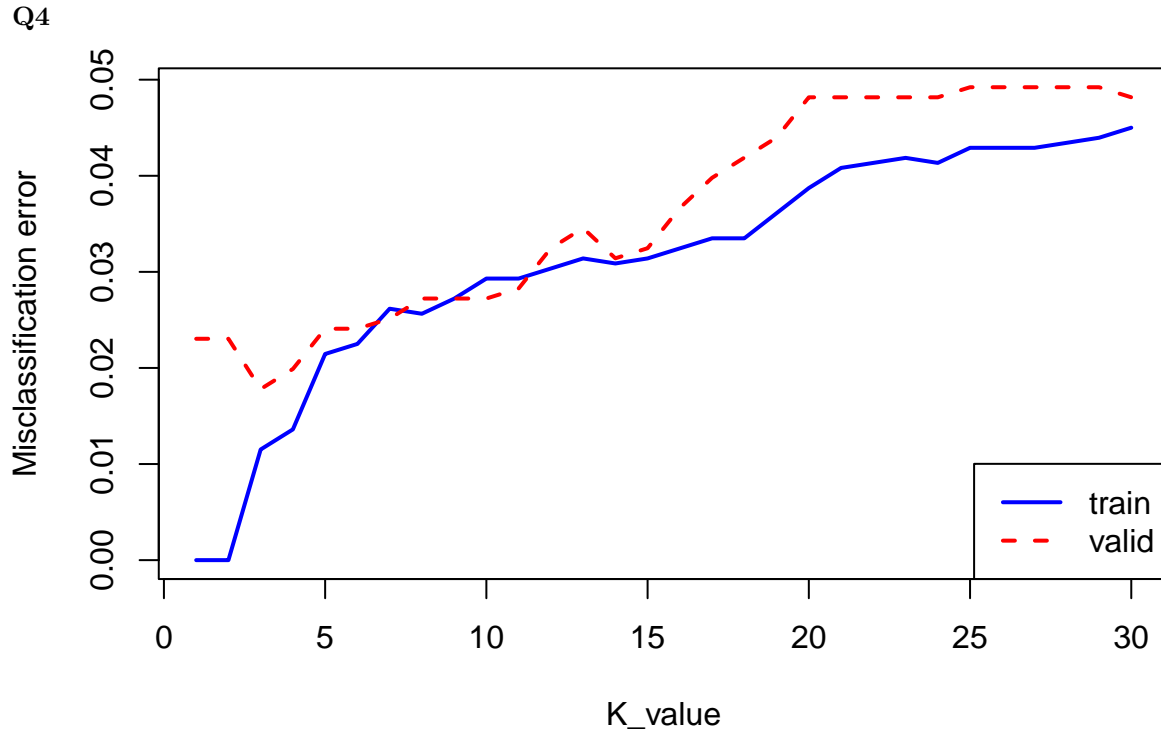
The model predicts 4 worst with 12.7 % error rate and predicts 6 best with 100 % accuracy. Both error rates for 4 and 5 are over 10 %, and error rates for 8, 3, 9, 1 are over 5 %. While, error rates for 2, 0, 7, 6 are below 5 %. Note that these error rates are computed based on the predictions on the test data.

Besides, misclassification error for the test data is 5.852 %. It is not bad but still has improvement room.

Q3



The two cases in the easiest group are easy to recognize as 8 visually. While, for the first two cases in the hardest group, they can recognize 8 roughly. But for the third one, it is quite hard to tell what digit it is.



When  $K$  increases, the model becomes less complex, as the predictions are based on a majority vote from more neighbors.

In general, the training error increases with the increase of  $K$  after  $K = 2$ , while the rate of increase gradually decreases. For validation error, it decreases at first and reaches the minimum at  $K = 3$  and then increases with the increase of  $K$ . These relationships indirectly indicate that larger  $K$  leads to simpler model.

According to this plot, the optimal  $K = 3$ .

