Used Car Analysis

Laurynas Kanopka, Jaden Smith, Olivia Caruso, Darina Serik

2025-04-10

Introduction

This analysis explores factors affecting used car prices using statistical modeling techniques. We examine how mileage, horsepower, accident history, and model year influence car prices, with the goal of building accurate predictive models.

Data Preparation

First, we'll load necessary packages and clean the raw data.

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

```r
  # 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
```

Handling Missing Values

Some of our data points are missing information such as if the car was in an accident or not. Since we are uncertain of the real value, we can't assume and set the values ourselves, so we'll remove rows with missing values to ensure that we have complete cases for our analysis/

```r
# 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
```

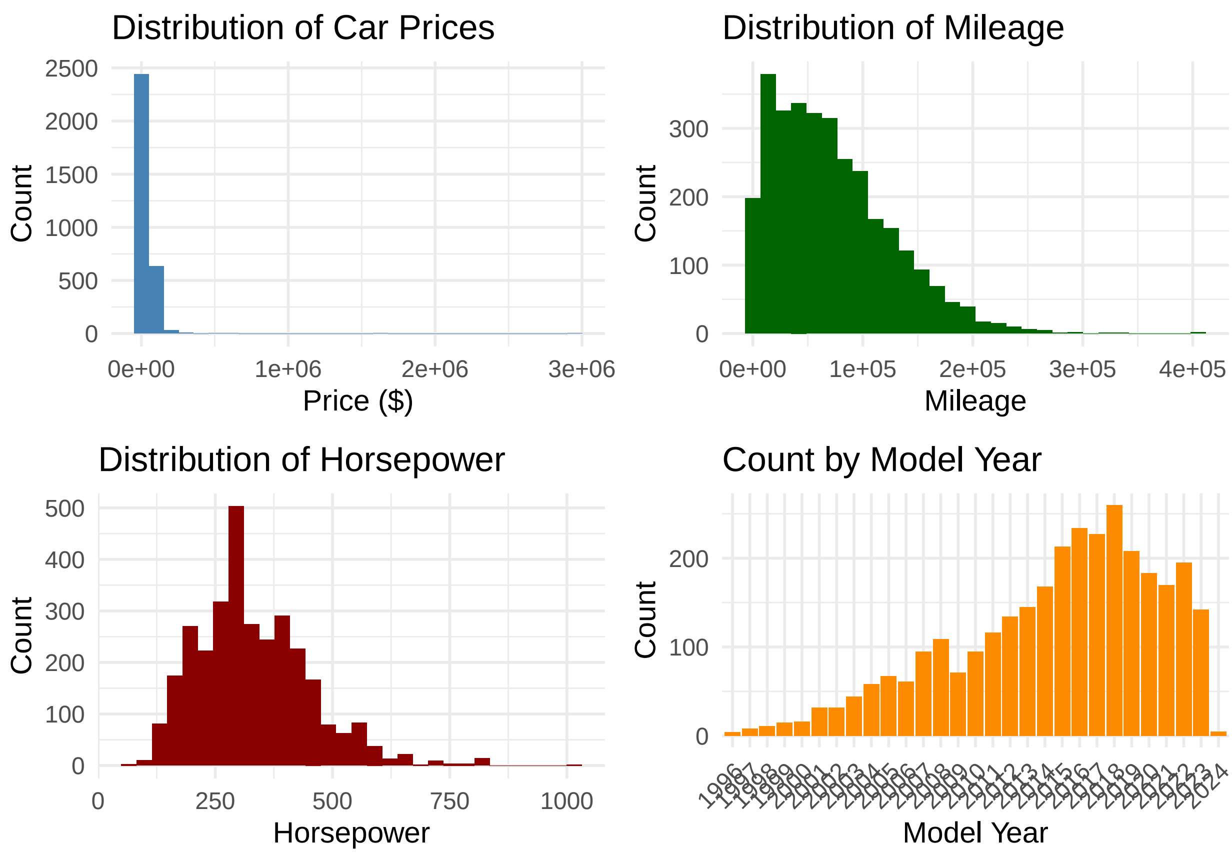
```r
car_data <- read.csv("cleaned_cars_no_null.csv")
attach(car_data)

# 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
```

Data Visualization

```r
# Create histograms for key variables
p1 <- ggplot(car_data, aes(x=Price)) +
  geom_histogram(bins=30, fill="steelblue") +
  theme_minimal() +
  labs(title="Distribution of Car Prices", x="Price ($)", y="Count")

p2 <- ggplot(car_data, aes(x=Mileage)) +
  geom_histogram(bins=30, fill="darkgreen") +
  theme_minimal() +
  labs(title="Distribution of Mileage", x="Mileage", y="Count")

p3 <- ggplot(car_data, aes(x=Horsepower)) +
  geom_histogram(bins=30, fill="darkred") +
  theme_minimal() +
```

