Used Car Analysis

Laurynas Kanopka, Jaden Smith, Olivia Caruso, Darina Serik

2025-04-10

R Markdown

```
# Clean up our data and put it into a new csv file.
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(readr)

# Read the original dataset
cars_data <- read.csv("used_cars.csv", stringsAsFactors = FALSE)

# Clean the data
cleaned_cars <- cars_data %>%
  mutate(
    # Extract just the numeric part from mileage
    Mileage = as.integer(gsub("[^0-9]", "", milage)),

    # Extract horsepower from engine column
    Horsepower = as.integer(str_extract(engine, "\\d+\\.?\\d*(?=HP|hp)")),

    # Clean model_year to ensure it's just the year
    Model_year = as.integer(model_year),

    # Clean price to extract just the number
    Price = as.integer(gsub("[^0-9]", "", price)),

    # Convert accident to binary (0 for none, 1 for everything else that's not null)
    Accident = ifelse(is.na(accident) | accident == "", NA,
```

```
                    ifelse(str_detect(accident, "None"), 0, 1)),

    # Keep original Brand and Model columns
    Brand = brand,
    Model = model
  ) %>%
  # Select only the columns we want in our final output
  select(Brand, Model, Model_year, Mileage, Horsepower, Accident, Price)

# Save the cleaned data to a new CSV file
write.csv(cleaned_cars, "cleaned_cars.csv", row.names = FALSE)

# Preview the first few rows of the cleaned data
head(cleaned_cars)
```

```
##      Brand                           Model Model_year Mileage Horsepower
## 1     Ford Utility Police Interceptor Base       2013   51000        300
## 2  Hyundai                     Palisade SEL       2021   34742         NA
## 3    Lexus                   RX 350 RX 350       2022   22372         NA
## 4 INFINITI                Q50 Hybrid Sport       2015   88900        354
## 5     Audi       Q3 45 S line Premium Plus       2021    9835         NA
## 6    Acura                         ILX 2.4L       2016  136397         NA
##   Accident Price
## 1        1 10300
## 2        1 38005
## 3        0 54598
## 4        0 15500
## 5        0 34999
## 6        0 14798
```

