# EDA - Diabetes analysis

2024-03-21

## Background

The dataset includes 9 baseline numeric variables: age, body mass index, average blood pressure, and six blood serum measurements for each of n = 442 diabetes patients. The response of interest is a quantitative measure of diabetes disease progression one year after baseline. The dataset is obtained from sklearn.datasets.

## Attribute Information:

age: age in years

bmi: body mass index

bp: average blood pressure

s1: tc, total serum cholesterol

s2: ldl, low-density lipoproteins

s3: hdl, high-density lipoproteins

s4: tch, total cholesterol / HDL

s5: ltg, possibly log of serum triglycerides level

s6: glu, blood sugar level

Target: quantitative measure of disease progression one year after baseline

*Note: All features have been standardized already.*

## Read Data

```r
set.seed(100)

library(olsrr)
```

```
## Warning: package 'olsrr' was built under R version 4.2.3
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
library(stats)
library(leaps)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:olsrr':
##
##     cement
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
diabetes = read.csv('diabetes_dataset.csv')

# Create an index for splitting the data (80% training, 20% test)
train_index <- createDataPartition(diabetes$Target, p = 0.8, list = FALSE)

# Split the data into training and test sets
diabetes <- diabetes[train_index, ]
test_data <- diabetes[-train_index, ]


n = nrow(diabetes)
head(diabetes)
```

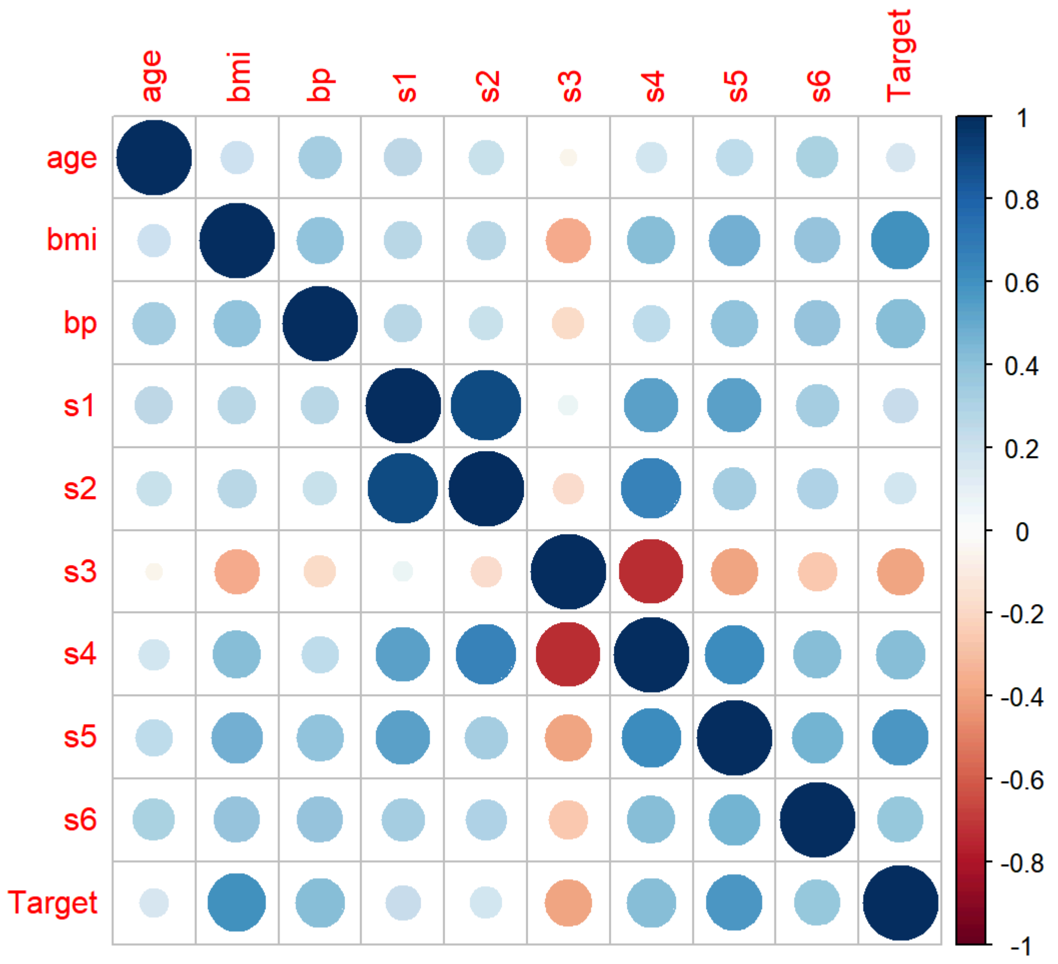
```
##           age         bmi           bp           s1          s2           s3
## 1  0.03807591  0.06169621  0.021872386 -0.044223498 -0.03482076 -0.043400846
## 3  0.08529891  0.04445121 -0.005670422 -0.045599451 -0.03419447 -0.032355932
## 4 -0.08906294 -0.01159502 -0.036656081  0.012190569  0.02499059 -0.036037570
## 5  0.00538306 -0.03638469  0.021872386  0.003934852  0.01559614  0.008142084
## 6 -0.09269548 -0.04069594 -0.019441826 -0.068990650 -0.07928784  0.041276824
## 7 -0.04547248 -0.04716281 -0.015998975 -0.040095640 -0.02480001  0.000778808
##             s4           s5           s6 Target
## 1 -0.002592262  0.019907486 -0.017646125    151
## 3 -0.002592262  0.002861309 -0.025930339    141
## 4  0.034308859  0.022687745 -0.009361911    206
## 5 -0.002592262 -0.031987639 -0.046640874    135
## 6 -0.076394504 -0.041176167 -0.096346157     97
## 7 -0.039493383 -0.062916879 -0.038356660    138
```

# EDA

## Correlation Matrix plot for variables (relationship between different variables)

```
# Calculate the correlation matrix
correlation_matrix <- cor(diabetes)

# Plot the correlation matrix
corrplot(correlation_matrix, method = "circle")
```



**The color intensity and the size of the circle indicate the strength of correlation: darker colors and larger circles represent stronger correlations.**

**Positive correlations are shown in blue, while negative correlations are in red.**

**Near-zero correlations are represented in white.**

**Variables that are highly correlated (either positively or negatively) will have larger circles and darker colors.**

- S2 & S1 has strong positive correltaion and S3 & S4 has strong negative correlation which suggest that changes in one variable are associated with change in another variable. These variables might indicate multicollinearity, which could affect the performance of certain predictive models. In such case, one of the correlated variables could be removed to avoid redundancy.
- bmi and S5 have relatively strong correlations with the target variable. These variable might be important predictors.

