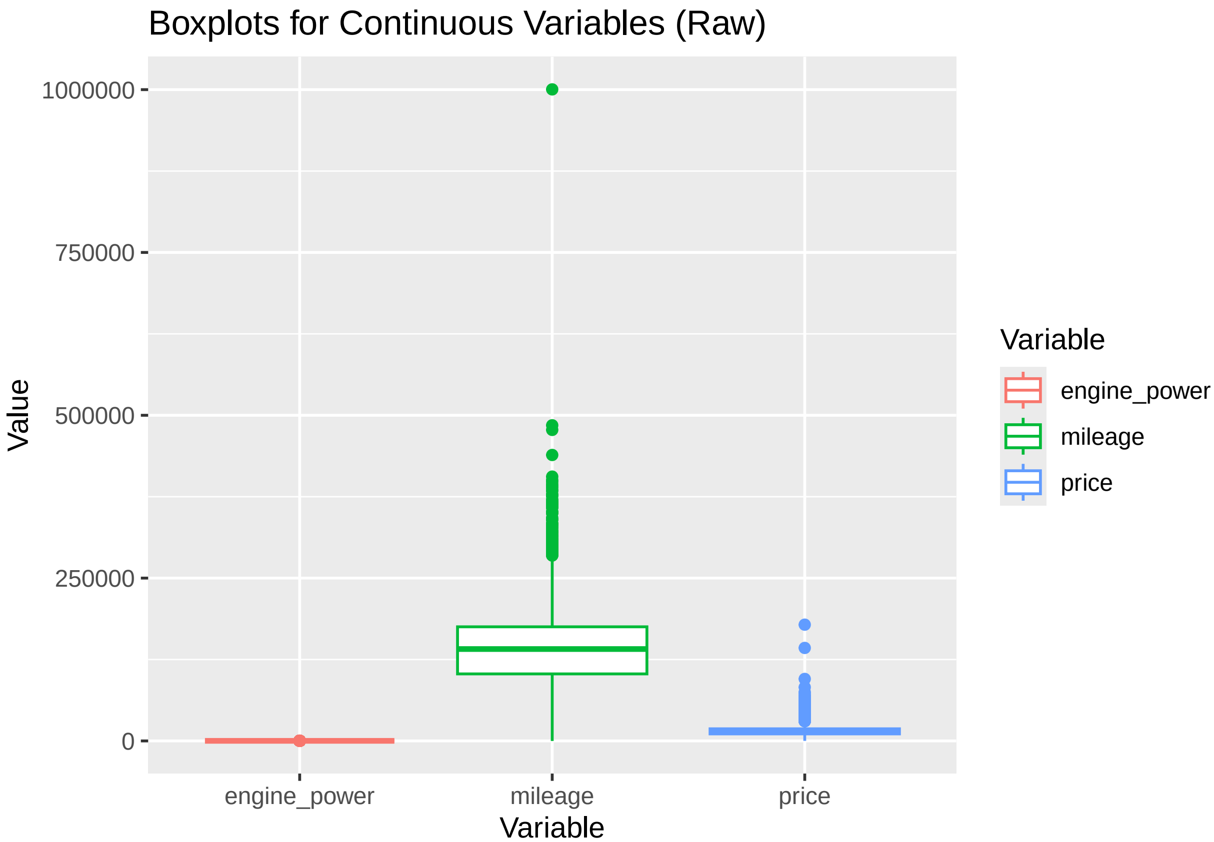
Report3

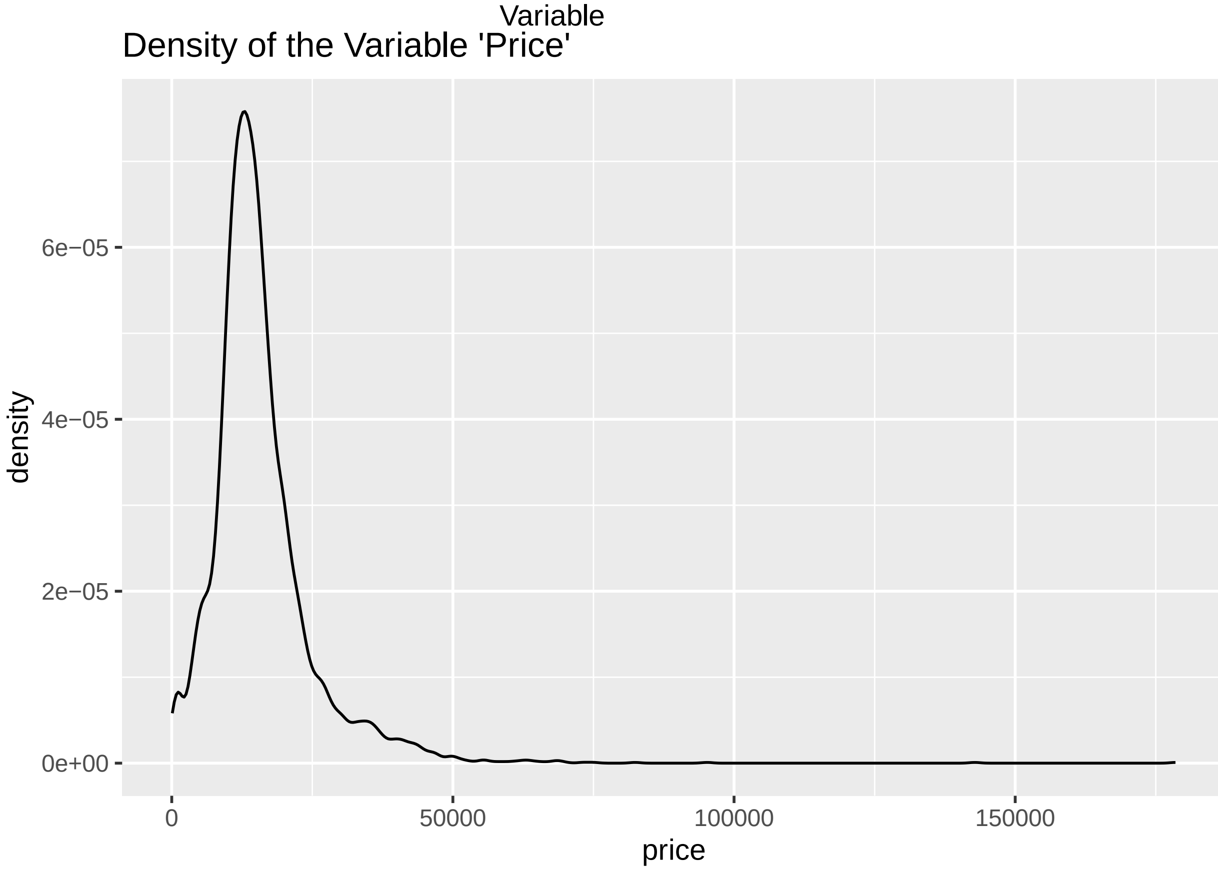
Lab2 Group A: Lee, Joshua; Liu, Kaiyi; Pulsone, Nathaniel; Wang, Mengyao; Xu, Zexian; Yang, Xiaojing

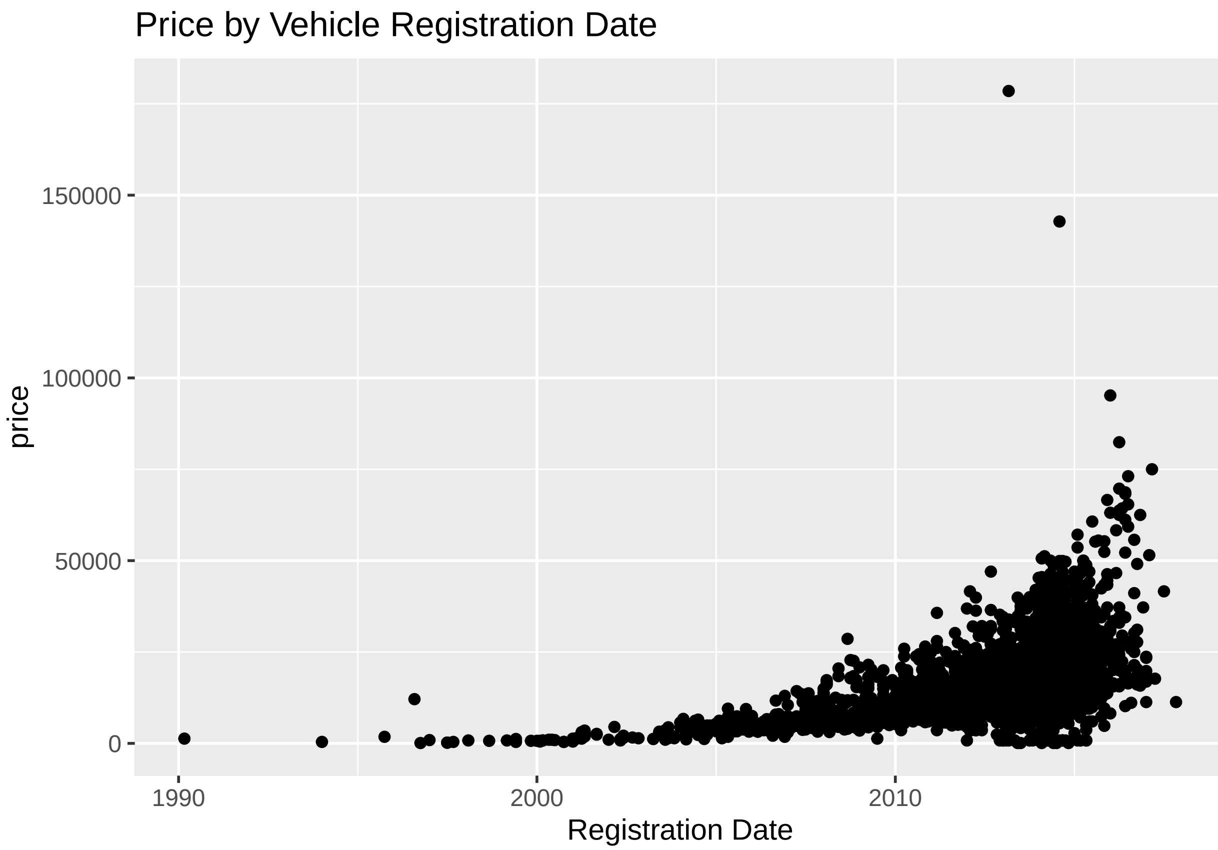
Loading the Library and DataSet

```
mylibrary <- c("tidyverse", "cowplot", "GGally", "MASS", "ggplot2", "glmnet", "data.table", "car", "MLm
invisible(lapply(mylibrary, library, character.only = TRUE))
### load the data
dat_bmw <- read.csv("BMWpricing_updated.csv")
```