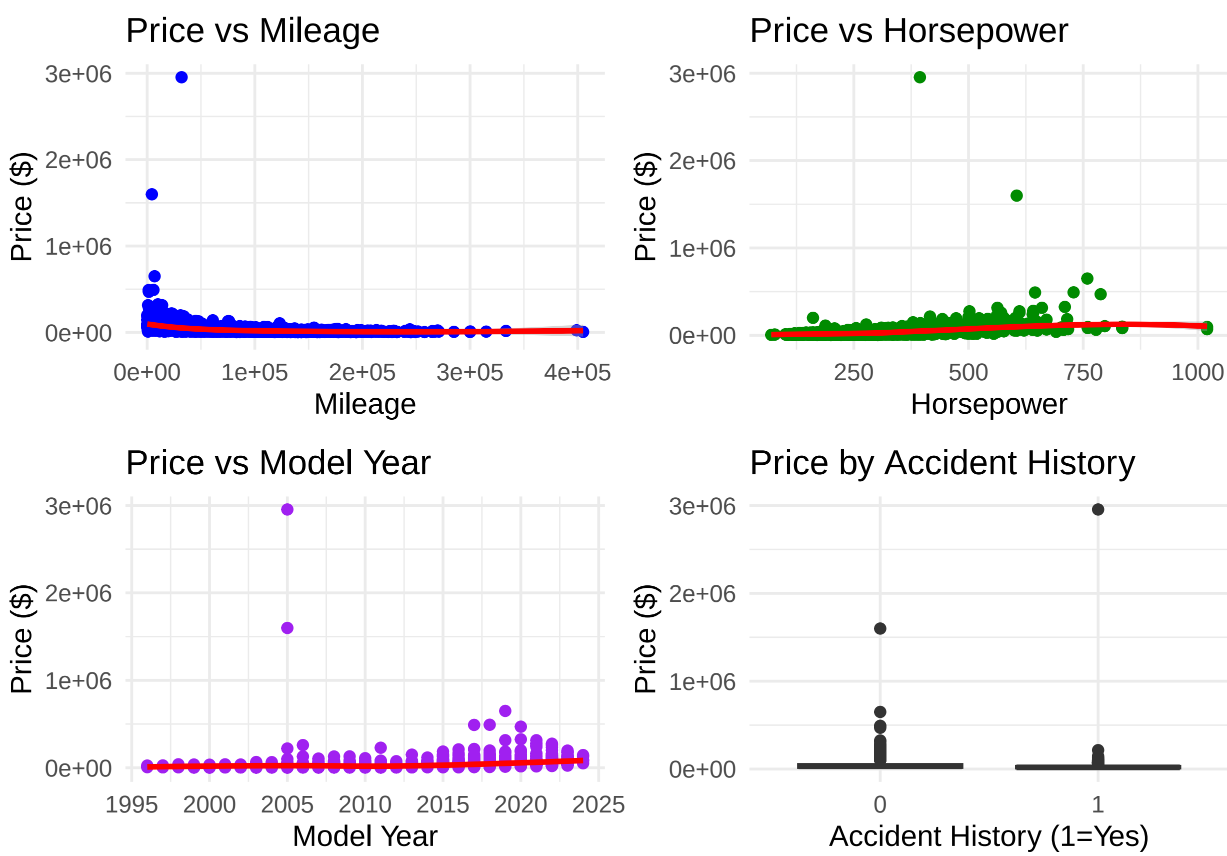
```r
  labs(title="Distribution of Horsepower", x="Horsepower", y="Count")

p4 <- ggplot(car_data, aes(x=factor(Model_year))) +
  geom_bar(fill="darkorange") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title="Count by Model Year", x="Model Year", y="Count")

grid.arrange(p1, p2, p3, p4, ncol=2)
```




```r
# Create scatter plots to explore relationships
s1 <- ggplot(car_data, aes(x=Mileage, y=Price)) +
  geom_point(color="blue") +
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Mileage", x="Mileage", y="Price ($)")

s2 <- ggplot(car_data, aes(x=Horsepower, y=Price)) +
  geom_point(color="green4") +
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Horsepower", x="Horsepower", y="Price ($)")

s3 <- ggplot(car_data, aes(x=Model_year, y=Price)) +
  geom_point(color="purple") +
```

```
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Model Year", x="Model Year", y="Price ($)")

s4 <- ggplot(car_data, aes(x=Accident, y=Price)) +
  geom_boxplot(fill="orange") +
  theme_minimal() +
  labs(title="Price by Accident History", x="Accident History (1=Yes)", y="Price ($)")

grid.arrange(s1, s2, s3, s4, ncol=2)
```