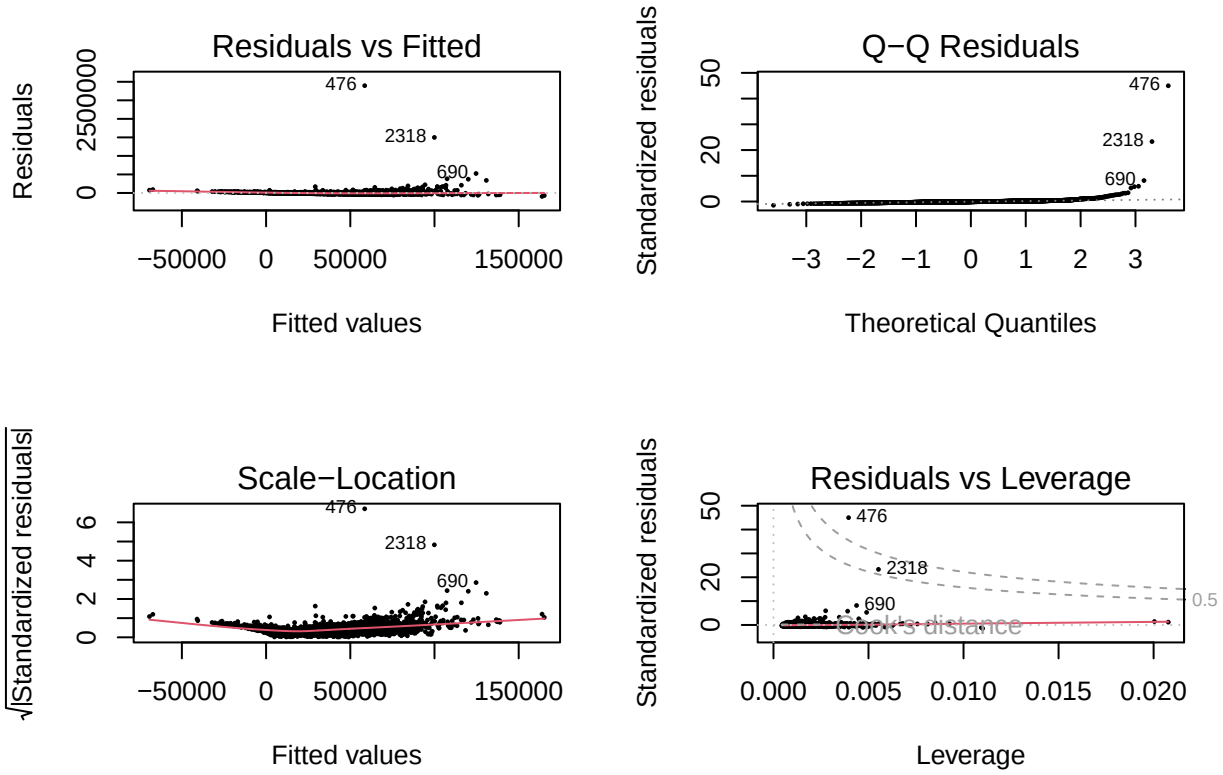
```
# Create a version with no null values

# Read the cleaned data
# We could use the cleaned_cars object directly, but reading from file ensures this chunk can run indep
cleaned_cars <- read.csv("cleaned_cars.csv", stringsAsFactors = FALSE)

# Remove rows with any NA values
cleaned_cars_no_null <- cleaned_cars %>%
  na.omit()

# Save to a new CSV file
write.csv(cleaned_cars_no_null, "cleaned_cars_no_null.csv", row.names = FALSE)

# Show how many rows were removed
cat("Original number of rows:", nrow(cleaned_cars), "\n")
cat("Number of rows after removing nulls:", nrow(cleaned_cars_no_null), "\n")
cat("Number of rows removed:", nrow(cleaned_cars) - nrow(cleaned_cars_no_null), "\n")

# Preview the first few rows
head(cleaned_cars_no_null)

#cat(nrow(cleaned_cars_no_null))
```

```
## Original number of rows: 4009
## Number of rows after removing nulls: 3118
## Number of rows removed: 891
##       Brand                          Model Model_year Mileage Horsepower
## 1      Ford Utility Police Interceptor Base       2013   51000        300
## 4  INFINITI                Q50 Hybrid Sport       2015   88900        354
## 7      Audi            S3 2.0T Premium Plus       2017   84000        292
## 8       BMW                          740 iL       2001  242000        282
## 9     Lexus                  RC 350 F Sport       2021   23436        311
## 10    Tesla         Model X Long Range Plus       2020   34000        534
##    Accident Price
## 1         1 10300
## 4         0 15500
## 7         0 31000
## 8         0  7300
## 9         0 41927
## 10        0 69950
```