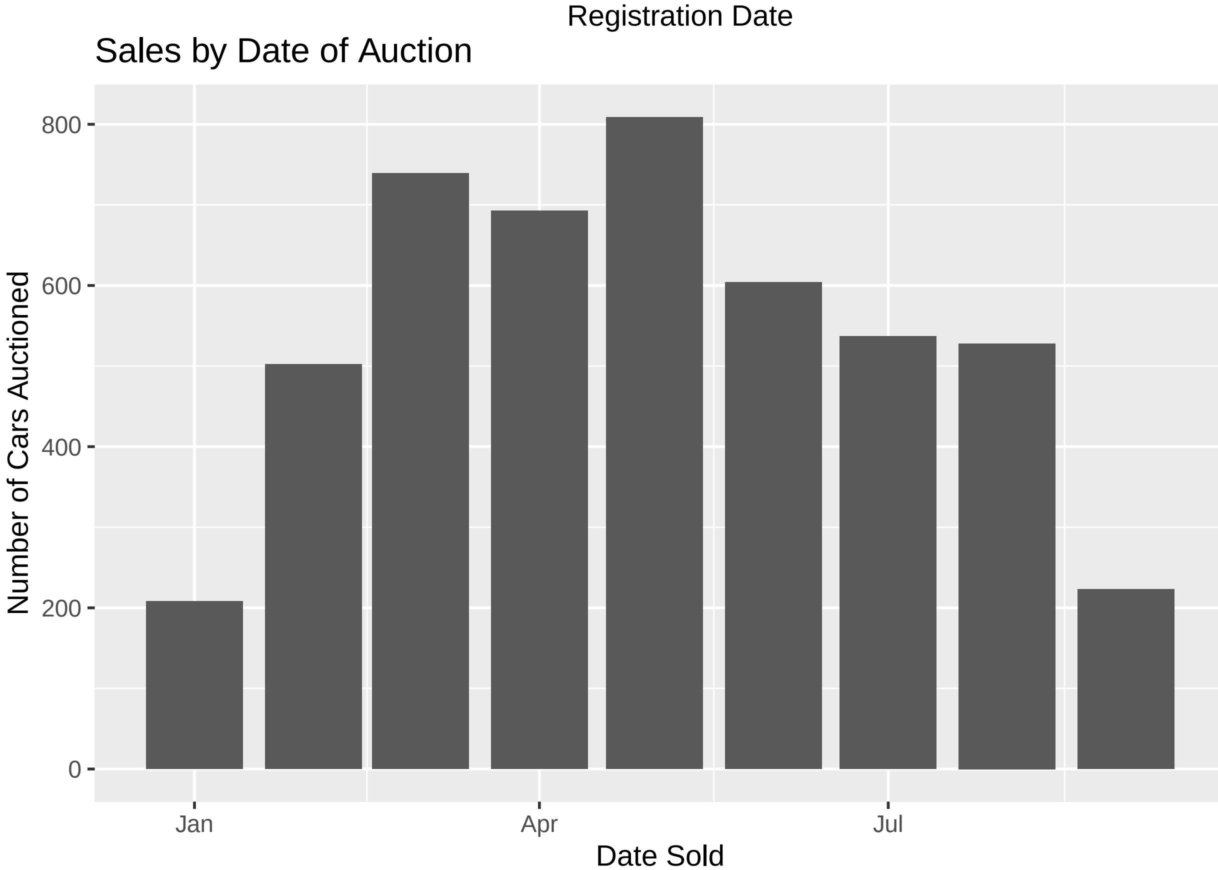
Data Overview







Price by Vehicle Registration Date
150000
100000
50000
0
price
1990
2000
2010
Registration Date



