assignment_2

2024-10-16

Contents

Problem 1. Regression

```r
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 spliting

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
set.seed(1)
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) Orginal 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
## -2.8444 -0.7768 -0.1022  0.5521  4.9831
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.596376   0.293581   8.844  < 2e-16 ***
## TPSA         0.028839   0.003333   8.653  < 2e-16 ***
## SAacc       -0.014815   0.002554  -5.800 1.42e-08 ***
## H050         0.037414   0.071245   0.525  0.59980
## MLOGP        0.487698   0.074228   6.570 1.69e-10 ***
## RDCHI        0.496235   0.162219   3.059  0.00238 **
## GATS1p      -0.580937   0.179735  -3.232  0.00134 **
## nN          -0.246680   0.060554  -4.074 5.65e-05 ***
## C040         0.002834   0.092694   0.031  0.97563
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.196 on 373 degrees of freedom
## Multiple R-squared:  0.5072, Adjusted R-squared:  0.4966
## F-statistic: 47.98 on 8 and 373 DF,  p-value: < 2.2e-16
```

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

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

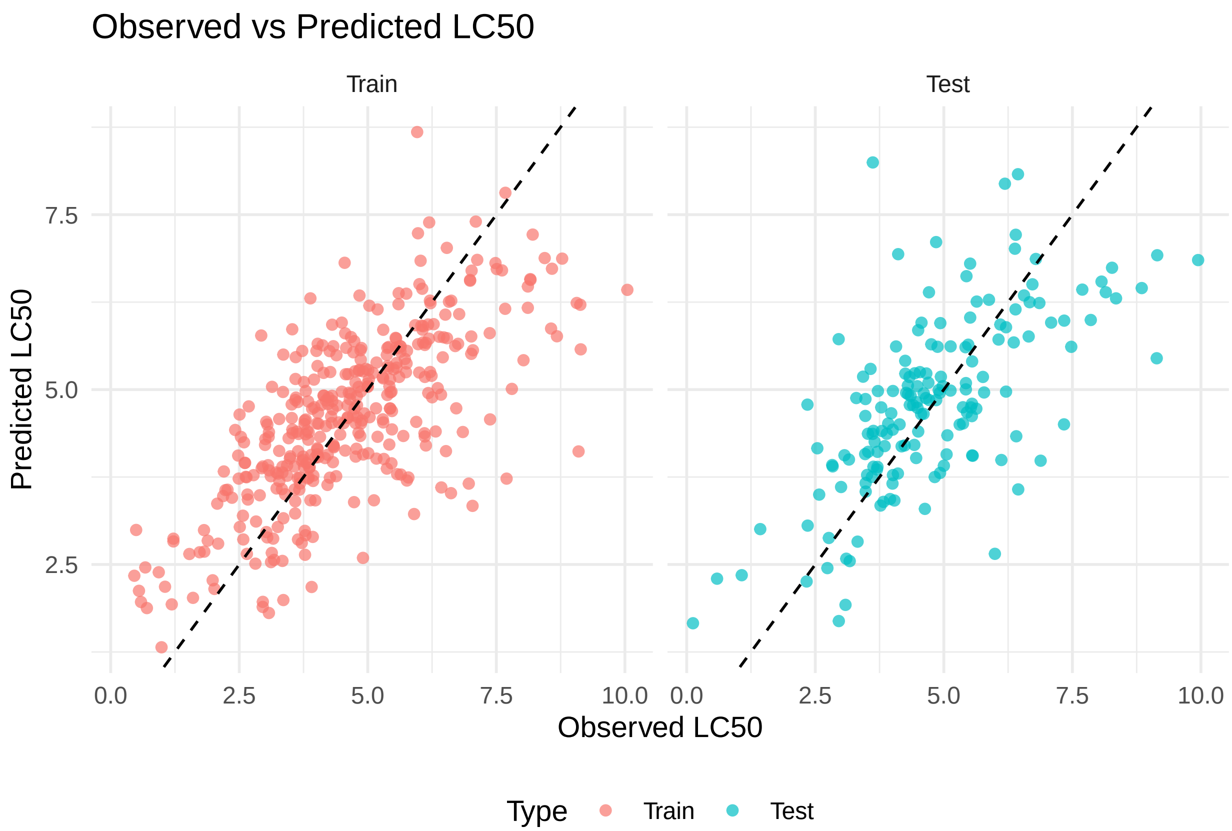
```r
# Evaluate model: calculate MSE, RMSE, and R-squared for training and test sets
mse_train <- mean((train_i$LC50 - train_i$predicted_LC50)^2)
rmse_train <- sqrt(mse_train)
r2_train <- 1 - (sum((train_i$LC50 - train_i$predicted_LC50)^2) / sum((train_i$LC50 - mean(train_i$LC50

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)
```

```r
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): 1.39723601460077
## RMSE (Train): 1.18204738255316
## R-squared (Train): 0.507173465901837
##
## Test Metrics:
## MSE (Test): 1.49726002233874
## RMSE (Test): 1.22362576890925
## R-squared (Test): 0.421486302559535
```

```r
# 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() +
```

```r
  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.

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