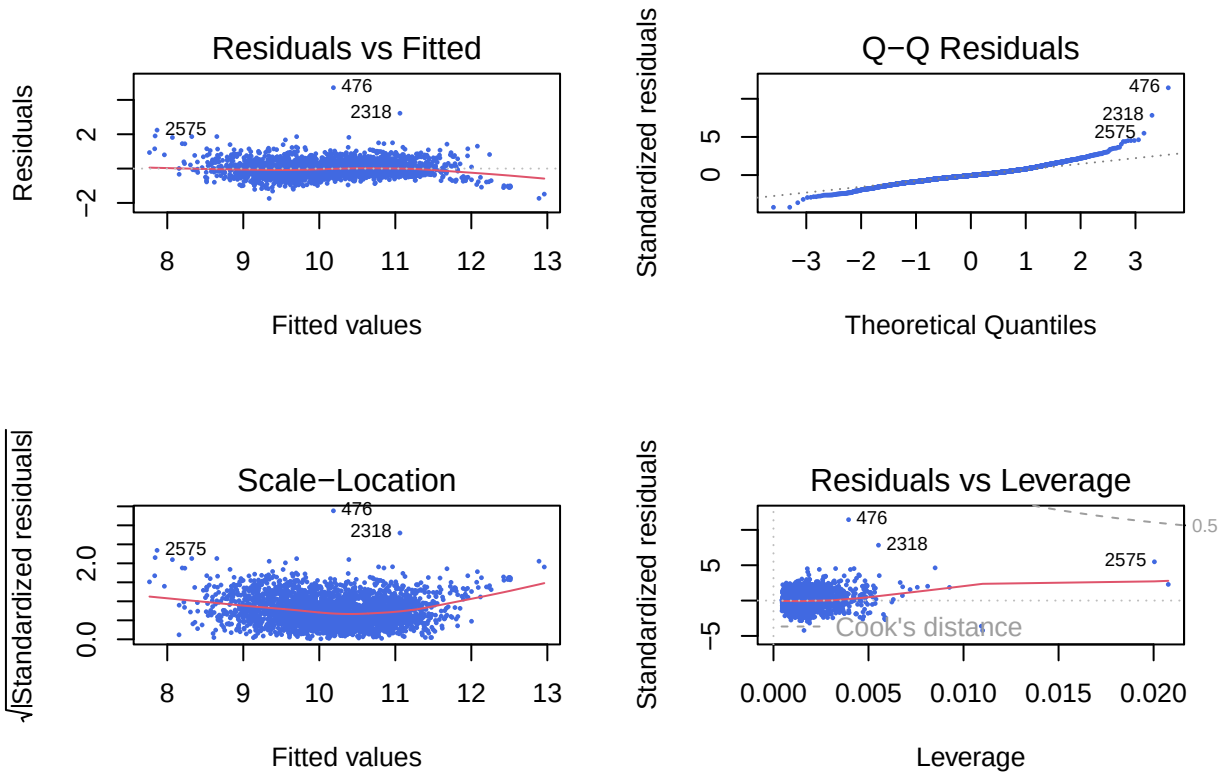
```r
car_data <- read.csv("cleaned_cars_no_null.csv")
attach(car_data)

# Convert Accident to factor
car_data$Accident <- factor(car_data$Accident)

# Summary statistics
summary(car_data[,c("Price","Mileage","Horsepower")])
```

```
##      Price            Mileage         Horsepower
##  Min.   :   2000   Min.   :   100   Min.   :  70.0
##  1st Qu.:  15461   1st Qu.: 29619   1st Qu.: 248.0
##  Median :  28000   Median : 62630   Median : 310.0
##  Mean   :  38602   Mean   : 71861   Mean   : 331.4
##  3rd Qu.:  47000   3rd Qu.:102342   3rd Qu.: 400.0
##  Max.   :2954083   Max.   :405000   Max.   :1020.0
```

```r
# Standard model
mod_std <- lm(Price ~ Mileage + Horsepower + Accident + Model_year, data=car_data)
summary(mod_std)

# Log-transformed model
mod_log <- lm(log(Price) ~ Mileage + Horsepower + Accident + Model_year, data=car_data)
summary(mod_log)

# Residual diagnostics for standard model
par(mfrow=c(2,2))
plot(mod_std, pch=16, cex=0.4, col="black")
```




```
par(mfrow=c(1,1))


# Residual diagnostics for log-transformed model
par(mfrow=c(2,2))
plot(mod_log, pch=16, cex=0.4, col="royalblue")
```

