Lab1- Group A 13

Arash Haratian, Yi Hung Chen

2022-11-15

Assignment 1

1) Importing And Spliting The Data

First we read the data and split it to three parts (training, validation and test sets) using the code from the lecture slides:

```
optdigits <- read.csv("D:/optdigits.csv", header = FALSE)
optdigits <- optdigits %>%
    mutate(y = as.factor(V65)) %>%
    select(!V65)
n <- nrow(optdigits)
set.seed(12345)
train_idx <- sample(seq_len(n), floor(n * 0.5))
train_data <- optdigits[train_idx, ]
remainder_idx <- setdiff(seq_len(n), train_idx)
set.seed(12345)
valid_idx <- sample(remainder_idx, floor(n * 0.25))
valid_data <- optdigits[valid_idx, ]
test_idx <- setdiff(remainder_idx, valid_idx)
test_data <- optdigits[test_idx, ]
```

2) Training the 30-NN

```
knn_model_train <- kknn(y ~ ., train = train_data, test = train_data,
    k = 30, kernel = "rectangular")
g_hat_train <- knn_model_train$fitted.values
cm_train <- table(train_data$y, g_hat_train)
knitr::kable(cm_train, caption = "Confusion matrix for 30-nn on train dataset")
```

Table 1: Confusion matrix for 30-nn on train dataset

	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

	0	1	2	3	4	5	6	7	8	9
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

```
error_train <- 1 - (sum(diag(cm_train))/length(g_hat_train))
```

```
knn_model_test <- kknn(y ~ ., train = train_data, test = test_data,
    k = 30, kernel = "rectangular")
g_hat_test <- knn_model_test$fitted.values
cm_test <- table(test_data$y, g_hat_test)
knitr::kable(cm_test, caption = "Confusion matrix for 30-nn on test dataset")
```

Table 2: Confusion matrix for 30-nn on test dataset

	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

```
error_test <- 1 - (sum(diag(cm_test))/length(g_hat_test))
```

As it is evidenced in the confusion matrix for train dataset, 17 observations with label `9` misclassified. Also classes `1`, `4`, and `8`, each has 16 observations that are misclassified. For instance, 10 of the observations from class `8` has classified as class `1`.

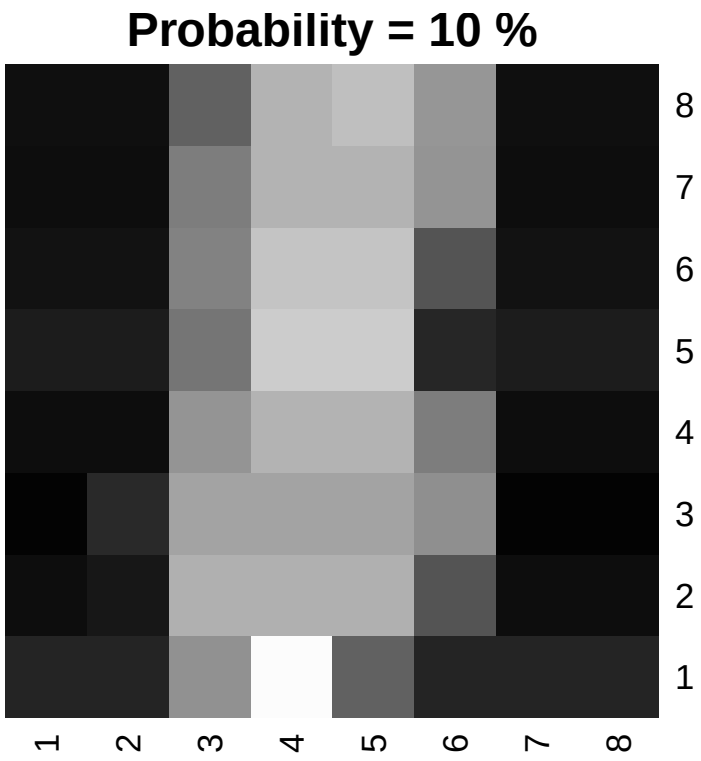
On the other hand, class `4` has the most number of misclassifications (15 in overall). Moreover, Classes `5` and `8` have the most number of misclassifications after class `4` (10 and 8 respectively).

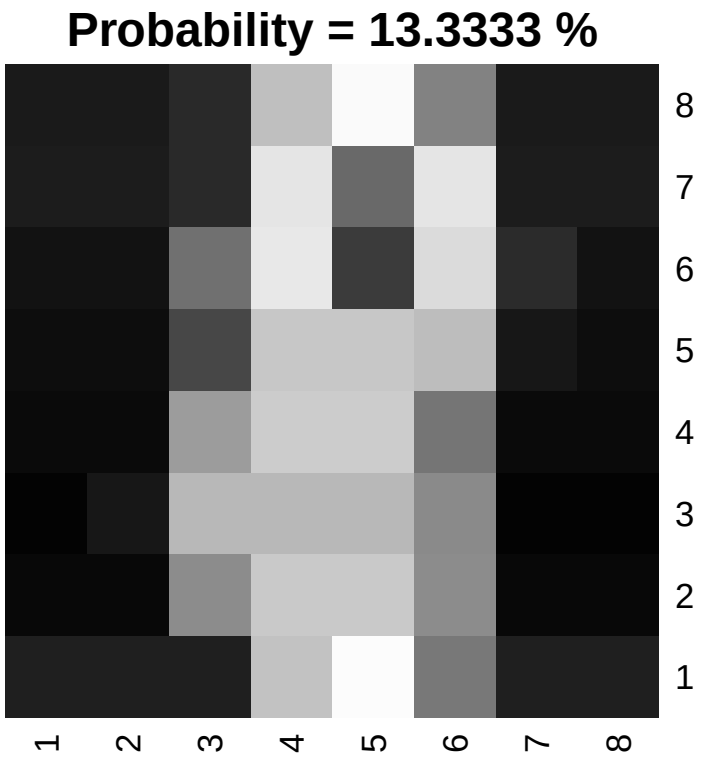
Only one `0` is misclassified in test data while there is no misclassification for class `0` in train dataset.

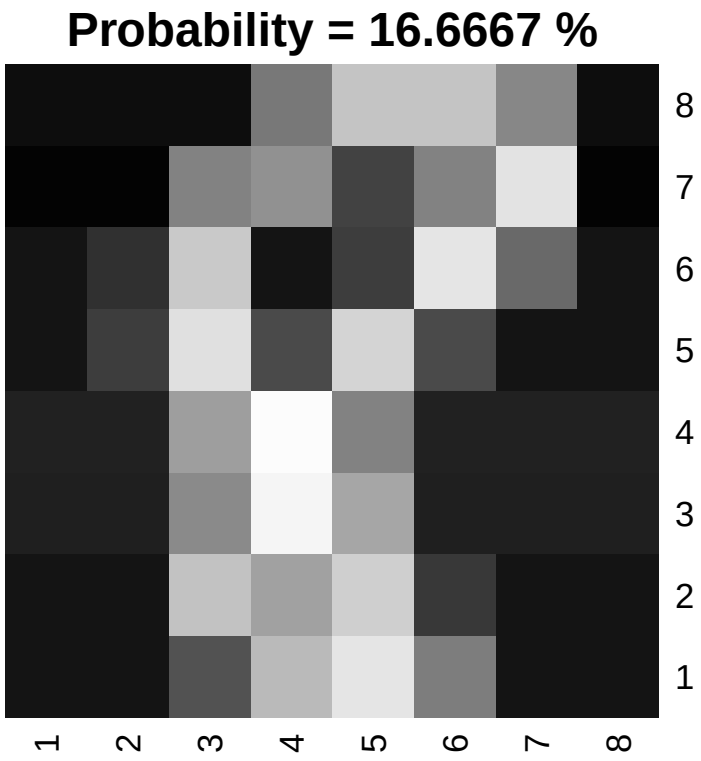
The misclassification error for the train dataset is 4.5002616 percent whereas the misclassification error for the test dataset is 5.3291536 percent‘. The model is biased on the training dataset as it has a very low error rate on train dataset compare to test dataset. ### 3) Plotting 5 Different Observations

