

# assignment\_2

2024-10-16

## Contents

<b>Problem 1. Regression</b>	<b>1</b>
a. Dataset splitting . . . . .	1
(i) Original Model . . . . .	2
(ii). Dummy encoding . . . . .	4
b. Repeating the procedure 200 times . . . . .	9

## Problem 1. Regression

```
data <- read.csv("qsar_aquatic_toxicity.csv", sep = ";", header = FALSE)
names(data) <- c(
  "TPSA",
  "SAacc",
  "H050",
  "MLOGP",
  "RDCHI",
  "GATS1p",
  "nN",
  "C040",
  "LC50"
)

head(data)
```

```
##      TPSA    SAacc H050 MLOGP RDCHI GATS1p nN C040  LC50
## 1    0.00    0.000   0 2.419 1.225 0.667  0    0 3.740
## 2    0.00    0.000   0 2.638 1.401 0.632  0    0 4.330
## 3    9.23   11.000   0 5.799 2.930 0.486  0    0 7.019
## 4    9.23   11.000   0 5.453 2.887 0.495  0    0 6.723
## 5    9.23   11.000   0 4.068 2.758 0.695  0    0 5.979
## 6 215.34 327.629   3 0.189 4.677 1.333  0    4 6.064
```

### a. Dataset splitting

Split the data into a training and a test set, with approximately 2/3 and 1/3 of the observations, respectively.

```
# Use 70% of dataset as training set and remaining 30% as testing set
sample <- sample.split(data$LC50, SplitRatio = 0.7)
train <- subset(data, sample == TRUE)
test <- subset(data, sample == FALSE)
```

```
cat("Dimension of Training Set:", paste(dim(train), collapse = "x"), "\nDimension of Test Set:", paste(
```

```
## Dimension of Training Set: 382x9
## Dimension of Test Set: 164x9
```

### (i) Original Model

Model each of them directly as a linear effect

```
train_i = train
test_i = test
```

```
# Fit linear regression model on training data
model <- lm(LC50 ~ ., data=train_i)

summary(model)
```

```
##
## Call:
## lm(formula = LC50 ~ ., data = train_i)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.3523 -0.7669 -0.1279  0.5547  4.9504
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.599932   0.308508   8.427 7.78e-16 ***
## TPSA         0.026443   0.003078   8.592 2.38e-16 ***
## SAacc        -0.015252   0.002473  -6.168 1.80e-09 ***
## H050         0.096989   0.071890   1.349  0.17812
## MLOGP        0.462983   0.075450   6.136 2.16e-09 ***
## RDCHI        0.505242   0.163742   3.086  0.00218 **
## GATS1p       -0.479643   0.189033  -2.537  0.01158 *
## nN           -0.228086   0.057635  -3.957 9.07e-05 ***
## C040         -0.025262   0.112645  -0.224  0.82267
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.213 on 373 degrees of freedom
## Multiple R-squared:  0.4685, Adjusted R-squared:  0.4571
## F-statistic: 41.1 on 8 and 373 DF, p-value: < 2.2e-16
```

```
# Predict on training and test datasets
pred_train <- predict(model, newdata=train_i)
pred_test <- predict(model, newdata=test_i)
```

```

# Adding predictions columns to the datasets
train_i$predicted_LC50 <- pred_train
test_i$predicted_LC50 <- pred_test

# Evaluate model: calculate MSE, RMSE, and R-squared for training and test sets
mse_train <- mean((train_i$LC50 - train_i$predicted_LC50)^2)
rmse_train <- sqrt(mse_train)
r2_train <- 1 - (sum((train_i$LC50 - train_i$predicted_LC50)^2) / sum((train_i$LC50 - mean(train_i$LC50))^2))

mse_test <- mean((test_i$LC50 - test_i$predicted_LC50)^2)
rmse_test <- sqrt(mse_test)
r2_test <- 1 - (sum((test_i$LC50 - test_i$predicted_LC50)^2) / sum((test_i$LC50 - mean(test_i$LC50))^2))

cat(paste0(
  "Training Metrics:\n",
  "MSE (Train): ", mse_train, "\n",
  "RMSE (Train): ", rmse_train, "\n",
  "R-squared (Train): ", r2_train, "\n\n",

  "Test Metrics:\n",
  "MSE (Test): ", mse_test, "\n",
  "RMSE (Test): ", rmse_test, "\n",
  "R-squared (Test): ", r2_test, "\n"
))

