Feature Selection

Exercise 01

In this exercise, we will work with the mtcars dataset, which is readily available in R.

Part 1 - Linear Regression with 'hp' as a Feature:

1. Perform linear regression with 'hp' as the independent variable and 'mpg' as the target variable using the 'lm' command in base R.
2. Use the 'summary' function to examine the regression results. Comment on the performance of the regression and the importance of the 'hp' feature.
3. Create a scatterplot and overlay the regression line using the `'geom_abline'` command from the ggplot2 library. Provide insights into the visual representation of the regression.

Part 2 - Linear Regression with All Features:

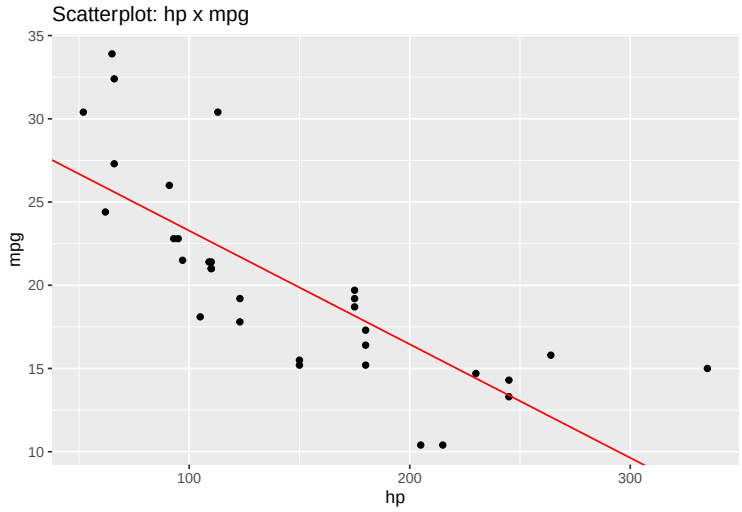
1. Use all features from the mtcars dataset to predict the response variable 'mpg.'
2. Once again, employ the 'summary' command to analyze the results. Assess whether the 'hp' variable maintains its importance compared to the previous step.
3. Discuss the results in terms of collinearity. You can calculate the Variance Inflation Factor (VIF) using the 'vif' command from the 'car' package.

```
library(tidymodels)
df <- as_tibble(mtcars)

# Perform Linear Regression
lin.model <- lm(mpg ~ hp, data = df)
summary(lin.model)
```

```
##
## Call:
## lm(formula = mpg ~ hp, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
## hp          -0.06823    0.01012  -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

```r
# Extract the coefficients (intercept = beta and slope = alfa)
beta <- coef(lin.model)[1]
alfa <- coef(lin.model)[2]

# Shows the scatterplot
ggplot(df, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_abline(intercept = beta, slope = alfa, color = "red") +
  labs(x = "hp", y = "mpg") +
  ggtitle("Scatterplot: hp x mpg")
```




```r
# Notice the negative correlation between these variables
# (the greater the value for hp, the lower the value for mpg)

library(car)
library(knitr)
lin.model <- lm(mpg ~ ., data = df)

# Summarize the results
summary(lin.model)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4506 -1.6044 -0.1196  1.2193  4.6271
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.30337   18.71788   0.657   0.5181
## cyl         -0.11144    1.04502  -0.107   0.9161
## disp         0.01334    0.01786   0.747   0.4635
## hp          -0.02148    0.02177  -0.987   0.3350
## drat         0.78711    1.63537   0.481   0.6353
## wt          -3.71530    1.89441  -1.961   0.0633 .
## qsec         0.82104    0.73084   1.123   0.2739
## vs           0.31776    2.10451   0.151   0.8814
## am           2.52023    2.05665   1.225   0.2340
## gear         0.65541    1.49326   0.439   0.6652
## carb        -0.19942    0.82875  -0.241   0.8122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.65 on 21 degrees of freedom
## Multiple R-squared:  0.869,  Adjusted R-squared:  0.8066
## F-statistic: 13.93 on 10 and 21 DF,  p-value: 3.793e-07
```