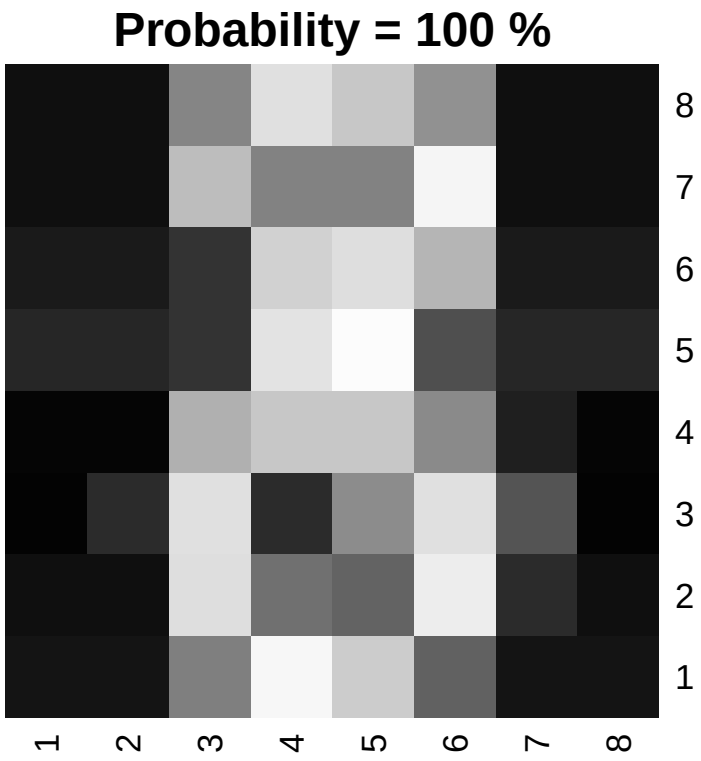
```
prob_g8 <- predict(knn_model_train, type = "prob")
prob_g8 <- prob_g8[, "8"]
observations <- train_data %>%
    mutate(prob = prob_g8) %>%
    filter(y == "8") %>%
    arrange(prob) %>%
    slice(c(1:3, length(y) - 1, length(y)))
```

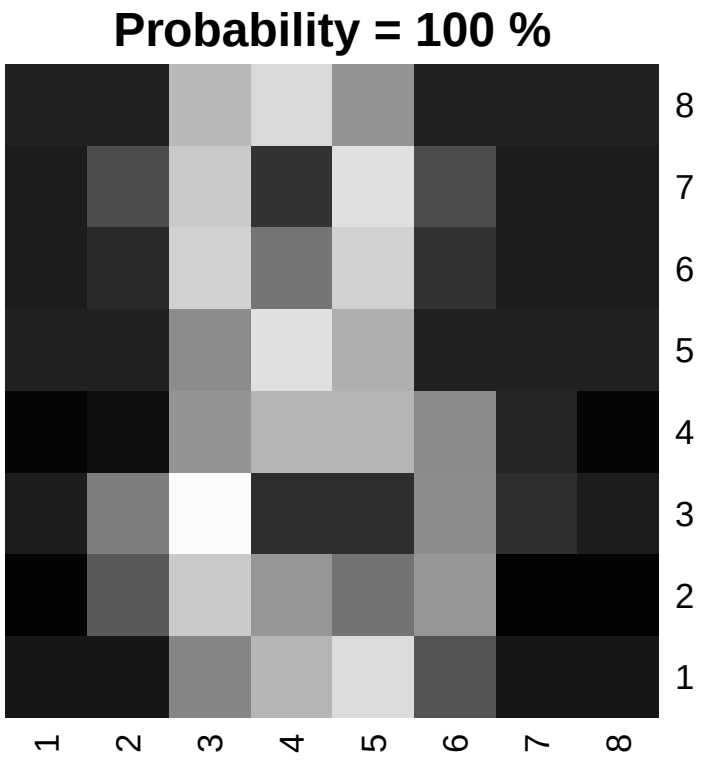
```r
for (i in 1:5) {
    fig_matrix <- matrix(as.numeric(observations[i, 1:64]),
        nrow = 8, byrow = T)
    fig_title <- paste("Probability =", round(observations[i,
        "prob"] * 100, 4), "%")
    heatmap(fig_matrix, Colv = NA, Rowv = NA, col = paste0("gray",
        1:99), main = fig_title)
}
```




Probability = 13.3333 %
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8


Probability = 16.6667 %
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8

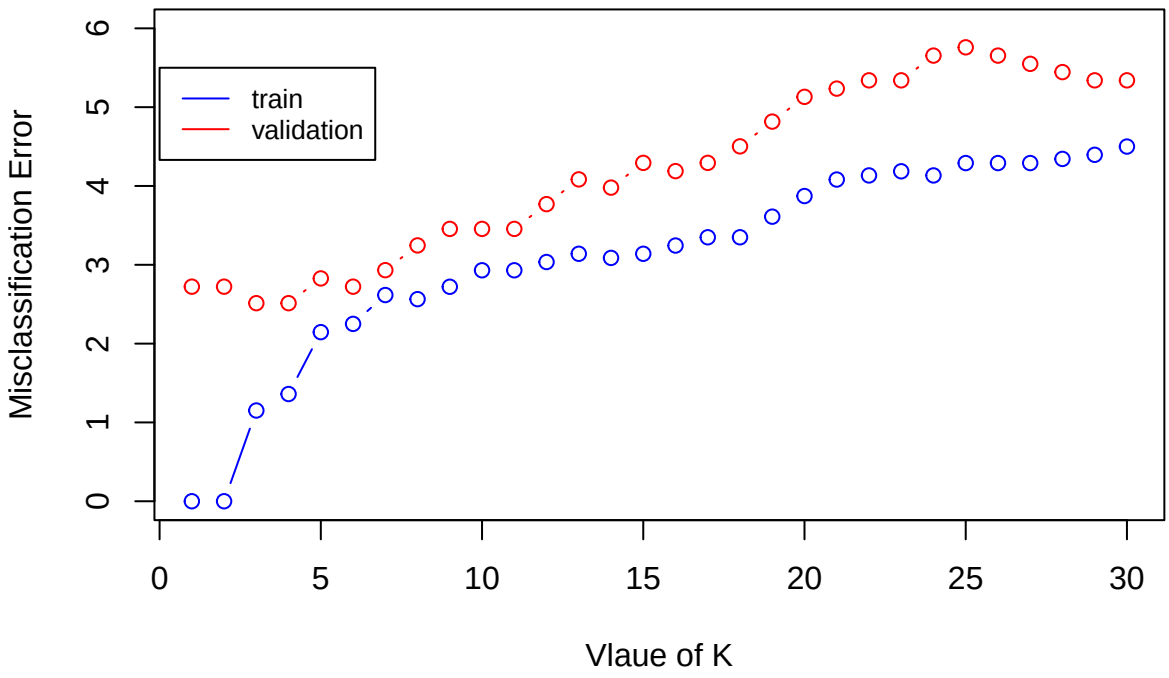

Probability = 100 %
8
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8



The last two plots are clearly resemble number 8, but the first three plots are barely similar to an 8. These first two plots are hard to classify visually since the top and the bottom circles of the 8 figure are blurred out.

