**Technical University of Dortmund** Summer Semester 2023
**Data Science** Exercise Sheet 03

Group member name(s): Ahmed Arian Sajid, Sabrina Sultana, Sharmin Ahmad
Group member UID(s): 235061, 235062, 230239

# Advanced Statistical Learning

## Answer to Exercise 1.a

Consider the hypothesis spaces defined by the rule

$$\mathcal{H}_\rangle := \{f(x) = \sum_{k=0}^{i} \theta_k x^k | \theta \in \mathbb{R}^{i+1}\} \tag{1}$$

For fitting model using the hypothesis spaces are,

$$H_1 = f(x) = \theta_0 + \theta_1 x \tag{2}$$

$$H_2 = f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \tag{3}$$

$$H_3 = f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \tag{4}$$

$$H_5 = f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 \tag{5}$$

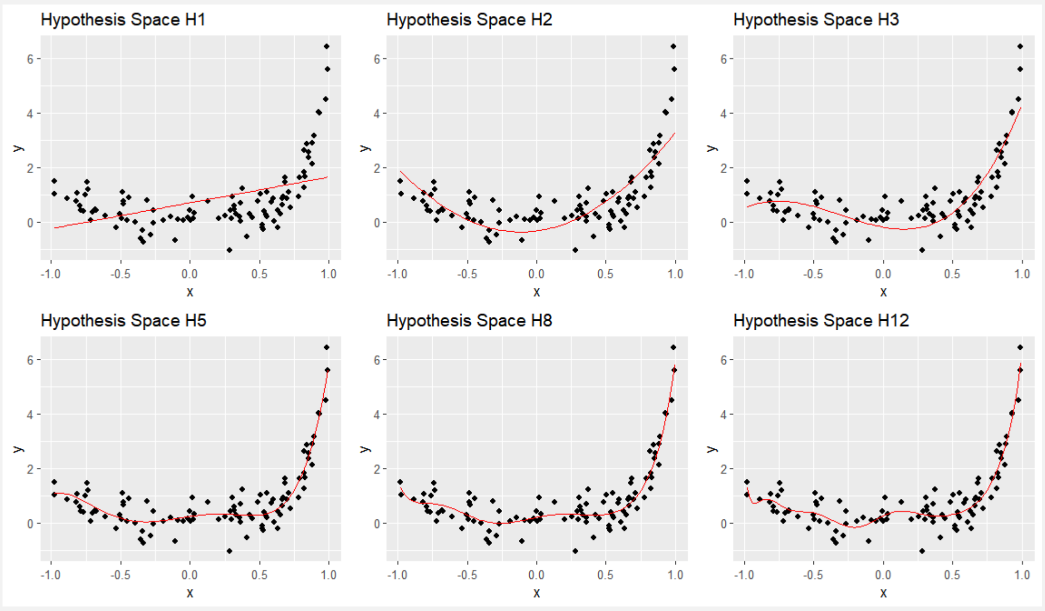
$$H_8 = f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8 \tag{6}$$

$$H_{12} = f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8 + \theta_9 x^9 + \theta_{10} x^{10} + \theta_{11} x^{11} + \theta_{12} x^{12} \tag{7}$$