```

```

## Training Metrics:
## MSE (Train): 1.43766516622101
## RMSE (Train): 1.19902675792537
## R-squared (Train): 0.468501834252579
##
## Test Metrics:
## MSE (Test): 1.40331663034852
## RMSE (Test): 1.1846166596619
## R-squared (Test): 0.510894263179344

```

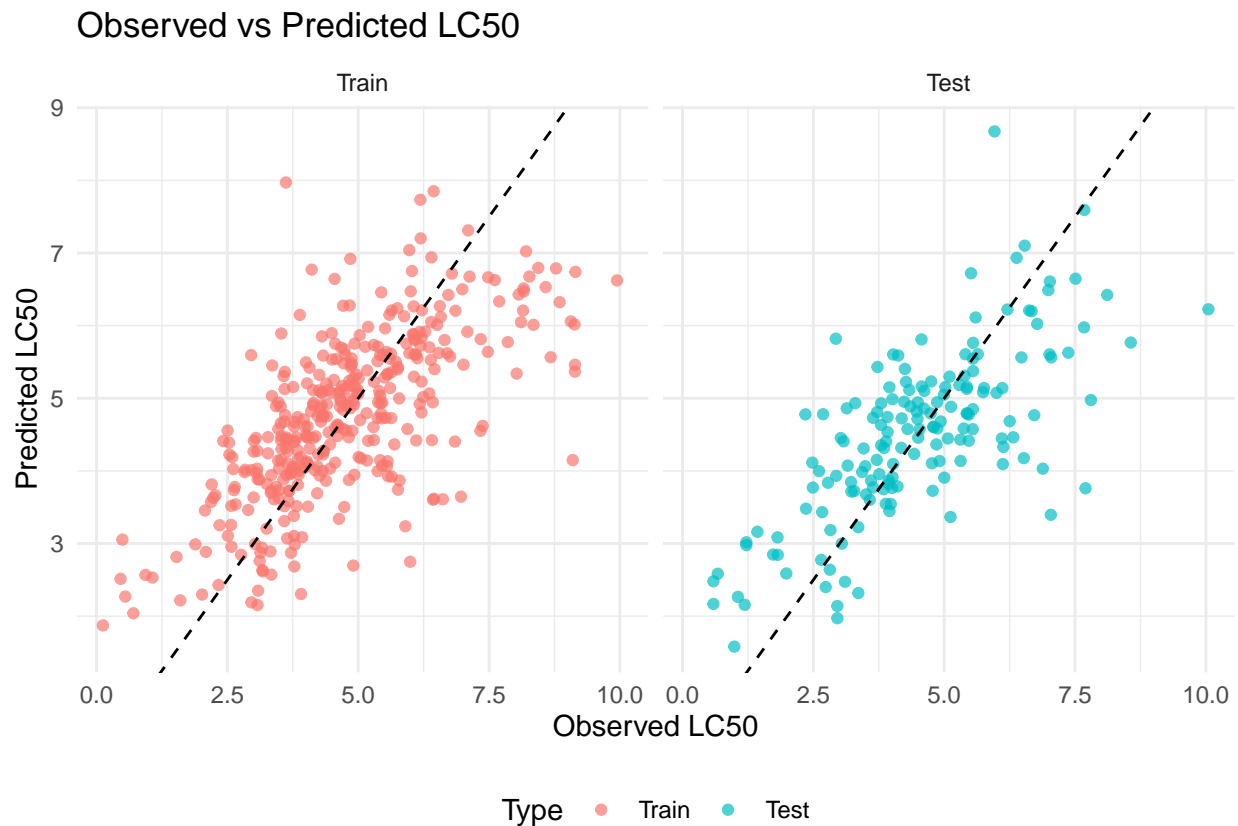
```

# Combine data for plotting
train_i$Type <- 'Train'
test_i$Type <- 'Test'
combined_data <- rbind(train_i, test_i)

combined_data$Type <- factor(combined_data$Type, levels = c('Train', 'Test'))

# Plotting observed vs predicted LC50 values
ggplot(combined_data, aes(x = LC50, y = predicted_LC50, color = Type)) +
  geom_point(alpha = 0.7) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  labs(title = "Observed vs Predicted LC50", x = "Observed LC50", y = "Predicted LC50") +
  theme_minimal() +
  facet_wrap(~Type) +
  theme(legend.position = "bottom")

