Technical University of Dortmund Summer Semester 2023
Data Science Exercise Sheet 02
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

Answer to Exercise 1.b

As we can see, the data-generating process of y has the square of $x1$; we need to include the effect of x_1^2 to our hypothesis space $\mathcal{H}$, which would result in a better fit of logistic regression. So, a new Hypothesis space would be:

$$\mathcal{H} := \{f(x) = x^T\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 | \theta \in \mathbb{R}^4\}$$

Answer to Exercise 1.c

To find the decision boundary, at first, we have to calculate the coefficients using the Hypothesis space from 1.b and the given data-generating process, we are using the following R Code to find the coefficients:

```
set.seed(123)
x1 <- runif(500, -1, 1)
x2 <- runif(500, -1, 1)
x3<-x1^2
y <- ifelse(x1^2 + rnorm(500, 0, 0.2) < x2, 0, 1)
fit<-glm(y ~ x1 + x2 + x3, family = "binomial")
(theta <- coef(fit))

(Intercept)          x1          x2          x3
-0.1178140   0.1665535  -8.5038775   8.6178639
```

here given $\pi(x) = 0.5$
therefore,

$$\begin{aligned}
&\frac{1}{1+\exp(-x^T\theta)} = 0.5 \\
&\implies 1+\exp(-x^T\theta) = 2 \\
&\implies \exp(-x^T\theta) = 1 \\
&\implies (-x^T\theta) = \log(1) \\
&\implies x^T\theta = 0 \\
&\implies \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 = 0 \\
&\implies \theta_0 + \theta_1 x_1 + \theta_3 x_1^2 = -\theta_2 x_2 \\
&\implies -\frac{\theta_0}{\theta 2} - \frac{\theta_1}{\theta_2} x_1 - \frac{\theta_3}{\theta_2} x_1^2 = x_2
\end{aligned}$$

using the above values of coefficients, we get:

$$\begin{aligned}
&-\frac{-0.1178}{-8.5039} - \frac{0.1666}{-8.5039} x_1 - \frac{8.6179}{-8.5039} x_1^2 = x_2 \\
&\implies -0.014 + 0.02 x_1 + 1.013 x_1^2 = x_2
\end{aligned}$$