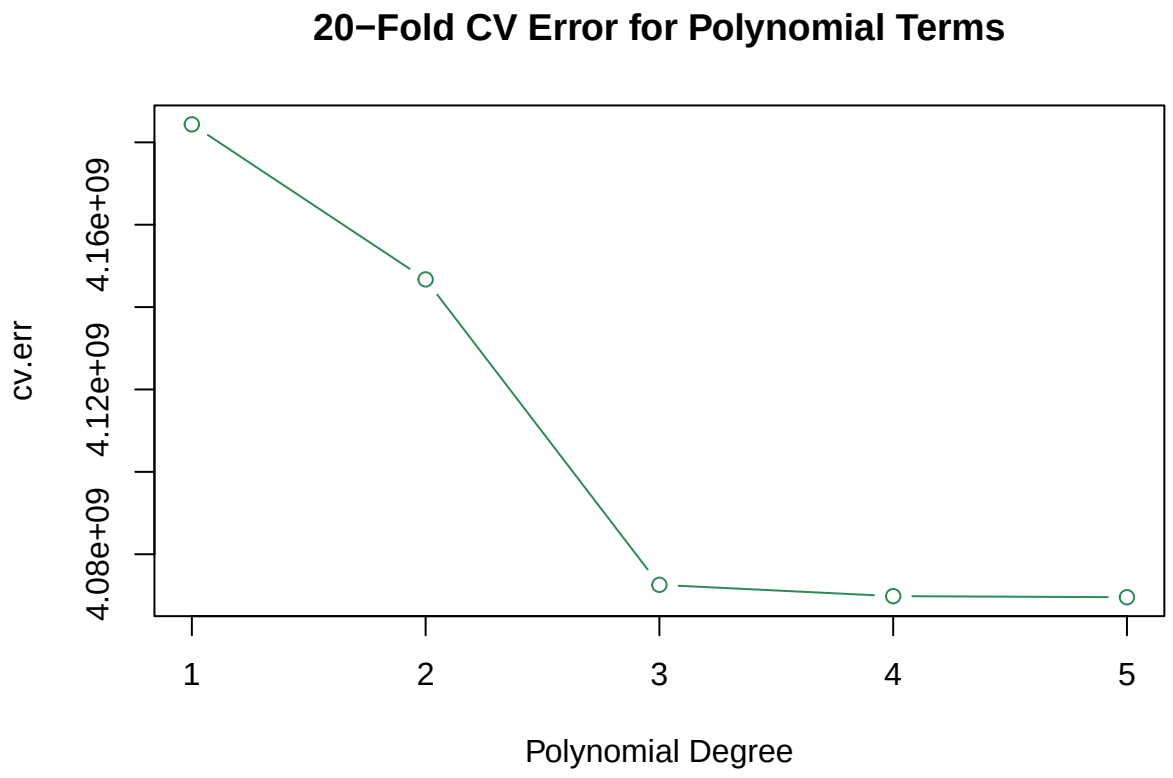



```
par(mfrow=c(1,1))
```

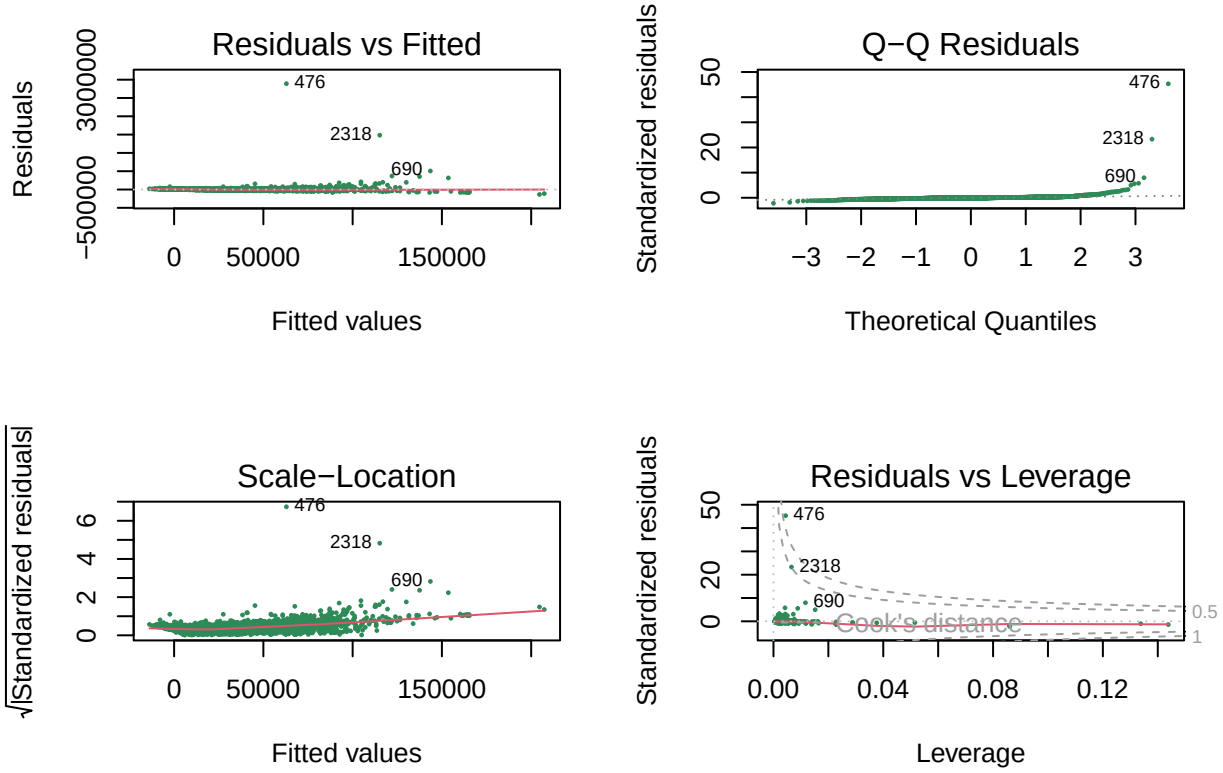
```
##
## Call:
## lm(formula = Price ~ Mileage + Horsepower + Accident + Model_year,
##     data = car_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -93877  -14036   -3810    6161 2895532
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.619e+04  5.008e+05  -0.092    0.927
## Mileage     -2.522e-01  2.829e-02  -8.916   <2e-16 ***
## Horsepower   1.627e+02  1.022e+01  15.912   <2e-16 ***
## Accident1   -8.351e+01  2.674e+03  -0.031    0.975
## Model_year   2.434e+01  2.483e+02   0.098    0.922
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64490 on 3113 degrees of freedom
## Multiple R-squared:  0.1545, Adjusted R-squared:  0.1534
## F-statistic: 142.2 on 4 and 3113 DF,  p-value: < 2.2e-16
##
```

```
##
## Call:
## lm(formula = log(Price) ~ Mileage + Horsepower + Accident + Model_year,
##     data = car_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7398 -0.2233 -0.0212  0.1949  4.7129
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.051e+01  3.204e+00 -22.005  < 2e-16 ***
## Mileage     -5.778e-06  1.810e-07 -31.920  < 2e-16 ***
## Horsepower   3.093e-03  6.542e-05  47.277  < 2e-16 ***
## Accident1   -6.343e-02  1.711e-02  -3.708 0.000213 ***
## Model_year   3.977e-02  1.588e-03  25.035  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4127 on 3113 degrees of freedom
## Multiple R-squared:  0.7542, Adjusted R-squared:  0.7539
## F-statistic:  2388 on 4 and 3113 DF,  p-value: < 2.2e-16
```

```r
set.seed(3180)
cv.err <- rep(0,5)
for (i in 1:5) {
  glm.fit <- glm(Price ~ poly(Mileage,i) + poly(Horsepower,i) + Accident + Model_year,
                 data=car_data)
  cv.err[i] <- cv.glm(car_data, glm.fit, K=20)$delta[1]
}
plot(1:5, cv.err, type="b", col="seagreen", xlab="Polynomial Degree",
     main="20-Fold CV Error for Polynomial Terms")
```