Sales by Date of Auction
800
600
400
200
0
Number of Cars Auctioned
Jan
Apr
Jul
Date Sold

```r
dat_bmw_clean <- dat_bmw|>
  filter(mileage < 500000 & mileage > 0)|>
  filter(price < 100000)|>
  filter(engine_power != 0)

head(dat_bmw_clean)
```

```
##   maker_key model_key mileage engine_power registration_date   fuel paint_color
## 1       BMW       118  140411          100          2/1/2012 diesel       black
## 2       BMW        M4   13929          317          4/1/2016 petrol        grey
## 3       BMW       320  183297          120          4/1/2012 diesel       white
## 4       BMW       420  128035          135          7/1/2014 diesel         red
## 5       BMW       425   97097          160         12/1/2014 diesel      silver
## 6       BMW       335  152352          225          5/1/2011 petrol       black
##      car_type feature_1 feature_2 feature_3 feature_4 feature_5 feature_6
## 1 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
## 2 convertible      TRUE      TRUE     FALSE     FALSE     FALSE      TRUE
## 3 convertible     FALSE     FALSE     FALSE     FALSE      TRUE     FALSE
## 4 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
## 5 convertible      TRUE      TRUE     FALSE     FALSE     FALSE      TRUE
## 6 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
##   feature_7 feature_8 price  sold_at obs_type
## 1      TRUE     FALSE 11300 1/1/2018 Training
## 2      TRUE      TRUE 69700 2/1/2018 Training
## 3      TRUE     FALSE 10200 2/1/2018 Training
## 4      TRUE      TRUE 25100 2/1/2018 Training
## 5      TRUE      TRUE 33400 4/1/2018 Training
## 6      TRUE      TRUE 17100 2/1/2018 Training
```

```r
### Create Model series variable
dat_bmw_clean <- dat_bmw_clean %>%
  mutate(model_series = case_when(
    grepl("^1", model_key) ~ "1_Series",
    grepl("^2", model_key) ~ "2_Series",
    grepl("^3", model_key) ~ "3_Series",
    grepl("^4", model_key) ~ "4_Series",
    grepl("^5", model_key) ~ "5_Series",
    grepl("^7", model_key) ~ "7_Series",
    grepl("^M|M$", model_key) ~ "M_Power",
    model_key %in% c("X1") ~ "X1",
    model_key %in% c("X3") ~ "X3",
    model_key %in% c("X5") ~ "X5",
    model_key %in% c("X6") ~ "X6",
    TRUE ~ "Other"
  ))
```

```r
dat_bmw_clean$sold_at <- as.Date(dat_bmw_clean$sold_at, format = "%m/%d/%Y")
dat_bmw_clean$registration_date <- as.Date(dat_bmw_clean$registration_date, format = "%m/%d/%Y")
dat_bmw_clean$age <- as.numeric((dat_bmw_clean$sold_at - dat_bmw_clean$registration_date) / 365.25)
```

```r
dat_bmw_clean$is_x3 <- ifelse(dat_bmw_clean$model_series == "X3", TRUE, FALSE)
```

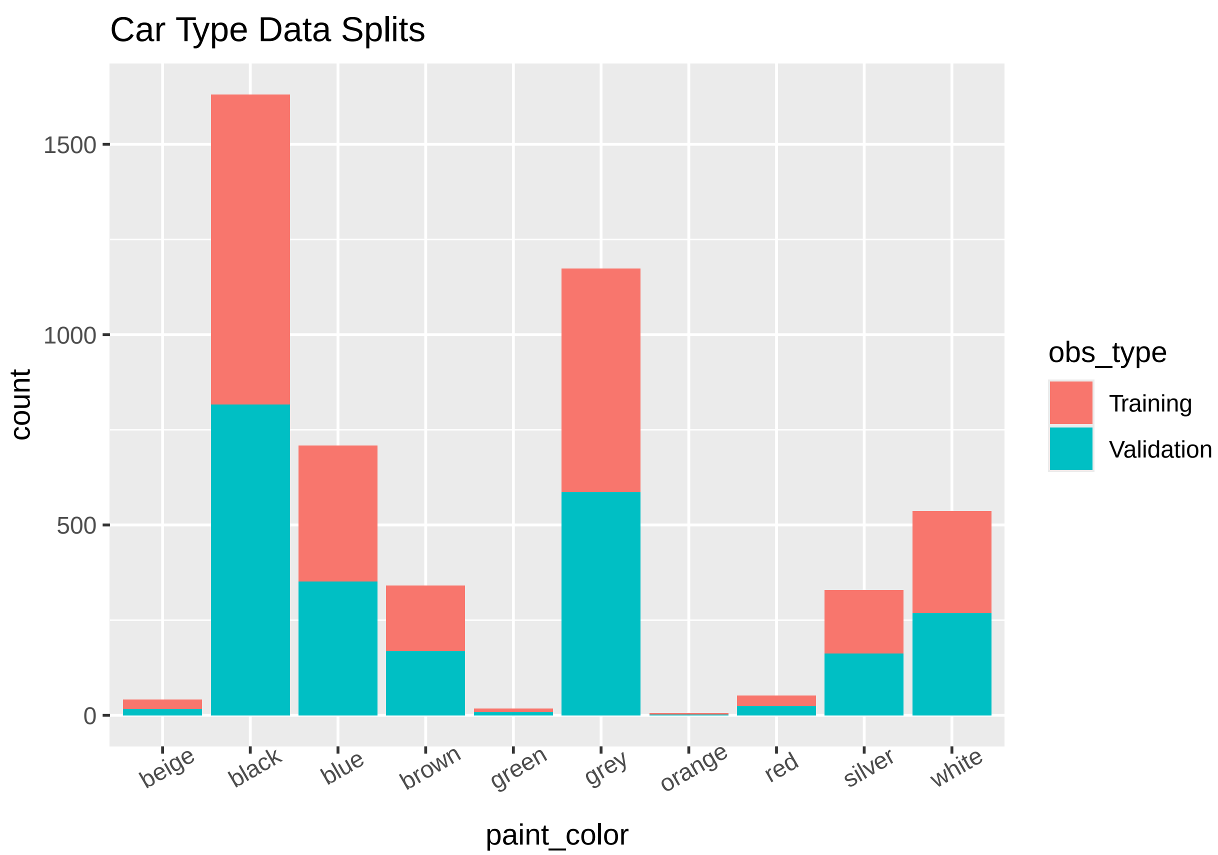
Train Test Split

```
train_index = which(dat_bmw_clean$obs_type == "Training")
test_index = which(dat_bmw_clean$obs_type == "Validation")
dat_bmw_train = dat_bmw_clean[train_index,]
dat_bmw_test = dat_bmw_clean[test_index,]
```