visualization of the data and fitted models below.

```
mydt <- read.csv("C:/Users/Sabrina/Downloads/fitting.csv")

hyp1 <- lm(y ~ x, data = mydt)
hyp2 <- lm(y ~ x + I(x^2),data = mydt)
hyp3 <- lm(y ~ x + I(x^2) + I(x^3), data = mydt)
hyp5 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5), data = mydt)
hyp8 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7)
+ I(x^8), data = mydt)
hyp12 <-lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7)
+ I(x^8) +  I(x^9) + I(x^10) + I(x^11) + I(x^12), data = mydt)

fn1 <- function(x) coef(hyp1)[1] + coef(hyp1)[2] * x
fn2 <- function(x) coef(hyp2)[1] + coef(hyp2)[2] * x + coef(hyp2)[3] * x^2
fn3 <- function(x) coef(hyp3)[1] + coef(hyp3)[2] * x + coef(hyp3)[3] * x^2 +
coef(hyp3)[4] * x^3
fn5 <- function(x) coef(hyp5)[1] + coef(hyp5)[2] * x + coef(hyp5)[3] * x^2 +
coef(hyp5)[4] * x^3 + coef(hyp5)[5] * x^4 + coef(hyp5)[6] * x^5
fn8 <- function(x) coef(hyp8)[1] + coef(hyp8)[2] * x + coef(hyp8)[3] * x^2 +
coef(hyp8)[4] * x^3 + coef(hyp8)[5] * x^4 + coef(hyp8)[6] * x^5 +
coef(hyp8)[7] * x^6 + coef(hyp8)[8] * x^7 + coef(hyp8)[9] * x^8
fn12 <- function(x) coef(hyp12)[1] + coef(hyp12)[2] * x + coef(hyp12)[3] * x^2 +
coef(hyp12)[4] * x^3 + coef(hyp12)[5] * x^4 + coef(hyp12)[6] * x^5 +
coef(hyp12)[7] * x^6 + coef(hyp12)[8] * x^7 + coef(hyp12)[9] * x^8 +
coef(hyp12)[10]*x^9 + coef(hyp12)[11]*x^10 +  coef(hyp12)[12]* x^11 +
coef(hyp12)[13]*x^12

library(ggplot2)
```

```
val <- ggplot(aes(y = y, x = x), data = mydt) + geom_point()

h1 <- val + geom_function(fun = fn1, col = "red") + ggtitle("Hypothesis Space H1")
h2 <- val + geom_function(fun = fn2, col = "red") + ggtitle("Hypothesis Space H2")
h3 <- val + geom_function(fun = fn3, col = "red") + ggtitle("Hypothesis Space H3")
h5 <- val + geom_function(fun = fn5, col = "red") + ggtitle("Hypothesis Space H5")
h8 <- val + geom_function(fun = fn8, col = "red") + ggtitle("Hypothesis Space H8")
h12 <- val + geom_function(fun = fn12, col = "red") + ggtitle("Hypothesis Space H12")
install.packages("gridExtra")
library(gridExtra)
grid.arrange(h1,h2,h3,h5,h8,h12, nrow = 2)
```



### Answer to Exercise 1.b

As we can see, there are 6 hypothesis spaces. Among them $H_1$ is linear regression line and $H_2$ is quadratic function line. These two line hardly touches the data points. These two hypothesis spaces are under fitting.
$H_3$ is better and but this also can be said under fitting .
$H_5$ is considered to be the best model . It represents all the dataset nicely.
$H_12$ and $H_8$ show 'wigglyness' in the data set. So these two hypothesis spaces are considered as over fitted model.

### Answer to Exercise 1.c

Here we first take the data set and fit the models ($H_1, H_2, H_3, H_5, H_8, H_{12}$) and split the data into train dataset and test dataset. Then calculate the error (MSE)

```
mydt <- read.csv("C:/Users/Sabrina/Downloads/test.csv")

train <- mydt[1:80,]
test <- mydt[81:100,]

hyp1 <- lm(y ~ x, data = train)
hyp2 <- lm(y ~ x + I(x^2),data = train)
hyp3 <- lm(y ~ x + I(x^2) + I(x^3), data = train)
```

```
hyp5 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5), data = train)
hyp8 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) +
I(x^7) + I(x^8), data = train)
hyp12 <-lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) +
I(x^7) + I(x^8) + I(x^9) + I(x^10) + I(x^11) + I(x^12), data = train)



fits<- list(hyp1, hyp2, hyp3, hyp5, hyp8, hyp12)
MSE <- lapply(fits, function(fit) sum((predict(fit, newdata = test) - test$y)^2))
MSE <- data.frame(MSE = unlist(MSE), models = c("H1", "H2", "H3", "H5","H8", "H12"))
MSE

           MSE models
            1 25.897546     H1
            2  9.410835     H2
            3  6.807164     H3
            4  4.041775     H5
            5  9.117652     H8
            6 23.569022    H12
```