```r
# 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)
```

```r
head(train_ii)
```

```
##   TPSA SAacc H050 MLOGP RDCHI GATS1p nN C040  LC50
## 1 0.00     0    0 2.419 1.225  0.667  0    0 3.740
## 2 0.00     0    0 2.638 1.401  0.632  0    0 4.330
## 3 9.23    11    0 5.799 2.930  0.486  0    0 7.019
## 5 9.23    11    0 4.068 2.758  0.695  0    0 5.979
## 8 0.00     0    0 3.267 2.318  0.963  0    0 4.100
## 9 0.00     0    0 2.067 1.800  1.250  0    0 3.941
```

```r
# 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
## -3.1077 -0.7722 -0.0985  0.5346  5.2865
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.621928   0.298781   8.775  < 2e-16 ***
## TPSA         0.022844   0.003346   6.827 3.53e-11 ***
## SAacc       -0.012485   0.002326  -5.367 1.41e-07 ***
## H050        -0.012262   0.155175  -0.079  0.93706
## MLOGP        0.530311   0.075388   7.034 9.62e-12 ***
## RDCHI        0.377759   0.163539   2.310  0.02144 *
## GATS1p      -0.529378   0.176963  -2.991  0.00296 **
## nN           0.179787   0.153256   1.173  0.24150
## C040        -0.121025   0.165130  -0.733  0.46408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.221 on 373 degrees of freedom
## Multiple R-squared:  0.4869, Adjusted R-squared:  0.4758
## F-statistic: 44.24 on 8 and 373 DF,  p-value: < 2.2e-16
```

```r
# Predict on training and test datasets
pred_train_transform_dummy <- predict(model, newdata=train_ii)
pred_test_transform_dummy <- predict(model, newdata=test_ii)
```

