503 HW1

Xinyang Qi (qxinyang)

2020/1/28

Question 1

(a) Flexible method is better. Because flexible method has less constraints and can be more closer to data. In addition, when sample size is extremely large, it is less possible to overfit.

(b) Inflexible method is better. Because flexible method is more likely to overfit, when the sample size is small.

(c) Flexible method is better. Because when the predictors and response is highly non-linear, their relationship may be complex. Inflexible method has more constraints and will lead to poor estimate of true relation.

(d) Inflexible method is better. Because when variance is very high, there may be a lot of outliers. Flexible method is more sensitive to outliers and may lead to poor performance.

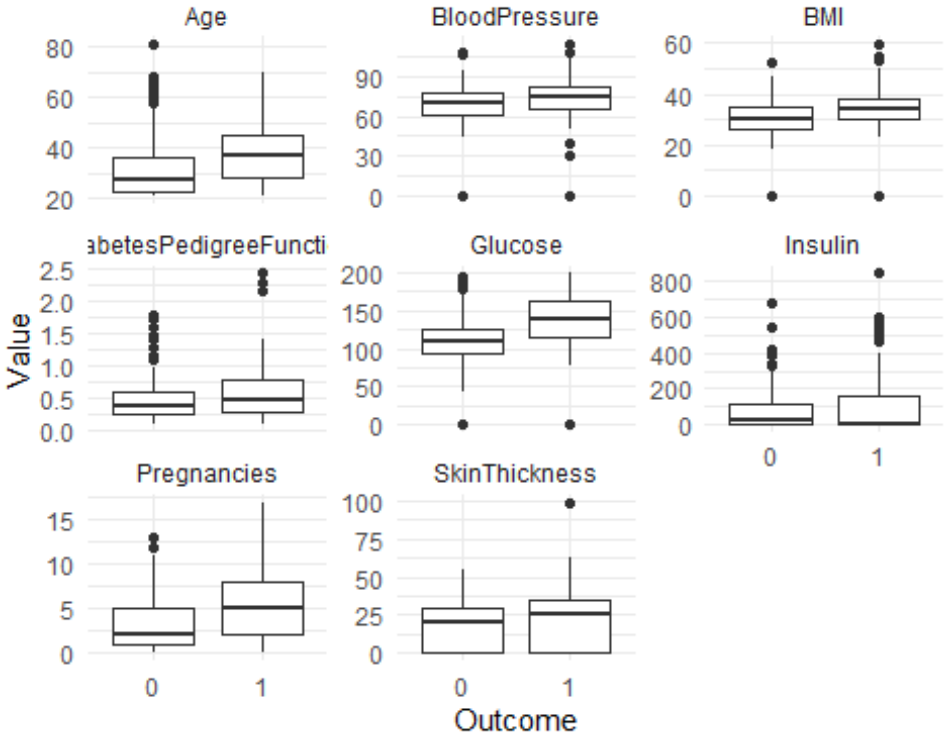
Question 2

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(class)
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.2
```

Firstly, we load the data and explore it.