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



Outlier Detection and Removal

Based on our exploratory analysis, we have a couple of outliers in the data that could skew our results. When trying to vsualize this data, the outlier's scale also makes it difficult to see the rest of the data. We're going to do this by removing values that are outside a zscore of our 3 in the data.

```
# Calculate z-scores for numerical variables
car_data <- car_data %>%
  mutate(
```

```r
    price_zscore = (Price - mean(Price)) / sd(Price),
    mileage_zscore = (Mileage - mean(Mileage)) / sd(Mileage),
    hp_zscore = (Horsepower - mean(Horsepower)) / sd(Horsepower)
  )

# Identify outliers (z-score > 3 or < -3)
outliers <- car_data %>%
  filter(abs(price_zscore) > 3 | abs(mileage_zscore) > 3 | abs(hp_zscore) > 3)

# Print number of outliers
cat("Number of outliers detected:", nrow(outliers), "\n")

# Show some examples of outliers
head(outliers[, c("Brand", "Model", "Price", "Mileage", "Horsepower", "Model_year")])

# Remove outliers
car_data_clean <- car_data %>%
  filter(abs(price_zscore) <= 3 & abs(mileage_zscore) <= 3 & abs(hp_zscore) <= 3) %>%
  select(-price_zscore, -mileage_zscore, -hp_zscore)  # Remove the z-score columns

# Print how many rows remain
cat("Rows after removing outliers:", nrow(car_data_clean), "\n")
cat("Number of rows removed:", nrow(car_data) - nrow(car_data_clean), "\n")

# Updated summary statistics
summary(car_data_clean[,c("Price","Mileage","Horsepower","Model_year")])
```

```
## Number of outliers detected: 71
##         Brand                    Model  Price Mileage Horsepower Model_year
## 1         BMW                   740 iL   7300  242000        282       2001
## 2       Aston   Martin DBS Superleggera 184606   22770        715       2019
## 3       Dodge Ram 1500 Laramie Mega Cab  10900  300183        345       2006
## 4      Rivian     R1S Adventure Package  92000    2800        835       2023
## 5      Rivian     R1S Adventure Package  94000    2500        835       2023
## 6 Lamborghini                Urus Base 257000   13692        641       2021
## Rows after removing outliers: 3047
## Number of rows removed: 71
##      Price           Mileage         Horsepower      Model_year
##  Min.   :  2000   Min.   :   100   Min.   : 70.0   Min.   :1996
##  1st Qu.: 15472   1st Qu.: 31000   1st Qu.:247.0   1st Qu.:2011
##  Median : 27950   Median : 63160   Median :310.0   Median :2016
##  Mean   : 35501   Mean   : 71073   Mean   :325.8   Mean   :2015
##  3rd Qu.: 46000   3rd Qu.:102000   3rd Qu.:400.0   3rd Qu.:2019
##  Max.   :240000   Max.   :232000   Max.   :691.0   Max.   :2024
```