## Multiple linear regression - Full model

```r
# Fit multiple linear regression model
model1 <- lm(Target ~ ., data = diabetes)

# Display model summary
summary(model1)
```

```
##
## Call:
## lm(formula = Target ~ ., data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -149.547  -40.131   -3.836   39.949  146.407
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.035      2.943  51.323  < 2e-16 ***
## age          -75.427     67.091  -1.124  0.26169
## bmi          593.410     76.403   7.767 9.28e-14 ***
## bp           260.434     74.337   3.503  0.00052 ***
## s1          -705.918    459.626  -1.536  0.12549
## s2           447.938    371.197   1.207  0.22836
## s3           115.654    239.369   0.483  0.62929
## s4            66.286    185.591   0.357  0.72119
## s5           769.059    192.943   3.986 8.21e-05 ***
## s6            56.534     76.502   0.739  0.46042
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.3 on 345 degrees of freedom
## Multiple R-squared:  0.5057, Adjusted R-squared:  0.4928
## F-statistic: 39.22 on 9 and 345 DF,  p-value: < 2.2e-16
```

- Intercept, bmi, bp and s5 are significant at conventional levels (p<0.05).
- The model as a whole is significant, as indicated by the small p-value of the F-statistic.

**Significant coefficients at 95% confidence level (alpha = 0.05): Intercept, bmi, bp, s5**

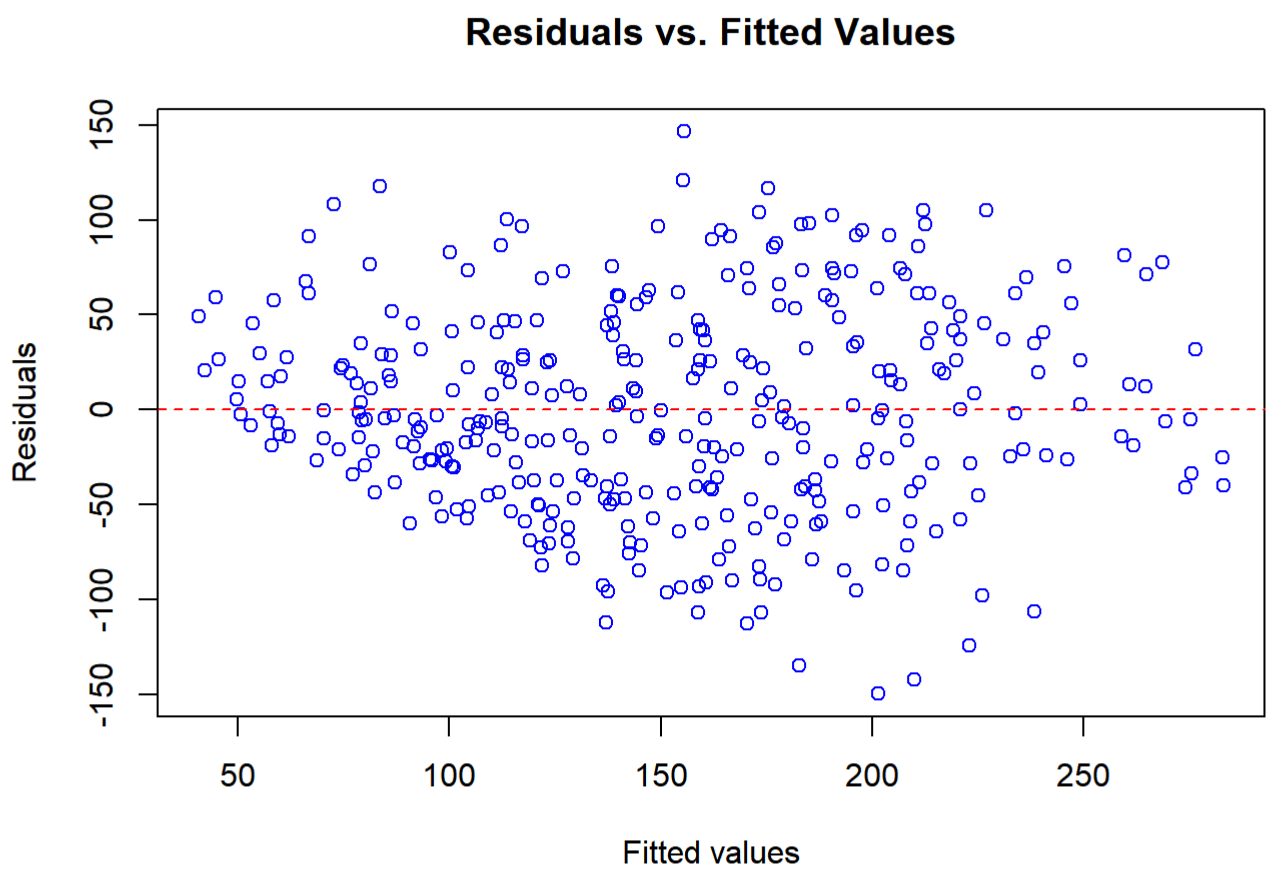
**Significant coefficients at 90% confidence level (alpha = 0.01): Intercept, bmi, bp, s1, s5**