```r
# Calculate the Variance Inflation Factor (VIF)
vif_vals <- vif(lin.model)
tibble( var_name = names(vif_vals)
, vif = vif_vals
) %>%
kable()
```

var_name	vif
cyl	15.373833
disp	21.620241
hp	9.832037
drat	3.374620
wt	15.164887
qsec	7.527958
vs	4.965873
am	4.648487
gear	5.357452
carb	7.908747

The p-value for 'hp' is 33.5%. Given that the standard threshold to reject the null hypothesis is 5%, it is evident that 'hp' is not significantly correlated with 'mpg.' Among the variables presented, 'wt' returns the lowest p-value, indicating a higher probability of being correlated with the target variable.

Regarding the VIF of the 'hp' variable, it exhibits a relatively high value. Typically, VIF values above 5 or 10 suggest high collinearity. This implies that 'hp' is likely correlated with another variable in the dataset, which lessens its importance in predicting 'mpg', as shown in the previous result.

Exercise 02

In this exercise, we will use a 2018 FIFA database of soccer players. Conduct a linear regression analysis with this dataset to identify the main predictor variables based on their p-values. Additionally, calculate the Variance Inflation Factor (VIF) using the 'vif' command and discuss the results in terms of collinearity and its implications for the analysis based on p-values.

```
library(car)
file_url = "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5F0ryUTq48kcDTKWTTuk"
df_orign <- read.csv(file_url) %>% as_tibble

# Clean the data
library(stringr)
df <- df_orign %>%
select(Age, Overall, Potential, Wage, Special,
Acceleration, Aggression, Agility, Balance, Ball.control,
Composure, Crossing, Curve, Dribbling, Finishing, Positioning,
Stamina, Interceptions, Strength, Vision, Volleys, Jumping, Penalties,
Shot.power, Sprint.speed, Heading.accuracy, Long.passing, Short.passing
) %>%
mutate( Wage = as.integer(str_extract(Wage,"[0-9]+")) ) %>%
mutate_if(is.character,as.integer) %>%
na.omit()


# Perform Linear Regression taking 'Wage' as the target variable
lin.model <- lm(Wage ~ ., data = df)
summary(lin.model) %>%
tidy() %>%
filter( p.value < 0.001, term != "(Intercept)" ) %>%
kable()
```

term	estimate	std.error	statistic	p.value
Age	-0.3613106	0.0643348	-5.616094	0.0000000
Overall	1.7208726	0.0641596	26.821728	0.0000000
Potential	0.5809598	0.0562196	10.333767	0.0000000
Composure	0.0883276	0.0208442	4.237520	0.0000227
Volleys	0.1068425	0.0202322	5.280813	0.0000001
Jumping	0.0571336	0.0153375	3.725093	0.0001959
Penalties	0.0672668	0.0189706	3.545845	0.0003924

```
# Calculate the Variance Inflation Factor (VIF)
vif_vals <- vif(lin.model)
tibble( var_name = names(vif_vals)
, vif = vif_vals
) %>%
arrange(desc(vif)) %>%
kable()
```

var_name	vif
Special	82.372238
Ball.control	16.445608
Dribbling	14.768679
Short.passing	12.056889
Overall	10.560309
Finishing	9.332033
Positioning	9.204990

var_name	vif
Acceleration	8.330485
Interceptions	8.045896
Long.passing	7.930655
Sprint.speed	7.029132
Volleys	6.767878
Crossing	6.750501
Curve	6.568314
Potential	6.148580
Shot.power	5.862025
Agility	4.837784
Penalties	4.752347
Age	4.642289
Heading.accuracy	4.547185
Vision	4.430197
Aggression	4.010046
Stamina	3.906742
Composure	3.834842
Balance	3.728890
Strength	2.675986
Jumping	1.750718

The summary indicates that the primary features have the lowest p-values: Age, Overall, Potential, Composure, Volleys, Jumping, and Penalties. All of these variables have p-values below 1%, indicating a strong correlation with the target variable (Wage).

It's important to note that some variables have a VIF (Variance Inflation Factor) exceeding 10, such as Overall, Ball.control, Special, Dribbling, and Short.passing. This suggests a strong correlation with other predictors in the dataset.

Other predictor variables have a VIF between 5 and 10, indicating a significant likelihood of collinearity: Positioning, Shot.power, Potential, Interceptions, Sprint.speed, Crossing, Acceleration, Curve, Long.passing, Volleys, and Finishing.

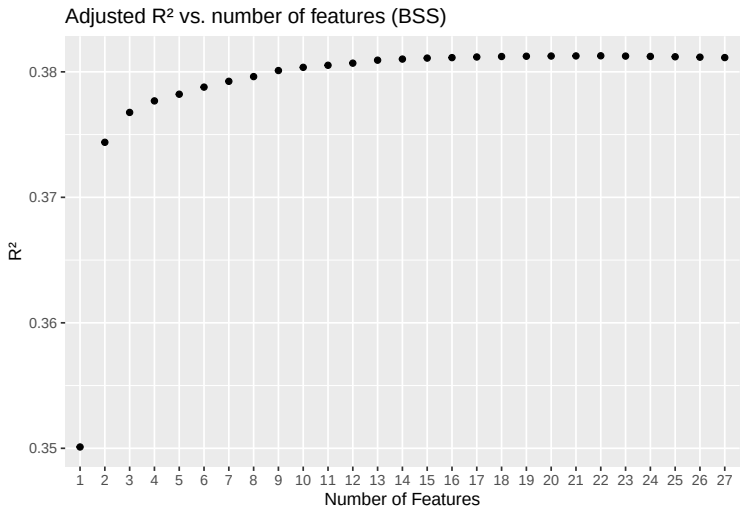
In summary, considering both results, Age, Composure, Jumping, and Penalties exhibit low values for both the p-value and the VIF. Therefore, these features are likely to have a strong correlation with the target variable and a low chance of collinearity with other features.

Exercise 03