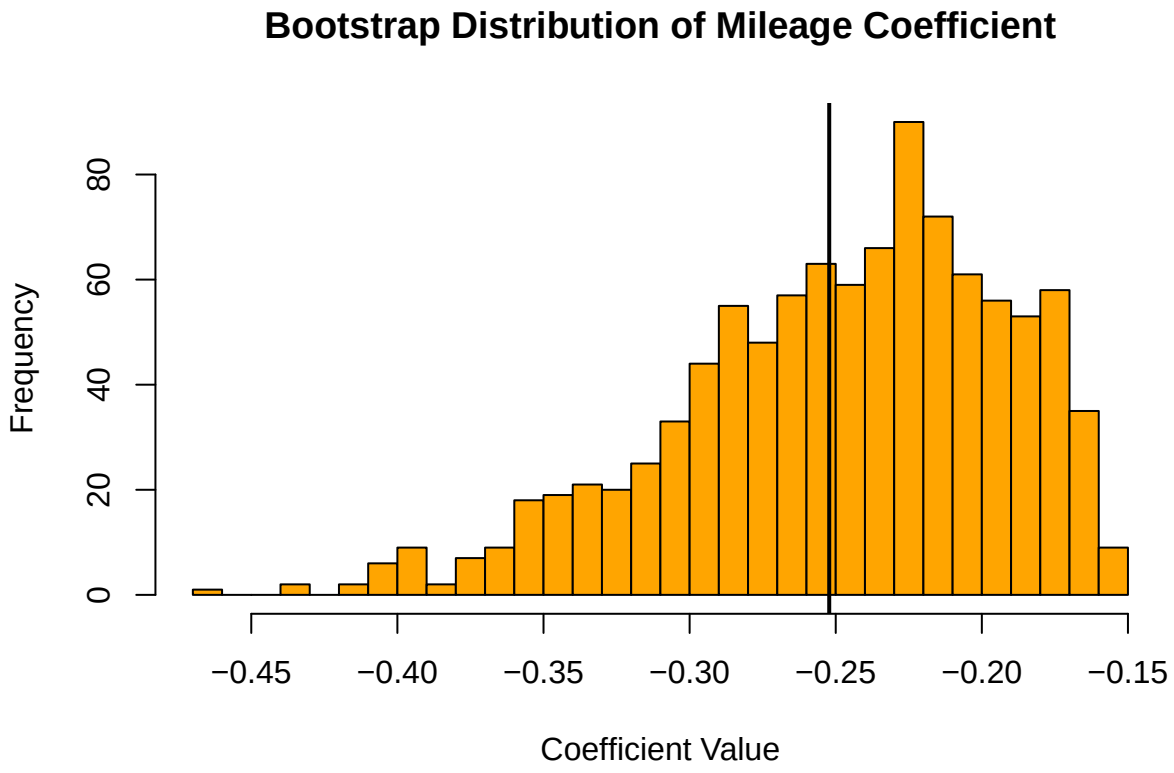

```r
# Fit optimal degree (example: 2)
mod_poly <- lm(Price ~ poly(Mileage,2) + poly(Horsepower,2) + Accident + Model_year,
               data=car_data)
summary(mod_poly)

# Residual diagnostics for log-transformed model
par(mfrow=c(2,2))
plot(mod_poly, pch=16, cex=0.4, col="seagreen")
```




```
par(mfrow=c(1,1))
```

```
##
## Call:
## lm(formula = Price ~ poly(Mileage, 2) + poly(Horsepower, 2) +
##     Accident + Model_year, data = car_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -134655  -12066   -2598    6473 2891185
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             598674.8   505748.3   1.184   0.2366
## poly(Mileage, 2)1      -841966.9    84996.0  -9.906  < 2e-16 ***
## poly(Mileage, 2)2       455935.4    66329.0   6.874 7.52e-12 ***
## poly(Horsepower, 2)1 1043123.7    69226.6  15.068  < 2e-16 ***
## poly(Horsepower, 2)2  157130.2    64244.3   2.446   0.0145 *
## Accident1               1461.1     2659.6   0.549   0.5828
## Model_year              -278.2      251.0  -1.108   0.2679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 63940 on 3111 degrees of freedom
## Multiple R-squared:  0.1696, Adjusted R-squared:  0.168
## F-statistic: 105.9 on 6 and 3111 DF,  p-value: < 2.2e-16
```

```r
alpha.fn <- function(data, index) {
  coef(lm(Price ~ Mileage + Horsepower + Accident + Model_year,
          data=data, subset=index))[2]  # Focus on Mileage ONLY
}

set.seed(123)