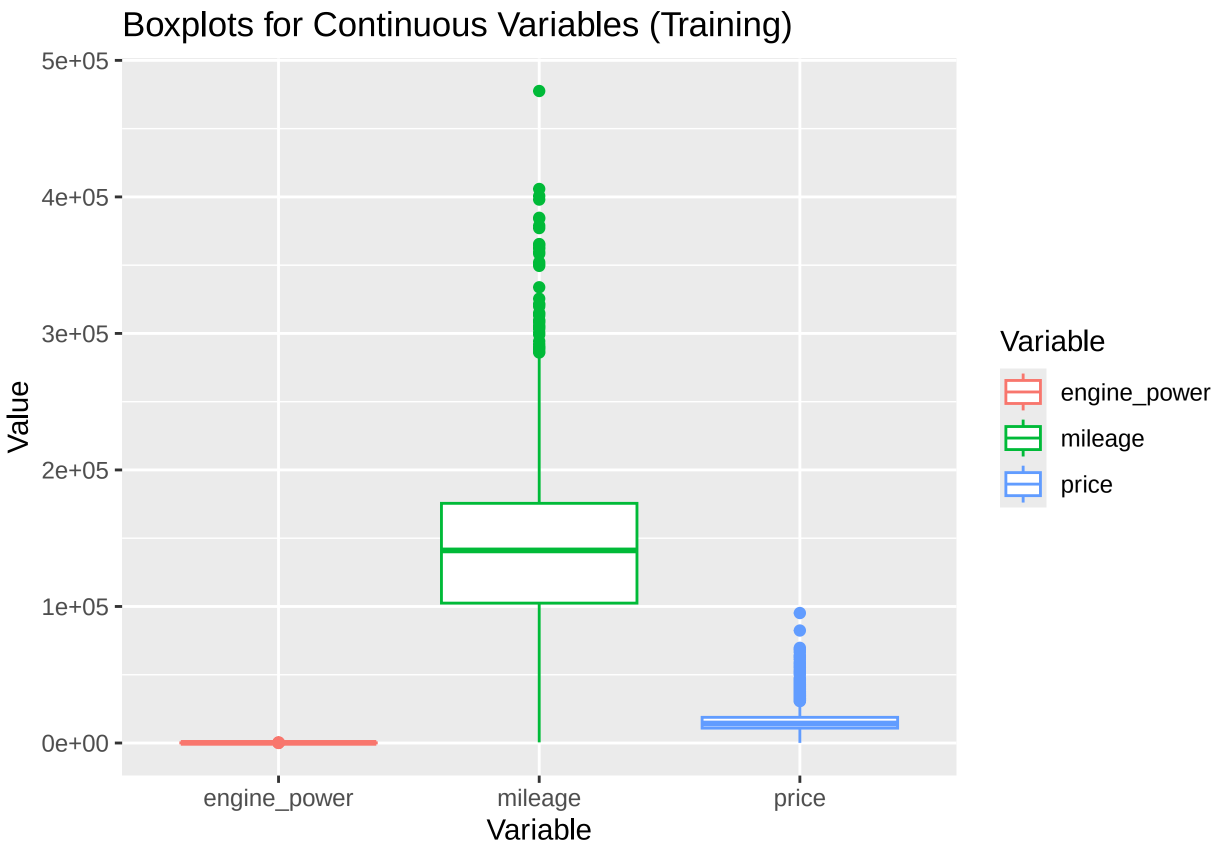
Features of the Training Data

```
dat_bmw_clean|>
  ggplot(aes(x = model_series, fill = obs_type)) +
  geom_bar() +
  labs(title = "Car Type Data Splits") +
  theme(axis.text.x = element_text(angle = 30))
```


```
dat_bmw_clean|>
  ggplot(aes(x = paint_color, fill = obs_type)) +
  geom_bar() +
  labs(title = "Car Type Data Splits") +
  theme(axis.text.x = element_text(angle = 30))
```




```r
dat_bmw_clean[,c("price", "mileage", "engine_power", "obs_type")]|>
  filter(obs_type == "Training")|>
  dplyr::select(price, mileage, engine_power)|>
  pivot_longer(everything(), values_to = "Value", names_to = "Variable") |>
  ggplot() + geom_boxplot(aes(x=Variable, y=Value, color=Variable)) +
  labs(title = "Boxplots for Continuous Variables (Training)")
```




```
### Function to create diagnostic plots
diagPlot<-function(model){
    p1<-ggplot(model, aes(.fitted, .resid))+geom_point()
    p1<-p1+stat_smooth(method="loess")+geom_hline(yintercept=0, col="red", linetype="dashed")
    p1<-p1+xlab("Fitted values")+ylab("Residuals")
    p1<-p1+ggtitle("Residual vs Fitted Plot")+theme_bw()

    p2 <- ggplot(model, aes(sample = .stdresid)) +
      stat_qq() +
      stat_qq_line() +
      xlab("Theoretical Quantiles") +
      ylab("Standardized Residuals") +
      ggtitle("Normal Q-Q") +
      theme_bw()

    p3<-ggplot(model, aes(.fitted, sqrt(abs(.stdresid))))+geom_point(na.rm=TRUE)
    p3<-p3+stat_smooth(method="loess", na.rm = TRUE)+xlab("Fitted Value")
    p3<-p3+ylab(expression(sqrt("|Standardized residuals|")))
    p3<-p3+ggtitle("Scale-Location")+theme_bw()

    p5<-ggplot(model, aes(.hat, .stdresid))+geom_point(na.rm=TRUE)
    p5<-p5+stat_smooth(method="loess", na.rm=TRUE)
    p5<-p5+xlab("Leverage")+ylab("Standardized Residuals")
    p5<-p5+ggtitle("Residual vs Leverage Plot")
    p5<-p5+scale_size_continuous("Cook's Distance", range=c(1,5))
```

```r
    p5<-p5+theme_bw()+theme(legend.position="bottom")

    #return(list(rvfPlot=p1, qqPlot=p2, sclLocPlot=p3, rvlevPlot=p5))
    plot_grid(p1, p2, p3, p5, align = "h")
}
```