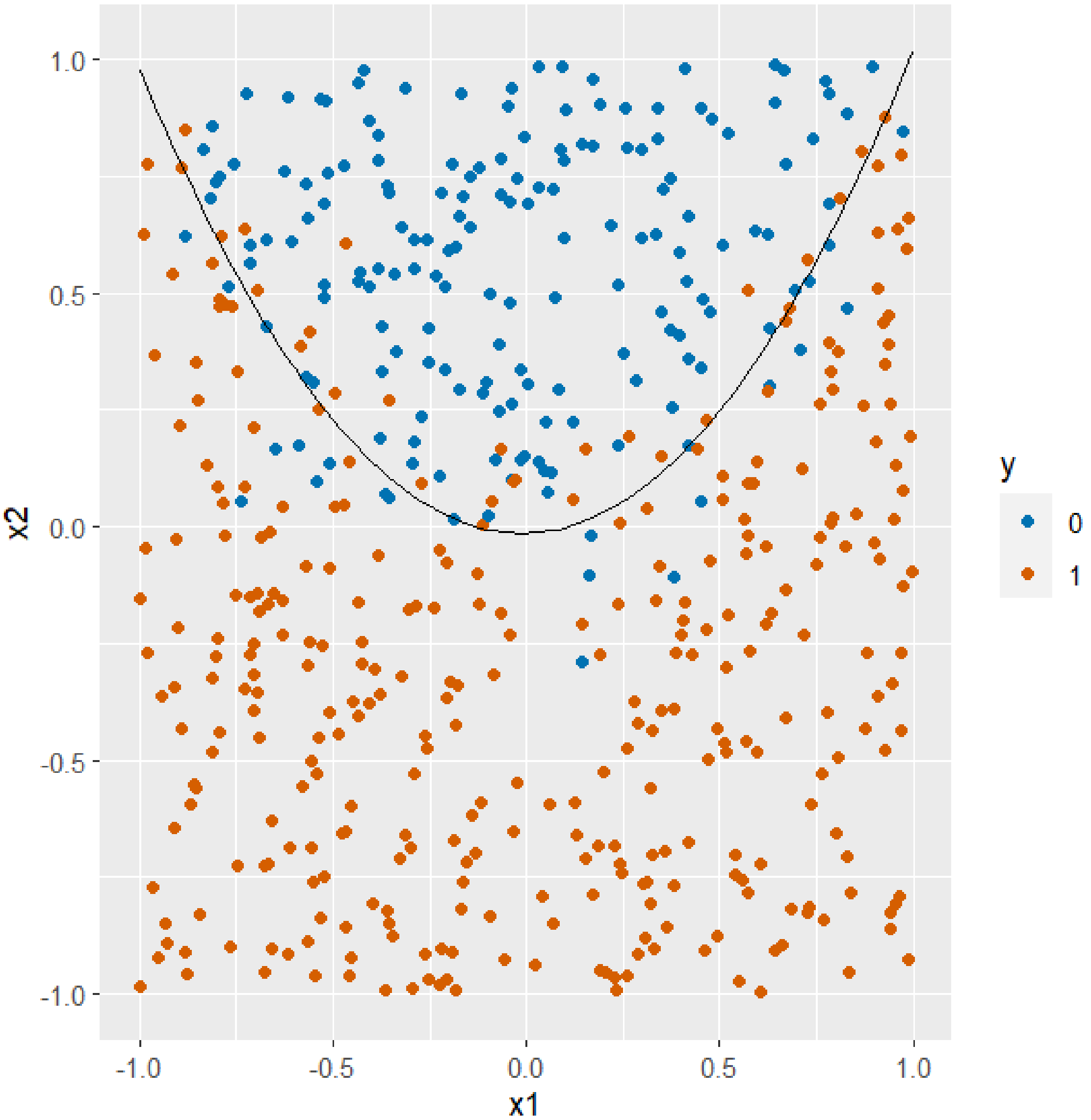
which is the decision boundary.

Answer to Exercise 1.d

We visualize the data and decision boundary found from 1.c using the following R Code:

```
set.seed(123)
x1 <- runif(500, -1, 1)
x2 <- runif(500, -1, 1)
y <- ifelse(x1^2 + rnorm(500, 0, 0.2) < x2, 0, 1)
df <- data.frame(x1 = x1, x2 = x2, y = factor(y))
boundary <- function(x){-0.014 + 0.02*x + 1.013*x^2}

library(ggplot2)
ggplot(df, aes(x = x1, y = x2, color = y)) +
geom_point() +
stat_function(fun = boundary,aes(x = x1,y=x2), color = "black")+
scale_color_manual(values = c("#0072B2", "#D55E00"))
```



Answer to Exercise 2

- (2 point) Write a function that gets a loss-function as its input and that calculates the optimal values 0 and 1. Do so by minimizing the empirical risk induced by the given loss function. As the optimizer, use a simple random search. (Sample some thousand (0, 1) in a suitable search interval, evaluate the empirical risk for all of them and use (0, 1) with the smallest risk value.