```
library(leaps)

# Run the BSS algorithm
regfit.full = regsubsets(Wage ~ ., df, method = "exhaustive", nvmax=nrow(df)-1)
summary.bss = tidy(regfit.full)
summary.bss$num.features <- row.names(summary.bss)
summary.bss %>% View
which.max(summary.bss$adj.r.squared)
```

```
## [1] 22
```

```r
# Shows the plot for adjusted R² vs. number of features
plot <- ggplot(summary.bss, aes(x = reorder(num.features, as.numeric(num.features)), y = adj.r.squared))
  geom_line() +
  geom_point() +
  xlab("Number of Features") +
  ylab("R²") +
  ggtitle("Adjusted R² vs. number of features (BSS)")

print(plot)
```



The results above suggest that the model with the highest adjusted R^2 among the best models with k features is the model with 22 features.

Exercise 04

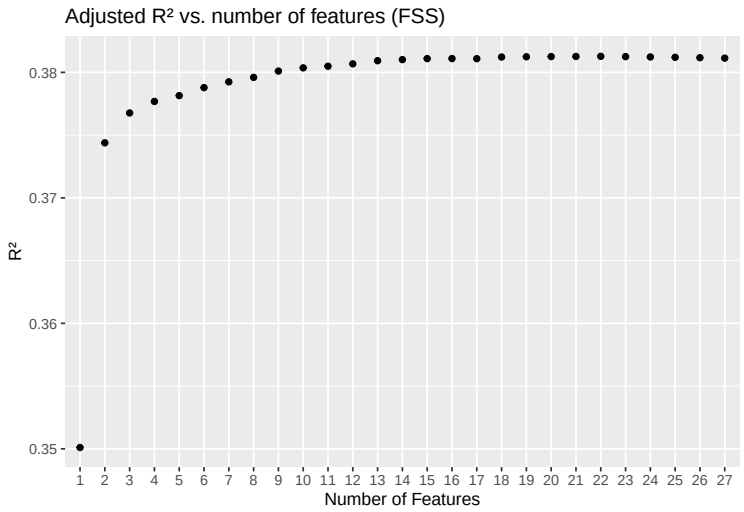
Instead of relying on p-values to identify the most relevant predictors, we will use the Best Subset Selection (BSS) method provided by the 'leaps' package. To view the variables chosen at each level (each one represents a number of features), simply call the 'tidy' function. It returns a tibble that shows the variables included at each level, with TRUE indicating inclusion. To select the best model among the levels, choose the one with the highest adjusted R^2 value.

For a visual representation of the results, create a plot of the adjusted R^2 against the number of features using the table generated by the 'tidy' command. How many predictor variables would you choose?