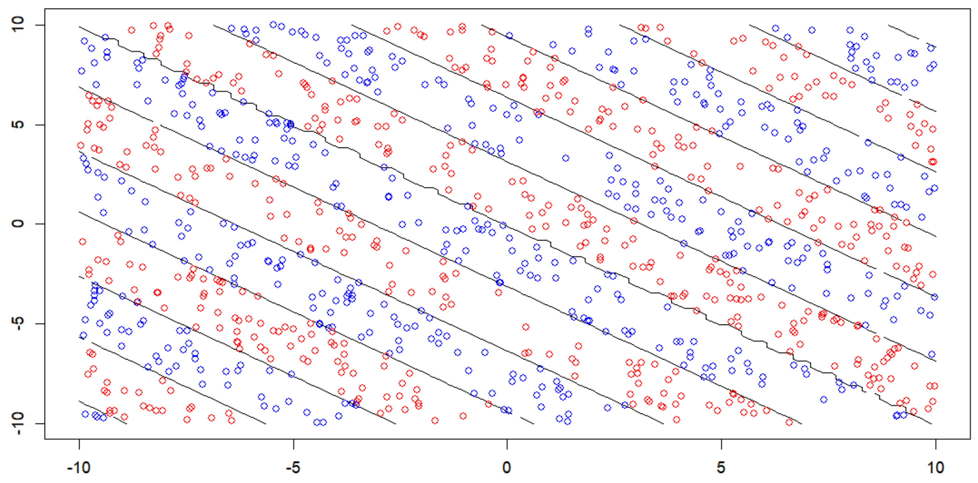
Here, we can see that $H_5$ gives the less error than the other model.

**Answer to Exercise 2**

- (1 point) Plot the true decision boundary for the data situation.

  **Solution:**

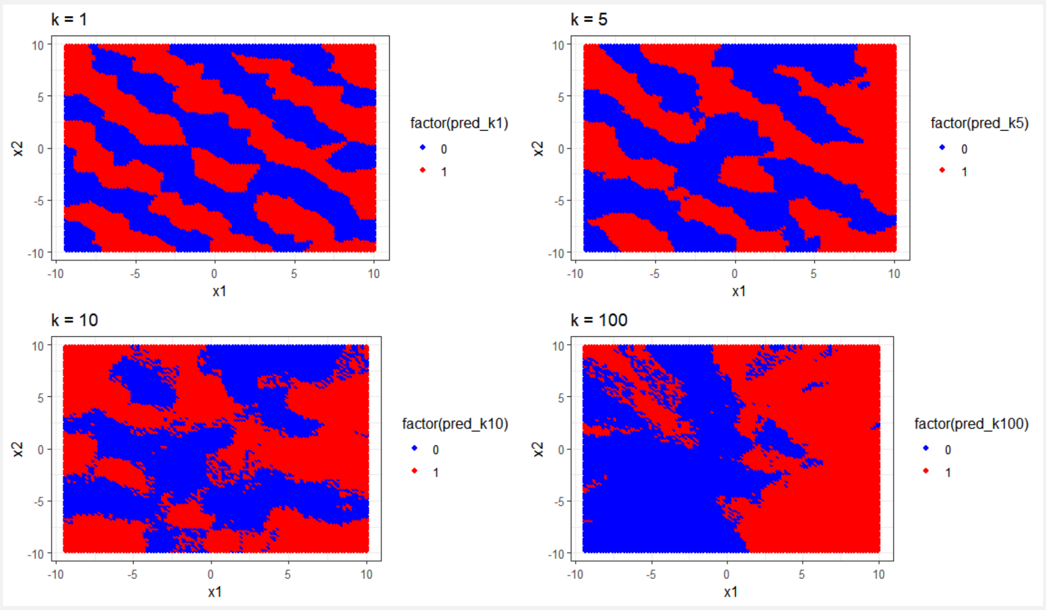
  ```
  n <- 1000
  x1 <- runif(n, -10, 10)
  x2 <- runif(n, -10, 10)
  y <- sin(x1 + x2) < 0

  x1_grid <- seq(-10, 10, length.out = 100)
  x2_grid <- seq(-10, 10, length.out = 100)
  grid <- expand.grid(x1_grid, x2_grid)

  z <- apply(grid, 1, function(p) as.numeric(sin(p[1] + p[2]) < 0))

  contour(x1_grid, x2_grid, matrix(z, nrow = length(x2_grid),
  ncol = length(x1_grid)),levels = 0.5, labels = "", xlim = c(-10, 10),
  ylim = c(-10, 10))
  points(x1[y], x2[y], col = "blue")
  points(x1[!y], x2[!y], col = "red")
  ```