```



## (ii). Dummy encoding

Transform 3 count variables (H050, nN, C040) using a 0/1 dummy encoding where 0 represents absence of the specific atom and 1 represents presence of the specific atoms.

```
# To make sure we use the same split in (i)
train_ii = train
test_ii = test

# Transform 3 count variables (H050, nN, C040) into 0/1 in train and test datasets

train_ii$H050 <- ifelse(train_ii$H050 > 0, 1, 0)
train_ii$nN <- ifelse(train_ii$nN > 0, 1, 0)
train_ii$C040 <- ifelse(train_ii$C040 > 0, 1, 0)

test_ii$H050 <- ifelse(test_ii$H050 > 0, 1, 0)
test_ii$nN <- ifelse(test_ii$nN > 0, 1, 0)
test_ii$C040 <- ifelse(test_ii$C040 > 0, 1, 0)

head(train_ii)
```

```
##      TPSA    SAacc H050 MLOGP RDCHI GATS1p nN C040  LC50
## 1    0.00    0.000   0 2.419 1.225  0.667  0   0 3.740
## 2    0.00    0.000   0 2.638 1.401  0.632  0   0 4.330
```

```
## 4    9.23  11.000    0 5.453 2.887  0.495  0    0 6.723
## 5    9.23  11.000    0 4.068 2.758  0.695  0    0 5.979
## 6 215.34 327.629    1 0.189 4.677  1.333  0    1 6.064
## 7    9.23  11.000    0 2.723 2.321  1.165  0    0 7.337
```

```
# Fit linear regression model on transformed training data
```

```
model_transform_dummy <- lm(LC50 ~ ., data = train_ii)
```

```
summary(model_transform_dummy)
```

```
##
## Call:
## lm(formula = LC50 ~ ., data = train_ii)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.2165 -0.8150 -0.1579  0.5693  5.0409
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.882543   0.315742   9.129 < 2e-16 ***
## TPSA         0.021496   0.003117   6.896 2.29e-11 ***
## SAacc       -0.011371   0.002107  -5.397 1.21e-07 ***
## H050        -0.251052   0.153688  -1.634  0.10320
## MLOGP        0.438238   0.075632   5.794 1.46e-08 ***
## RDCHI        0.451885   0.167133   2.704  0.00717 **
## GATS1p       -0.543773   0.185734  -2.928  0.00362 **
## nN          -0.053328   0.147166  -0.362  0.71728
## C040        -0.070479   0.157156  -0.448  0.65408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.234 on 373 degrees of freedom
## Multiple R-squared:  0.4499, Adjusted R-squared:  0.4381
## F-statistic: 38.14 on 8 and 373 DF, p-value: < 2.2e-16
```

```
# Predict on training and test datasets
```

```
pred_train_transform_dummy <- predict(model, newdata=train_ii)
```

```
pred_test_transform_dummy <- predict(model, newdata=test_ii)
```

```
# Adding predictions columns to the datasets
```

```
train_ii$predicted_LC50 <- pred_train_transform_dummy
```

```
test_ii$predicted_LC50 <- pred_test_transform_dummy
```

```
# Evaluate model: calculate MSE, RMSE, and R-squared for training and test sets
```

```
mse_train_transform_dummy <- mean((train_ii$LC50 - train_ii$predicted_LC50)^2)
```

```
rmse_train_transform_dummy <- sqrt(mse_train_transform_dummy)
```

```
r2_train_transform_dummy <- 1 - (sum((train_ii$LC50 - train_ii$predicted_LC50)^2) / sum((train_ii$LC50 -
```

```
mse_test_transform_dummy <- mean((test_ii$LC50 - test_ii$predicted_LC50)^2)
```

```
rmse_test_transform_dummy <- sqrt(mse_test_transform_dummy)
```

```
r2_test_transform_dummy <- 1 - (sum((test_ii$LC50 - test_ii$predicted_LC50)^2) / sum((test_ii$LC50 - me
```

```

cat(paste0(
  "Training Metrics:\n",
  "MSE (Train): ", mse_train_transform_dummy, "\n",
  "RMSE (Train): ", rmse_train_transform_dummy, "\n",
  "R-squared (Train): ", r2_train_transform_dummy, "\n\n",

  "Test Metrics:\n",
  "MSE (Test): ", mse_test_transform_dummy, "\n",
  "RMSE (Test): ", rmse_test_transform_dummy, "\n",
  "R-squared (Test): ", r2_test_transform_dummy, "\n"
))

