library(stats) library(caret) library(glmnet)

df <- read.csv(“tecator.csv”) n <- dim(df)[1] source(“split_data.R”) sets <- split_data(data_frame = df, k = 2, seed = 12345, distributions = c(1,1)) train <- sets[[1]] test <- sets[[2]]

#Channels as features train_features <- train[1:(ncol(train)-3)] test_features <- test[1:(ncol(test)-3)]

#fitting linear regression model lin_model <- lm(train$Fat ~. +0, data = train_features) # +0 takes away the intercept

#predictions on the training data and test data pred_train <- predict(lin_model, train_features) pred_test <- predict(lin_model, test_features)

Training and test errors

linear_mse_train <- mean((train$Fat - pred_{t}rain)^{2}) linear_{m}se_{t}est < -mean((test$Fat - pred_test) ^ 2)

summary(lin_model)

#==part1==================================

#Lasso regression X_train <- as.matrix(train_features) X_test <- as.matrix(test_features)

Fit LASSO regression model on the training data

lasso_model_train <- cv.glmnet(X_train, train$Fat , alpha = 1) # Setting alpha = 1 for LASSO

LASSO Cost Function

lasso_cost_function <- function(X,y, beta, lambda) { N <- length(y) prediction <- X %*% beta residuals <- y - prediction mse_term <- sum(residuals^2) / (2* N) l1_penalty <- lambda * sum(abs(beta)) cost <- mse_term + l1_penalty return(cost) }

Example usage of the cost function

optimal_lambda <- as.matrix(lasso_model_train$lambda.min) optimal_beta <- coef(lasso_model_train, s = optimal_lambda) #The discrepancy in the dimensions suggests that the optimal_beta vector includes an additional element, #which typically corresponds to the intercept term. #In glmnet models, the intercept is included by default.

coefficients_optimal <- as.matrix(tail(optimal_beta, -1)) # Exclude the first element (intercept)

Ensure that X_train_lasso and optimal_beta have compatible dimensions

if (ncol(X_train) == length(coefficients_optimal)) { cost <- lasso_cost_function(X_train, train$Fat, coefficients_optimal, optimal_lambda) cat("LASSO Cost Function Value:", cost, "") } else { cat("Dimensions of X_train_lasso and optimal_beta are not compatible for matrix multiplication.") }

#=================================part2==

Fit LASSO regression model on the training data

summary(lasso_model_train)

Plot the coefficient paths

plot(lasso_model_train$glmnet.fit, "lambda", main = "LASSO Coefficient Paths", xlab = "Log(lambda)", ylab = "Coefficients")

Identify the value of lambda for a model with only three features

desired_num_features <- 3 lambda_index <- which.min(lasso_model_train$glmnet.fit$dev.ratio > desired_num_features / ncol(X_train))

Add a vertical line at the selected lambda value

abline(v = log(lasso_model_train$glmnet.fit$lambda[lambda_index]), col = "red", lty = 2)

#===part3======================

Fit Ridge regression model

ridge_model_train <- cv.glmnet(X_train, train$Fat , alpha = 0) # Setting alpha = 0 for ridge

Plotting regression coefficients vs. log(λ)

plot(ridge_model_train$glmnet.fit, "lambda", main = "Ridge Coefficient Paths", xlab = "Log(lambda)", ylab = "Coefficients")

#===============================part4==================================

#cross-validation to compute optimal lasso model cv_lasso_model <- cv.glmnet(X_test, test$Fat , alpha = 1)

plot(cv_lasso_model)

Optimal λ and number of variables

opt_lambda <- cv_lasso_model$lambda.minn_variables_chosen < -sum(coef(cv_lasso_model, s = optimal_lambda)! = 0)lambda_four < -cv_lasso_model$cvm[which.min(abs(log(cv_lasso_model$lambda) - (-4)))]lambda_optimal < -cv_lasso_model$cvm[which.min(abs(log(cv_lasso_model$lambda) - opt_lambda))]

test_pred_lasso <- predict(cv_lasso_model, newx = X_test, s = opt_lambda) plot(test$Fat, test_pred_lasso, main = "Actual vs. Predicted (LASSO)", xlab = "Actual", ylab = "Predicted", col = c("blue", "red")) legend("topleft", legend = c("Actual", "Predicted"), col = c("blue", "red"), pch = 1)

#===============================part5==================================