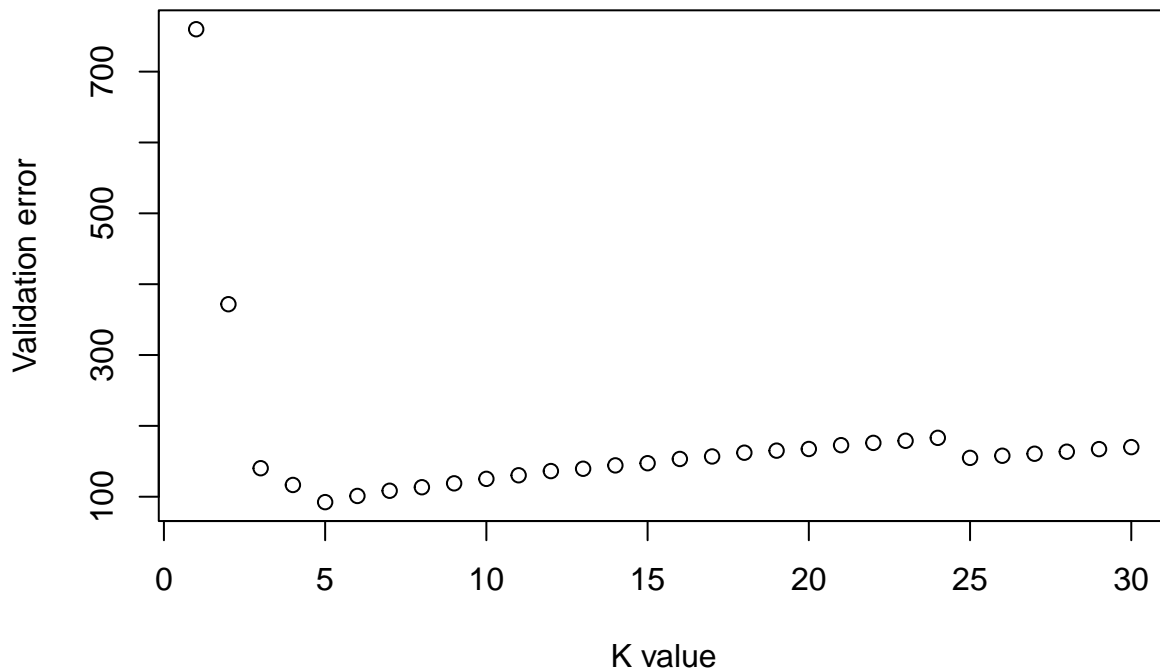
```
## Misclassification error for the test data: 0.0313479623824452
```

```
## K train_error valid_error test_error
```

```
## 3 3 0.0115123 0.01780105 0.03134796
```

It can be computed that `test_error` is 1.76 times of `valid_error` and 2.72 times of `train_error`. In other words, `test_error` is quite larger than the other two errors. Nonetheless, the absolute value of `test_error` is small. Therefore, this model has a good prediction quality.

Q5



The optimal  $K = 5$  since validation error reaches minimum here.

In a multinomial classification task, misclassification error cannot reflect the extent of mistakes. Specifically, when an observation is misclassified, the number of misclassified cases, which is used to compute the misclassification error, will always add 1 no matter how large the probability that the model calculates about this observation is.

However, cross-entropy can reflect the extent of mistakes by function  $-\log(\text{probability})$ . For example, two observations are misclassified with probabilities of 0.1 and 0.2, respectively. Their contributions to the whole cross-entropy are different. And this difference provides a better capability of measuring the quality of the model.

## Assignment 2. Linear regression and ridge regression

Q1

See appendix.

```
## Loading required package: ggplot2
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:knn':
##
##   contr.dummy
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##   filter, lag
```

```
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union
```

## Q2

```
## Training MSE is: 0.878543102826276.
## Test MSE is: 0.935447712156708.
##
## Call:
## lm(formula = motor_UPDRS ~ ., data = train_scaled)
##
## 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
```

The significant contributors are Jitter.Abs., Shimmer.APQ5, Shimmer.APQ11, NHR, HNR, DFA, and PPE, which are marked \*\*\* in summary output.

## Q3

See appendix.

## Q4

```
##      lambda train_MSE test_MSE      DoF      sigma
## 1         1 0.8786272 0.9350013 13.860736 0.9373538
## 2        100 0.8844134 0.9323346  9.924887 0.9404627
```