boot.res <- boot(car_data, alpha.fn, R=1000)

# Use percentile method instead of bca
boot.ci(boot.res, type="perc", index=1)

# Plot bootstrap distribution
hist(boot.res$t, breaks=30, col="orange",
     main="Bootstrap Distribution of Mileage Coefficient",
     xlab="Coefficient Value")
abline(v=boot.res$t0, col="black", lwd=2)
```




```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
```

```
## 95%   (-0.3764, -0.1657 )
## Calculations and Intervals on Original Scale
```

```
# Function to calculate metrics
calc_metrics <- function(model, data) {
  pred <- predict(model, data)
  residuals <- data$Price - pred
  rmse <- sqrt(mean(residuals^2))
  mae <- mean(abs(residuals))
  r2 <- summary(model)$r.squared
  return(c(RMSE=rmse, MAE=mae, R2=r2))
}

metrics <- rbind(
  Standard = calc_metrics(mod_std, car_data),
  Log_Transformed = calc_metrics(mod_log, car_data),
  Polynomial = calc_metrics(mod_poly, car_data)
)
print(metrics)
```

```
##                      RMSE      MAE        R2
## Standard        64442.22 16657.87 0.1544548
## Log_Transformed 80004.08 38591.49 0.7542018
## Polynomial      63863.94 15723.78 0.1695621
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

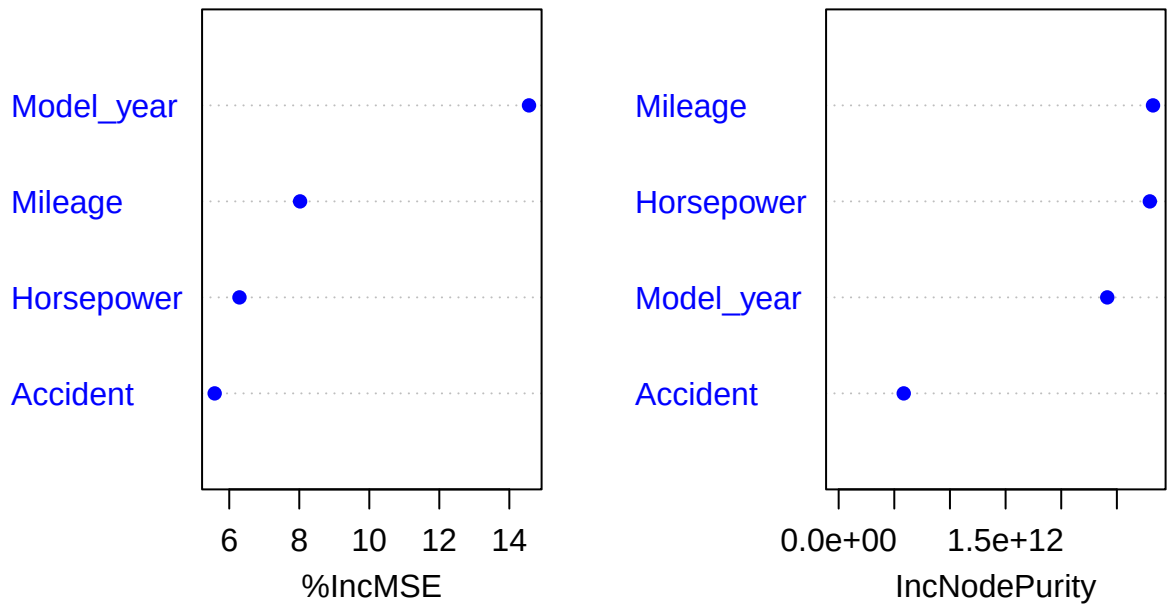
```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
set.seed(3180)
rf.fit <- randomForest(Price ~ Mileage + Horsepower + Accident + Model_year,
                       data=car_data,
                       importance=TRUE,
                       ntree=500)