  Solution:

```
squared_loss <- function(y_true, y_pred) {
 mean((y_true - y_pred)^2)}

absolute_loss <- function(y_true, y_pred) {
  mean(abs(y_true - y_pred))}

huber_loss <- function(y_true, y_pred, delta = 1.0) {
  error <- y_true - y_pred
```

```
  absolute_error <- abs(error)
  quadratic_error <- error^2
  mask <- absolute_error <= delta
  mean(ifelse(mask, quadratic_error/2.0, delta * (absolute_error-delta/2.0)))
}

calculate_optimal_theta <- function(loss_func)
  {
  data <- read.csv("C:/Users/Sabrina/Downloads/fitting.csv")
  x <- data$x
  y <- data$y

  search_interval <- c(-10.0, 10.0)
  num_samples <- 1000

  best_loss <- Inf
  best_theta <- NULL

  for (i in 1:num_samples) {
    theta <- runif(2, min = search_interval[1], max = search_interval[2])
    y_pred <- theta[1] + theta[2] * x
    loss <- loss_func(y, y_pred)
    if (loss < best_loss) {
      best_loss <- loss
      best_theta <- theta
    }
  }
  return(best_theta)
}
optimal_theta_squared_loss <- calculate_optimal_theta(squared_loss)
print("Optimal Theta (Squared Loss):")
print(optimal_theta_squared_loss)

optimal_theta_absolute_loss <- calculate_optimal_theta(absolute_loss)
print("Optimal Theta (Absolute Loss):")
print(optimal_theta_absolute_loss)

optimal_theta_huber_loss <- calculate_optimal_theta(huber_loss)
print("Optimal Theta (Huber Loss):")
print(optimal_theta_huber_loss)
```

Optimal Values for θ_0 and θ_1:

```
> print(optimal_theta_squared_loss)
[1] 0.2959996 1.1404840

> print(optimal_theta_absolute_loss)
[1] 0.5366411 0.1776845

> print(optimal_theta_huber_loss)
[1] 0.4298614 0.2177412
```

- (1 point) Instantiate the following risk functions: L1-loss, L2-Loss, Quantile-Loss with $\alpha \in 0.05, 0.95$, Huber-Loss with $\delta \in 1, 2$ and Epsilon-Loss with $\varepsilon \in 0.5, 3$.

  Solution:

```
  data <- read.csv("C:/Users/Sabrina/Downloads/fitting.csv")
x <- data$x
y <- data$y

l1_loss <- function(theta0, theta1) {
  y_pred <- theta0 + theta1 * x
  return(mean(abs(y - y_pred))) }

l2_loss <- function(theta0, theta1) {
  y_pred <- theta0 + theta1 * x
  return(mean((y - y_pred)^2)) }

quantile_loss <- function(theta0, theta1, alpha) {
  y_pred <- theta0 + theta1 * x
  residual <- y - y_pred
  return(mean(ifelse(residual >= 0, alpha * residual, (alpha - 1) * residual)))
  }

huber_loss <- function(theta0, theta1, delta) {
  y_pred <- theta0 + theta1 * x
  absolute_error <- abs(y - y_pred)
  quadratic_error <- (y - y_pred)^2
  mask <- absolute_error <= delta
  return(mean(ifelse(mask, quadratic_error/2, delta*(absolute_error - delta/2))))
}

epsilon_loss <- function(theta0, theta1, epsilon) {
  y_pred <- theta0 + theta1 * x
  absolute_error <- abs(y - y_pred)
  return(mean(ifelse(absolute_error <= epsilon, absolute_error^2 / 2,
  epsilon * (absolute_error - epsilon / 2))))
}


calculate_optimal_model <- function(loss_func) {
  search_interval <- c(-10.0, 10.0)
  num_samples <- 1000

  best_loss <- Inf
  best_theta <- NULL

  for (i in 1:num_samples) {
    theta <- runif(2, min = search_interval[1], max = search_interval[2])
    loss <- loss_func(theta[1], theta[2])

    if (loss < best_loss) {
      best_loss <- loss
      best_theta <- theta
    }
  }
  return(best_theta)
```

```
}

optimal_theta_l1_loss <- calculate_optimal_model(l1_loss)
optimal_theta_l2_loss <- calculate_optimal_model(l2_loss)
optimal_theta_quantile_05_loss <- calculate_optimal_model
(function(theta0, theta1)quantile_loss(theta0, theta1, 0.05))
optimal_theta_quantile_95_loss <- calculate_optimal_model
(function(theta0,theta1)quantile_loss(theta0,theta1,0.95))
optimal_theta_huber_1_loss <-calculate_optimal_model
(function(theta0,theta1)huber_loss(theta0,theta1,1))
optimal_theta_huber_2_loss <- calculate_optimal_model
(function(theta0,theta1)huber_loss(theta0,theta1,2))
optimal_theta_epsilon_05_loss <- calculate_optimal_model
(function(theta0,theta1)epsilon_loss(theta0,theta1,0.5))
optimal_theta_epsilon_3_loss  <- calculate_optimal_model
(function(theta0,theta1)epsilon_loss(theta0,theta1,3))
```