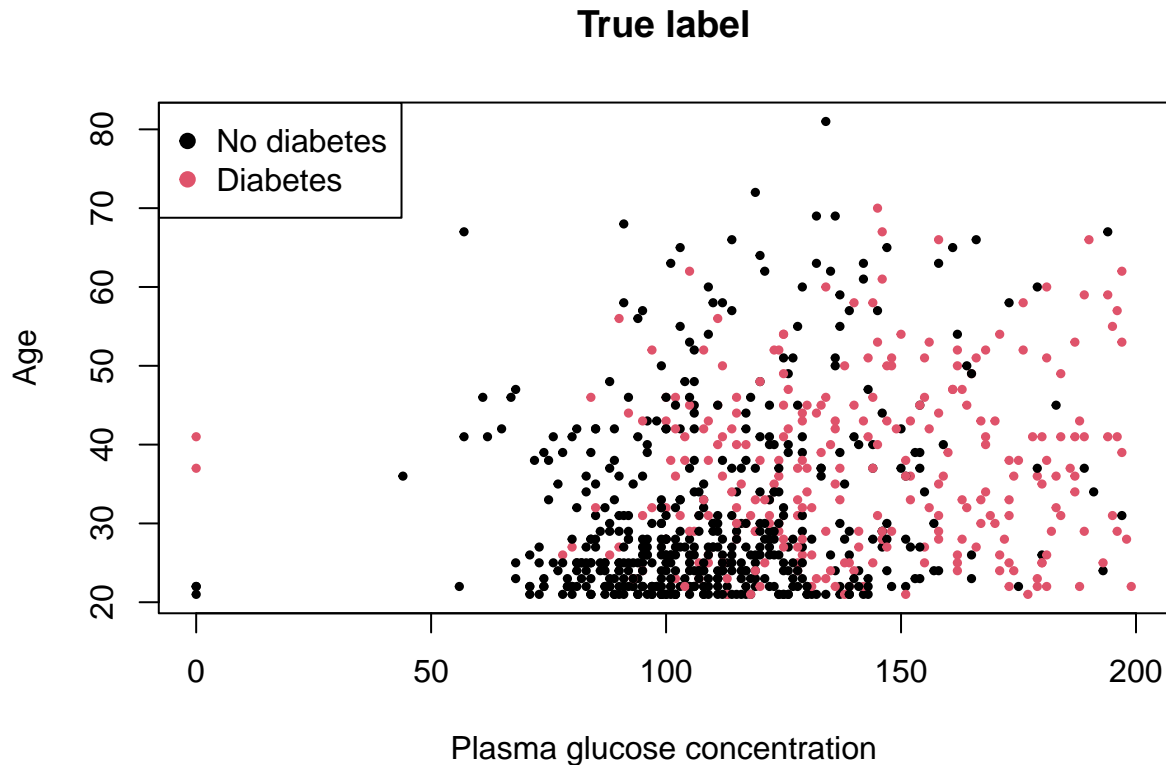
```
## 3    1000 0.9211180 0.9539465  5.643925 0.9597496
```

As the table shown, train\_MSE increases slightly as  $\lambda$  increases, while test\_MSE reaches minimum at  $\lambda = 100$ , which is regarded as the optimal penalty parameter among these three.

Larger  $\lambda$  results to smaller DoF. As  $\lambda$  increases, the parameters are heavily constrained and the degrees of freedom will effectively be lower, tending to 0 as  $\lambda \rightarrow \infty$ . In summary,  $\lambda$  controls the complexity of the model, when it increases (DoF decreases), model becomes simpler; when it decreases (DoF increases), model becomes more complex. Since there is a trade-off of model complexity for the best model, there exists optimal values of  $\lambda$  and DoF.

### Assignment 3. Logistic regression and basis function expansion

Q1



The scatterplot shows that it is quite hard to form a convincing decision boundary. In the black point-clustering area (considering no diabetes), there also exists many red points (with diabetes). This will result in *false negative*, which should avoid as much as possible in this diagnosis problem.

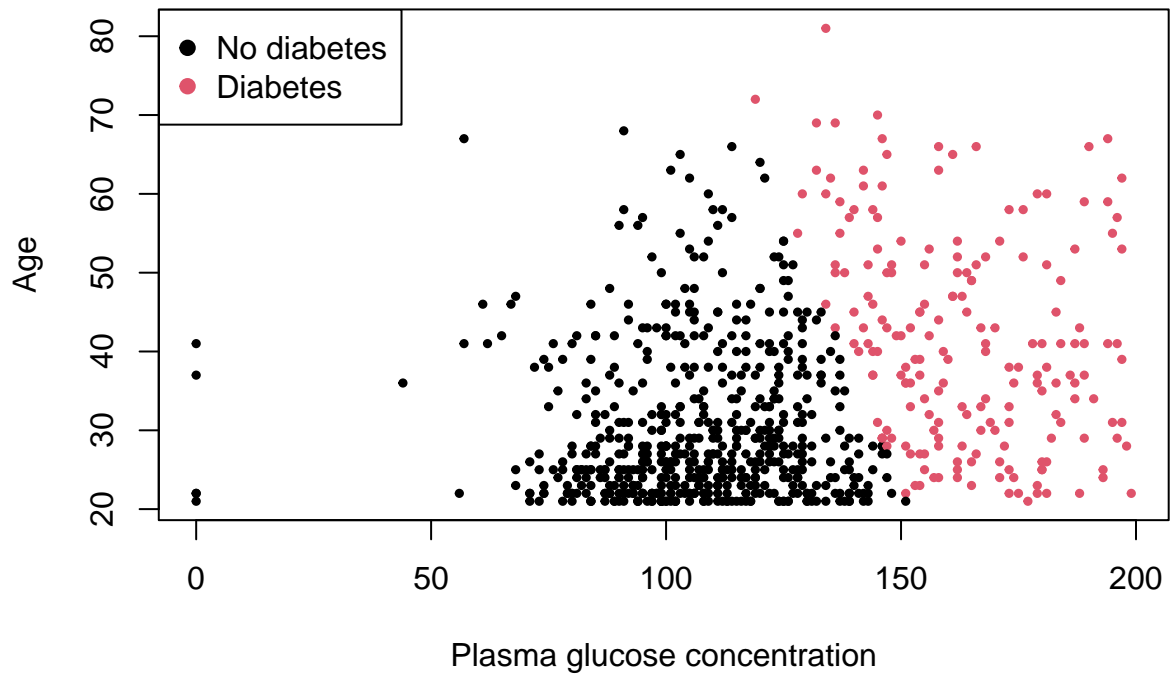
Q2

```
## (Intercept)          V2          V8
## -5.91244906  0.03564404  0.02477835

##      pred
## true  0   1
##      0 436  64
##      1 138 130

## The training misclassification error is: 0.2630208
```