## Residual analysis for full model

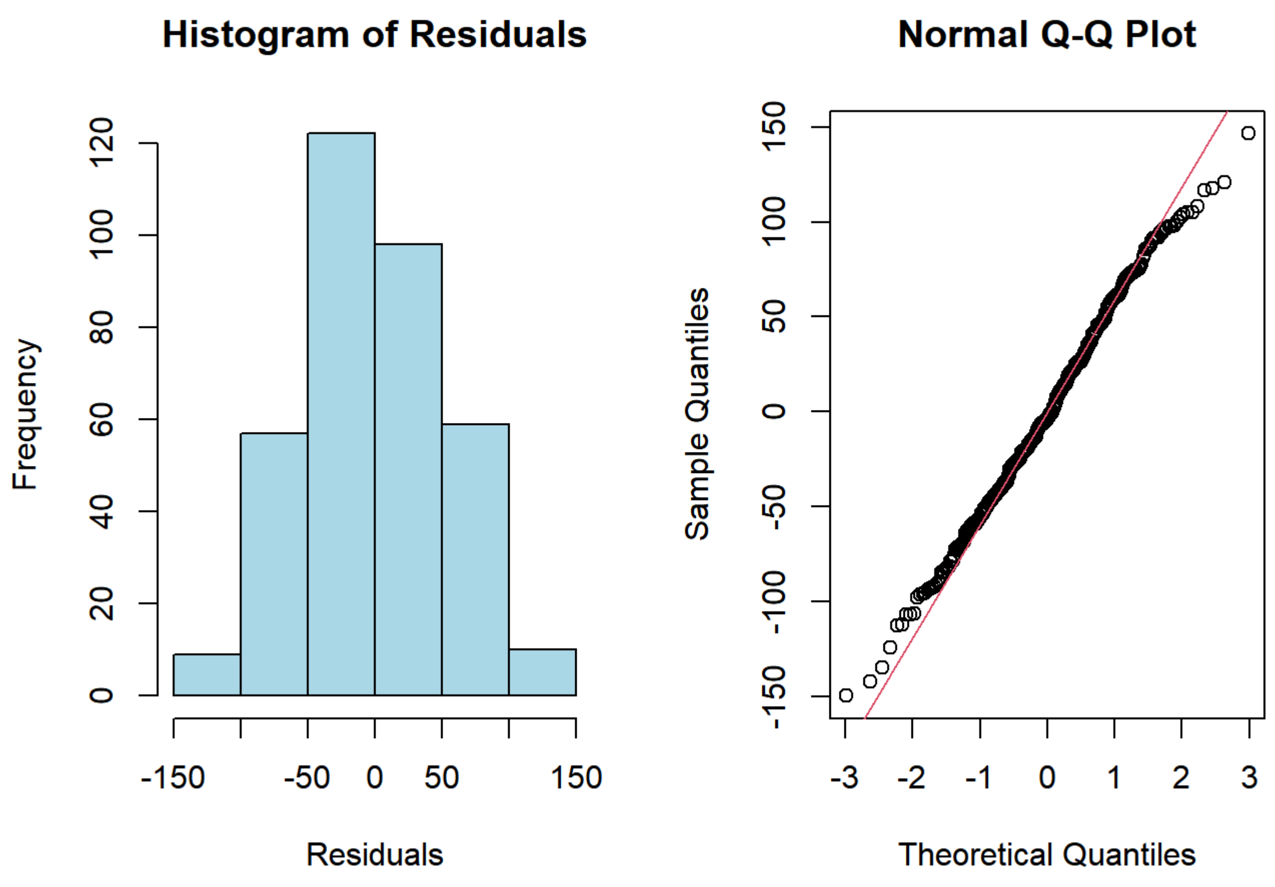
```
# Obtain residuals
residuals_model1 <- residuals(model1)

# Plot residuals vs. fitted values
plot(fitted(model1), residuals_model1, xlab = "Fitted values", ylab = "Residuals", main = "Residuals vs. Fitted Values", col
= "blue")

# Add a horizontal line at y = 0
abline(h = 0, col = "red", lty = 2)
```



```r
# Plot a density plot of residuals
par(mfrow = c(1, 2))
hist(residuals_model1, main = "Histogram of Residuals", xlab = "Residuals", col = "lightblue", border = "black")
qqnorm(residuals_model1, main = "Normal Q-Q Plot")
qqline(residuals_model1, col = 2)
```



```r
par(mfrow = c(1, 1))  # Reset to default plotting layout
```

- Residuals vs. Fitted values plot: Random scatter of points around the horizontal line at y=0. There is no specific patterns (eg., curved pattern, funnel shape), the constant variance (Homoscedasticity) or linearity assumption holds.
- Histogram Residuals: Symmetric distribution centered around zero (normally distributed).

- Normal Q-Q plot: The points fall approximately along the diagonal line (normality assumption holds).

Therefore, the assumptions of linearity, constant variance, and normality hold (No need to transform data).

## Mellow’s CP, AIC and BIC values for the full model

***Compare different models and select the one that has the optimal balance between bias and variance.***

```
# Calculate Mallow's Cp
Cp <- (sum(residuals(model1)^2) / (nrow(diabetes) - length(coef(model1)) + 2)) * (nrow(diabetes) - length(coef(model1)))

# Calculate AIC
AIC_value <- AIC(model1)

# Calculate BIC
BIC_value <- BIC(model1)

# Display the values
print(paste("Mallow's Cp:", Cp))
```

```
## [1] "Mallow's Cp: 1049087.23692245"
```

```
print(paste("AIC:", AIC_value))
```

```
## [1] "AIC: 3868.41459344312"
```

```
print(paste("BIC:", BIC_value))
```

```
## [1] "BIC: 3911.00788912735"
```

The high Mallow’s Cp, AIC, and BIC values for the full model indicate that the model could potentially be improved or simplified.

## Reduced model (significant coefficients at the 95% confidence level)

```r
# Fit new multiple linear regression model with significant variables
model2 <- lm(Target ~ bmi + bp + s5, data = diabetes)

# Display model summary
summary(model2)
```

```
##
## Call:
## lm(formula = Target ~ bmi + bp + s5, data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -141.892  -39.074   -4.866   37.002  139.033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.363      2.971  50.940  < 2e-16 ***
## bmi          640.779     74.036   8.655  < 2e-16 ***
## bp           236.287     71.265   3.316  0.00101 **
## s5           532.596     72.368   7.360 1.32e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.98 on 351 degrees of freedom
## Multiple R-squared:  0.4848, Adjusted R-squared:  0.4804
## F-statistic: 110.1 on 3 and 351 DF,  p-value: < 2.2e-16
```

The adjusted R-squared value is 0.4765, indicating that approximately 47.65% of the variance in the target variable is explained by the predictors in the model.

The F-statistic is significant ($p < 0.001$), indicating that the model as a whole is significant.

### Partial F-test to compare the reduced model with the full model.

```
# Perform Partial F-test
partial_f_test <- anova(model2, model1)

# Display the Partial F-test result
print(partial_f_test)
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ bmi + bp + s5
## Model 2: Target ~ age + bmi + bp + s1 + s2 + s3 + s4 + s5 + s6
##   Res.Df     RSS Df Sum of Sq      F  Pr(>F)
## 1    351 1099824
## 2    345 1055169  6     44655 2.4334 0.02564 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value associated with the Partial F-test is 0.007745, which is less than 0.05. Since the p-value is less than 0.05, we reject the null hypothesis, indicating that the full model (model1) significantly improves the fit compared to the reduced model (model2) at the 95% confidence level.

While statistical significance is important, it's not the sole criterion for variable selection. It's essential to consider other factors such as theoretical relevance, practical significance, model fit, interpretability, and predictive performance. Relying solely on statistical significance for variable selection can lead to over-fitting, omitted variable bias, variable selection bias, and loss of information, as previously discussed.

## Stepwise Regression (Forward) - Full model

BIC is favored for forward stepwise regression to select simpler models, while AIC is favored for backward stepwise regression to balance model fit and complexity. However, the choice between BIC and AIC ultimately depends on the specific goals and assumptions of the analysis.