```

```

## Training Metrics:
## MSE (Train): 1.52097492292195
## RMSE (Train): 1.23327812066944
## R-squared (Train): 0.437702602334203
##
## Test Metrics:
## MSE (Test): 1.44890775251117
## RMSE (Test): 1.20370584135459
## R-squared (Test): 0.495004136236074

```

```

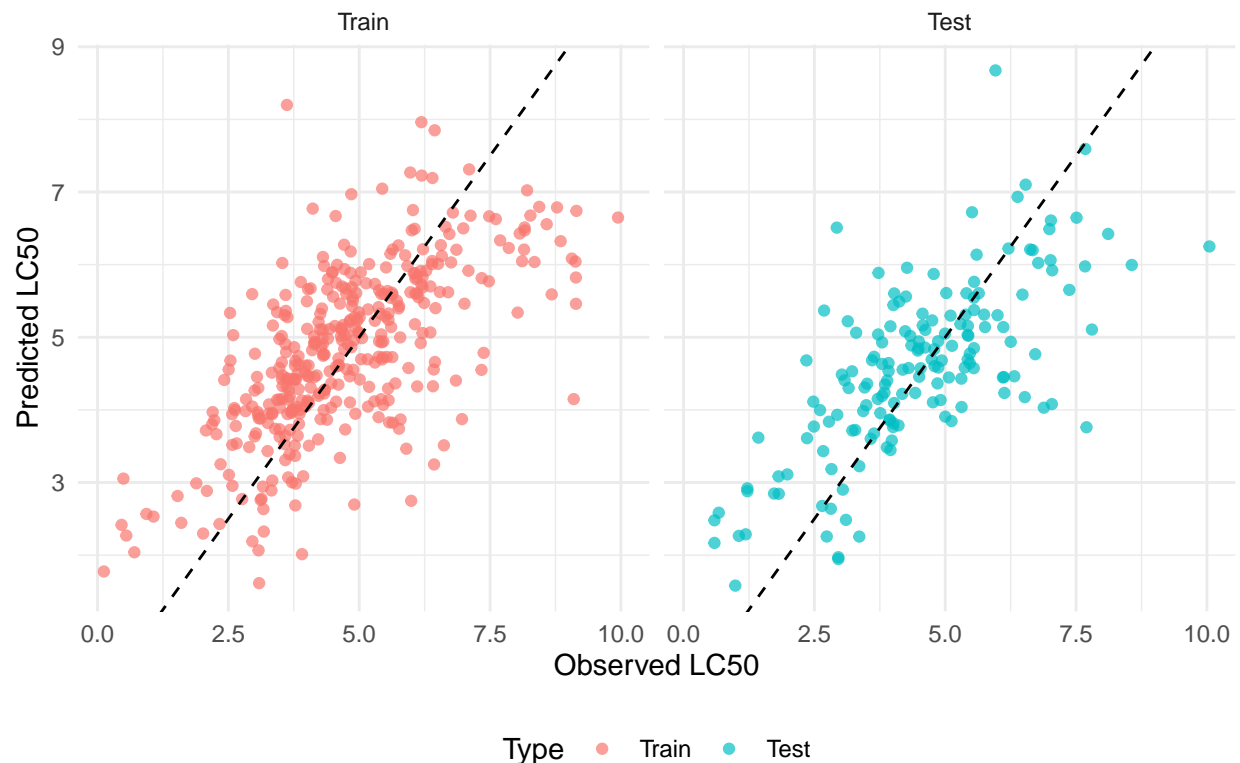
# Combine data for plotting
train_ii$Type <- 'Train'
test_ii$Type <- 'Test'
combined_data <- rbind(train_ii, test_ii)

combined_data$Type <- factor(combined_data$Type, levels = c('Train', 'Test'))

# Plotting observed vs predicted LC50 values
ggplot(combined_data, aes(x = LC50, y = predicted_LC50, color = Type)) +
  geom_point(alpha = 0.7) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  labs(title = "Dummy Encoding: Observed vs Predicted LC50", x = "Observed LC50", y = "Predicted LC50")
  theme_minimal() +
  facet_wrap(~Type) +
  theme(legend.position = "bottom")