4) Best Value of K

```
results <- map(1:30, ~{
    model <- kknn(y ~ ., train = train_data, test = train_data,
        k = .x, kernel = "rectangular")
    train_sum <- sum(diag(table(train_data$y, model$fitted.values)))
    model <- kknn(y ~ ., train = train_data, test = valid_data,
        k = .x, kernel = "rectangular")
    valid_sum <- sum(diag(table(valid_data$y, model$fitted.values)))
    c(train_sum, valid_sum)
})
results <- as.data.frame(results)
names(results) <- 1:30
lengths <- c(nrow(train_data), nrow(valid_data))
errors <- (lengths - results)/lengths * 100
plot(t(errors)[, 1], type = "b", col = "blue", ylim = c(0,
    6), xlab = "Vlaue of K", ylab = "Misclassification Error")
points(t(errors)[, 2], type = "b", col = "red")
legend(0, 5.5, legend = c("train", "validation"), col = c("blue",
    "red"), lty = 1, cex = 0.8)
```



By decreasing the value of `k`, the model predicts the train dataset more accurately than validation dataset (due to overfitting), while the error rate of the validation data first decrease but then it increases slightly (around `k = 3` to `k = 1`).

The best value of hyperparameter `k` is 3.

```
best_k <- which.min(t(errors)[, 2])
knn_model_test <- kknn(y ~ ., train = train_data, test = test_data,
    k = best_k, kernel = "rectangular")
g_hat_test <- knn_model_test$fitted.values
cm_test <- table(test_data$y, g_hat_test)
knitr::kable(cm_test, caption = "Confusion matrix for 30-nn on test dataset")
```

Table 3: Confusion matrix for 30-nn on test dataset

	0	1	2	3	4	5	6	7	8	9
0	77	0	0	0	1	0	0	0	0	0
1	0	84	1	0	0	0	0	0	0	1
2	0	1	100	0	0	0	0	0	0	0
3	0	0	0	109	0	1	0	0	1	0
4	0	0	0	0	103	0	1	2	0	3
5	0	0	1	0	0	100	0	0	0	2
6	0	0	0	0	0	0	90	0	0	0
7	0	0	0	1	0	0	0	111	0	0
8	0	4	0	0	0	0	0	0	74	0

	0	1	2	3	4	5	6	7	8	9
9	0	0	0	0	0	3	0	0	0	86

```
error_test <- 1 - (sum(diag(cm_test))/length(g_hat_test))
```

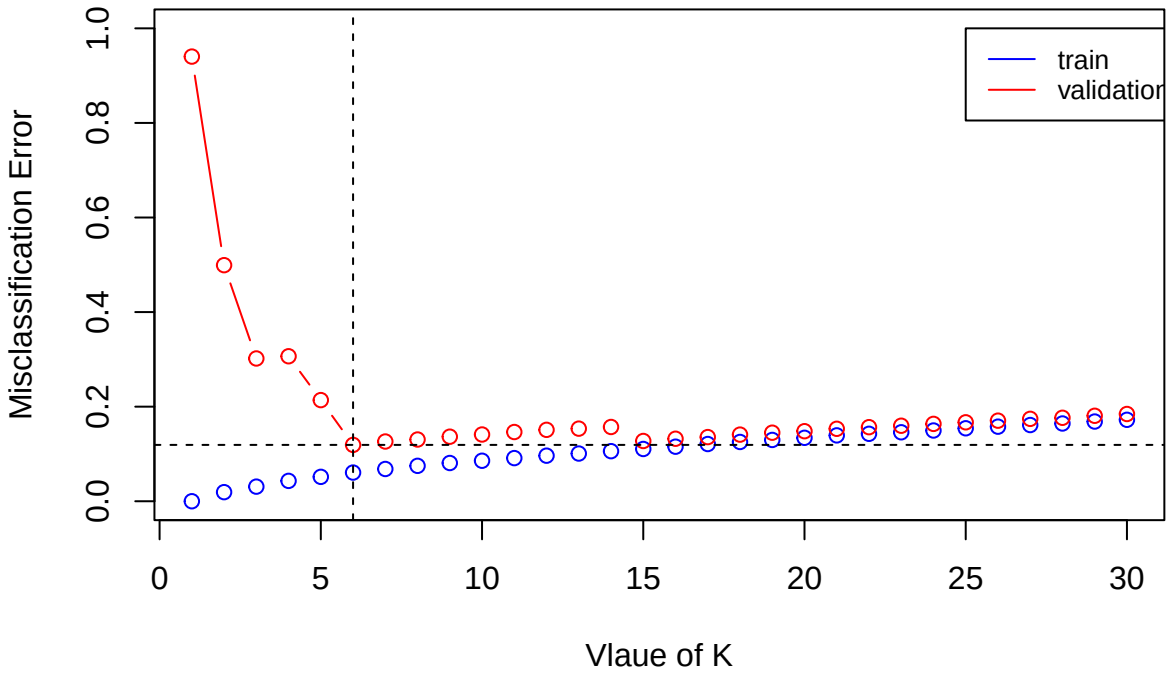
The misclassification error for test dataset is 2.4033% while for validation and train dataset is 2.513089% and 1.1512297% respectively. The error for the test dataset is colse to the one for the validation dataset which is a good sign (it means that model will have predict new data points with high accuracy as well). However, misclassification error for train dataset is smaller which means that the model has overfitted on the train dataset.