### Predict label (r = 0.5)



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

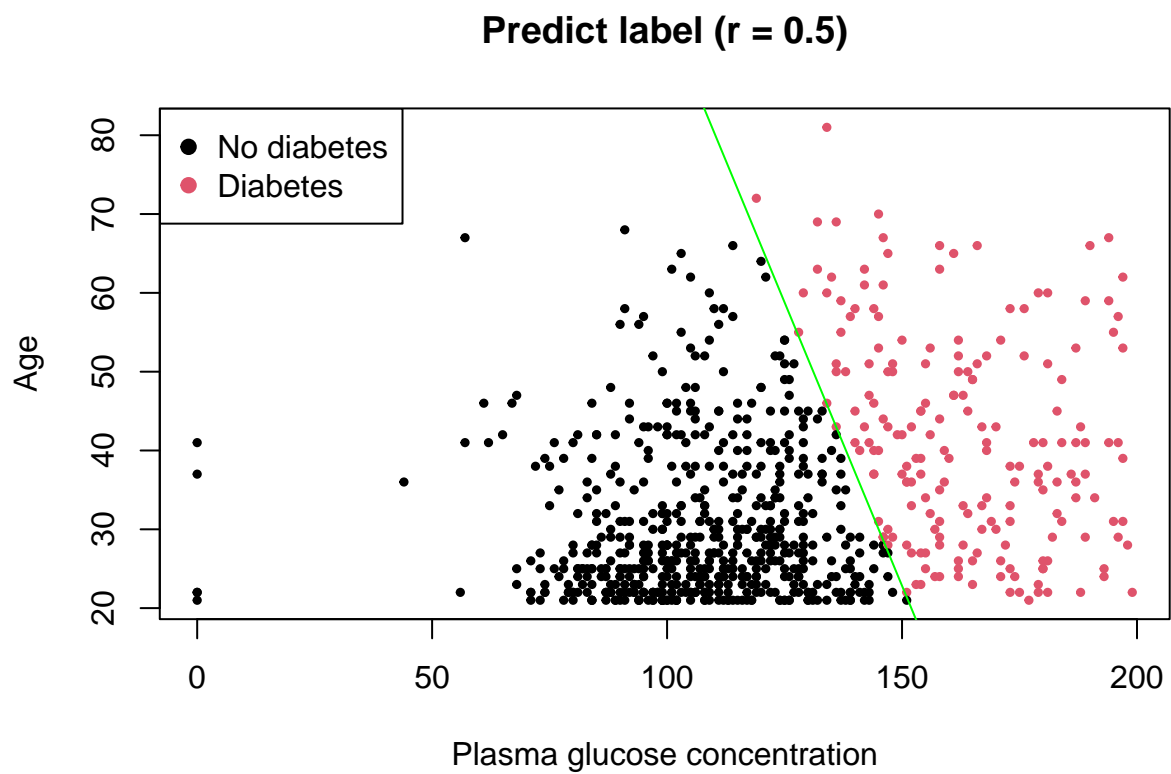
The training misclassification error means that more than a quarter of cases are misclassified. And the confusion table shows that more than half of the people with diabetes are not diagnosed.

### Q3

Set the above probability equation equals 0.5 to get the equation of the decision boundary:

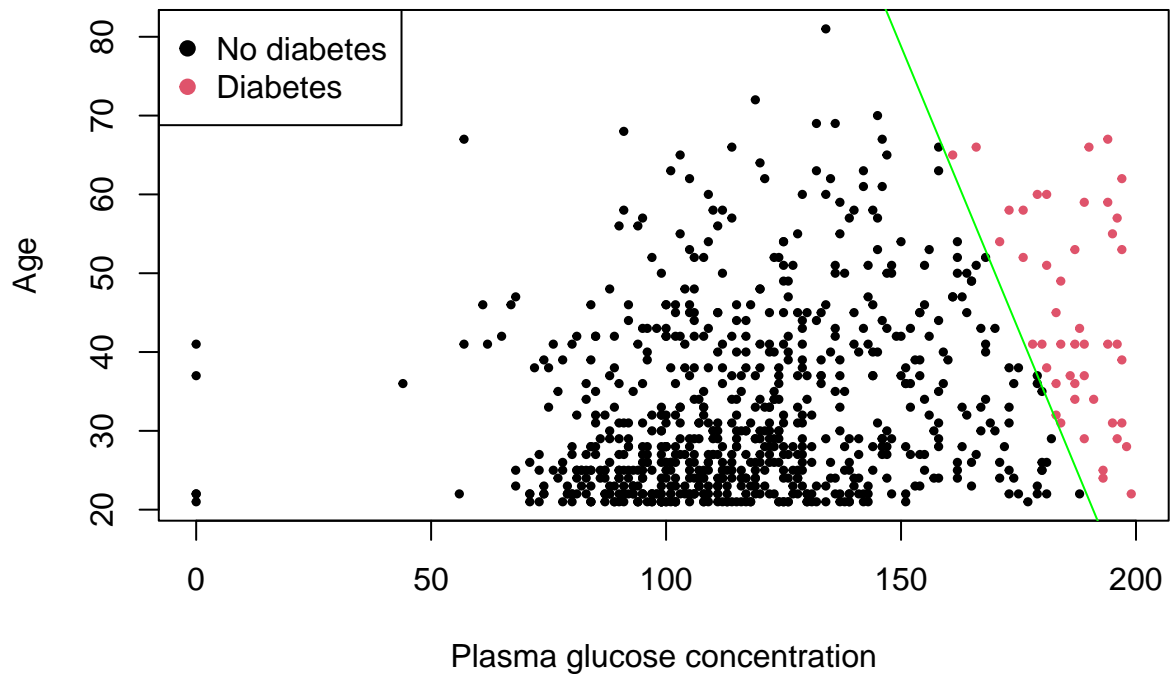
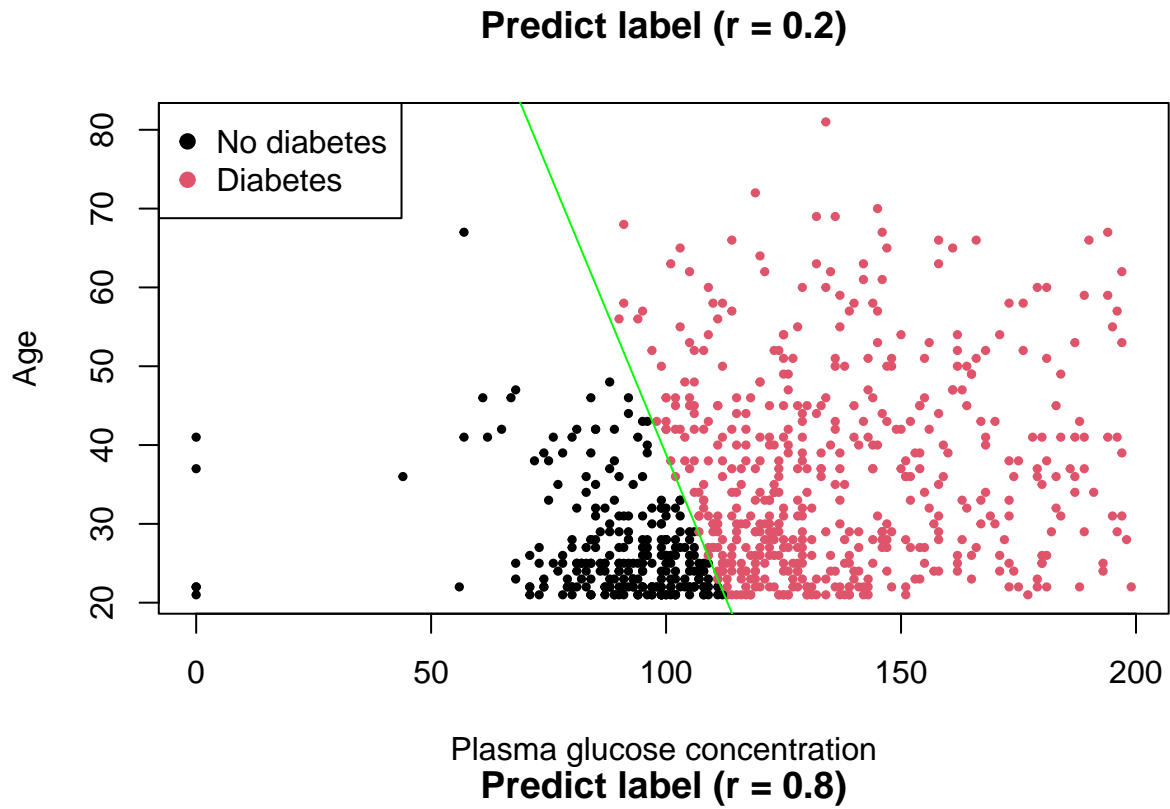
$$5.912 - 0.036x_1 - 0.025x_2 = 0$$





Catch well.

Q4



```
##      r train_error
## 1 0.2  0.3723958
## 2 0.8  0.3151042
```

The decision boundary moves right as  $r$  value increases, making less people can be predicted as diabetes.