```

## Dummy Encoding: Observed vs Predicted LC50



```
# Prepare combined data
train_combined <- train_i[, c("LC50", "predicted_LC50")]
train_combined$Method <- 'Original'
train_combined$Type <- 'Train'
train_ii_combined <- train_ii[, c("LC50", "predicted_LC50")]
train_ii_combined$Method <- 'Dummy'
train_ii_combined$Type <- 'Train'
train_combined_all <- rbind(train_combined, train_ii_combined)

test_combined <- test_i[, c("LC50", "predicted_LC50")]
test_combined$Method <- 'Original'
test_combined$Type <- 'Test'
test_ii_combined <- test_ii[, c("LC50", "predicted_LC50")]
test_ii_combined$Method <- 'Dummy'
test_ii_combined$Type <- 'Test'
test_combined_all <- rbind(test_combined, test_ii_combined)

# Convert 'Method' and 'Type' to factors
train_combined_all$Method <- factor(train_combined_all$Method, levels = c('Original', 'Dummy'))
test_combined_all$Method <- factor(test_combined_all$Method, levels = c('Original', 'Dummy'))

# Function to draw regression lines
add_regression_lines <- function(df, original_model, dummy_model) {
  ggplot(df, aes(x = LC50, y = predicted_LC50, color = Method)) +
    geom_point(alpha = 0.7) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
```

```

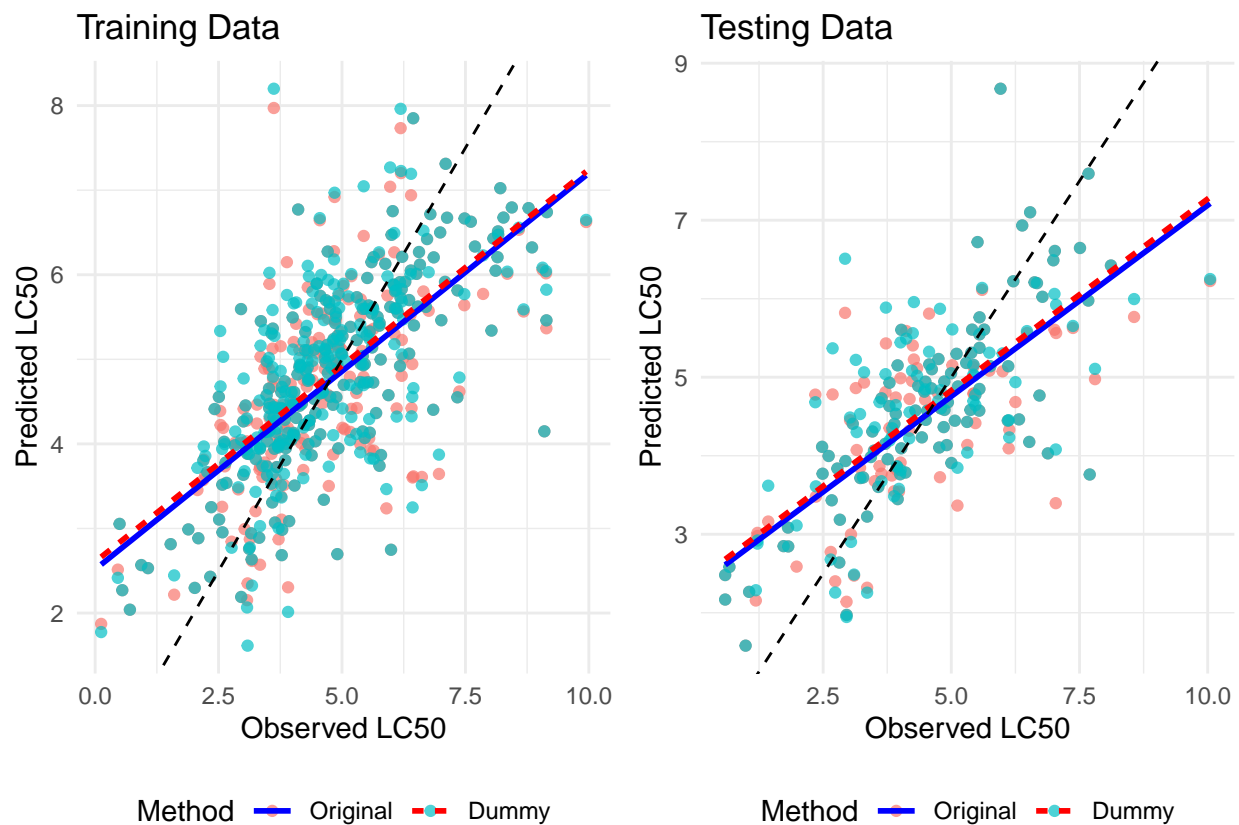
    aes(linetype = Method),
    data = df[df$Method == 'Original', ],
    color = 'blue') +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE,
    aes(linetype = Method),
    data = df[df$Method == 'Dummy', ],
    color = 'red') +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  labs(x = "Observed LC50", y = "Predicted LC50", title = df$Type[1]) +
  theme_minimal() +
  theme(legend.position = "bottom")
}

# Plot training data with both regression lines
train_plot <- add_regression_lines(train_combined_all, model, model_transform_dummy)
train_plot <- train_plot + labs(title = "Training Data")

# Plot testing data with both regression lines
test_plot <- add_regression_lines(test_combined_all, model, model_transform_dummy)
test_plot <- test_plot + labs(title = "Testing Data")

# Display plots side by side
grid.arrange(train_plot, test_plot, ncol = 2)