```r
# Run the FSS algorithm
regfit.forward <- regsubsets(Wage ~., df, method = "forward", nvmax=ncol(df)-1)
summary.fss <- tidy(regfit.forward)
summary.fss$num.features <- row.names(summary.fss)
summary.fss %>% View
which.max(summary.fss$adj.r.squared)
```

```
## [1] 22
```

```r
# Shows the plot for adjusted R² vs. number of features
plot <- ggplot(summary.fss, aes(x = reorder(num.features, as.numeric(num.features)), y = adj.r.squared))
  geom_line(aes(x = reorder(num.features, as.numeric(num.features)), y = adj.r.squared )) +
  geom_point() +
  xlab("Number of Features") +
  ylab("R²") +
  ggtitle("Adjusted R² vs. number of features (FSS)")

print(plot)
```



Similar to the previous exercise, the model with the highest adjusted R^2 among the best models with k features is the model with 22 features.

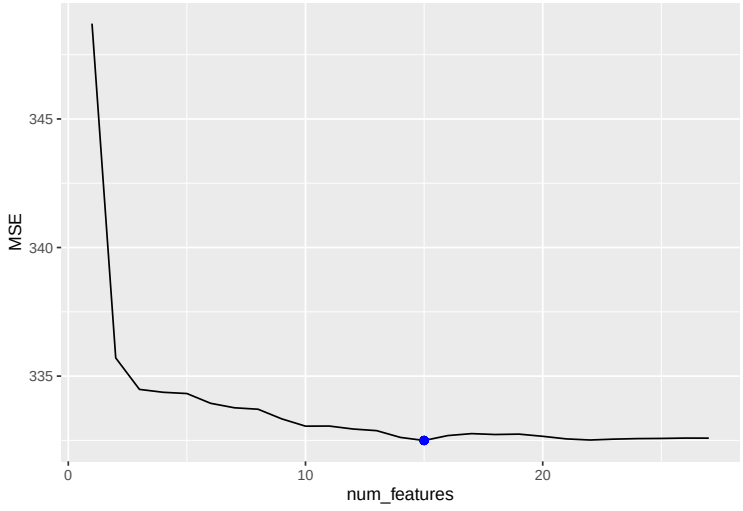
The primary advantage of FSS (Forward Stepwise Selection) over BSS (Backward Stepwise Selection) is its greater computational efficiency. To illustrate this, let's assume d = 20. Using BSS, there would be 1,048,576 different models to be tested. In contrast, FSS reduces this number to 211. However, FSS tests models incrementally, limiting its ability to compare every possible model with k features. Therefore, FSS may not yield the optimal combination of features for each value of k.

Exercise 5

In the previous exercise, we used the adjusted R^2 to select the best set of predictor variables for our model. For more accurate error measurement, it is best practice to employ cross-validation when determining the number of features. You can achieve this by using the 'vfold_cv' command to generate data partitions into folds. Choose 'v = 10' to create a 10-fold cross-validation.

```r
library(rsample)
library(tidyr)
library(leaps)

# Generate 10-fold cross-validation sets
cv.split = vfold_cv(df,v=10)
results <- matrix(0,nrow=nrow(cv.split),ncol=ncol(df)-1)

# Run FSS on each fold
```

```
for( i in 1:nrow(cv.split) ) {
  s = cv.split$splits[[i]]
  train = analysis(s)
  test = assessment(s)
  rss.fit= regsubsets(
      Wage ~., train,
      method = "forward",
      nvmax=ncol(df)-1)

  # Fit and evaluate the error
  rss.td = tidy(rss.fit)

  for( j in 1:nrow(rss.td) ) {
    coefs <- coef(rss.fit,id = j)
    v.names <- names(coefs)
    test.mat<- model.matrix(Wage ~ ., data = test)
    pred <- test.mat[,v.names] %*% coefs
    MSE <- mean(( test$Wage - pred )**2)
    results[i,j] = MSE
  }
}


plot.df <- tibble(num_features = 1:ncol(results),
MSE = colMeans(results))
ggplot(plot.df, aes(x = num_features, y = MSE)) +
geom_line() +
geom_point(aes(x = 15, y = MSE[15]), color = "blue", size = 2)
```



According to the graph, d = 15 appears to be a more appropriate number of features. Below this level, there is little reduction in the MSE (Mean Squared Error).