```
# Perform forward stepwise regression using BIC
forward_model <- step(model1, direction = "forward", k = log(nrow(diabetes)), trace = 0)

# Display model summary
summary(forward_model)
```

```
##
## Call:
## lm(formula = Target ~ age + bmi + bp + s1 + s2 + s3 + s4 + s5 +
##     s6, data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -149.547  -40.131   -3.836   39.949  146.407
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.035      2.943  51.323  < 2e-16 ***
## age          -75.427     67.091  -1.124  0.26169
## bmi          593.410     76.403   7.767 9.28e-14 ***
## bp           260.434     74.337   3.503  0.00052 ***
## s1          -705.918    459.626  -1.536  0.12549
## s2           447.938    371.197   1.207  0.22836
## s3           115.654    239.369   0.483  0.62929
## s4            66.286    185.591   0.357  0.72119
## s5           769.059    192.943   3.986 8.21e-05 ***
## s6            56.534     76.502   0.739  0.46042
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.3 on 345 degrees of freedom
## Multiple R-squared:  0.5057, Adjusted R-squared:  0.4928
## F-statistic: 39.22 on 9 and 345 DF,  p-value: < 2.2e-16
```

```
# Extract p-values from the model summary
p_values <- summary(forward_model)$coefficients[, 4]

# Determine significant coefficients at the 99% confidence level
significant_99 <- names(p_values[p_values < 0.01])

# Display significant coefficients at the 99% confidence level
print("Significant coefficients at the 99% confidence level:")
```

```
## [1] "Significant coefficients at the 99% confidence level:"
```

```
print(significant_99)
```

```
## [1] "(Intercept)" "bmi"         "bp"          "s5"
```

- For forward model, Intercept, bmi, bp and s5 were selected.

## Stepwise Regression (Backward) - Full model

```
# Perform backward stepwise regression using AIC
backward_model <- step(model1, direction = "backward", k = 2, trace = 0)

# Display model summary
summary(backward_model)
```

```
##
## Call:
## lm(formula = Target ~ bmi + bp + s1 + s2 + s5, data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -148.497  -39.157   -3.511   40.191  152.984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.252      2.928  51.649  < 2e-16 ***
## bmi          598.122     75.505   7.922 3.17e-14 ***
## bp           248.854     70.353   3.537 0.000459 ***
## s1          -537.100    168.991  -3.178 0.001614 **
## s2           335.483    156.309   2.146 0.032539 *
## s5           720.661     89.006   8.097 9.52e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.13 on 349 degrees of freedom
## Multiple R-squared:  0.5031, Adjusted R-squared:  0.496
## F-statistic: 70.66 on 5 and 349 DF,  p-value: < 2.2e-16
```

```
# Extract p-values from the model summary
p_values_backward <- summary(backward_model)$coefficients[, 4]

# Determine significant coefficients at the 99% confidence level
significant_99_backward <- names(p_values_backward[p_values_backward < 0.01])

# Display significant coefficients at the 99% confidence level
print("Significant coefficients at the 99% confidence level in backward_model:")
```

```
## [1] "Significant coefficients at the 99% confidence level in backward_model:"
```

```
print(significant_99_backward)
```

```
## [1] "(Intercept)" "bmi"         "bp"          "s1"          "s5"
```

- For backward model,Intercept, bmi, bp, s1 and s5 were selected.

## Partial F-test for forward_model vs. full model & backward_model vs. full model

```
# Partial F-test for forward_model vs. model1
partial_f_test_forward <- anova(forward_model, model1)

partial_f_test_forward
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ age + bmi + bp + s1 + s2 + s3 + s4 + s5 + s6
## Model 2: Target ~ age + bmi + bp + s1 + s2 + s3 + s4 + s5 + s6
##   Res.Df     RSS Df Sum of Sq F Pr(>F)
## 1    345 1055169
## 2    345 1055169  0         0
```

```
# Partial F-test for backward_model vs. model1
partial_f_test_backward <- anova(backward_model, model1)

partial_f_test_backward
```

```
## Analysis of Variance Table
##
## Model 1: Target ~ bmi + bp + s1 + s2 + s5
## Model 2: Target ~ age + bmi + bp + s1 + s2 + s3 + s4 + s5 + s6
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1    349 1060765
## 2    345 1055169  4    5596.1 0.4574  0.767
```

**Partial F-test for forward_model vs. model1:** The p-value is 0.This indicates that the observed F-statistic is highly significant at the 95% confidence level.Therefore, we reject the null hypothesis, indicating that the full model provides a significantly better fit to the data compared.

**Partial F-test for backward_model vs. model1:** The p-value is 0.8369. This indicates that the observed F-statistic is not significant at the 95% confidence level.Therefore, we fail to reject the null hypothesis, suggesting that there is no significant difference in model fit between the backward_model and model1. It suggests that there may not be a substantial loss in model performance when using the subset of predictors identified by backward stepwise regression, indicating that this subset of predictors captures a significant portion of the variability in the response variable.

## Full model search - Possible Model

With 9 predictor variables in the dataset, there are

$$2^9 = 512$$

possible models that can be constructed using subsets drawn from the full set of predictor variables.