```
train = read.csv("diabetes_train.csv")
test = read.csv("diabetes_test.csv")
explore = gather(train, key = "Variable", value = "Value", -c("Outcome
"))
explore$Outcome = as.factor(explore$Outcome)
ggplot(explore) + geom_boxplot(aes(x = Outcome, y = Value)) +
  facet_wrap(.~Variable, scales = "free_y") + theme_minimal()
```



From the graph above, we can find that there are many zeros in “Glucose”, “BloodPressure”, “SkinThickness”, “Insulin” and “BMI”. This is unusual pattern and very likely to be missing values in fact. So we need to drop them before we step into data analyse.

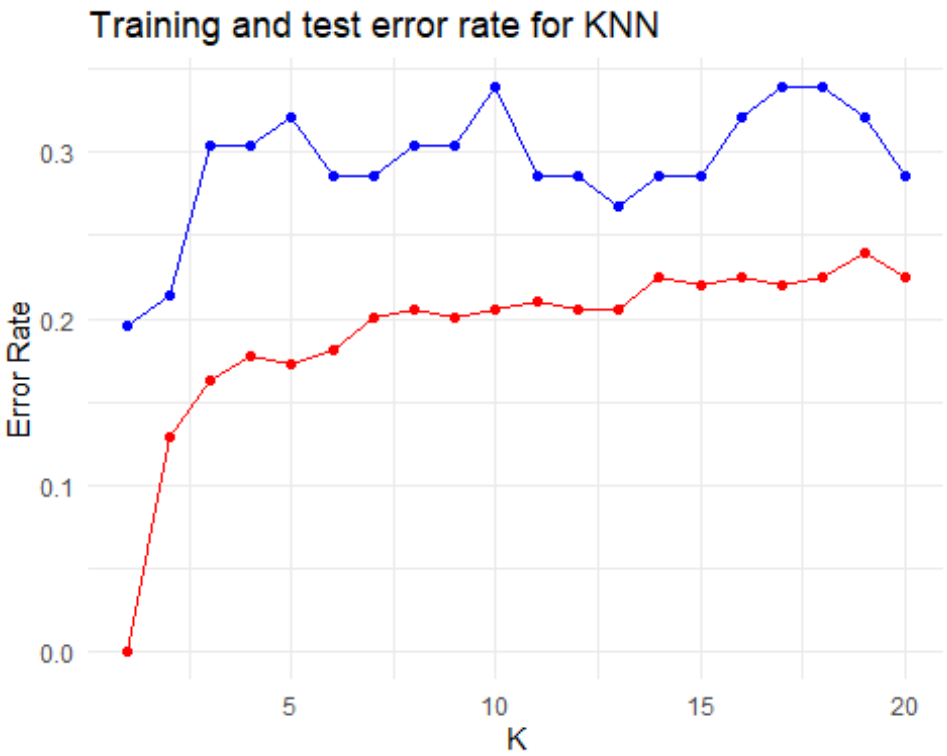
```
train$BloodPressure[train$BloodPressure == 0] = NA
train$Glucose[train$Glucose == 0] = NA
train$BMI[train$BMI == 0] = NA
train$SkinThickness[train$SkinThickness == 0] = NA
train$Insulin[train$Insulin == 0] = NA
test$BloodPressure[test$BloodPressure == 0] = NA
test$Glucose[test$Glucose == 0] = NA
test$BMI[test$BMI == 0] = NA
test$SkinThickness[test$SkinThickness == 0] = NA
test$Insulin[test$Insulin == 0] = NA
train = drop_na(train)
test = drop_na(test)

train_label = train %>% .$Outcome
train_x = train %>% select(-c("Outcome"))
test_label = test %>% .$Outcome
test_x = test %>% select(-c("Outcome"))

mean_train = colMeans(train_x)
std_train = sqrt(diag(var(train_x)))
# training data
```

```r
train_x = scale(train_x, center = mean_train, scale = std_train)
# test data
test_x = scale(test_x, center = mean_train, scale = std_train)