Diagonostic of the baseline model

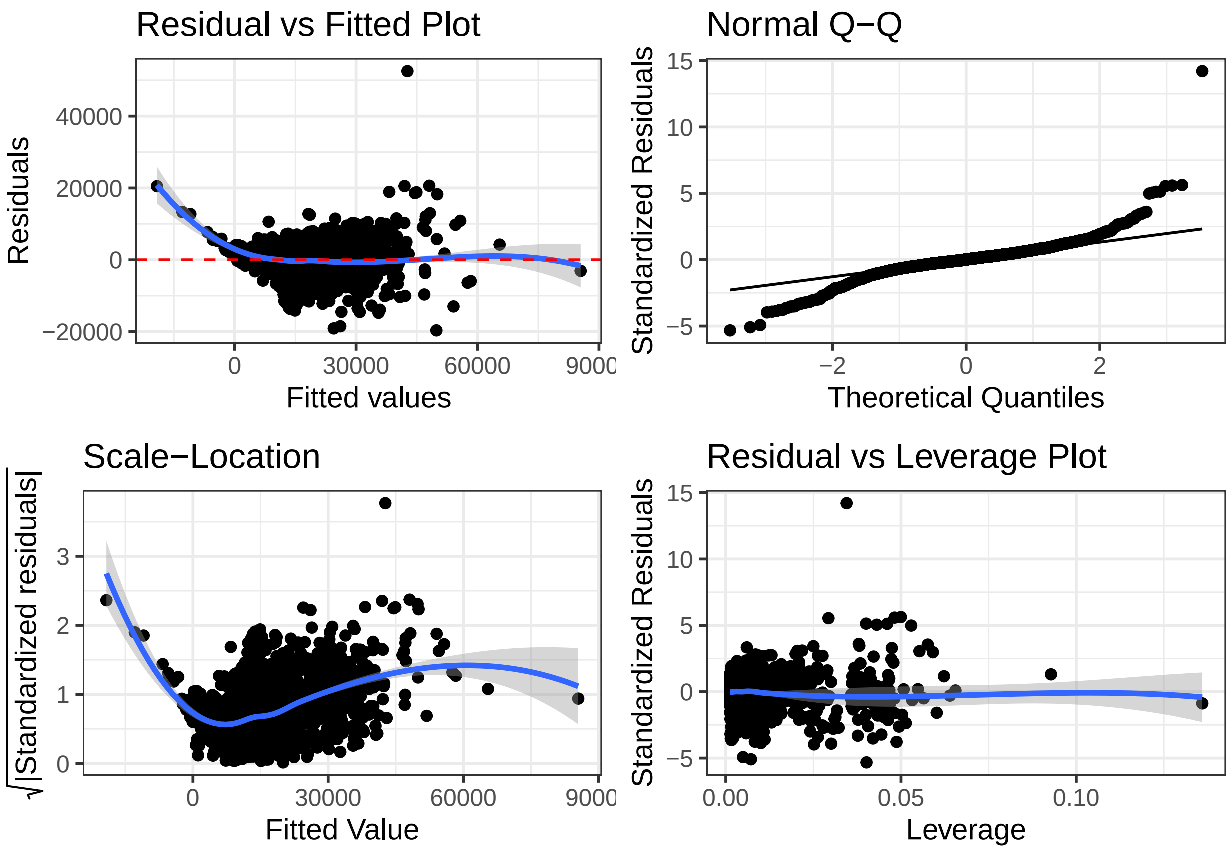
```r
Full_model_train <- lm(price ~ mileage + engine_power + model_series + age + I(mileage^(1/2)) + mileage
summary(Full_model_train)
```