- (1 point) Create an explicit dataset with $n = 250$. Visualize the decision boundary for $k =$ 1,5,10,100. What do you observe?

  **Solution:**

```
n <- 250
x1 <- runif(n, -10, 10)
x2 <- runif(n, -10, 10)
y <- as.numeric(sin(x1 + x2) < 0)
data <- data.frame(x1, x2, y)

x1_range <- seq(min(x1), max(x1), length.out = 100)
x2_range <- seq(min(x2), max(x2), length.out = 100)
grid <- expand.grid(x1 = x1_range, x2 = x2_range)

library(class)
k_values <- c(1, 5, 10, 100)
pred_list <- lapply(k_values, function(k) {
  knn(train = data[, c("x1", "x2")], test = grid, cl = data$y, k = k)
})
names(pred_list) <- paste0("k", k_values)

library(ggplot2)
grid$pred_k1 <- pred_list[["k1"]]
grid$pred_k5 <- pred_list[["k5"]]
grid$pred_k10 <- pred_list[["k10"]]
grid$pred_k100 <- pred_list[["k100"]]
p1 <- ggplot(grid, aes(x = x1, y = x2, color = factor(pred_k1))) +
  geom_point() +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw() +
  ggtitle("k = 1")
p2 <- ggplot(grid, aes(x = x1, y = x2, color = factor(pred_k5))) +
  geom_point() +
  scale_color_manual(values = c("blue", "red")) +
      theme_bw() +
      ggtitle("k = 5")
    p3 <- ggplot(grid, aes(x = x1, y = x2, color = factor(pred_k10))) +
      geom_point() +
      scale_color_manual(values = c("blue", "red")) +
      theme_bw() +
      ggtitle("k = 10")
    p4 <- ggplot(grid, aes(x = x1, y = x2, color = factor(pred_k100))) +
      geom_point() +
      scale_color_manual(values = c("blue", "red")) +
      theme_bw() +
      ggtitle("k = 100")
    library(gridExtra)
    grid.arrange(p1, p2, p3, p4, ncol = 2)
```



we can observe that,
For k=1, the decision boundary is very complex and follows the contours of the data.
As k increases, the decision boundary becomes smoother. For k =10 this approximates the true decision boundary much better.
k=100, the decision boundary becomes too simple and smooth. So, it will overfit.

- (2 points) For k 2 f1; 2; 3; :::; 100g estimate the error rate of the model. At first, do so using the training error rate, i.e., the error the model has by predicting the original training data. At second, do so by using subsampling: Split the data into 2/3 training and 1/3 test data, train the model on the train data, test it on the test date. Repeat 100 times and use the mean value as an estimator for the error.