# Variable importance plot
varImpPlot(rf.fit, pch=16, col="blue",
           main="Variable Importance in Random Forest")
```

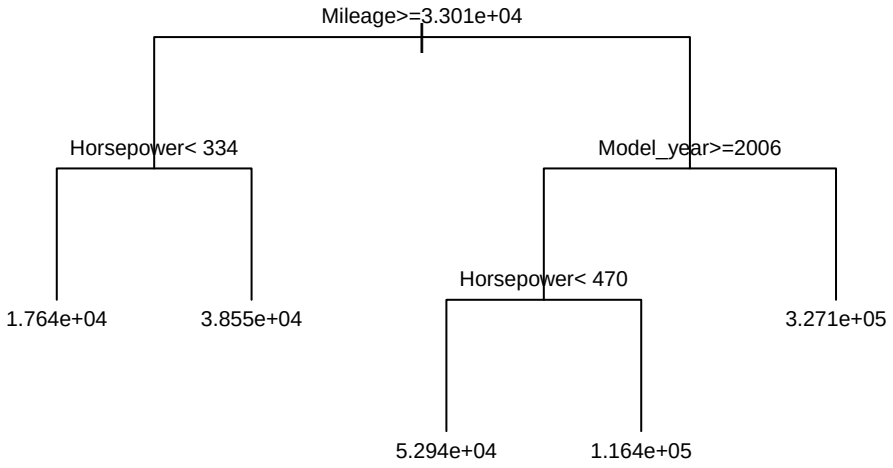