```
# Load necessary libraries
library(leaps)

# Fit all possible models using leaps package
all_models <- regsubsets(Target ~ ., data = diabetes)

# Extract Cp values for all models
all_cp <- summary(all_models)$cp
all_cp
```

```
## [1] 93.468621 21.863215 12.600438  6.407600  3.829708  4.818824  6.234589
## [8]  8.127563
```

```
# Find the index of the model with the lowest Cp value
best_model_index <- which.min(all_cp)
best_model_index
```

```
## [1] 5
```

```
# Get the variables included in the best model
best_model_variables <- names(coef(all_models, id = best_model_index))

# Display the variables included in the best model and the corresponding Cp value
cat("Variables included in the best model:", best_model_variables, "\n")
```

```
## Variables included in the best model: (Intercept) bmi bp s1 s2 s5
```

```
cat("Corresponding Mallow's Cp value:", all_cp[best_model_index], "\n")
```

```
## Corresponding Mallow's Cp value: 3.829708
```

The result indicates that the best model, as determined by Mallow's Cp statistic, includes the intercept along with the variables 'bmi', 'bp', 's1', 's1', 's2', and 's5'.

The Mallow's Cp value for this best model is approximately 3.44089.

- Mellow's Cp is a measure of model fit that penalizes models for their complexity, similar to other model selection critria such as AIC and BIC.
- Lower values of Cp indicate better model fit while balancing the trade-off between model complexity and goodness of fit.

## Model with selected variables for best model

```
# Fit the best model using selected variables
best_model <- lm(Target ~ bmi + bp + s1 + s2 + s5, data = diabetes)

# Display the model summary
summary(best_model)
```

```
##
## Call:
## lm(formula = Target ~ bmi + bp + s1 + s2 + s5, data = diabetes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -148.497  -39.157   -3.511   40.191  152.984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  151.252      2.928  51.649  < 2e-16 ***
## bmi          598.122     75.505   7.922 3.17e-14 ***
## bp           248.854     70.353   3.537 0.000459 ***
## s1          -537.100    168.991  -3.178 0.001614 **
## s2           335.483    156.309   2.146 0.032539 *
## s5           720.661     89.006   8.097 9.52e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.13 on 349 degrees of freedom
## Multiple R-squared:  0.5031, Adjusted R-squared:  0.496
## F-statistic: 70.66 on 5 and 349 DF,  p-value: < 2.2e-16
```

## Compare the models (model1, model2, forward_model, backward_model, best_model)

```
# Create a dataframe to store model information
model_info <- data.frame(
  Model = c("model1", "model2", "forward_model","backward_model", "best_model"),
  Adjusted_R_squared = c(summary(model1)$adj.r.squared, summary(model2)$adj.r.squared,
                         summary(forward_model)$adj.r.squared,summary(backward_model)$adj.r.squared, summary(best_model)$ad
j.r.squared),
  AIC = c(AIC(model1), AIC(model2), AIC(forward_model), AIC(backward_model),AIC(best_model))
)

# Display the model comparison
print(model_info)
```

```
##             Model Adjusted_R_squared      AIC
## 1          model1          0.4928052 3868.415
## 2          model2          0.4803776 3871.129
## 3   forward_model          0.4928052 3868.415
## 4  backward_model          0.4959593 3862.292
## 5      best_model          0.4959593 3862.292
```

Based on this comparison, the backward_model and best_model seem to perform slightly better than the other models in terms of adjusted R-squared and AIC. However, the differences in adjusted R-squared and AIC values between these models are relatively small, indicating that they may provide similar levels of model fit.

- model1 and forward_model perform similarly in terms of Adjusted R-squared and AIC.
- model2 performs the worst among the compared models in terms of both Adjusted R-squared and AIC.

## Ridge and LASSO regularization

### Ridge regression using 10-fold CV to find the optimal lambda value

```
# Load necessary library
library(glmnet)

# Prepare the data
x <- as.matrix(diabetes[, -ncol(diabetes)])  # Predictor variables
y <- diabetes$Target  # Response variable

# Perform ridge regression with 10-fold cross-validation
ridge.cv <- cv.glmnet(x, y, alpha = 0, nfolds = 10)

ridge <- glmnet(x, log(y), alpha = 0, nlambda = 100)

# Get the optimal lambda value
optimal_lambda_ridge <- ridge.cv$lambda.min

# Display the optimal lambda value
cat("Optimal lambda value (Lambda_min):", optimal_lambda_ridge, "\n")
```

```
## Optimal lambda value (Lambda_min): 6.78004
```

```
# Get the coefficients at the optimal lambda value
coefficients_at_optimal_lambda <- coef(ridge.cv, s = optimal_lambda_ridge)

# Display the coefficients
print(coefficients_at_optimal_lambda)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept)  151.09669
## age          -59.25916
## bmi          557.46721
## bp           253.83624
## s1           -62.70125
## s2           -62.34385
## s3          -146.92522
## s4            45.76856
## s5           470.29048
## s6            75.41568
```

```
# Count the number of non-zero coefficients (variables selected by ridge regression)
num_selected_variables <- sum(coefficients_at_optimal_lambda != 0)
cat("Number of variables selected by ridge regression:", num_selected_variables, "\n")
```