  **Solution:**

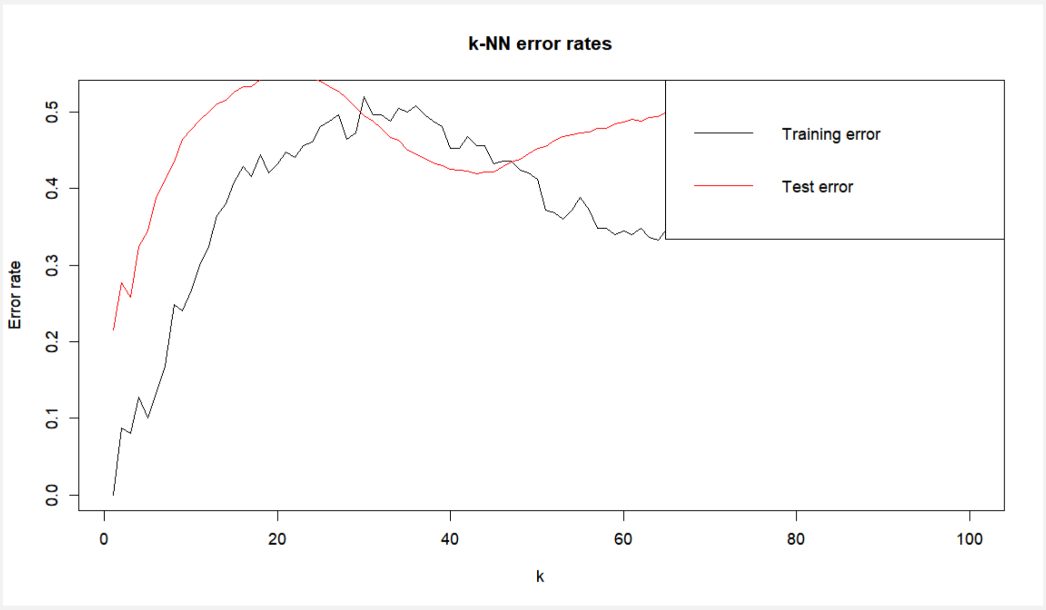
```
        n <- 250
        x1 <- runif(n, -10, 10)
        x2 <- runif(n, -10, 10)
        y <- sin(x1 + x2) < 0

        error_rate <- function(y_true, y_pred) { mean(y_true != y_pred) }
        train_errors <- rep(NA, 100)
```

```
test_errors <- rep(NA, 100)

for (k in 1:100) {
   knn_model <- knn(train = cbind(x1, x2), test = cbind(x1, x2),
   cl = y, k = k)
  train_errors[k] <- error_rate(y, knn_model)
  test_errors_rep <- rep(NA, 100)
  for (i in 1:100) {
    set.seed(i)
    train_idx <- sample(1:n, size = round(2*n/3), replace = FALSE)
    test_idx <- setdiff(1:n, train_idx)

    knn_model <- knn(train = cbind(x1[train_idx], x2[train_idx]),
                     test = cbind(x1[test_idx], x2[test_idx]),
                     cl = y[train_idx], k = k)
    test_errors_rep[i] <- error_rate(y[test_idx], knn_model)
  }
  test_errors[k] <- mean(test_errors_rep)
}
```

- Visualize both error rates with respect to k. What differences do you observe? How can you explain them?

  <u>**Solution:**</u>

```
plot(1:100, train_errors, type = "l", xlab = "k",
ylab = "Error rate", main = "k-NN error rates")
lines(1:100, test_errors, col = "red")
legend("topright", legend = c("Training error", "Test error"),
lty = 1, col = c("black", "red"))
```



We can observe that the training error rate is always lower than the test error rate for all values of k. As we splited the data into 2/3 training and 1/3 test data, it will perform better in training data.

And as the value of k increases, both the training and test error rates first increases upto a peak point and then tend to decrease with larger value of k.
Test error rates fluctuates in the middle range value of k.