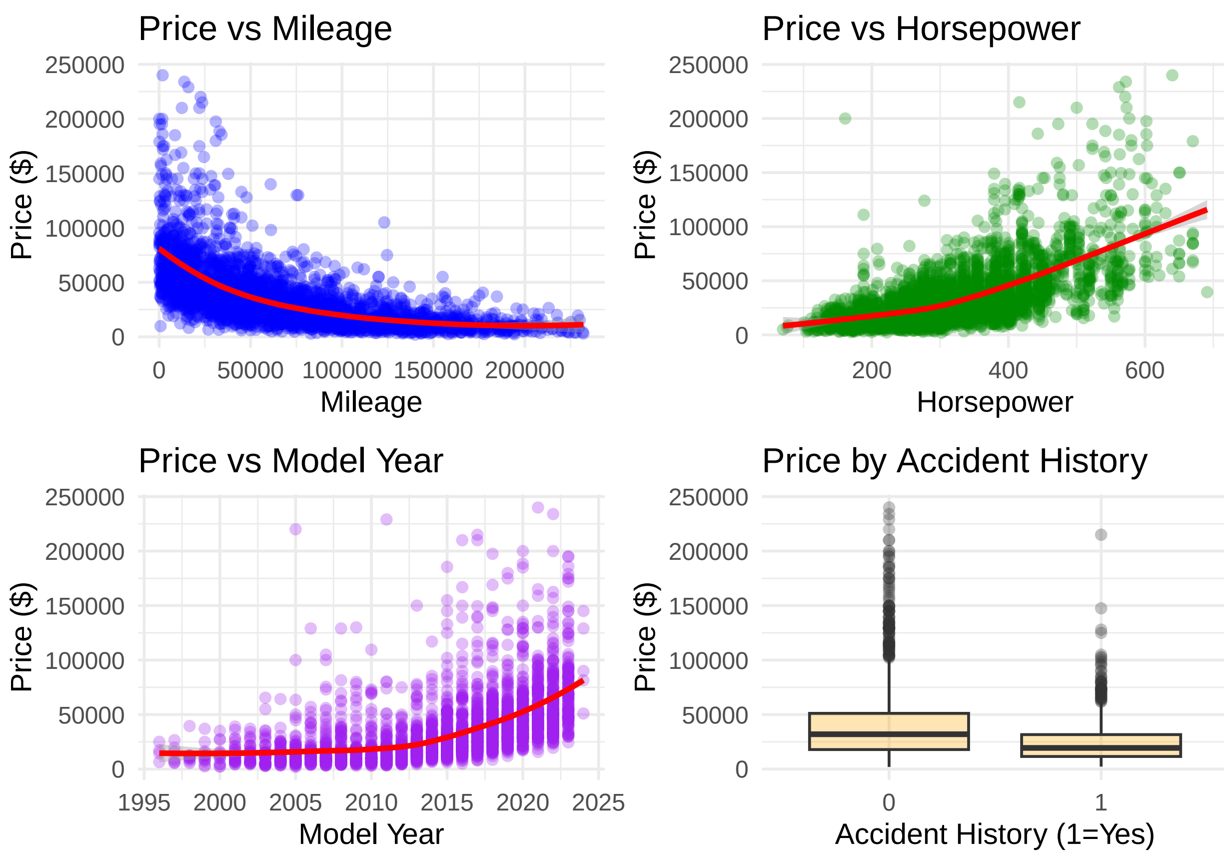
Outlier analysis

Looking at the outliers show us that some of these are some rare cars, we see supercars, old classics, and a lot of electric cars as well. It's interesting to note that electric cars may be an outlier because of their ability to have such high horsepower in regular production cars when compared to gasoline engines. Car's like the Rivian R1S, a regular family SUV, have 835 horsepower that rival and surpass many supercars in this specific metric.

Lets look at our exploratory data again without these outliers

```
# Create scatter plots to explore relationships
s1 <- ggplot(car_data_clean, aes(x=Mileage, y=Price)) +
  geom_point(color="blue", alpha=0.3) +
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Mileage", x="Mileage", y="Price ($)")