```r
# Adding predictions columns to the datasets
train_ii$predicted_LC50 <- pred_train_transform_dummy
test_ii$predicted_LC50 <- pred_test_transform_dummy
```

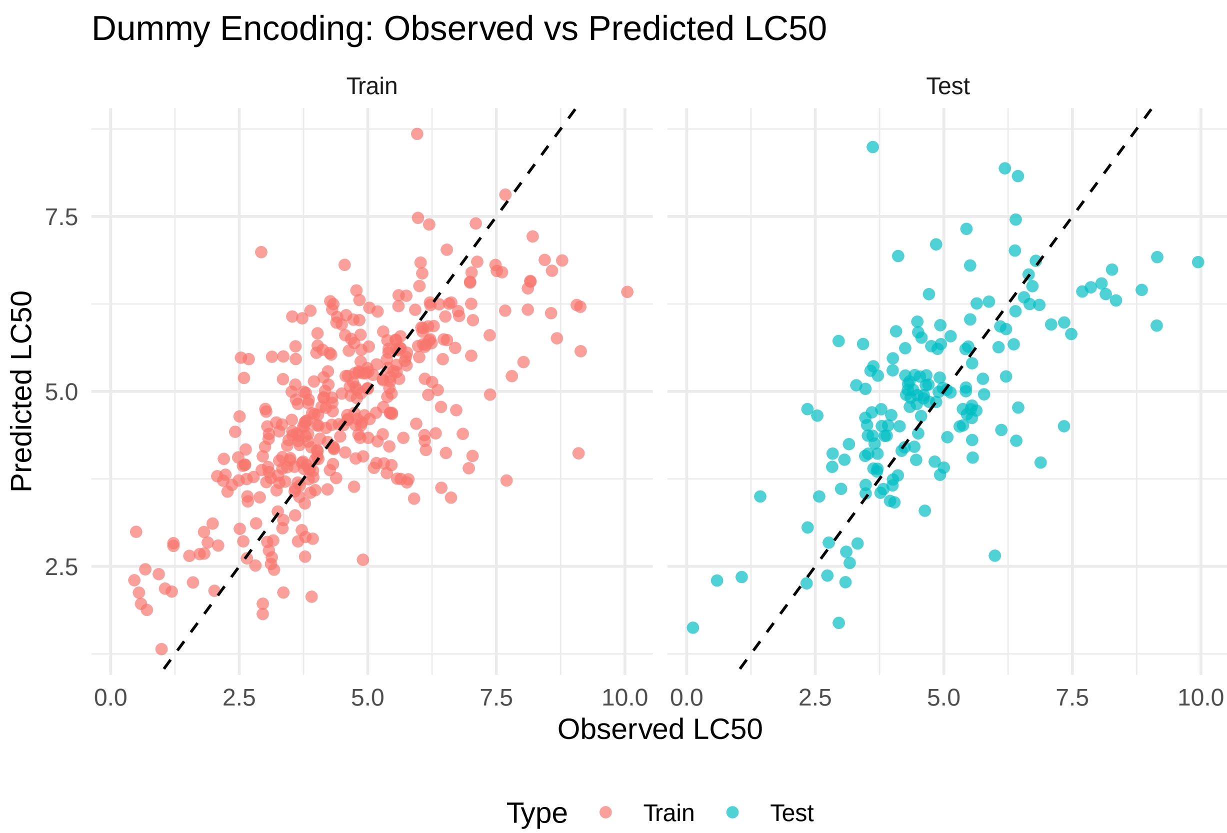
```r
# Evaluate model: calculate MSE, RMSE, and R-squared for training and test sets
mse_train_transform_dummy <- mean((train_ii$LC50 - train_ii$predicted_LC50)^2)
rmse_train_transform_dummy <- sqrt(mse_train_transform_dummy)
```