```
## Number of variables selected by ridge regression: 10
```

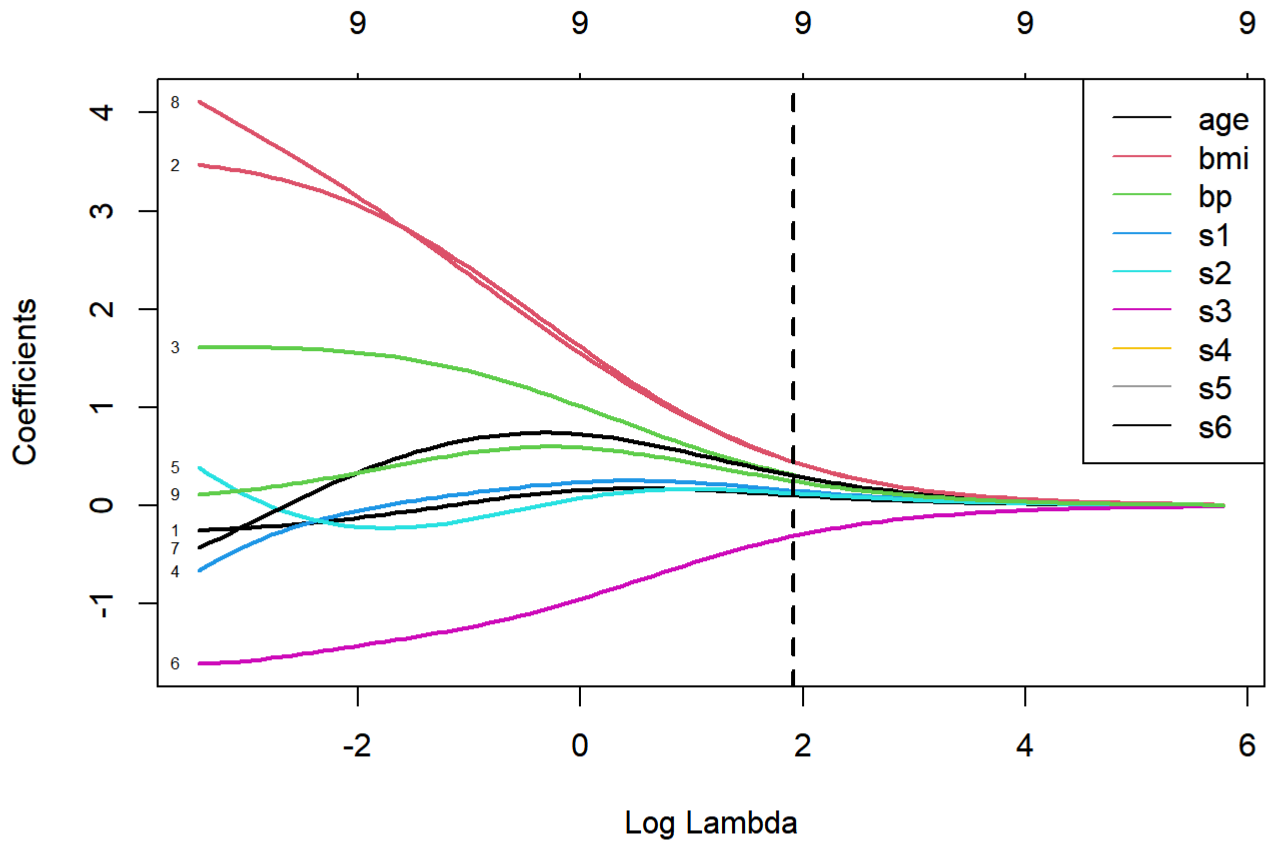
## Coefficient path for Lasso Regression

When lambda is small (far left), the model is less regularized, and the coefficients are large. As lambda increases (moving towards the right), the coefficients shrink towards zero, and some coefficients may eventually reach zero, indicating variable selection.

The plot helps identify the optimal lambda value where the model achieves a balance between bias and variance, typically selected using cross-validation.

```
# Plot the coefficient path for Lasso regression with coefficients on the y-axis
plot(ridge, xvar = "lambda", label = TRUE, lwd=2)
abline(v=log(ridge.cv$lambda.min), col='black', lty=2, lwd=2)

# Add a legend
legend("topright", legend = colnames(x), col = 1:ncol(x), lty = 1)
```



The result is expected for ridge regression. Ridge regression shrinks the coefficients towards zero while still keeping all variables in the model. It's rare for ridge regression to completely eliminate variables by setting their coefficients to exactly zero, except when lambda is very large.

The fact that one variable ("s3") had its coefficient reduced but not set to exactly zero is typical behavior for ridge regression.

Ridge regression’s regularization penalty is designed to control the magnitude of coefficients without causing variable selection in the same way as lasso regression. Therefore, it’s expected that ridge regression would retain all variables in the model, albeit with some coefficients shrunk towards zero.

## Lasso regression

```r
# Prepare the data
x <- as.matrix(diabetes[, -ncol(diabetes)])  # Predictor variables
y <- diabetes$Target  # Response variable

# Perform lasso regression with 10-fold cross-validation
lasso.cv<- cv.glmnet(x, log(y), alpha = 1, nfolds = 10)

# Fit lasso model with 100 values for lambda
lasso <- glmnet(x,log(y), alpha = 1, nlambda = 100 )

# Get the optimal lambda value
optimal_lambda_lasso <- lasso.cv$lambda.min

# Display the optimal lambda value
cat("Optimal lambda value for lasso regression (Lambda_min):", optimal_lambda_lasso, "\n")
```

```
## Optimal lambda value for lasso regression (Lambda_min): 0.01647888
```

```r
# Get the coefficients at the optimal lambda value
coefficients_at_optimal_lambda_lasso <- coef(lasso.cv, s = "lambda.min")

# Display the coefficients
print(coefficients_at_optimal_lambda_lasso)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept)  4.874131
## age          .
## bmi          3.448520
## bp           1.320900
## s1           .
## s2           .
## s3          -1.389234
## s4           .
## s5           3.720776
## s6           .
```

```
# Count the number of non-zero coefficients (variables selected by lasso regression)
num_selected_variables_lasso <- sum(coefficients_at_optimal_lambda_lasso != 0)
cat("Number of variables selected by lasso regression:", num_selected_variables_lasso, "\n")
```