5) Best Value of K and Cross Entropy

First we define the cross entropy as a function in R:

```
cross_entropy <- function(y, prob) {
    one_hot <- model.matrix(~0 + y)
    result <- sum(-one_hot * log(prob + 1e-15))
    return(result/length(y))
}
```

```
results <- map(1:30, ~{
    model <- kknn(y ~ ., train = train_data, test = train_data,
        k = .x, kernel = "rectangular")
    ce_train <- cross_entropy(train_data$y, model$prob)
    model <- kknn(y ~ ., train = train_data, test = valid_data,
        k = .x, kernel = "rectangular")
    ce_valid <- cross_entropy(valid_data$y, model$prob)
    c(ce_train, ce_valid)
})
results <- as.data.frame(results)
names(results) <- 1:30
best_k <- which.min(t(results)[, 2])
ce_lowest <- t(results)[best_k, 2]
plot(t(results)[, 1], type = "b", col = "blue", ylim = c(0,
    1), xlab = "Vlaue of K", ylab = "Misclassification Error")
points(t(results)[, 2], type = "b", col = "red")
abline(v = 6, h = ce_lowest, lty = 2)
legend(25, 1, legend = c("train", "validation"), col = c("blue",
    "red"), lty = 1, cex = 0.8)
```



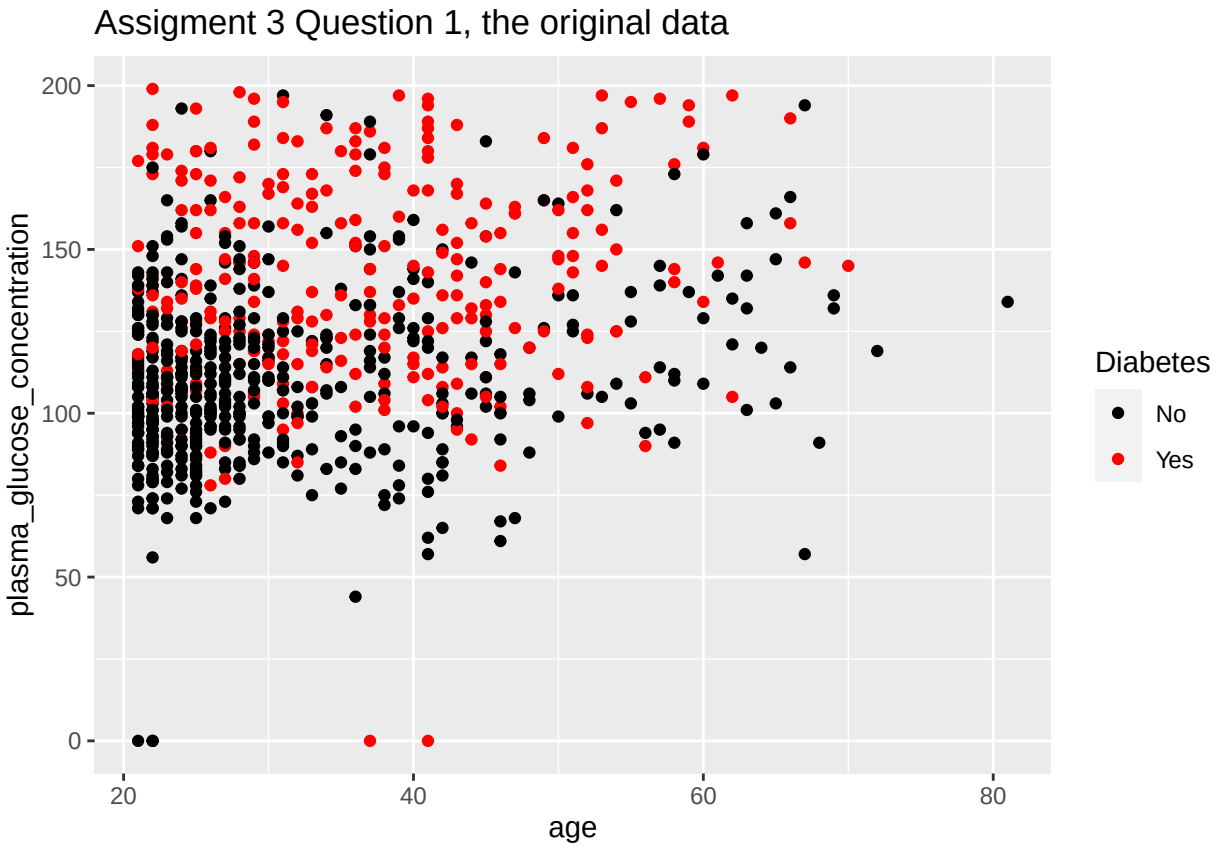
The best value of k is 6 if we use cross-entropy as computing the error for validation dataset. The reason that cross entropy is more useful for datasets with multinomial distribution for the response is that we use probabilities (instead of final labels) to study the quality of the model.

Assignment 3

First, the data from the Excel file *pima-indians-diabetes* will be imported and the column names are changed

Ex.3.1

Make a scatter plot showing a Plasma glucose concentration on Age where observations are colored by Diabetes levels.



Q. Do you think that Diabetes is easy to classify by a standard logistic regression model that uses these two variables as features?

A. In my opinion, it is not easy to classify Diabetes using standard logistic regression model (using age and Plasma glucose concentration as model features). As we can observed on the plot, although the one who "does not" have Diabetes are more concentrate on the bottom left side of the graph, it is still not easy to classify if the age gets older.

Ex.3.2

```
# Use 'glm' function with family = binomial to train
# the logistic regression model
model_1 <- glm(diabetes ~ plasma_glucose_concentration +
    age, data = diabetes_data_1, family = binomial)
summary(model_1)$coef
```

Train a logistic regression model with y =Diabetes as target, x_1 =Plasma glucoseconcentration and x_2 =Age as features. Make a prediction for all observations by using r = 0.5

```
##                               Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)                 -5.91244906 0.46261970 -12.780366 2.110740e-37
## plasma_glucose_concentration  0.03564404 0.00328999  10.834088 2.373176e-27
## age                           0.02477835 0.00737371   3.360364 7.783984e-04
```

Q. Report the probabilistic equation of the estimated model According to the coefficient, the probabilistic equation is

$$p = \frac{1}{1 + e^{5.9124 + 0.0356 * plasma\ glucose\ concentration + 0.0247 age}}$$