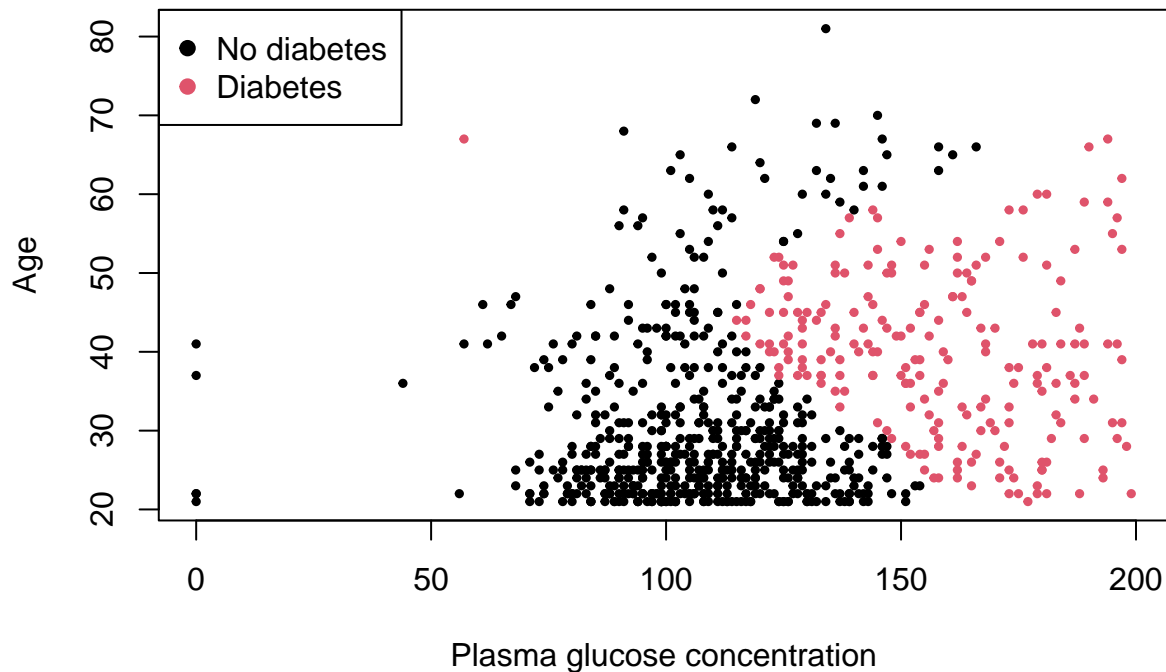
## Q5

```
##      (Intercept)          V2          V8          Z1          Z2
## -9.309821e+00  3.793014e-02  1.456805e-01  1.278015e-08 -1.779600e-07
##           Z3          Z4          Z5
##  8.515150e-07 -1.698011e-06  8.126623e-07

##      pred
## true    0    1
##      0 433  67
##      1 121 147

## The training misclassification error is: 0.2447917
```

### Predict label ( $r = 0.5$ )



The basis expansion trick makes the decision boundary change from straight line to curved line. But the training misclassification rate does not improve much, still remaining closed to a quarter. From the confusion table, it can be seen that there are 45.1% people with diabetes who are not diagnosed. Therefore, this model still does a poor classification work.

## Appendix

### Code for Assignment 1

#### Q1

```
rawdata <- read.csv("optdigits.csv", header = F)
n <- nrow(rawdata)
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
```

```

id1 <- setdiff(1:n,id)
id2 <- sample(id1,floor(n*0.25))
id3 <- setdiff(id1,id2)

train <- rawdata[id,]
valid <- rawdata[id2,]
test <- rawdata[id3,]

```

## Q2

```

suppressWarnings({
  library(kknn)
})
m1 <- kknn(as.factor(train$V65)~.,train,train,k=30,kernel="rectangular")
train_true <- train$V65
train_predict <- m1$fitted.values
train_table <- table(train_true, train_predict)
cat("Confusion matrix for the training data:\n")
print(train_table)

m2 <- kknn(as.factor(train$V65)~.,train,test,k=30,kernel="rectangular")
test_true <- test$V65
test_predict <- m2$fitted.values
test_table <- table(test_true, test_predict)
cat("Confusion matrix for the test data:\n")
print(test_table)

train_mis <- 1-sum(diag(train_table))/sum(train_table)
cat(paste0("Misclassification error for the training data: ",round(train_mis,5),"n"))

test_mis <- 1-sum(diag(test_table))/sum(test_table)
cat(paste0("Misclassification error for the test data: ",round(test_mis,5),"n"))

error_list <- list()
for (i in 1:10){
  error_Rate <- 1-test_table[i,i]/sum(test_table[i,])
  error_list$digit[i] <- i-1
  error_list$error_rate[i] <- error_Rate
}
error_df <- as.data.frame(error_list)
error_df <- error_df[order(error_df$error,decreasing = T),]
cat("Misclassification error for each digit in the test data:\n")
print(error_df)

