# Load required libraries

library(datasets) #Get the data

library(stats) # Computations

library(class) # binary outcome

library(caret) # Cross-validation

### 1. Gradient Descent for Linear Regression using mtcars

data(mtcars)

View(mtcars)

X <- cbind(1, mtcars$wt)

y <- mtcars$mpg

theta <- matrix(0, nrow = 2)

alpha <- 0.01

num\_iter <- 1000

m <- length(y)

gradient <- function(X, y, theta) {

(1 / m) \* t(X) %\*% (X %\*% theta - y)

}

for (i in 1:num\_iter) {

theta <- theta - alpha \* gradient(X, y, theta)

}

cat('Gradient Descent Linear Regression Coefficients:', theta, '\\n')

### 2. Gradient Descent for Logistic Regression using iris

data(iris)

View(iris)

iris\_bin <- iris[iris$Species != 'setosa', ]

iris\_bin$y <- ifelse(iris\_bin$Species == 'versicolor', 1, 0)

X <- as.matrix(cbind(1, iris\_bin[, c("Petal.Length", "Petal.Width")]))

y <- iris\_bin$y

theta <- matrix(0, nrow = ncol(X))

sigmoid <- function(z) 1 / (1 + exp(-z))

cost <- function(X, y, theta) {

m <- length(y)

h <- sigmoid(X %\*% theta)

- (1 / m) \* sum(y \* log(h) + (1 - y) \* log(1 - h))

}

grad <- function(X, y, theta) {

m <- length(y)

(1 / m) \* t(X) %\*% (sigmoid(X %\*% theta) - y)

}

for (i in 1:1000) {

theta <- theta - 0.01 \* grad(X, y, theta)

}

cat('Gradient Descent Logistic Regression Coefficients:', theta, '\\n')

### 3. Gradient Descent on Simulated Polynomial Regression

set.seed(123)

x <- runif(100, -3, 3)

y <- 2 + 3 \* x - 1.5 \* x^2 + rnorm(100, 0, 1)

X <- cbind(1, x, x^2)

theta <- matrix(0, nrow = 3)

for (i in 1:1000) {

theta <- theta - 0.01 \* gradient(X, y, theta)

}

cat('Gradient Descent Polynomial Coefficients:', theta, '\\n')

### 4. Nelder-Mead for minimizing MSE in Linear Regression (mtcars)

mse <- function(params) {

intercept <- params[1]

slope <- params[2]

y\_pred <- intercept + slope \* mtcars$wt

mean((mtcars$mpg - y\_pred)^2)

}

opt <- optim(c(0, 0), mse, method = "Nelder-Mead")

cat('Nelder-Mead Linear Regression Coefficients:', opt$par, '\\n')

### 5. Nelder-Mead for Logistic Regression Log Loss (iris)

log\_loss <- function(params) {

b0 <- params[1]

b1 <- params[2]

b2 <- params[3]

z <- b0 + b1 \* iris\_bin$Petal.Length + b2 \* iris\_bin$Petal.Width

p <- sigmoid(z)

-mean(iris\_bin$y \* log(p + 1e-9) + (1 - iris\_bin$y) \* log(1 - p + 1e-9))

}

opt2 <- optim(c(0, 0, 0), log\_loss, method = "Nelder-Mead")

cat('Nelder-Mead Logistic Regression Coefficients:', opt2$par, '\\n')

### 6. Nelder-Mead to tune k in kNN (simulated binary classification)

set.seed(123)

train\_idx <- createDataPartition(iris\_bin$y, p = 0.7, list = FALSE)

train <- iris\_bin[train\_idx, ]

test <- iris\_bin[-train\_idx, ]

cv\_error <- function(k\_val) {

k <- round(k\_val[1])

if (k < 1) return(Inf)

pred <- knn(train[, 1:4], test[, 1:4], cl = train$y, k = k)

mean(pred != test$y)

}

opt\_k <- optim(c(3), cv\_error, method = "Nelder-Mead")

cat('Optimal k for kNN:', round(opt\_k$par), '\\n')