Full model got with the pooled sample (from Report 3)

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series +
##     age + I(mileage^(1/2)) + mileage:engine_power, data = dat_bmw_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -19635  -1583     57   1716  52516
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.423e+04  1.173e+03  12.128  < 2e-16 ***
## mileage               1.289e-01  7.851e-03  16.419  < 2e-16 ***
## engine_power          1.504e+02  5.023e+00  29.945  < 2e-16 ***
## model_series2_Series -3.297e+02  7.598e+02  -0.434 0.664418
## model_series3_Series  1.097e+03  2.676e+02   4.098 4.31e-05 ***
## model_series4_Series  6.372e+03  6.026e+02  10.574  < 2e-16 ***
## model_series5_Series  4.348e+03  3.117e+02  13.952  < 2e-16 ***
## model_series7_Series  1.281e+04  8.726e+02  14.677  < 2e-16 ***
## model_seriesM_Power   1.372e+04  9.143e+02  15.008  < 2e-16 ***
## model_seriesOther     1.198e+04  6.530e+02  18.343  < 2e-16 ***
## model_seriesX1        1.391e+03  3.909e+02   3.559 0.000379 ***
## model_seriesX3        5.376e+03  3.598e+02  14.942  < 2e-16 ***
## model_seriesX5        1.302e+04  4.848e+02  26.852  < 2e-16 ***
## model_seriesX6        1.409e+04  8.929e+02  15.775  < 2e-16 ***
## age                  -1.235e+03  3.590e+01 -34.399  < 2e-16 ***
## I(mileage^(1/2))     -5.973e+01  5.061e+00 -11.803  < 2e-16 ***
## mileage:engine_power -5.994e-04  3.097e-05 -19.353  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3763 on 2414 degrees of freedom
## Multiple R-squared:  0.8231, Adjusted R-squared:  0.8219
## F-statistic:   702 on 16 and 2414 DF,  p-value: < 2.2e-16
```

```r
diagPlot(Full_model_train)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