```





## b. Repeating the procedure 200 times

Procedure

- Randomly splitting training/test (70%/30%).
- Fit the models with 2 options (i) Original model and (ii) Dummy encoding.
- Record the test errors (MSE/RMSE/ $R^2$ ).

```
# Initialize vectors to store test errors
mse_test_errors_i <- numeric(200)
rmse_test_errors_i <- numeric(200)
r2_test_errors_i <- numeric(200)
mse_test_errors_ii <- numeric(200)
rmse_test_errors_ii <- numeric(200)
r2_test_errors_ii <- numeric(200)

# Repeat the procedure 200 times
set.seed(2)
for (i in 1:200) {
  # Split the data
  sample <- sample.split(data$LC50, SplitRatio = 0.7)
  train <- subset(data, sample == TRUE)
  test <- subset(data, sample == FALSE)

  # Option (i): Original model
  model <- lm(LC50 ~ ., data=train)
  pred_test_i <- predict(model, newdata=test)
  mse_test_i <- mean((test$LC50 - pred_test_i)^2)
  rmse_test_i <- sqrt(mse_test_i)
  r2_test_i <- 1 - (sum((test$LC50 - pred_test_i)^2) / sum((test$LC50 - mean(test$LC50))^2))

  # Option (ii): Dummy encoding
  train$H050 <- ifelse(train$H050 > 0, 1, 0)
  train$nN <- ifelse(train$nN > 0, 1, 0)
  train$C040 <- ifelse(train$C040 > 0, 1, 0)

  test$H050 <- ifelse(test$H050 > 0, 1, 0)
  test$nN <- ifelse(test$nN > 0, 1, 0)
  test$C040 <- ifelse(test$C040 > 0, 1, 0)

  model_ii <- lm(LC50 ~ ., data = train)
  pred_test_ii <- predict(model_ii, newdata = test)
  mse_test_ii <- mean((test$LC50 - pred_test_ii)^2)
  rmse_test_ii <- sqrt(mse_test_ii)
  r2_test_ii <- 1 - (sum((test$LC50 - pred_test_ii)^2) / sum((test$LC50 - mean(test$LC50))^2))

  # Record the test errors
  mse_test_errors_i[i] <- mse_test_i
  rmse_test_errors_i[i] <- rmse_test_i
  r2_test_errors_i[i] <- r2_test_i

  mse_test_errors_ii[i] <- mse_test_ii
  rmse_test_errors_ii[i] <- rmse_test_ii
  r2_test_errors_ii[i] <- r2_test_ii
}
```

```
rmse_test_errors_ii[i] <- rmse_test_ii
r2_test_errors_ii[i] <- r2_test_ii
}
```

Make a plot that illustrates the empirical distributions of the test error for each modelling option and compare the average test error. What is the point of repeating the experiment in this way before drawing any conclusions? Try to explain why one often obtains, like in this case, a worse result by using option (ii).

Initials insight:

- Method 1: performs better in term of MSE
- Method 2: better in reduce over fitting

```
# Calculate and print average test errors
average_test_error_i <- mean(mse_test_errors_i)
average_rmse_error_i <- mean(rmse_test_errors_i)
average_r2_error_i <- mean(r2_test_errors_i)

average_test_error_ii <- mean(mse_test_errors_ii)
average_rmse_error_ii <- mean(rmse_test_errors_ii)
average_r2_error_ii <- mean(r2_test_errors_ii)

cat(paste0(
  "Average Test Errors (Original Model):\n",
  "MSE: ", average_test_error_i, "\n",
  "RMSE: ", average_rmse_error_i, "\n",
  "R-squared: ", average_r2_error_i, "\n\n",
  "Average Test Errors (Dummy Model):\n",
  "MSE: ", average_test_error_ii, "\n",
  "RMSE: ", average_rmse_error_ii, "\n",
  "R-squared: ", average_r2_error_ii, "\n"
))
```

```
## Average Test Errors (Original Model):
## MSE: 1.47416671253053
## RMSE: 1.2118365144871
## R-squared: 0.461029936280147
##
## Average Test Errors (Dummy Model):
## MSE: 1.52473049238122
## RMSE: 1.23264425343633
## R-squared: 0.442463420670575
```

```
# Create data frames for plotting
errors_df_mse <- data.frame(
  Error = c(mse_test_errors_i, mse_test_errors_ii),
  Metric = 'MSE',
  Model = factor(rep(c("Original", "Dummy"), each = 200))
)
errors_df_rmse <- data.frame(