- (2 points) Calculate the optimal (θ_0, θ_1) for all above loss functions. Visualize them with the data.Compare the values / regressions lines. What do you observe?
  Solution:

```
    > print(optimal_theta_l1_loss)
[1] 0.5502889 0.4817652
> print(optimal_theta_l2_loss)
[1] 0.2839772 1.1548081
> print(optimal_theta_quantile_05_loss)
[1] -0.2767742  0.2637494
> print(optimal_theta_quantile_95_loss)
[1] 2.788339 1.413778
> print(optimal_theta_huber_1_loss)
[1] 0.4335429 0.7449206
> print(optimal_theta_huber_2_loss)
[1] 0.7202876 0.6412855
> print(optimal_theta_epsilon_05_loss)
[1] 0.4538465 0.6615618
> print(optimal_theta_epsilon_3_loss)
[1] 0.770620 0.452974
```

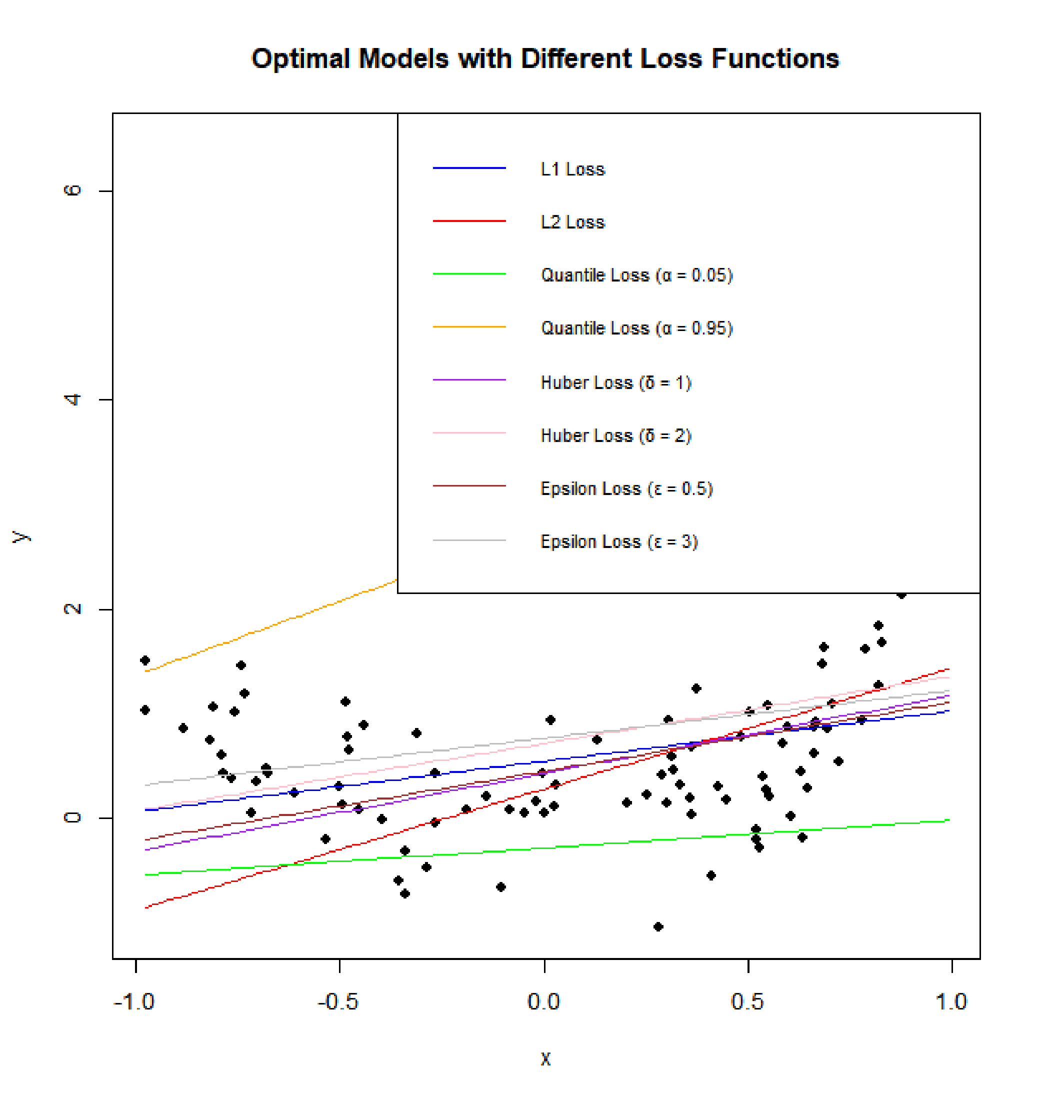
Visualize them with data. Compare the values/regression line.

```
plot(x, y, main = "Optimal Models with Different Loss Functions",
xlab = "x", ylab = "y", pch = 16)
plot_reg_line <- function(theta0, theta1, color) {
  x_range <- range(x)
  x_vals <- seq(x_range[1], x_range[2], length.out = 100)
  y_vals <- theta0 + theta1 * x_vals
  lines(x_vals, y_vals, col = color)
}

plot_reg_line(optimal_theta_l1_loss[1], optimal_theta_l1_loss[2], "blue")
plot_reg_line(optimal_theta_l2_loss[1], optimal_theta_l2_loss[2], "red")
plot_reg_line(optimal_theta_quantile_05_loss[1],
    optimal_theta_quantile_05_loss[2], "green")
plot_reg_line(optimal_theta_quantile_95_loss[1],
    optimal_theta_quantile_95_loss[2], "orange")
```

```
plot_reg_line(optimal_theta_huber_1_loss[1],
    optimal_theta_huber_1_loss[2], "purple")
plot_reg_line(optimal_theta_huber_2_loss[1],