s2 <- ggplot(car_data_clean, aes(x=Horsepower, y=Price)) +
  geom_point(color="green4", alpha=0.3) +
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Horsepower", x="Horsepower", y="Price ($)")

s3 <- ggplot(car_data_clean, aes(x=Model_year, y=Price)) +
  geom_point(color="purple", alpha=0.3) +
  geom_smooth(method="loess", color="red") +
  theme_minimal() +
  labs(title="Price vs Model Year", x="Model Year", y="Price ($)")

s4 <- ggplot(car_data_clean, aes(x=Accident, y=Price)) +
  geom_boxplot(fill="orange", alpha=0.3) +
  theme_minimal() +
  labs(title="Price by Accident History", x="Accident History (1=Yes)", y="Price ($)")

grid.arrange(s1, s2, s3, s4, ncol=2)
```

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



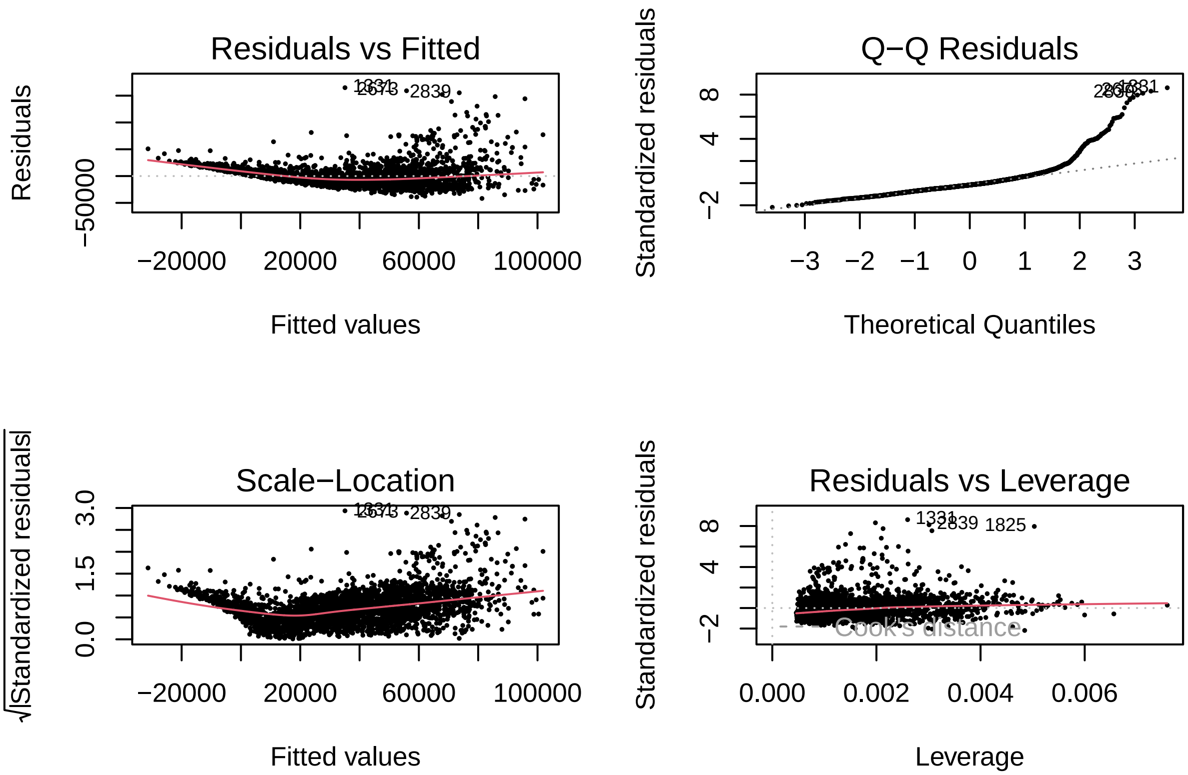
This makes the data much easier to see and analyze.