  Error = c(rmse_test_errors_i, rmse_test_errors_ii),
```

```

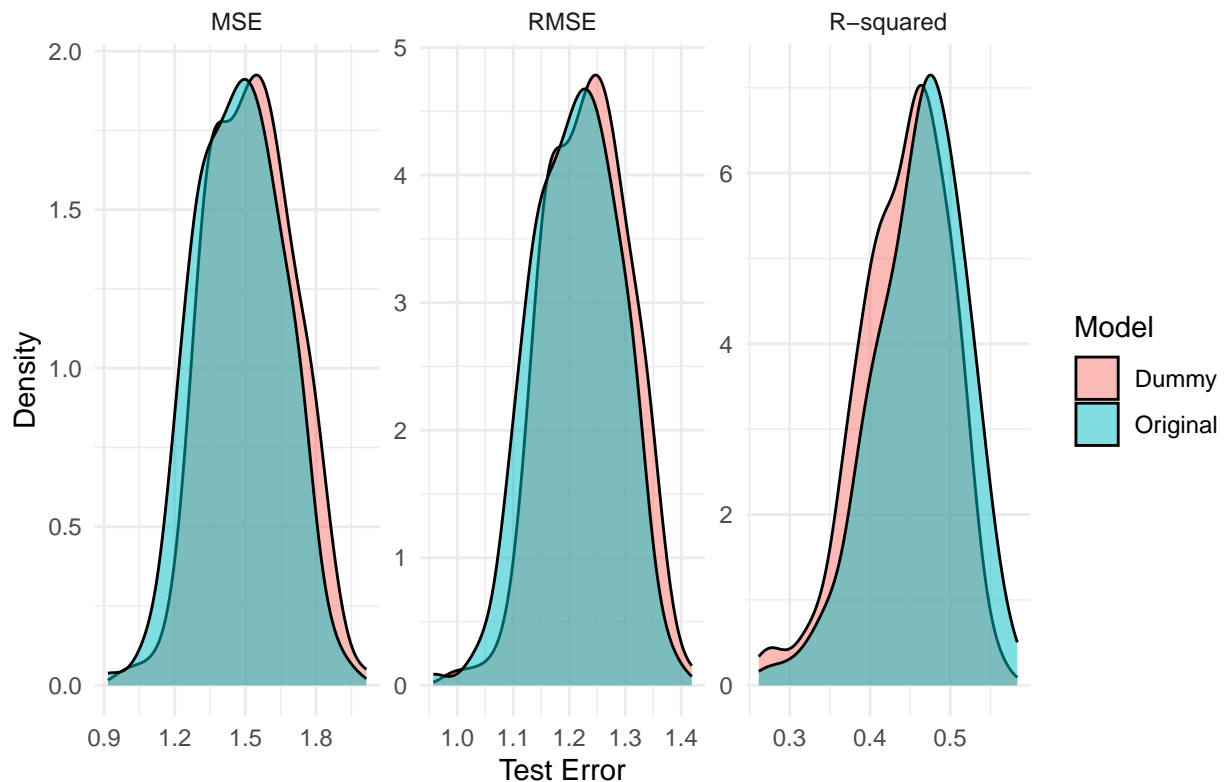
Metric = 'RMSE',
Model = factor(rep(c("Original", "Dummy"), each = 200))
)
errors_df_r2 <- data.frame(
  Error = c(r2_test_errors_i, r2_test_errors_ii),
  Metric = 'R-squared',
  Model = factor(rep(c("Original", "Dummy"), each = 200))
)
errors_df <- rbind(errors_df_mse, errors_df_rmse, errors_df_r2)

# Ensure the 'Metric' factor has the correct level order
errors_df$Metric <- factor(errors_df$Metric, levels = c('MSE', 'RMSE', 'R-squared'))

# Plot the empirical distributions of the test errors
ggplot(errors_df, aes(x = Error, fill = Model)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ Metric, scales = "free") +
  labs(title = "Empirical Distributions of Test Errors", x = "Test Error", y = "Density") +
  theme_minimal()

```

## Empirical Distributions of Test Errors



```

# Plot the empirical distributions of the test errors using boxplots
ggplot(errors_df, aes(x = Metric, y = Error, fill = Model)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Boxplots of Test Errors", x = "Error Metric", y = "Error Value") +
  theme_minimal() +

```

```
theme(legend.position = "top")
```

## Boxplots of Test Errors