```

## Q3

```

correct_index <- which(train_predict==8 & train_true==8)

correst_prob <- m1$prob[correct_index,9]

easy_index <- order(correst_prob, decreasing = TRUE)[1:2]
hard_index <- order(correst_prob)[1:3]

```

```

for (i in 1:length(easy_index)){
  df <- train[correct_index[easy_index[i]],-ncol(train)]
  mat <- matrix(as.numeric(df),8,byrow = T)
  heatmap(mat, Colv=NA, Rowv=NA, main=paste0("Easiest_", i))
}

for (i in 1:length(hard_index)){
  df <- train[correct_index[hard_index[i]],-ncol(train)]
  mat <- matrix(as.numeric(df),8,byrow = T)
  heatmap(mat, Colv=NA, Rowv=NA, main=paste0("Hardest_", i))
}

```

#### Q4

```

store_list <- list()
for (i in 1:30){
  store_list$K[i] <- i
  train_model <- kknk(as.factor(train$V65)~.,train,train,k=i,kernel="rectangular")
  train_table <- table(train$V65,train_model$fitted.values)
  store_list$train_error[i] <- 1-sum(diag(train_table))/sum(train_table)

  valid_model <- kknk(as.factor(train$V65)~.,train,valid,k=i,kernel="rectangular")
  valid_table <- table(valid$V65,valid_model$fitted.values)
  store_list$valid_error[i] <- 1-sum(diag(valid_table))/sum(valid_table)
}
store_df <- as.data.frame(store_list)

plot(store_df$K, store_df$train_error, type = "l", col = "blue", lty = 1, lwd = 2, ylim = range(c(store,
lines(store_df$K, store_df$valid_error, col = "red", lty = 2, lwd = 2)
legend("bottomright", legend = c("train", "valid"), col = c("blue", "red"), lty = 1:2, lwd = 2)

m3 <- kknk(as.factor(train$V65)~.,train,test,k=3,kernel="rectangular")
test_true <- test$V65
test_predict <- m3$fitted.values
test_table <- table(test_true, test_predict)
test_error <- 1-sum(diag(test_table))/sum(test_table)
cat(paste0("Misclassification error for the test data: ",test_error))

errors <- cbind(store_df[3,],test_error)
print(errors)

```

#### Q5

```

compute_cross_entropy <- function(true_labels, predicted_probs) {
  -sum(log(predicted_probs[cbind(1:length(true_labels), true_labels + 1)] + 1e-15))
}

store_list <- list()
for (i in 1:30){
  store_list$K[i] <- i
  valid_model <- kknk(as.factor(train$V65)~.,train,valid,k=i,kernel="rectangular")
  store_list$valid_error[i] <- compute_cross_entropy(valid$V65,valid_model$prob)
}

```

```
store_df <- as.data.frame(store_list)
plot(store_df$K, store_df$valid_error, xlab = "K value", ylab = "Validation error")
```

## Code for Assignment 2

### Q1

```
# load necessary package
library(caret)
library(dplyr)
# import data
rawdata <- read.csv("parkinsons.csv")

rawdata <- rawdata %>% select(-c(1:4,6))
n <- nrow(rawdata)
set.seed(12345)
id <- sample(1:n,floor(n*0.6))
train <- rawdata[id,]
test <- rawdata[-id,]

# scale data
param <- preProcess(train)
train_scaled <- predict(param,train)
test_scaled <- predict(param,test)
```

### Q2

```
# calculate mse
mse <- function(true_value, predict_value){
  mean((true_value - predict_value)^2)
}

# linear regression model
model <- lm(motor_UPDRS ~ ., data = train_scaled)

# Training and test MSE
train_mse <- mse(train_scaled$motor_UPDRS, predict(model, train_scaled))
cat(paste0("Training MSE is: ",train_mse,".\n"))

test_mse <- mse(test_scaled$motor_UPDRS, predict(model, test_scaled))
cat(paste0("Test MSE is: ",test_mse,".\n"))

# Significant variables
summary(model)
```