    optimal_theta_huber_2_loss[2], "pink")
plot_reg_line(optimal_theta_epsilon_05_loss[1],
    optimal_theta_epsilon_05_loss[2], "brown")
plot_reg_line(optimal_theta_epsilon_3_loss[1],
    optimal_theta_epsilon_3_loss[2], "gray")

legend("topright", legend = c("L1 Loss", "L2 Loss",
"Quantile Loss ($\alpha$ = 0.05)", "Quantile Loss ($\alpha$ = 0.95)",
"Huber Loss ($\delta $ = 1)", "Huber Loss ($\delta $ = 2)",
"Epsilon Loss ($\varepsilon$)", "Epsilon Loss ($\varepsilon$ = 3)"),
col = c("blue", "red", "green", "orange", "purple", "pink", "brown", "gray"),
lty = 1, cex = 0.8)
```



Obsrvations:

- The L1 loss function (blue line) tends to produce a regression line that is less affected by outliers, as it focuses on the absolute differences between the predicted and true values.

- The L2 loss function (red line) produces a regression line that minimizes the mean squared error and is sensitive to large errors.
- The quantile loss functions with $\alpha = 0.05$ (green line) and $\alpha = 0.95$ (orange line) generate regression lines that capture the lower and upper quantiles of the data, respectively.
- The Huber loss functions with $\delta = 1$ (purple line) and $\delta = 2$ (pink line) produce regression lines that balance between the L1 and L2 losses, providing a compromise between robustness and sensitivity to outliers. i
- The epsilon loss function with $\varepsilon = 0.5$ (brown line) places more emphasis on smaller errors, while the epsilon loss function with $\varepsilon = 3$ (gray line) allows for larger errors.