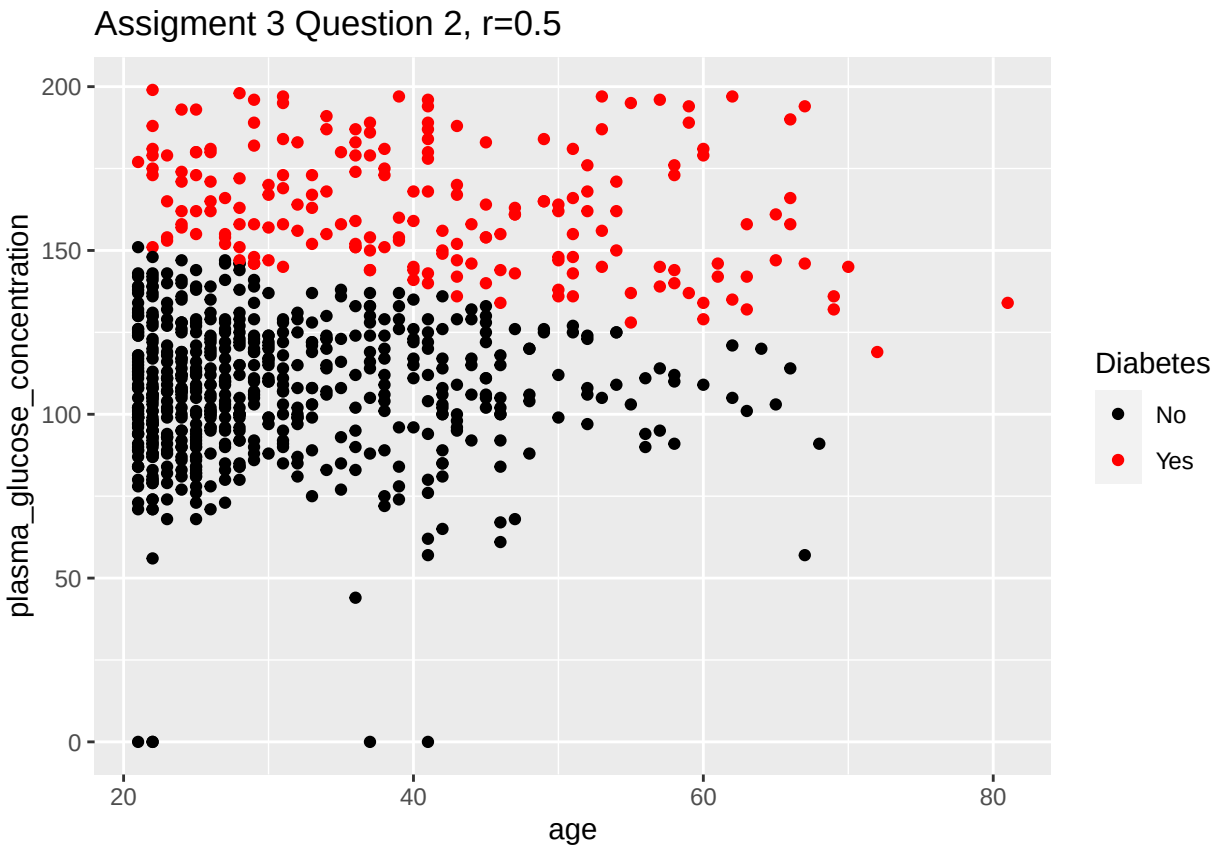
2. Compute training misclassification error

```
missclassification_ex2
```

```
## [1] 0.2630208
```

The missclassification error is 0.2659713

3. Plot the scatter plot showing the predicted values of Diabetes



Q. Comment on the quality of the classification by using these results

A. In my opinion, the quality of the classification is mediocre. Although the overall missclafication rate (26.59713%) is not high, the prediction of older people is not ideal compare to the original data.

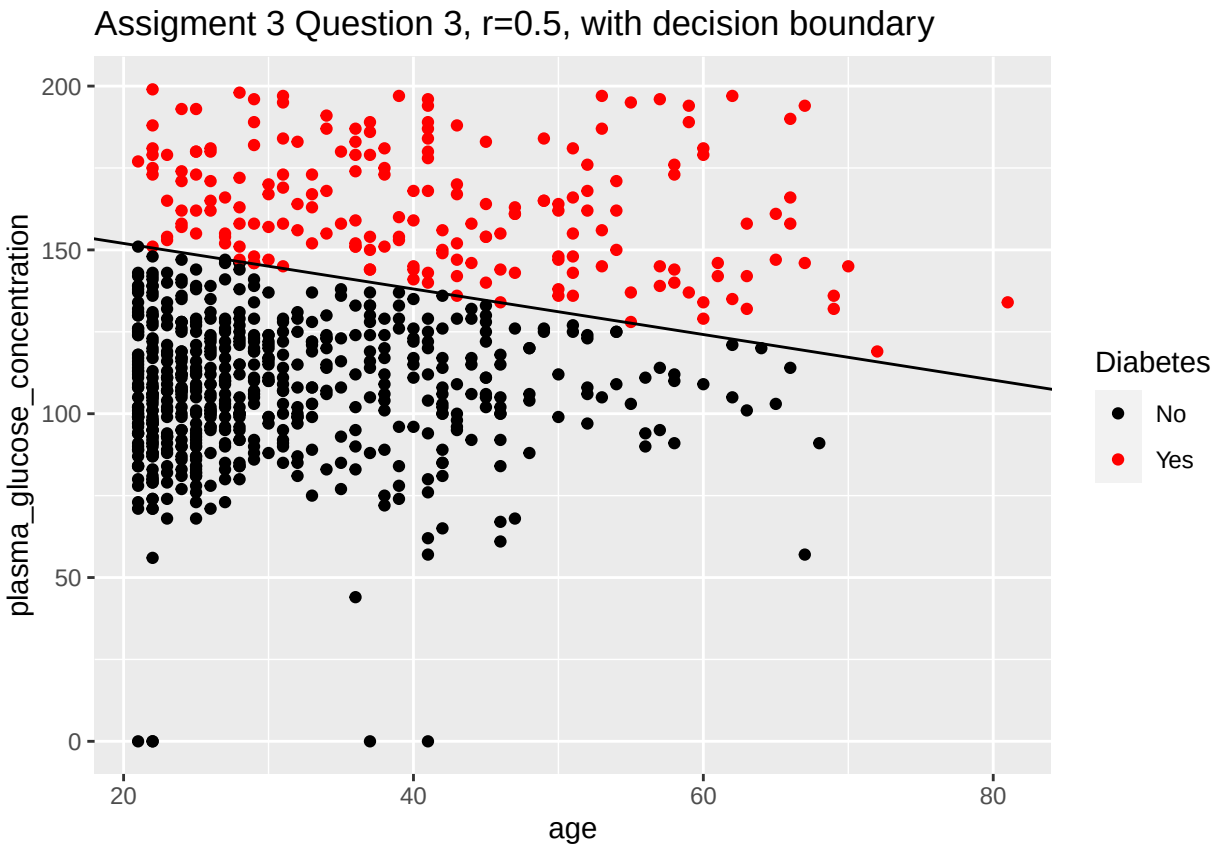
EX.3.3

(a) Report the equation of the decision boundary between the two classes of step 2

The decision boundary equation of step 2 is

$$plasma\ glucose\ concentration = \frac{5.912}{0.03565} + \frac{-0.0247}{0.0356} age = 165.8345 - 0.6938 age$$