### Q3

```
# Log-likelihood function
Loglikelihood <- function(theta_vec, sigma){
  y <- train_scaled[,1]
  x <- as.matrix(train_scaled[,-1])
  n <- nrow(train_scaled)
```

```

value <- -(n/2*log(sigma^2*2*pi)+sum((y-x%%theta_vec)^2)/(2*sigma^2))
return(value)
}
# Ridge log-likelihood function
Ridge <- function(theta_vec, sigma, lambda){
  -Loglikelihood(theta_vec, sigma)+lambda*sum(theta_vec^2)
}
# Ridge log-likelihood optimization function
RidgeOpt <- function(lambda){
  init_theta <- rep(0, ncol(train_scaled)-1)
  init_sigma <- 0.9

  objective_function <- function(params) {
    theta_vec <- params[-length(params)]
    sigma <- params[length(params)]
    return(Ridge(theta_vec, sigma, lambda))
  }

  rlt <- optim(c(init_theta, init_sigma), fn = objective_function, method = "BFGS")

  optimal_theta_vec <- rlt$par[-length(rlt$par)]
  optimal_sigma <- rlt$par[length(rlt$par)]

  return(list(theta_vec = optimal_theta_vec, sigma = optimal_sigma))
}
# Degrees of freedom function
DF <- function(lambda){
  x <- as.matrix(train_scaled[, -1])
  df <- sum(diag(x%%solve(t(x)%*%x+lambda*diag(ncol(x)))%*%t(x)))
  return(df)
}

```

#### Q4

```

train_x <- as.matrix(train_scaled[, -1])
train_y <- train_scaled[, 1]
test_x <- as.matrix(test_scaled[, -1])
test_y <- test_scaled[, 1]

# Ridge optimization for different lambdas
Lambda <- c(1, 100, 1000)
store_list <- list()

for (i in 1:length(Lambda)){
  store_list$lambda[i] <- Lambda[i]
  optimal_theta_vec <- RidgeOpt(Lambda[i])$theta_vec
  store_list$train_MSE[i] <- mse(train_y, train_x%%optimal_theta_vec)
  store_list$test_MSE[i] <- mse(test_y, test_x%%optimal_theta_vec)
  store_list$DoF[i] <- DF(Lambda[i])
  store_list$sigma[i] <- RidgeOpt(Lambda[i])$sigma
}
store_df <- as.data.frame(store_list)
print(store_df)

```

## Code for Assignment 3

### Q1

```
rawdata <- read.csv("pima-indians-diabetes.csv", header = F)

data1 <- rawdata[,c(2,8,9)]
plot(data1$V2, data1$V8, col = data1$V9 + 1, pch = 19, cex = 0.5, xlab = "Plasma glucose concentration",
legend("topleft", legend = c("No diabetes", "Diabetes"), col = c(1, 2), pch = 19)
```

### Q2

```
m1=glm(V9~., data1, family = "binomial")
m1$coefficients

Prob=predict(m1, type="response")
pred=ifelse(Prob>0.5, 1, 0)

true <- data1$V9
con_table <- table(true, pred)
con_table

train_error <- 1-sum(diag(con_table))/sum(con_table)
cat("The training misclassification error is:",train_error,"\n")

plot(data1$V2, data1$V8, col = pred + 1, pch = 19, cex = 0.5, xlab = "Plasma glucose concentration", ylab = "Diabetes",
legend("topleft", legend = c("No diabetes", "Diabetes"), col = c(1, 2), pch = 19)
```

### Q3

```
plot(data1$V2, data1$V8, col = pred + 1, pch = 19, cex = 0.5, xlab = "Plasma glucose concentration", ylab = "Diabetes",
legend("topleft", legend = c("No diabetes", "Diabetes"), col = c(1, 2), pch = 19)
abline(a = 5.91244906/0.02477835, b = -0.03564404/0.02477835, col = "green")
```

### Q4

```
threshold <- c(0.2,0.8)
store_list <- list()

for (i in 1:length(threshold)){
  store_list$r[i] <- threshold[i]
  pred=ifelse(Prob>threshold[i], 1, 0)
  con_table <- table(true, pred)
  #store_list$confusion_table[i] <- con_table
  store_list$train_error[i] <- 1-sum(diag(con_table))/sum(con_table)

  plot(data1$V2, data1$V8, col = pred + 1, pch = 19, cex = 0.5, xlab = "Plasma glucose concentration", ylab = "Diabetes",
  legend("topleft", legend = c("No diabetes", "Diabetes"), col = c(1, 2), pch = 19)
  abline(a = -(log((1-threshold[i])/threshold[i]))-5.91244906)/0.02477835, b = -0.03564404/0.02477835, col = "green")
}

store_df <- as.data.frame(store_list)
print(store_df)
```



## Q5

```
library(dplyr)
data1 <- data1%>%mutate(Z1=V2^4, Z2=V2^3*V8, Z3=V2^2*V8^2, Z4=V2*V8^3, Z5=V8^4)

m1=glm(V9~., data1, family = "binomial")
m1$coefficients

Prob=predict(m1, type="response")
pred=ifelse(Prob>0.5, 1, 0)

true <- data1$V9
con_table <- table(true, pred)
con_table

train_error <- 1-sum(diag(con_table))/sum(con_table)
cat("The training misclassification error is:",train_error,"\n")

plot(data1$V2, data1$V8, col = pred + 1, pch = 19, cex = 0.5, xlab = "Plasma glucose concentration", ylab = "Diabetes",
legend("topleft", legend = c("No diabetes", "Diabetes"), col = c(1, 2), pch = 19)
```