```r
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 - mea
```

```r
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"
))
```

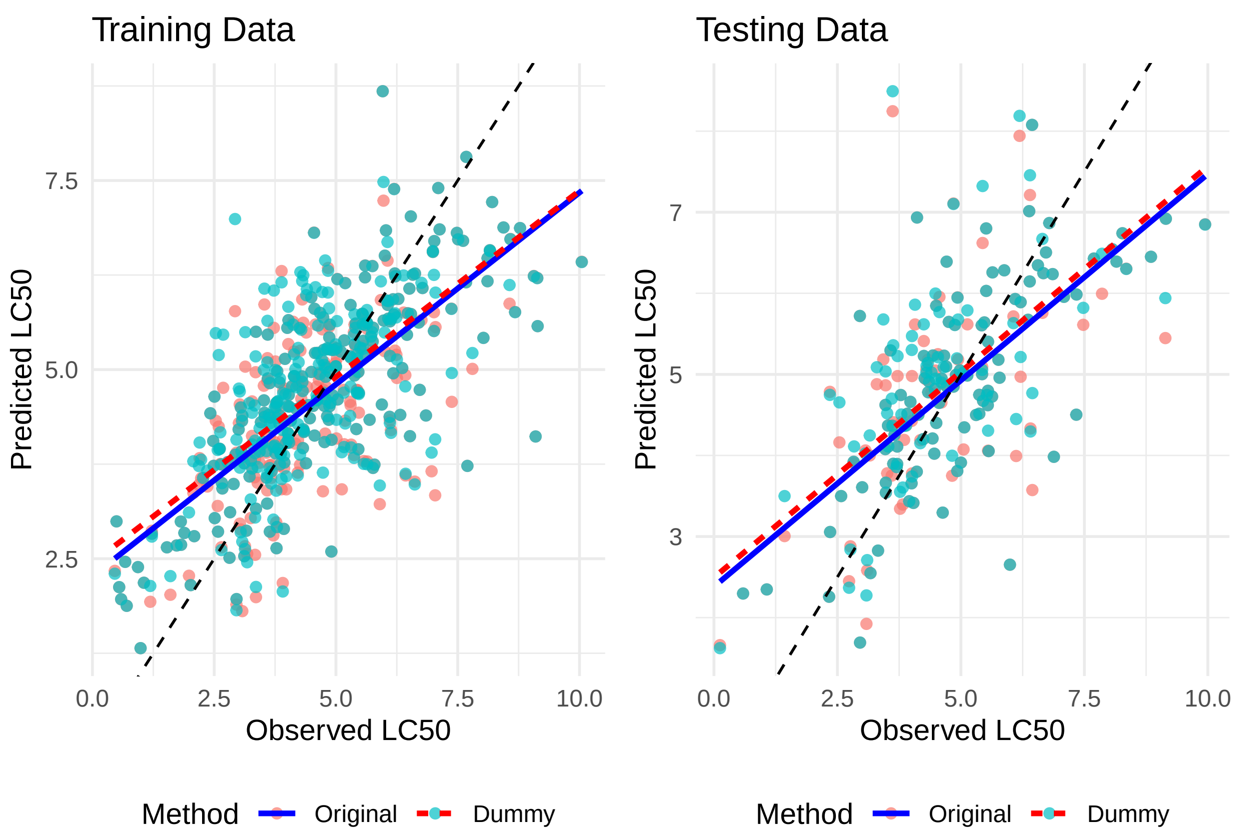
```
## Training Metrics:
## MSE (Train): 1.51792788422074
## RMSE (Train): 1.23204216008249
## R-squared (Train): 0.464603595688728
##
## Test Metrics:
## MSE (Test): 1.54980939282952
## RMSE (Test): 1.24491340776358
## R-squared (Test): 0.401182193609039
```

```r
# 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")
```




```
# 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 spiting 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
```

```r
  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:

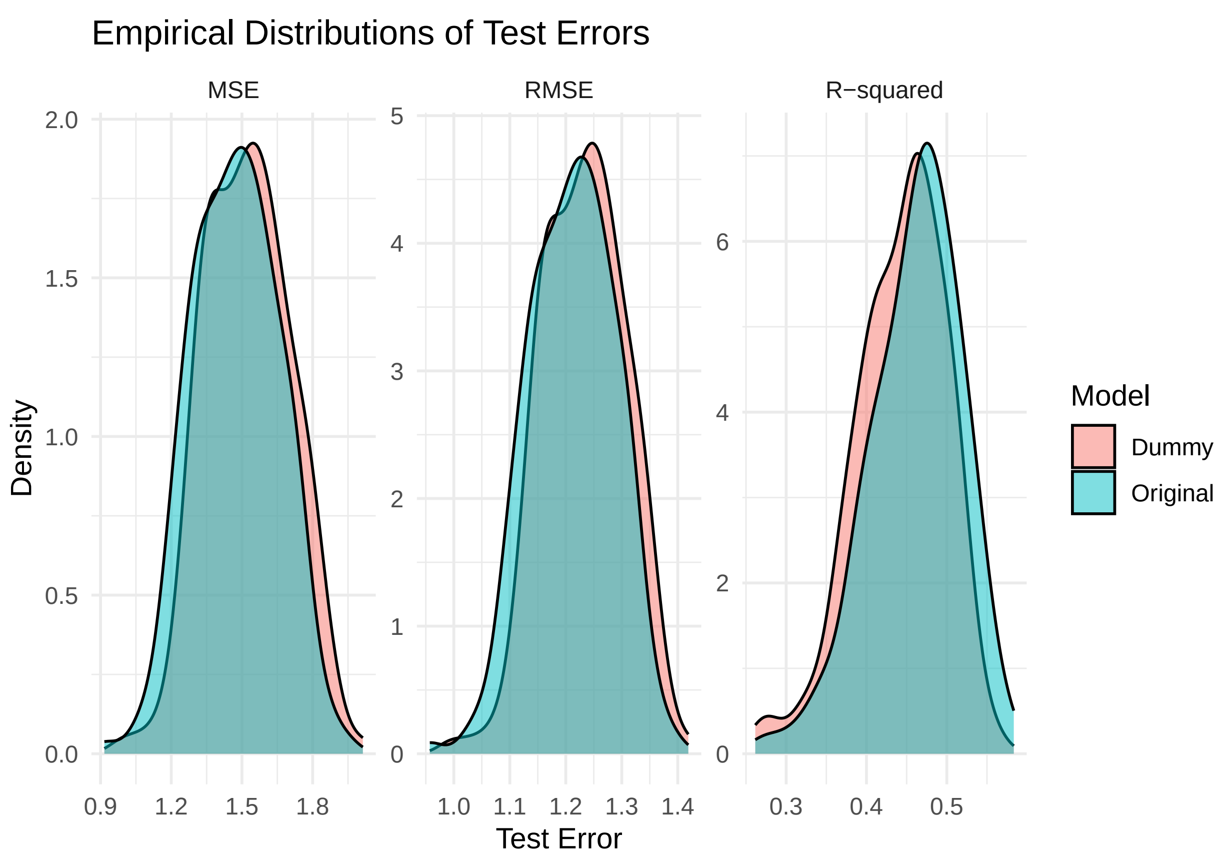
- Method 1: performs better in term of MSE
- Method 2: better in reduce over fitting

```r
# 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
```

```r
# 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()
```




```
# 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() +
```