```
## Number of variables selected by lasso regression: 5
```

age, s2, s4: These coefficients are set to zero at the optimal lambda value, indicating that these variables were not selected by lasso regression. These variables are considered non-significant or less influential in predicting the response variable.

bmi, bp, s1, s3, s5, s6: These coefficients are non-zero, indicating that these variables were selected by lasso regression. They have a non-zero influence on predicting the response variable. (The signs of these coefficients indicate the direction of their effect on the response variable: positive or negative.)

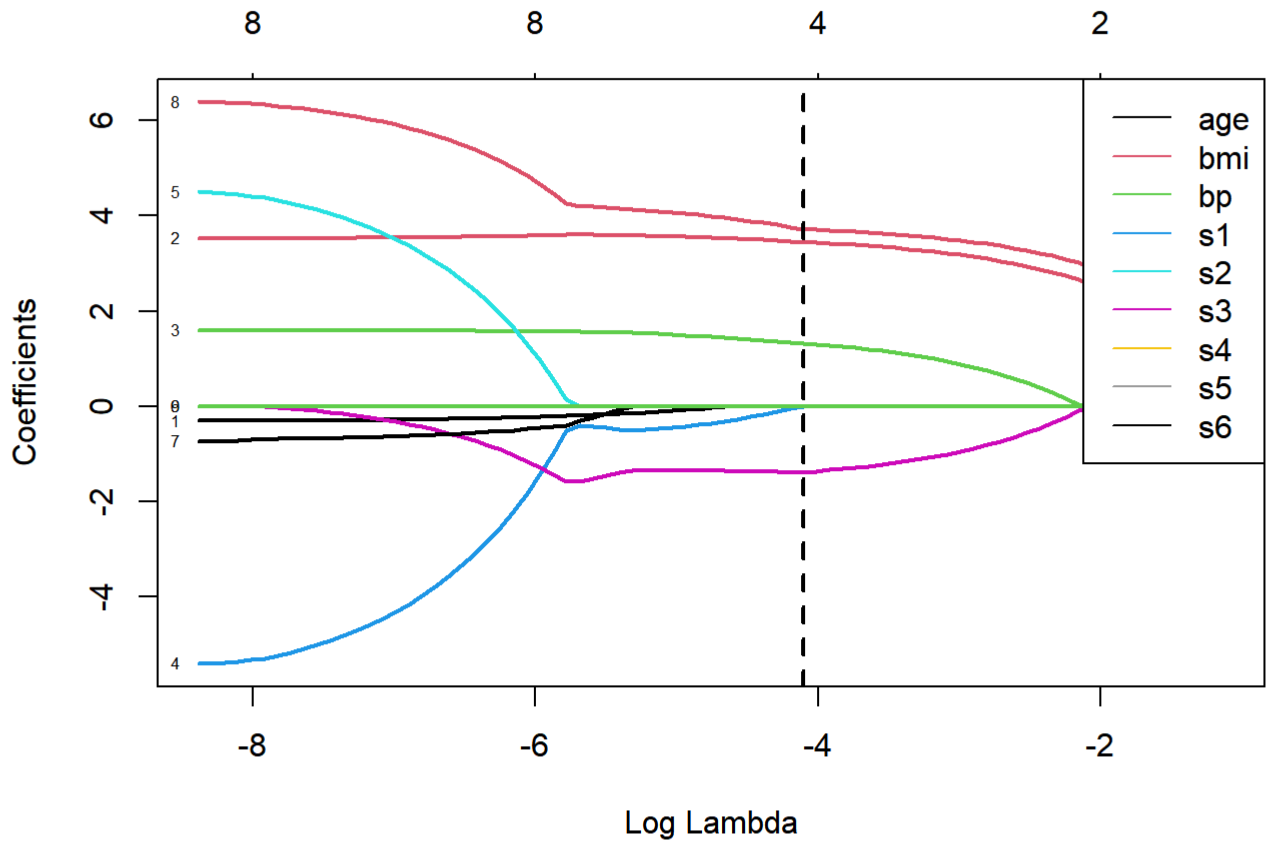
## Coefficient path for Lasso Regression

When lambda is small (far left), the model is less regularized, and the coefficients are large. As lambda increases (moving towards the right), the coefficients shrink towards zero, and some coefficients may eventually reach zero, indicating variable selection.

The plot helps identify the optimal lambda value where the model achieves a balance between bias and variance, typically selected using cross-validation.

```r
# Plot the coefficient path for Lasso regression with coefficients on the y-axis
plot(lasso, xvar = "lambda", label = TRUE, lwd=2)
abline(v=log(lasso.cv$lambda.min), col='black', lty=2, lwd=2)

# Add a legend
legend("topright", legend = colnames(x), col = 1:ncol(x), lty = 1)
```



Variable Selection:

Variables with non-zero coefficients at higher lambda values are retained in the model, indicating their importance in predicting the response variable. Variables with coefficients that shrink to zero at lower lambda values are considered less important and may be dropped from the model.

## Elastic Net Regularization

```
## 10-fold CV to find the optimal lambda
enetmodel.cv = cv.glmnet(x, log(y), alpha=0.5, nfolds=10)

## Fit elastic net model with 100 values for lambda
enetmodel = glmnet(x, log(y), alpha=0.5, nlambda=100)

# Get the optimal lambda value
optimal_lambda_elastic_net <- enetmodel.cv $lambda.min

# Display the optimal lambda value
cat("Optimal lambda value for elastic net regression (Lambda_min):", optimal_lambda_elastic_net, "\n")
```

```
## Optimal lambda value for elastic net regression (Lambda_min): 0.03617111
```

```
# Get the coefficients at the optimal lambda value
coefficients_at_optimal_lambda_enet <- coef(enetmodel.cv, s = "lambda.min")

# Display the coefficients
print(coefficients_at_optimal_lambda_enet)
```

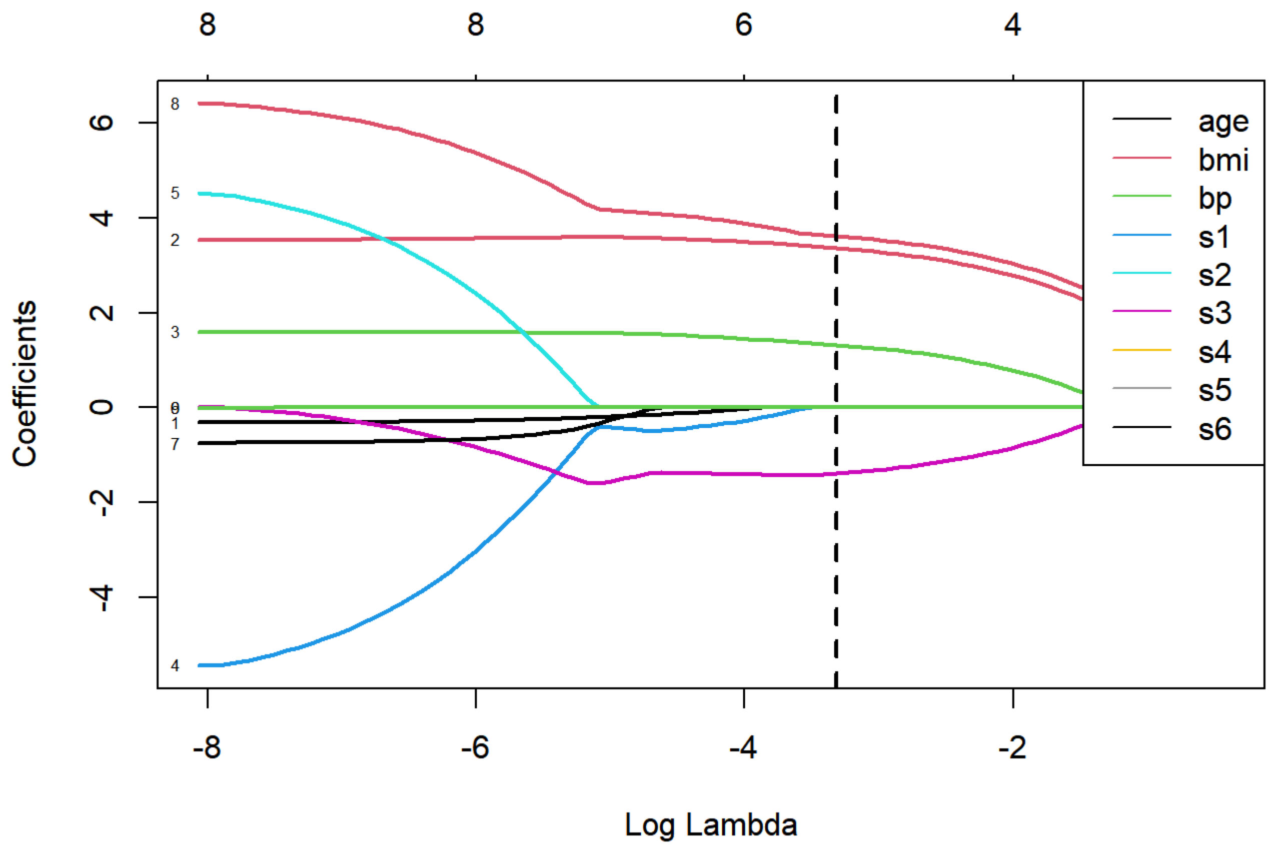
```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept)  4.874264
## age          .
## bmi          3.359092
## bp           1.323940
## s1           .
## s2           .
## s3          -1.386567
## s4           .
## s5           3.612716
## s6           .
```