k_range = 1:20
train_error = c()
test_error = c()
set.seed(1)
for (i in 1:20) {
  pred_train = knn(train_x, train_x, train_label, k = i)
  train_error[i] = mean(pred_train != train_label)
  pred_test = knn(train_x, test_x, train_label, k = i)
  test_error[i] = mean(pred_test != test_label)
}
errors = data.frame(train_error, test_error, k_range)
ggplot(errors, aes(x = k_range)) +
  geom_line(aes(y = train_error), col = "red") + geom_point(aes(y = tra
in_error), col = "red") +
  geom_line(aes(y = test_error), col = "blue") + geom_point(aes(y = tes
t_error), col = "blue") +
  ylab("Error Rate") + xlab("K") + ggtitle("Training and test error rat
e for KNN") + theme_minimal()
```



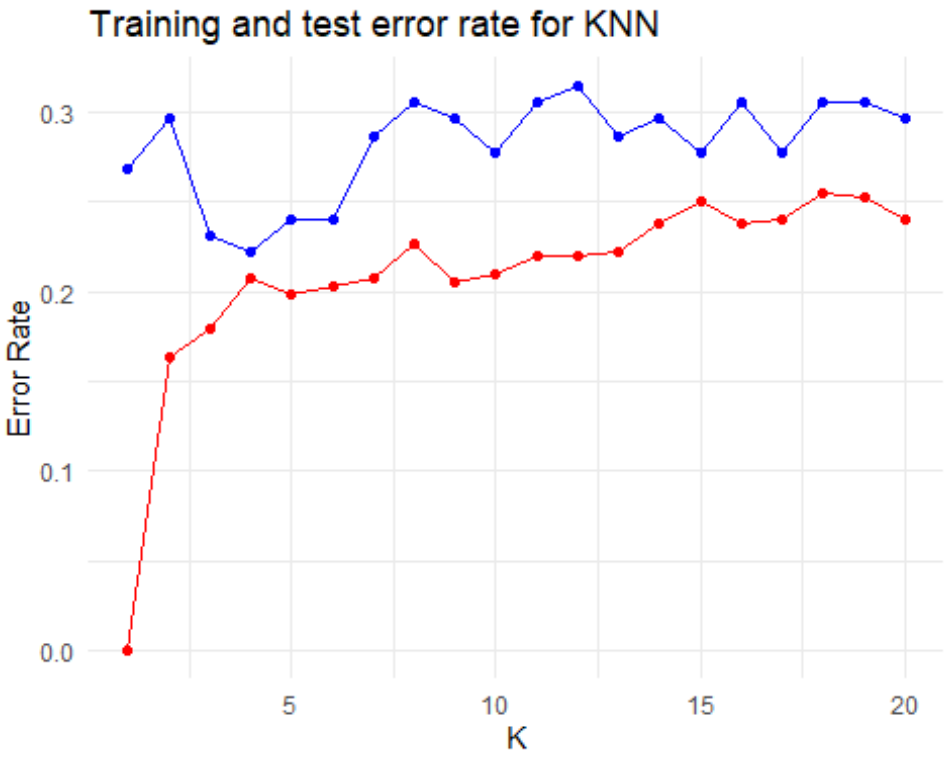
From the output above, we can find that when $k = 1$, the train error and test error are both the smallest. However, as we know, the smaller the k is, the more complex the model is. And a complex model is more likely to overfit. So I think $k = 2$ is

optimal. Because when $k = 2$, the test error is nearly smallest and can reduce the probability of overfitting.

In addition, in order to avoid dropping so many rows from the data set, I try to replace the zeros with medians.

```
train = read.csv("diabetes_train.csv")
test = read.csv("diabetes_test.csv")
train$BloodPressure[train$BloodPressure == 0] =
  median(train$BloodPressure[train$BloodPressure != 0])
train$Glucose[train$Glucose == 0] =
  median(train$Glucose[train$Glucose != 0])
train$BMI[train$BMI == 0] = median(train$BMI[train$BMI != 0])
train$SkinThickness[train$SkinThickness == 0] =
  median(train$SkinThickness[train$SkinThickness != 0])
train$Insulin[train$Insulin == 0] =
  median(train$Insulin[train$Insulin != 0])
test$BloodPressure[test$BloodPressure == 0] =
  median(train$BloodPressure[train$BloodPressure != 0])
test$Glucose[test$Glucose == 0] =
  median(train$Glucose[train$Glucose != 0])
test$BMI[test$BMI == 0] = median(train$BMI[train$BMI != 0])
test$SkinThickness[test$SkinThickness == 0] =
  median(train$SkinThickness[train$SkinThickness != 0])
test$Insulin[test$Insulin == 0] =
  median(train$Insulin[train$Insulin != 0])
train_label = train %>% .$Outcome
train_x = train %>% select(-c("Outcome"))
test_label = test %>% .$Outcome
test_x = test %>% select(-c("Outcome"))
mean_train = colMeans(train_x)
std_train = sqrt(diag(var(train_x)))
# training data
train_x = scale(train_x, center = mean_train, scale = std_train)
# test data
test_x = scale(test_x, center = mean_train, scale = std_train)
set.seed(1)
for (i in 1:20) {
  pred_train = knn(train_x, train_x, train_label, k = i)
  train_error[i] = mean(pred_train != train_label)
  pred_test = knn(train_x, test_x, train_label, k = i)
  test_error[i] = mean(pred_test != test_label)
}
errors = data.frame(train_error, test_error, k_range)
ggplot(errors, aes(x = k_range)) +
  geom_line(aes(y = train_error), col = "red") + geom_point(aes(y = tra
in_error), col = "red") +
  geom_line(aes(y = test_error), col = "blue") + geom_point(aes(y = tes
t_error), col = "blue") +
```

```
  ylab("Error Rate") + xlab("K") + ggtitle("Training and test error rat
e for KNN") + theme_minimal()
```



From the graph above, we can find that when $k = 4$, the test error is smallest. So we should choose $k = 4$.

Compared with dropping all the zeros, I think this method is better. Although the test errors of two methods are almost the same, the model of $k = 2$ is more complex and has smaller sample size. Thus it is more likely to overfit.