Model Building and Evaluation

Now we'll build different models to predict car prices and compare their performance.

Standard Linear Regression Model

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

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



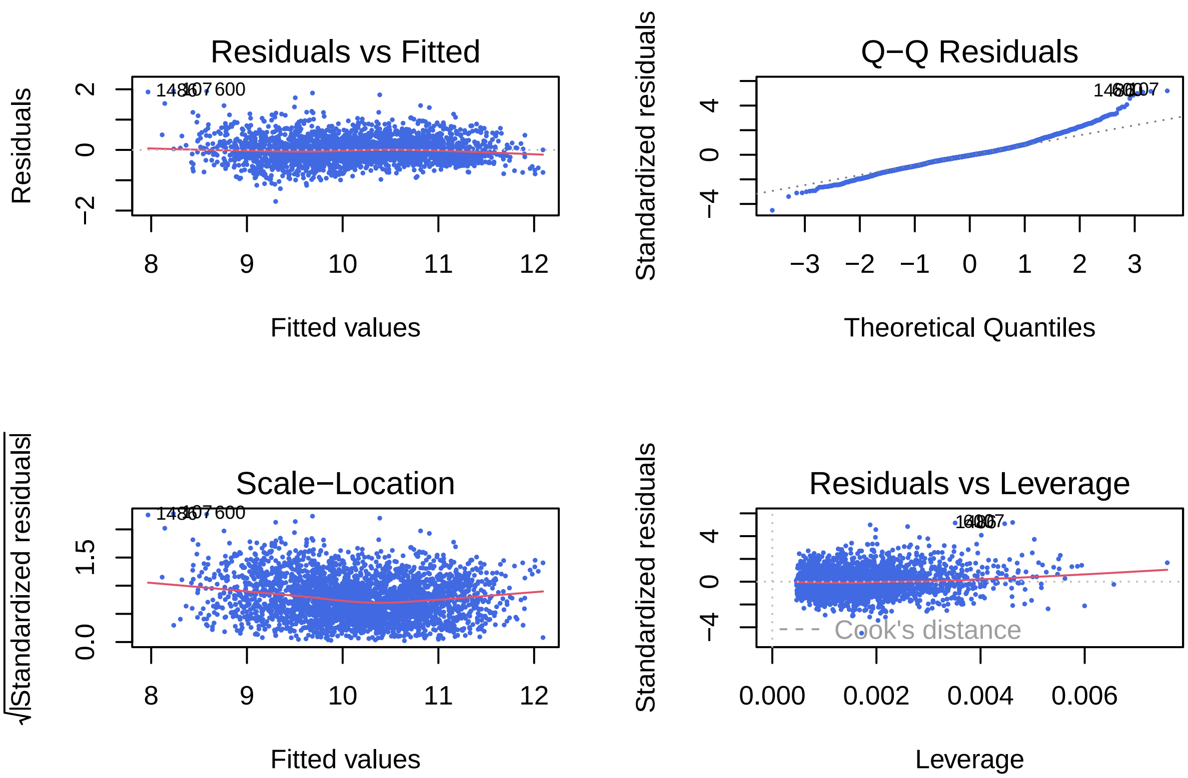

```r
par(mfrow=c(1,1))
```

```
##
## Call:
## lm(formula = Price ~ Mileage + Horsepower + Accident + Model_year,
##     data = car_data_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41779 -10136  -3276   6041 164918
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.900e+06  1.514e+05 -12.549  < 2e-16 ***
## Mileage     -1.836e-01  9.039e-03 -20.309  < 2e-16 ***
## Horsepower   1.294e+02  3.337e+00  38.761  < 2e-16 ***
## Accident1   -3.186e+03  8.015e+02  -3.975 7.19e-05 ***
## Model_year   9.465e+02  7.502e+01  12.617  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19160 on 3042 degrees of freedom
## Multiple R-squared:  0.589,  Adjusted R-squared:  0.5885
## F-statistic:  1090 on 4 and 3042 DF,  p-value: < 2.2e-16
```