```
# Count the number of non-zero coefficients (variables selected by lasso regression)
num_selected_variables_enet <- sum(coefficients_at_optimal_lambda_enet != 0)
cat("Number of variables selected by lasso regression:", num_selected_variables_enet, "\n")
```

```
## Number of variables selected by lasso regression: 5
```

```
## Plot coefficient paths
plot(enetmodel, xvar="lambda", label=TRUE, lwd=2)
abline(v=log(enetmodel.cv$lambda.min), col='black', lty=2, lwd=2)

# Add a legend
legend("topright", legend = colnames(x), col = 1:ncol(x), lty = 1)
```



MODEL COMPARISON (model1, model2, Forward_model,

## Backward_model, Best_model)

```
# Assuming you have a separate test dataset stored in 'test_data'
# Predictions using Model1
predictions_model1 <- predict(model1, newdata = test_data)

# Predictions using Model2
predictions_model2 <- predict(model2, newdata = test_data)

# Predictions using Forward_model
predictions_forward <- predict(forward_model, newdata = test_data)

# Predictions using Backward_model
predictions_backward <- predict(backward_model, newdata = test_data)

# Predictions using Best_model
predictions_best <- predict(best_model, newdata = test_data)
```

## MSE comparison - Predictive Accuracy

```
# Calculate mean squared error for each model
mse_model1 <- mean((test_data$Target - predictions_model1)^2)
mse_model2 <- mean((test_data$Target - predictions_model2)^2)
mse_forward <- mean((test_data$Target - predictions_forward)^2)
mse_backward <- mean((test_data$Target - predictions_backward)^2)
mse_best <- mean((test_data$Target - predictions_best)^2)

# Print MSE for each model
cat("Mean Squared Error (MSE) for Model1:", mse_model1, "\n")
```

```
## Mean Squared Error (MSE) for Model1: 3135.175
```

```
cat("Mean Squared Error (MSE) for Model2:", mse_model2, "\n")
```

```
## Mean Squared Error (MSE) for Model2: 3393.35
```

```
cat("Mean Squared Error (MSE) for Forward_model:", mse_forward, "\n")
```

```
## Mean Squared Error (MSE) for Forward_model: 3135.175
```

```
cat("Mean Squared Error (MSE) for Backward_model:", mse_backward, "\n")
```

```
## Mean Squared Error (MSE) for Backward_model: 3167.069
```

```
cat("Mean Squared Error (MSE) for Best_model:", mse_best, "\n")
```

```
## Mean Squared Error (MSE) for Best_model: 3167.069
```

The Mean Squared Error (MSE) values provide a measure of the average squared difference between the predicted and actual values in the test set.

Lower values of MSE indicate better model performance because they suggest that the model's predictions are closer to the actual values. In summary, Model1 and Forward_model seem to be the best-performing models based on MSE.

## R2 & Adjusted-R2 comparison

```
# Create a dataframe to store model information
model_info <- data.frame(
  Model = c("model1", "model2", "forward_model", "backward_model", "best_model"),
  R_squared = c(summary(model1)$r.squared, summary(model2)$r.squared,
                summary(forward_model)$r.squared, summary(backward_model)$r.squared, summary(best_model)$r.squared),
  Adjusted_R_squared = c(summary(model1)$adj.r.squared, summary(model2)$adj.r.squared,
                         summary(forward_model)$adj.r.squared, summary(backward_model)$adj.r.squared, summary(best_model)$ad
j.r.squared)
)

# Display the model comparison
print(model_info)
```

```
##             Model R_squared Adjusted_R_squared
## 1         model1 0.5057000          0.4928052
## 2         model2 0.4847812          0.4803776
## 3  forward_model 0.5057000          0.4928052
## 4 backward_model 0.5030785          0.4959593
## 5     best_model 0.5030785          0.4959593
```

Comparing the R-squared and Adjusted R-squared values across different models provides insights into how well each model explains the variance in the target variable and how well it generalizes to new data, respectively.

A higher R-squared value suggests that the model fits the data better. (Adjusted R-squared adjusts for the number of predictors in the model. It penalizes model with more predictors, preventing overfitting.)

In summary, comparing R-squared and Adjusted R-squared values helps identify which model provides a better balance between goodness of fit and complexity, aiding in model selection.

## AIC & BIC comparison

```
# Create a dataframe to store model information
model_info <- data.frame(
  Model = c("model1", "model2", "forward_model", "backward_model", "best_model"),
  AIC = c(AIC(model1), AIC(model2), AIC(forward_model), AIC(backward_model), AIC(best_model)),
  BIC = c(BIC(model1), BIC(model2), BIC(forward_model), BIC(backward_model), BIC(best_model))
)

# Display the model comparison
print(model_info)
```

```
##             Model      AIC      BIC
## 1         model1 3868.415 3911.008
## 2         model2 3871.129 3890.490
## 3  forward_model 3868.415 3911.008
## 4 backward_model 3862.292 3889.397
## 5     best_model 3862.292 3889.397
```

Again, Model 1 and Forward Model seem to be the preferred choices due to their lower AIC and BIC values, indicating better model fit and parsimony.

Model 2 could be considered as an alternative if we prioritize simplicity over slightly better model fit.

Backward Model and Best Model, while having lower AIC and BIC values compared to Model 2, are still less preferable than Model 1 and Forward Model.

Overall, Model 1 and Forward Model emerge as the top choices based on AIC and BIC criteria.