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

c. Variable selection procedures

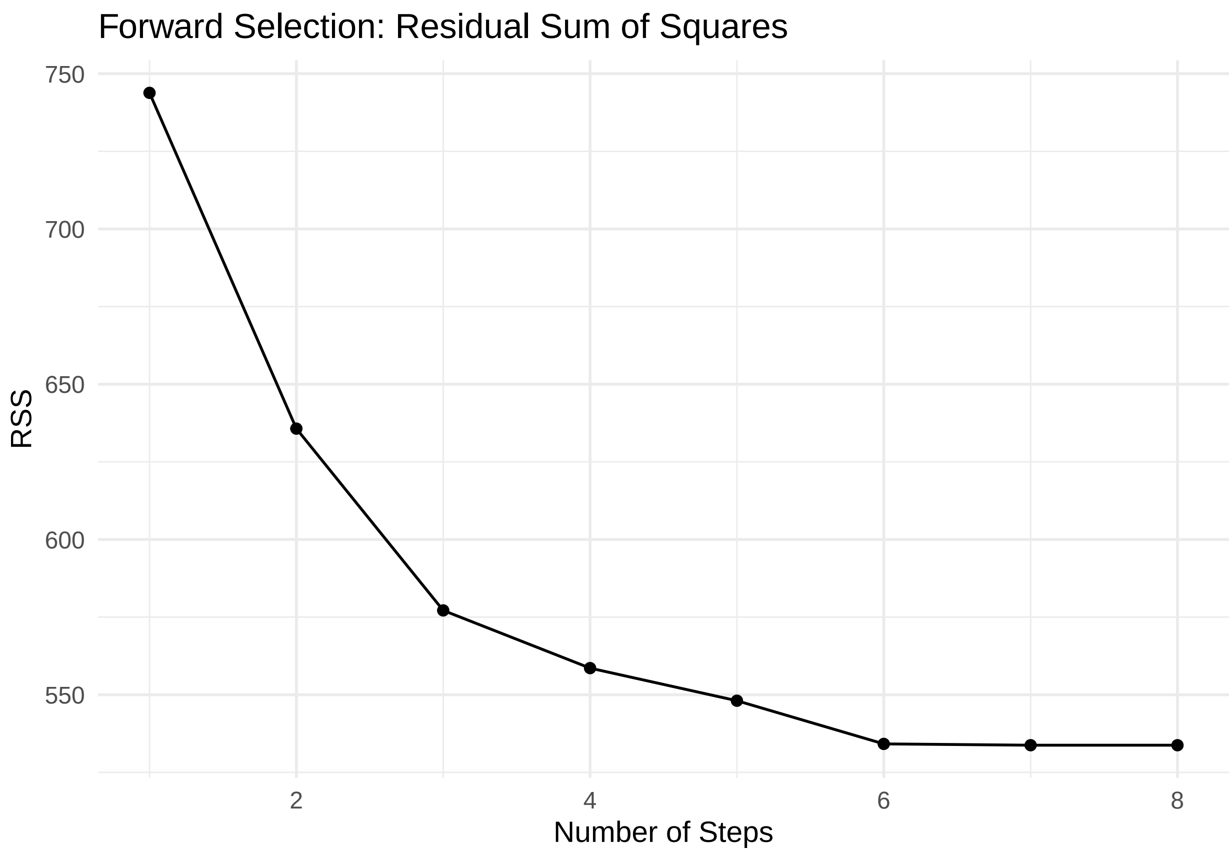
(at least backward elimination and forward selection) with different stopping criteria (at least AIC and BIC) and compare the results. Do you obtain the same model?

```r
# Split the data into training (70%) and test (30%) sets
set.seed(1)
sample <- sample.split(data$LC50, SplitRatio = 0.7)
train <- subset(data, sample == TRUE)
test <- subset(data, sample == FALSE)
```