The standard linear model shows that all predictors are statistically significant, but the R-squared value is relatively low at around 0.46, indicating moderate explanatory power. Residual plots suggest non-linear patterns, and our QQ residuals are very off at the right end.

Log-Transformed Model

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

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




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

# Back-transform predictions for comparison
log_pred <- exp(predict(mod_log, car_data_clean))
```

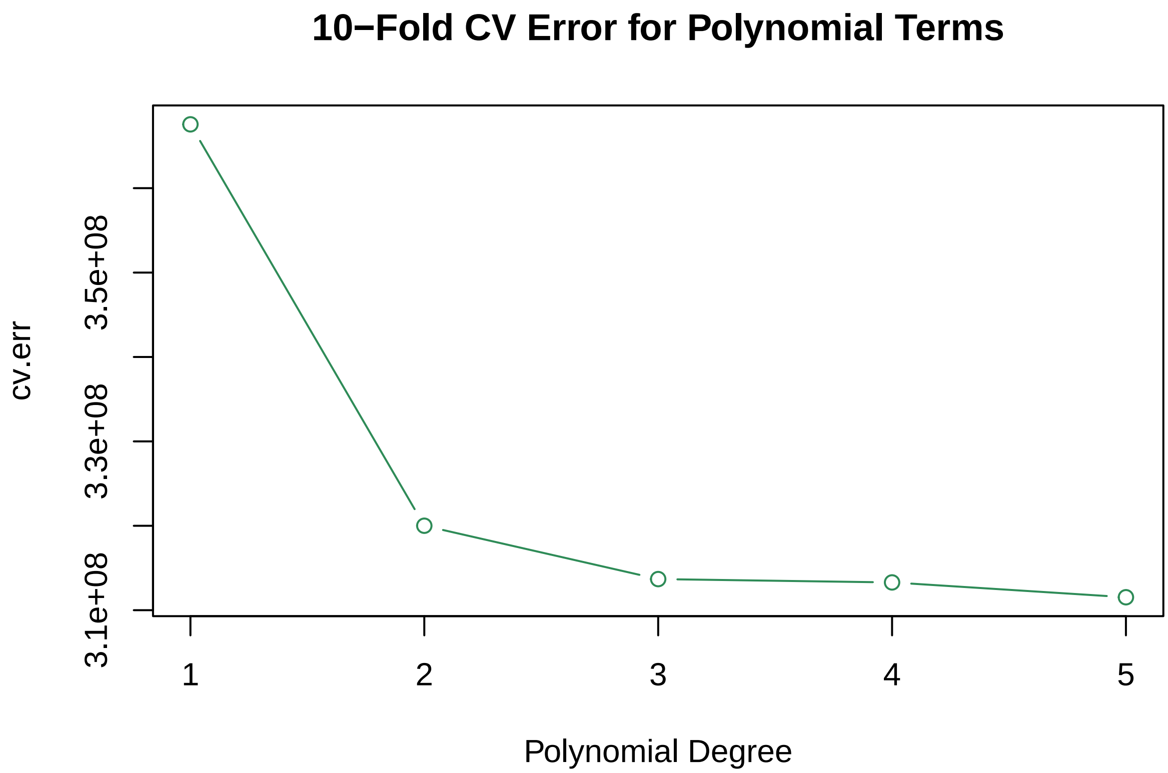
```
##
## Call:
## lm(formula = log(Price) ~ Mileage + Horsepower + Accident + Model_year,
##     data = car_data_clean)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.69920 -0.21705 -0.02003  0.19396  1.95249
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.149e+01  2.973e+00 -24.045  < 2e-16 ***
## Mileage     -6.182e-06  1.775e-07 -34.821  < 2e-16 ***
## Horsepower   3.272e-03  6.555e-05  49.916  < 2e-16 ***
## Accident1   -6.185e-02  1.574e-02  -3.928 8.75e-05 ***
## Model_year   4.023e-02  1.474e-03  27.304  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3764 on 3042 degrees of freedom
## Multiple R-squared:  0.7786, Adjusted R-squared:  0.7783
## F-statistic:  2674 on 4 and 3042 DF,  p-value: < 2.2e-16
```

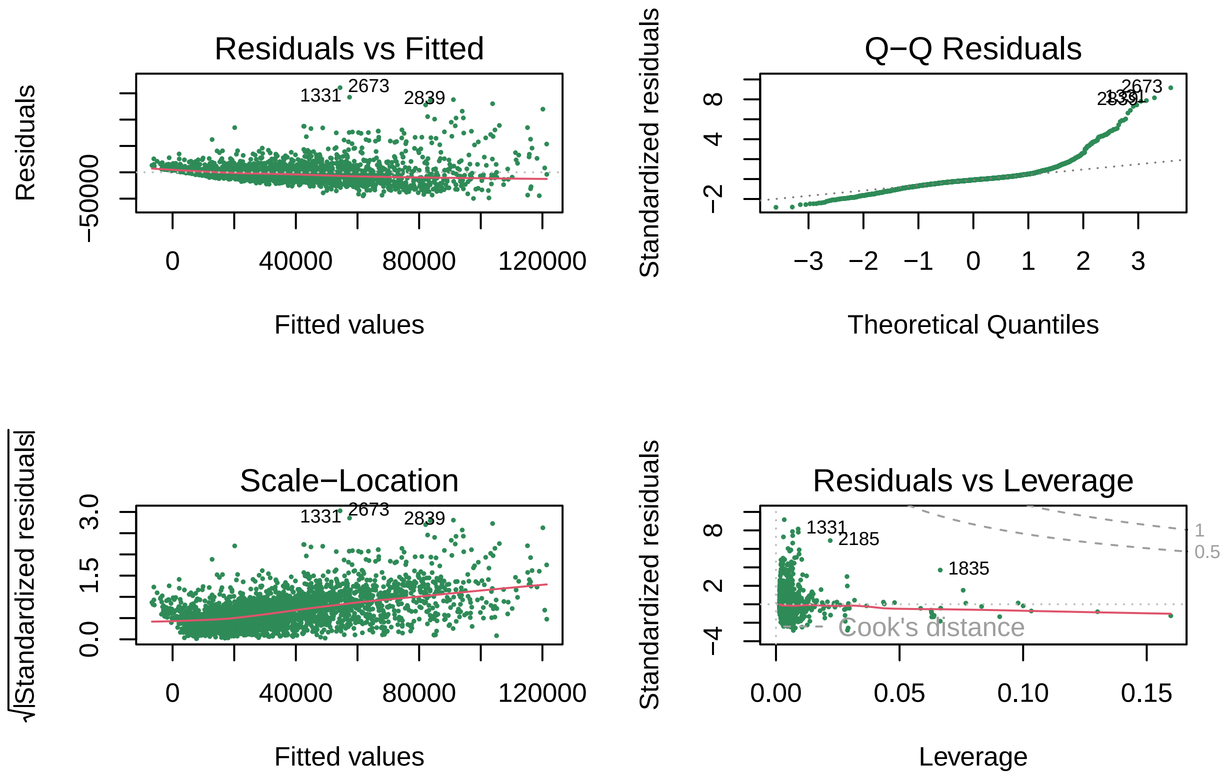
The log-transformed model shows a much better fit with an R-squared of around 0.78. All variables are highly significant, and residual plots show improved patterns.

Polynomial Model