(b) Add a curve showing this boundary to the scatter plot

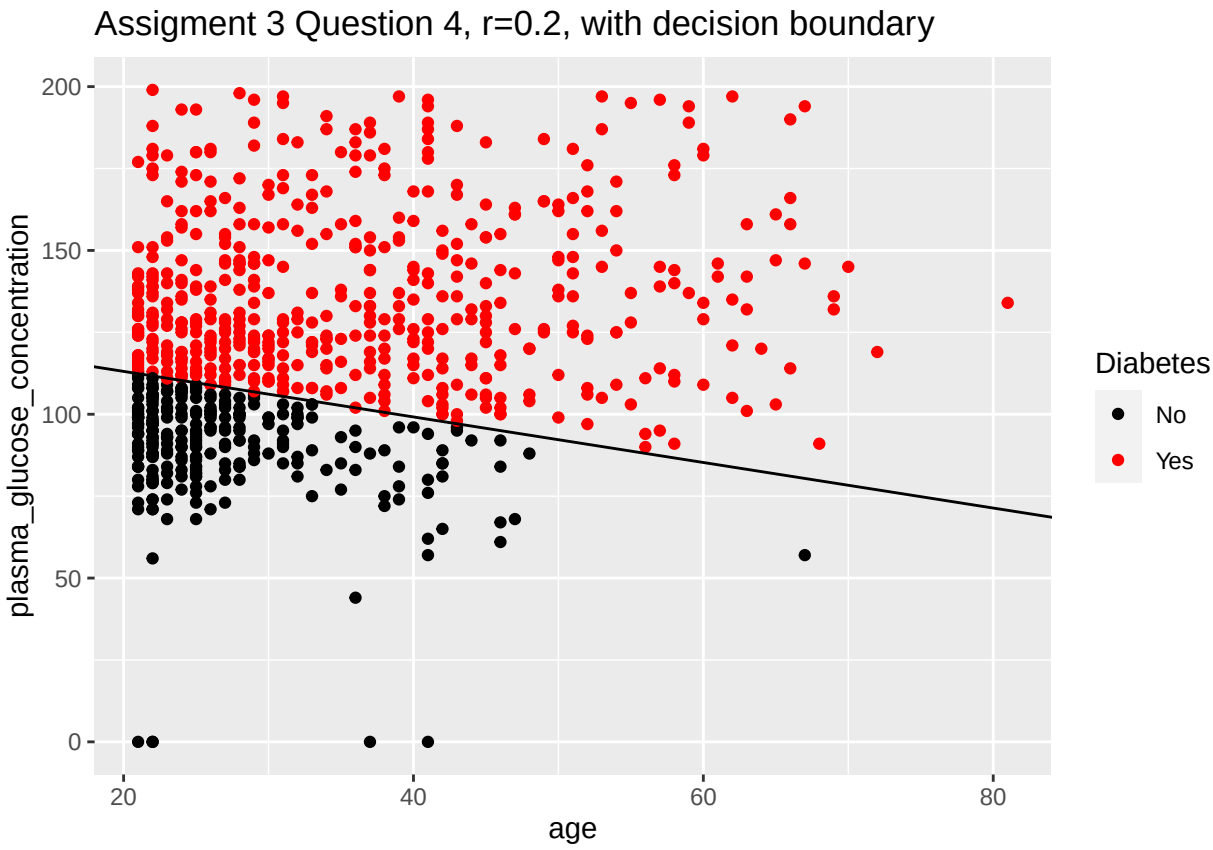


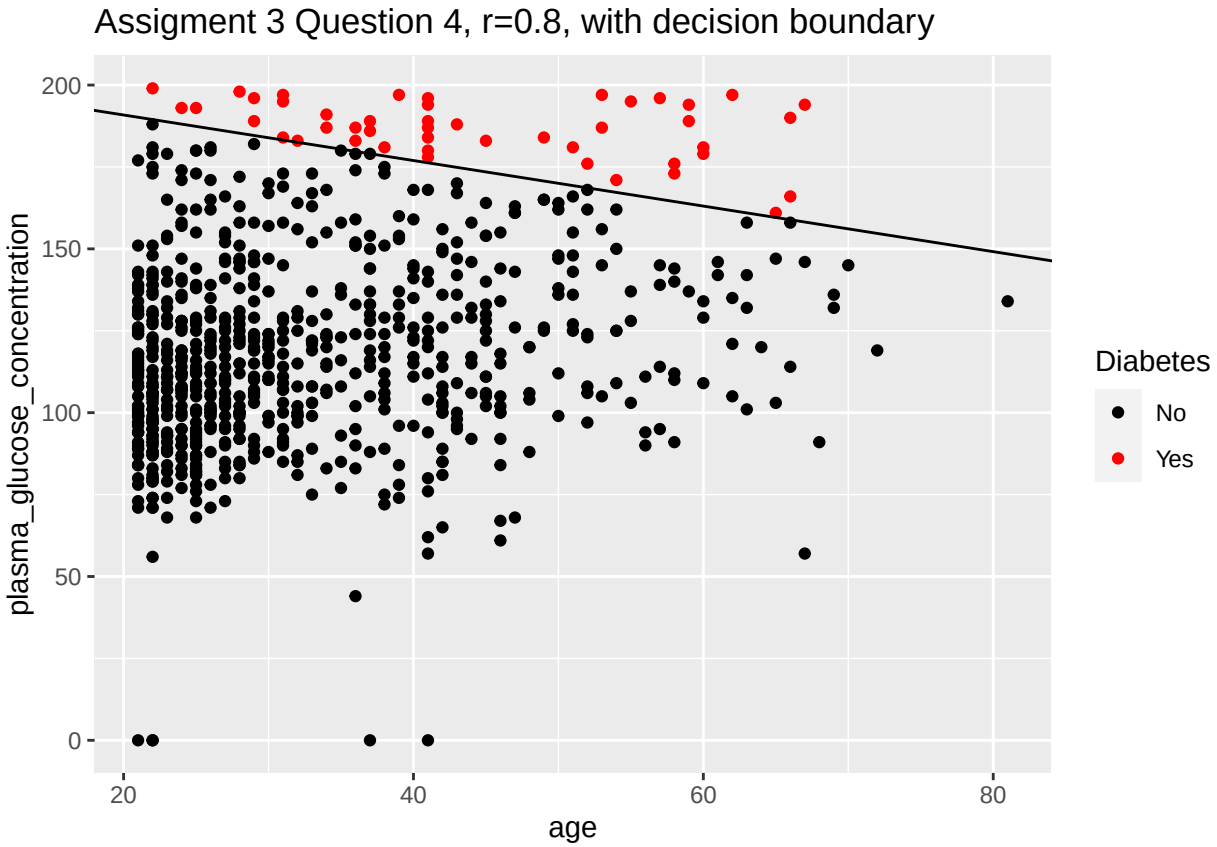
Q. Comment whether the decision boundary seems to catch the data distribution well.

A. As the graph shown, the decision boundary separate the prediction of Diabetes very well in this case. However, because the dot on this particular graph is the predicted data, not the original data. So we can not conclude if the decision boundary catch the original data well or not.