Forward Selection

```r
# Fit models and collect RSS at each step for forward selection
full.model <- lm(LC50 ~ ., data = train)
null.model <- lm(LC50 ~ 1, data = train)
y <- train$LC50
num_vars <- ncol(train) - 1  # exclude the response variable column

scp <- full.model$terms
```

```r
rss_forward <- vector("numeric", num_vars)

for (j in 1:num_vars) {
  mdl <- stepAIC(object = null.model, scope = scp, direction = 'forward', k = 0, steps = j, trace = FALS
  rss_forward[j] <- sum((mdl$fitted.values - y)^2)
}

# Plot RSS for forward selection
forward_df <- data.frame(Step = 1:num_vars, RSS = rss_forward)

ggplot(forward_df, aes(x = Step, y = RSS)) +
  geom_line() + geom_point() +
  labs(title = "Forward Selection: Residual Sum of Squares", x = "Number of Steps", y = "RSS") +
  theme_minimal()
```



AIC

- Backward Elimination Using AIC

```r
# Fit full model
full.model <- lm(LC50 ~ ., data = train)
summary(full.model)
```

```
##
```

```
## Call:
## lm(formula = LC50 ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8444 -0.7768 -0.1022  0.5521  4.9831
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.596376   0.293581   8.844  < 2e-16 ***
## TPSA         0.028839   0.003333   8.653  < 2e-16 ***
## SAacc       -0.014815   0.002554  -5.800 1.42e-08 ***
## H050         0.037414   0.071245   0.525  0.59980
## MLOGP        0.487698   0.074228   6.570 1.69e-10 ***
## RDCHI        0.496235   0.162219   3.059  0.00238 **
## GATS1p      -0.580937   0.179735  -3.232  0.00134 **
## nN          -0.246680   0.060554  -4.074 5.65e-05 ***
## C040         0.002834   0.092694   0.031  0.97563
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.196 on 373 degrees of freedom
## Multiple R-squared:  0.5072, Adjusted R-squared:  0.4966
## F-statistic: 47.98 on 8 and 373 DF,  p-value: < 2.2e-16
```

```r
# Apply backward elimination using AIC
model.backward.aic <- stepAIC(full.model, direction = 'backward', trace = FALSE)
summary(model.backward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

- Forward Selection Using AIC

```r
# Fit null model
null.model <- lm(LC50 ~ 1, data = train)

# Apply forward selection using AIC
model.forward.aic <- stepAIC(null.model, scope = list(lower = null.model, upper = full.model), directio
summary(model.forward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

- Stepwise Selection Using AIC

```r
# Apply stepwise selection using AIC
model.stepwise.aic <- stepAIC(null.model, scope = list(lower = null.model, upper = full.model), directi
summary(model.stepwise.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
```

```
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

BIC

- Backward Elimination