```
# Get predictions and calculate metrics
rf.pred <- predict(rf.fit, car_data)
rf.resid <- car_data$Price - rf.pred

# Add to model comparison table
rfmetrics <- rbind(
  metrics,
  Random_Forest = c(
    RMSE = sqrt(mean(rf.resid^2)),
    MAE = mean(abs(rf.resid)),
    R2 = cor(car_data$Price, rf.pred)^2  # Pseudo R^2
  )
)

# Update comparison table
print(rfmetrics)
```

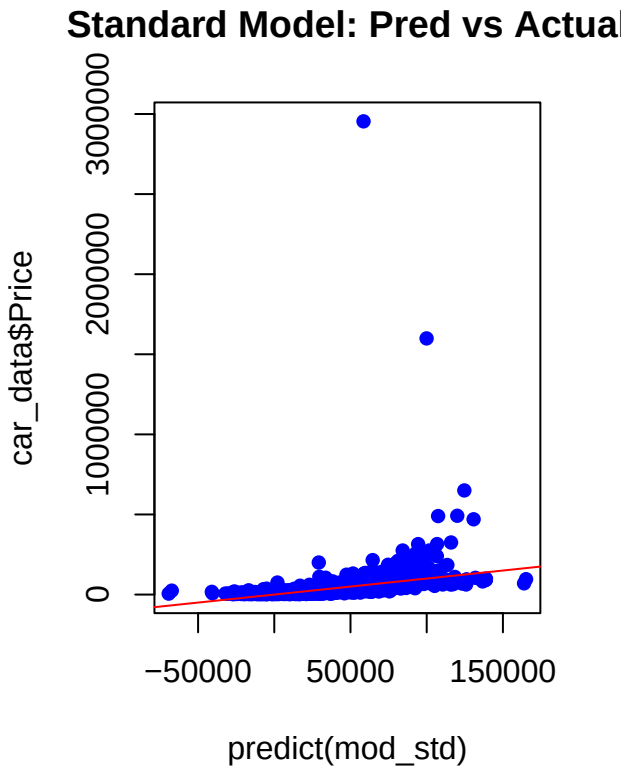
```
##                     RMSE      MAE        R2
## Standard        64442.22 16657.87 0.1544548
## Log_Transformed 80004.08 38591.49 0.7542018
## Polynomial      63863.94 15723.78 0.1695621
## Random_Forest   44070.88 11508.85 0.7952305
```

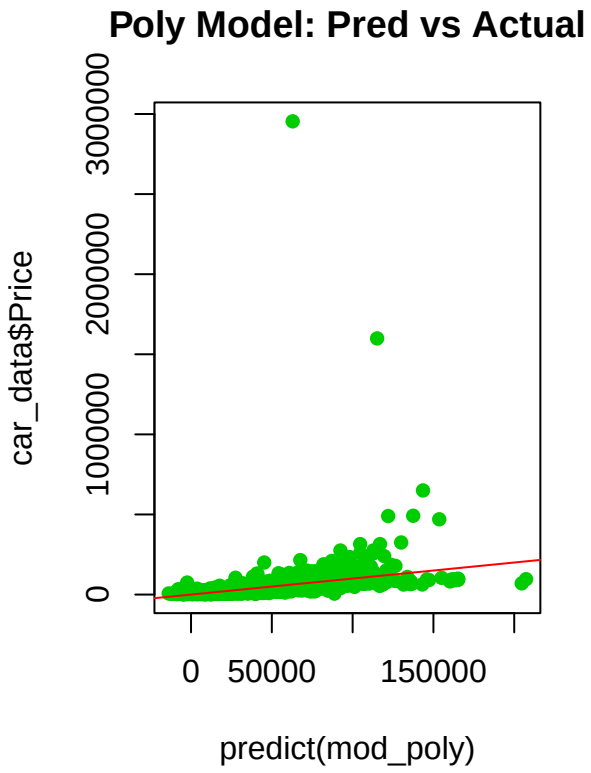
```
library(rpart)
tree.mod <- rpart(Price ~ Mileage + Horsepower + Accident + Model_year,
```

```
                  data=car_data, method="anova")
plot(tree.mod, uniform=TRUE, margin=0.1)
text(tree.mod, cex=0.7)
```




```
# Predicted vs Actual
par(mfrow=c(1,2))
plot(predict(mod_std), car_data$Price, pch=16, col="blue",
     main="Standard Model: Pred vs Actual")
abline(0,1, col="red")

plot(predict(mod_poly), car_data$Price, pch=16, col="green3",
     main="Poly Model: Pred vs Actual")
abline(0,1, col="red")
```






```
par(mfrow=c(1,1))
```