EX.3.4

Make same kind of plots as in step 2 but use thresholds r = 0.2 and r = 0.8



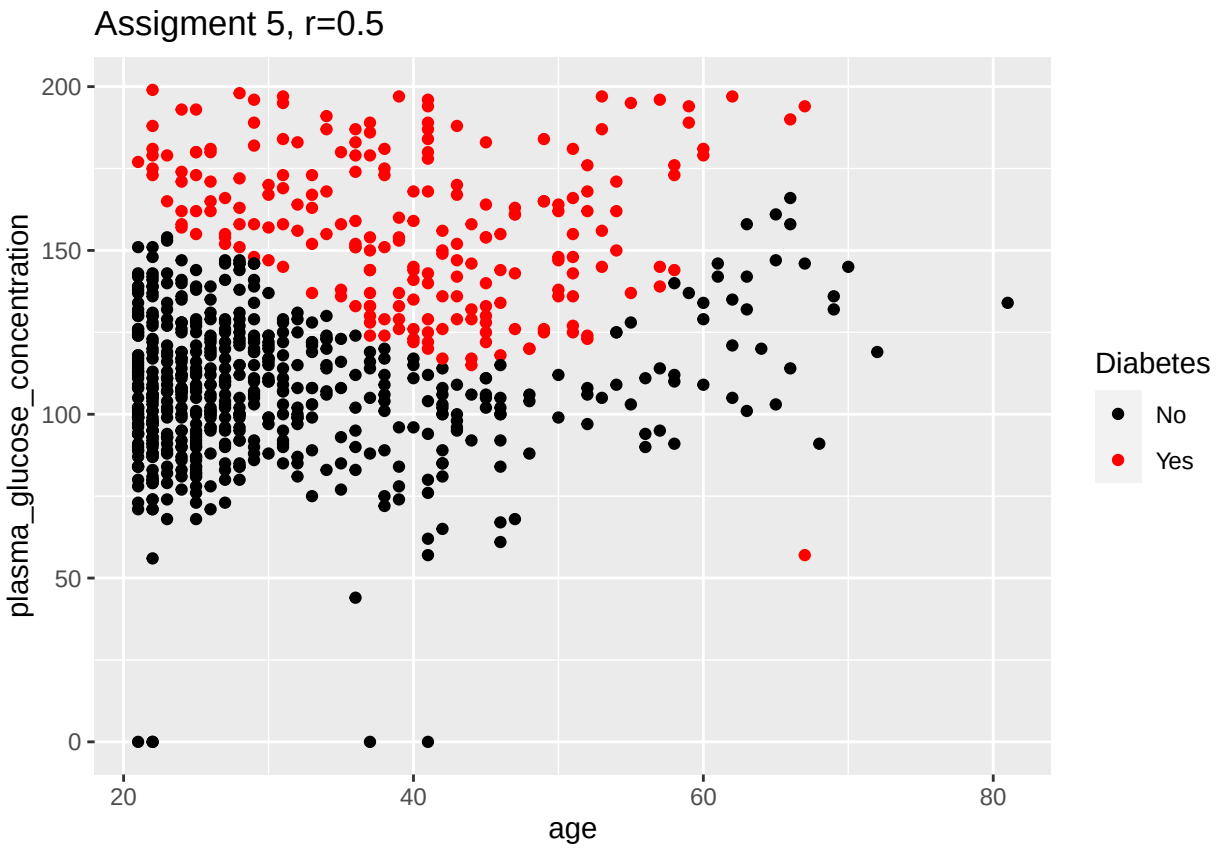


Q. comment on what happens with the prediction when r value changes

A. When r = 0.8, the decision boundary of having diabetes will move toward to the top, which leaves less people being predict to have Diabetes. For r = 0.2 the opposite happen the decision boundary of having diabetes will move toward to the bottom, , which predict less people having Diabetes.
Both cases will result with higher missclasification errors, which means the prediction became less accurate.

EX.3.5

Perform a basis function expansion trick by computing new features




```
## The missclassification is 0.2447917
```

Q. What can you say about the quality of this model compared to the previous logistic regression model? How have the basis expansion trick affected the shape of the decision boundary and the prediction accuracy?

A. According to the missclasification error(0.2464146), this model is by far the best quality one compares to other model in this assignment. The basis expansion trick change the decision boundary to a "U" shape and the prediction is closer to the original data. However, due to higher dimension of the features, the decision boundary is hard to visualize on a 2-D graph, but we can still observe by looking at the color difference. To sum up, using the basis expansion improves the prediction accuracy.