```
# Apply backward elimination using BIC
model.backward.bic <- stepAIC(full.model, direction = 'backward', k = log(nrow(train)), trace = FALSE)
summary(model.backward.bic)
```

```
##
## Call:
## lm(formula = LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

- Forward Selection

```
# Apply forward selection using BIC
model.forward.bic <- stepAIC(null.model, scope = list(lower = null.model, upper = full.model), directio
summary(model.forward.bic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

- Stepwise Selection

```r
# Apply stepwise selection using BIC
model.stepwise.bic <- stepAIC(null.model, scope = list(lower = null.model, upper = full.model), directi
summary(model.stepwise.bic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7029 -0.7797 -0.0956  0.5643  4.9808
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.657047   0.268474   9.897  < 2e-16 ***
## MLOGP        0.479421   0.071536   6.702 7.57e-11 ***
## TPSA         0.028584   0.003270   8.741  < 2e-16 ***
## SAacc       -0.014051   0.002051  -6.850 3.04e-11 ***
## nN          -0.241351   0.058505  -4.125 4.56e-05 ***
## GATS1p      -0.610543   0.170729  -3.576 0.000394 ***
## RDCHI        0.491458   0.157271   3.125 0.001917 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.193 on 375 degrees of freedom
```

```
## Multiple R-squared:  0.5068, Adjusted R-squared:  0.4989
## F-statistic: 64.22 on 6 and 375 DF,  p-value: < 2.2e-16
```

Model Comparison

```r
# Predict on the test set using all models
test$pred_backward_aic <- predict(model.backward.aic, newdata = test)
test$pred_forward_aic <- predict(model.forward.aic, newdata = test)
test$pred_stepwise_aic <- predict(model.stepwise.aic, newdata = test)
test$pred_backward_bic <- predict(model.backward.bic, newdata = test)
test$pred_forward_bic <- predict(model.forward.bic, newdata = test)
test$pred_stepwise_bic <- predict(model.stepwise.bic, newdata = test)

# Calculate MSE, RMSE, and R-squared for each model
mse <- function(actual, predicted) mean((actual - predicted)^2)
rmse <- function(actual, predicted) sqrt(mse(actual, predicted))
r2 <- function(actual, predicted) 1 - (sum((actual - predicted)^2) / sum((actual - mean(actual))^2))

metrics <- data.frame(
  Model = c("Backward AIC", "Forward AIC", "Stepwise AIC", "Backward BIC", "Forward BIC", "Stepwise BIC
  MSE = c(
    mse(test$LC50, test$pred_backward_aic),
    mse(test$LC50, test$pred_forward_aic),
    mse(test$LC50, test$pred_stepwise_aic),
    mse(test$LC50, test$pred_backward_bic),
    mse(test$LC50, test$pred_forward_bic),
    mse(test$LC50, test$pred_stepwise_bic)
  ),
  RMSE = c(
    rmse(test$LC50, test$pred_backward_aic),
    rmse(test$LC50, test$pred_forward_aic),
    rmse(test$LC50, test$pred_stepwise_aic),
    rmse(test$LC50, test$pred_backward_bic),
    rmse(test$LC50, test$pred_forward_bic),
    rmse(test$LC50, test$pred_stepwise_bic)
  ),
  R2 = c(
    r2(test$LC50, test$pred_backward_aic),
    r2(test$LC50, test$pred_forward_aic),
    r2(test$LC50, test$pred_stepwise_aic),
    r2(test$LC50, test$pred_backward_bic),
    r2(test$LC50, test$pred_forward_bic),
    r2(test$LC50, test$pred_stepwise_bic)
  )
)
print(metrics)
```

```
##          Model      MSE     RMSE        R2
## 1 Backward AIC 1.499497 1.224539 0.4206221
## 2  Forward AIC 1.499497 1.224539 0.4206221
## 3 Stepwise AIC 1.499497 1.224539 0.4206221
## 4 Backward BIC 1.499497 1.224539 0.4206221
```

```
## 5  Forward BIC 1.499497 1.224539 0.4206221
## 6 Stepwise BIC 1.499497 1.224539 0.4206221
```