Lasso Regression for variable Selection

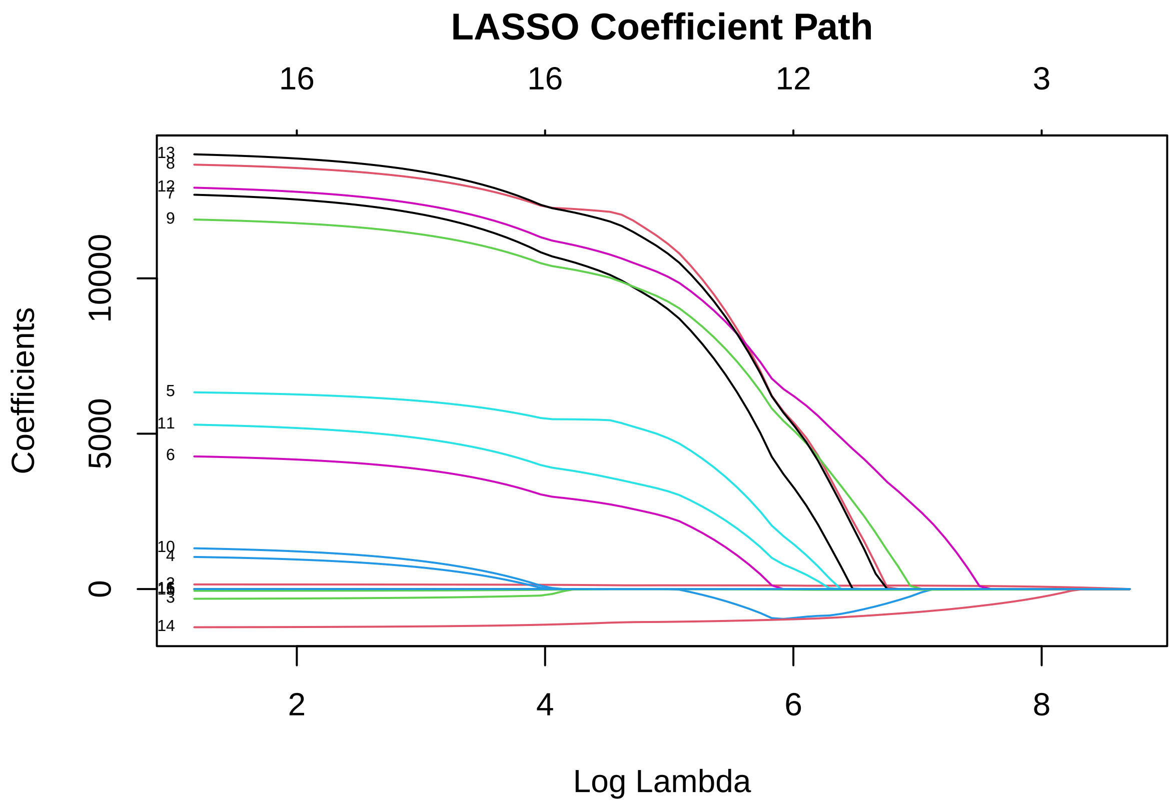
```
# Create a model matrix
X <- model.matrix(price ~ mileage + engine_power + model_series + age + I(mileage^(1/2)) + mileage:engi

# Create the response variable
Y <- dat_bmw_train$price

# set seed for reproductivity
set.seed(20250408)

# Fit the LASSO regression model using glmnet with alpha = 1
lasso_model <- glmnet(x = X, y = Y, alpha = 1)

# The x-axis shows log(lambda) and each line corresponds to a coefficient.
plot(lasso_model, xvar = "lambda", label = TRUE)
title("LASSO Coefficient Path", line = 2.5)
```




```r
# Use cross-validation to determine the lambda that minimizes the mean squared error
cv_model <- cv.glmnet(x = X, y = Y, alpha = 1)
best_lambda <- cv_model$lambda.min

cat("Best Lambda - LASSO:", best_lambda)
```

```
## Best Lambda - LASSO: 3.234687
```

```r
# Display the coefficients at the best lambda value.
lasso_coef <- coef(lasso_model, s = best_lambda)
print(lasso_coef)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                                   s1
## (Intercept)             1.397773e+04
## mileage                 1.233594e-01
## engine_power            1.491882e+02
## model_series2_Series -3.124235e+02
## model_series3_Series  1.033387e+03
## model_series4_Series  6.331205e+03
## model_series5_Series  4.269069e+03
## model_series7_Series  1.269119e+04
## model_seriesM_Power   1.365871e+04
## model_seriesOther     1.189258e+04
## model_seriesX1        1.311942e+03
## model_seriesX3        5.290716e+03
## model_seriesX5        1.291696e+04
## model_seriesX6        1.399076e+04
## age                  -1.228902e+03
## I(mileage^(1/2))     -5.700250e+01
```

```
## mileage:engine_power -5.870496e-04
```

```r
reduced_model = lm(price ~ mileage + engine_power + model_series + age + I(mileage^(1/2)), data = dat_br
anova(Full_model_train, reduced_model)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ mileage + engine_power + model_series + age + I(mileage^(1/2)) +
##     mileage:engine_power
## Model 2: price ~ mileage + engine_power + model_series + age + I(mileage^(1/2))
##   Res.Df        RSS Df   Sum of Sq      F    Pr(>F)
## 1   2414 3.4175e+10
## 2   2415 3.9477e+10 -1 -5302067949 374.52 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Validation and Prediction Power Metrics

```r
validation_data <- subset(dat_bmw_clean, obs_type == "Validation")
newX <- model.matrix(price ~ mileage + engine_power + model_series + age + I(mileage^(1/2)) + mileage:en

best_model <- glmnet(x = X, y = Y, alpha = 1, lambda = best_lambda)

pred_values <- predict(best_model, newx = newX)
obs_values <- validation_data$price


RMSE <- RMSE(pred_values, obs_values)
MAE <- MAE(pred_values, obs_values)
MAPE <- MAPE(pred_values, obs_values)

rbind(RMSE, MAE, MAPE)
```

```
##              [,1]
## RMSE 3826.6398506
## MAE  2538.6284046
## MAPE    0.6997168
```