Appendix

```r
# =====Assignment 1=====

library(tidyverse)
library(kknn)
optdigits <- read.csv("./dataset/optdigits.csv", header = FALSE)
optdigits <- optdigits %>%
    mutate(y = as.factor(V65)) %>%
    select(!V65)
n <- nrow(optdigits)
set.seed(12345)
train_idx <- sample(seq_len(n), floor(n * 0.5))
train_data <- optdigits[train_idx, ]
remainder_idx <- setdiff(seq_len(n), train_idx)
set.seed(12345)
valid_idx <- sample(remainder_idx, floor(n * 0.25))
valid_data <- optdigits[valid_idx, ]
test_idx <- setdiff(remainder_idx, valid_idx)
test_data <- optdigits[test_idx, ]
# 2
knn_model_train <- kknn(y ~ ., train = train_data, test = train_data,
    k = 30, kernel = "rectangular")
g_hat_train <- knn_model_train$fitted.values
cm_train <- table(train_data$y, g_hat_train)
knitr::kable(cm_train, caption = "Confusion matrix for 30-nn on train dataset")
error_train <- 1 - (sum(diag(cm_train))/length(g_hat_train))
knn_model_test <- kknn(y ~ ., train = train_data, test = test_data,
    k = 30, kernel = "rectangular")
g_hat_test <- knn_model_test$fitted.values
cm_test <- table(test_data$y, g_hat_test)
knitr::kable(cm_test, caption = "Confusion matrix for 30-nn on test dataset")
error_test <- 1 - (sum(diag(cm_test))/length(g_hat_test))
# 3
prob_g8 <- predict(knn_model_train, type = "prob")
prob_g8 <- prob_g8[, "8"]
observations <- train_data %>%
    mutate(prob = prob_g8) %>%
    filter(y == "8") %>%
    arrange(prob) %>%
    slice(c(1:3, length(y) - 1, length(y)))
for (i in 1:5) {
    fig_matrix <- matrix(as.numeric(observations[i, 1:64]),
        nrow = 8, byrow = T)
    fig_title <- paste("Probability =", round(observations[i,
        "prob"] * 100, 4), "%")
    heatmap(fig_matrix, Colv = NA, Rowv = NA, col = paste0("gray",
        1:99), main = fig_title)
}
# 4
results <- map(1:30, ~{
    model <- kknn(y ~ ., train = train_data, test = train_data,
        k = .x, kernel = "rectangular")
```

```
    train_sum <- sum(diag(table(train_data$y, model$fitted.values)))
    model <- kknn(y ~ ., train = train_data, test = valid_data,
        k = .x, kernel = "rectangular")
    valid_sum <- sum(diag(table(valid_data$y, model$fitted.values)))
    c(train_sum, valid_sum)
})
results <- as.data.frame(results)
names(results) <- 1:30
lengths <- c(nrow(train_data), nrow(valid_data))
errors <- (lengths - results)/lengths * 100
plot(t(errors)[, 1], type = "b", col = "blue", ylim = c(0,
    6), xlab = "Vlaue of K", ylab = "Misclassification Error")
points(t(errors)[, 2], type = "b", col = "red")
legend(0, 5.5, legend = c("train", "validation"), col = c("blue",
    "red"), lty = 1, cex = 0.8)
# 5
cross_entropy <- function(y, prob) {
    one_hot <- model.matrix(~0 + y)
    result <- sum(-one_hot * log(prob + 1e-15))
    return(result/length(y))
}
results <- map(1:30, ~{
    model <- kknn(y ~ ., train = train_data, test = train_data,
        k = .x, kernel = "rectangular")
    ce_train <- cross_entropy(train_data$y, model$prob)
    model <- kknn(y ~ ., train = train_data, test = valid_data,
        k = .x, kernel = "rectangular")
    ce_valid <- cross_entropy(valid_data$y, model$prob)
    c(ce_train, ce_valid)
})
results <- as.data.frame(results)
names(results) <- 1:30
best_k <- which.min(t(results)[, 2])
ce_lowest <- t(results)[best_k, 2]
plot(t(results)[, 1], type = "b", col = "blue", ylim = c(0,
    1), xlab = "Vlaue of K", ylab = "Misclassification Error")
points(t(results)[, 2], type = "b", col = "red")
abline(v = 6, h = ce_lowest, lty = 2)
legend(25, 1, legend = c("train", "validation"), col = c("blue",
    "red"), lty = 1, cex = 0.8)
```

```r
# =====Assignment 3===== =====Assignment 3=====

# =====Set Up=====
diabetes_data <- read.csv("D:/pima-indians-diabetes.csv",
    header = FALSE)
colnames(diabetes_data) <- c("number_of_times_pregnant",
    "plasma_glucose_concentration", "blood_pressure", "triceps_skinfold_thickness",
    "serum_insulin", "bmi", "diabetes_pedigree_function",
    "age", "diabetes")
library(ggplot2)
# ===== ÉX 3.1===== create 'diabetes_data_1' so the
# original data wont be affect
diabetes_data_1 <- diabetes_data

# use 'as.factor' so 1 means has diabetes, 0 means no
# diabetes
diabetes_data_1$diabetes <- as.factor(ifelse(diabetes_data_1$diabetes ==
    1, "Yes", "No"))  # use 'as.factor' so 1 means has diabetes, 0 means no diabetes
plot_assigment3_q1 <- ggplot(diabetes_data_1, aes(x = age,
    y = plasma_glucose_concentration, color = diabetes)) +
    geom_point() + labs(title = "Assigment 3 Question 1, the original data",
    colour = "Diabetes") + scale_color_manual(values = c("#000000",
    "#ff0000"))

plot_assigment3_q1
# ===== ÉX 3.2===== =====1=====
model_1 <- glm(diabetes ~ plasma_glucose_concentration +
    age, data = diabetes_data_1, family = binomial)
summary(model_1)$coef
diabetes_data_1$probabilities <- predict(model_1, diabetes_data_1,
    type = "response")
# The type='response' option tells R to output
# probabilities of the form P(Y = 1|X), as opposed to
# other information such as the logit.

diabetes_data_1$predicted_classes_0.5 <- as.factor(ifelse(diabetes_data_1$probabilities >
    0.5, "Yes", "No"))

# =====2=====
missclass = function(X, X1) {
    n = length(X)
    return(1 - sum(diag(table(X, X1)))/n)
}
missclassification_ex2 <- missclass(diabetes_data_1$diabetes,
    diabetes_data_1$predicted_classes_0.5)
missclassification_ex2
# =====3=====
plot_assigment3_q2 <- ggplot(diabetes_data_1) + geom_point(aes(x = age,
    y = plasma_glucose_concentration, color = predicted_classes_0.5)) +
    labs(title = "Assigment 3 Question 2, r=0.5", colour = "Diabetes") +
    scale_color_manual(values = c("#000000", "#ff0000"))

plot_assigment3_q2
```

```r
# ===== ÉX 3.3===== To correct the intercept on the
# plot if the threshold is not 0.5
inverse_logit <- function(threshold) {
    return(-log((1 - threshold)/threshold))

}
decision_boundary <- function(a, b, c, ...) {
    # function to plot decision boundary
    slope <- -a/b
    intercept <- -c/b
    geom_abline(slope = slope, intercept = intercept, ...)

}
plot_assigment3_q3 <- ggplot(diabetes_data_1) + geom_point(aes(x = age,
    y = plasma_glucose_concentration, color = predicted_classes_0.5)) +
    labs(title = "Assigment 3 Question 3, r=0.5, with decision boundary",
        colour = "Diabetes") + scale_color_manual(values = c("#000000",
    "#ff0000")) + decision_boundary(model_1$coefficients[3],
    model_1$coefficients[2], model_1$coefficients[1] - inverse_logit(0.5))
plot_assigment3_q3


# ===== ÉX 3.4===== ===== r=0.2 =====
diabetes_data_1$predicted_classes_0.2 <- as.factor(ifelse(diabetes_data_1$probabilities >
    0.2, "Yes", "No"))

plot_assigment3_q4_r0.2 <- ggplot(diabetes_data_1) + geom_point(aes(x = age,
    y = plasma_glucose_concentration, color = predicted_classes_0.2)) +
    labs(title = "Assigment 3 Question 4, r=0.2, with decision boundary",
        colour = "Diabetes") + scale_color_manual(values = c("#000000",
    "#ff0000")) + decision_boundary(model_1$coefficients[3],
    model_1$coefficients[2], model_1$coefficients[1] - inverse_logit(0.2))

plot_assigment3_q4_r0.2

# ===== r=0.8 =====
diabetes_data_1$predicted_classes_0.8 <- as.factor(ifelse(diabetes_data_1$probabilities >
    0.8, "Yes", "No"))

plot_assigment3_q4_r0.8 <- ggplot(diabetes_data_1) + geom_point(aes(x = age,
    y = plasma_glucose_concentration, color = predicted_classes_0.8)) +
    labs(title = "Assigment 3 Question 4, r=0.8, with decision boundary",
        colour = "Diabetes") + scale_color_manual(values = c("#000000",
    "#ff0000")) + decision_boundary(model_1$coefficients[3],
    model_1$coefficients[2], model_1$coefficients[1] - inverse_logit(0.8))
plot_assigment3_q4_r0.8

# ===== ÉX 3.5=====
diabetes_data_ex5 <- diabetes_data
# Create new data frame so it won't affect the
# previous data frame

# Add new features
```

```r
diabetes_data_ex5$z1 <- (diabetes_data_ex5$plasma_glucose_concentration)^4
diabetes_data_ex5$z2 <- (diabetes_data_ex5$plasma_glucose_concentration)^3 *
    diabetes_data_ex5$age
diabetes_data_ex5$z3 <- (diabetes_data_ex5$plasma_glucose_concentration)^2 *
    (diabetes_data_ex5$age)^2
diabetes_data_ex5$z4 <- (diabetes_data_ex5$plasma_glucose_concentration)^1 *
    (diabetes_data_ex5$age)^3
diabetes_data_ex5$z5 <- (diabetes_data_ex5$age)^4

# Do the model using glm with new features
model_2 <- glm(diabetes ~ plasma_glucose_concentration +
    age + z1 + z2 + z3 + z4 + z5, data = diabetes_data_ex5,
    family = binomial)

diabetes_data_ex5$probabilities <- predict(model_2, diabetes_data_ex5,
    type = "response")
diabetes_data_ex5$predicted_classes_0.5 <- as.factor(ifelse(diabetes_data_ex5$probabilities >
    0.5, "Yes", "No"))
plot_assigment3_q5 <- ggplot(diabetes_data_ex5, aes(x = age,
    y = plasma_glucose_concentration)) + geom_point(aes(x = age,
    y = plasma_glucose_concentration, color = predicted_classes_0.5)) +
    scale_color_manual(values = c("#000000", "#ff0000")) +
    labs(title = "Assigment 5, r=0.5", colour = "Diabetes")
plot_assigment3_q5

missclassification_ex5 <- missclass(diabetes_data_ex5$diabetes,
    diabetes_data_ex5$predicted_classes_0.5)
cat("The missclassification is", missclassification_ex5)
```