```r
# Cross-validation to find optimal polynomial degree
library(boot)
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_clean)
  cv.err[i] <- cv.glm(car_data_clean, glm.fit, K=10)$delta[1]
}

# Plot CV error by polynomial degree
plot(1:5, cv.err, type="b", col="seagreen", xlab="Polynomial Degree",
     main="10-Fold CV Error for Polynomial Terms")
```




```
# Determine best degree (find minimum CV error)
best_degree <- which.min(cv.err)
cat("Best polynomial degree based on CV:", best_degree, "\n")

# Fit optimal degree polynomial model
mod_poly <- lm(Price ~ poly(Mileage, best_degree) + poly(Horsepower, best_degree) +
                 Accident + Model_year, data=car_data_clean)
summary(mod_poly)

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




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

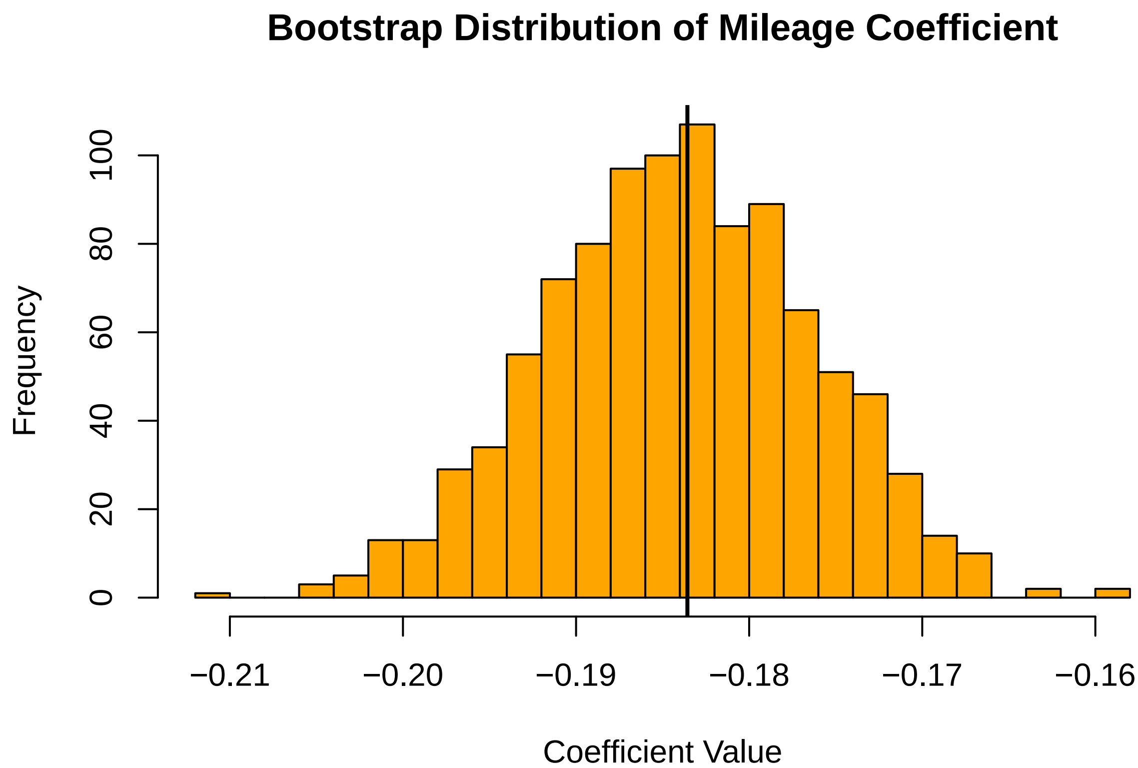
```
## Best polynomial degree based on CV: 5
##
## Call:
## lm(formula = Price ~ poly(Mileage, best_degree) + poly(Horsepower,
##     best_degree) + Accident + Model_year, data = car_data_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -49628  -7951  -1501   4551 160645
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -1.561e+06  1.417e+05 -11.011  < 2e-16 ***
## poly(Mileage, best_degree)1    -5.402e+05  2.326e+04 -23.229  < 2e-16 ***
## poly(Mileage, best_degree)2     3.177e+05  1.799e+04  17.661  < 2e-16 ***
## poly(Mileage, best_degree)3    -1.339e+05  1.763e+04  -7.593 4.14e-14 ***
## poly(Mileage, best_degree)4     5.550e+04  1.761e+04   3.151  0.00164 **
## poly(Mileage, best_degree)5    -2.076e+04  1.763e+04  -1.178  0.23899
## poly(Horsepower, best_degree)1  7.681e+05  1.875e+04  40.971  < 2e-16 ***
## poly(Horsepower, best_degree)2  2.061e+05  1.769e+04  11.649  < 2e-16 ***
## poly(Horsepower, best_degree)3 -2.449e+04  1.761e+04  -1.391  0.16444
## poly(Horsepower, best_degree)4 -7.483e+04  1.761e+04  -4.250 2.20e-05 ***
## poly(Horsepower, best_degree)5 -7.397e+04  1.762e+04  -4.198 2.78e-05 ***
```

```
## Accident1                      -2.105e+03  7.399e+02  -2.845  0.00447 **
## Model_year                      7.926e+02  7.036e+01  11.266  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17560 on 3034 degrees of freedom
## Multiple R-squared:  0.6556, Adjusted R-squared:  0.6543
## F-statistic: 481.3 on 12 and 3034 DF,  p-value: < 2.2e-16
```

The polynomial model with degree 5 improves on the standard linear model but still shows some residual issues.

Bootstrap Analysis

```r
# Bootstrap analysis for model stability
alpha.fn <- function(data, index) {
  coef(lm(Price ~ Mileage + Horsepower + Accident + Model_year,
          data=data, subset=index))
}

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

# Get confidence intervals for all coefficients, we probably won't need to put this in the actualy pape
for (i in 1:5) {
  cat("Bootstrap CI for", c("Intercept", "Mileage", "Horsepower", "Accident1", "Model_year")[i], ":\n")
  print(boot.ci(boot.res, type="perc", index=i))
}

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



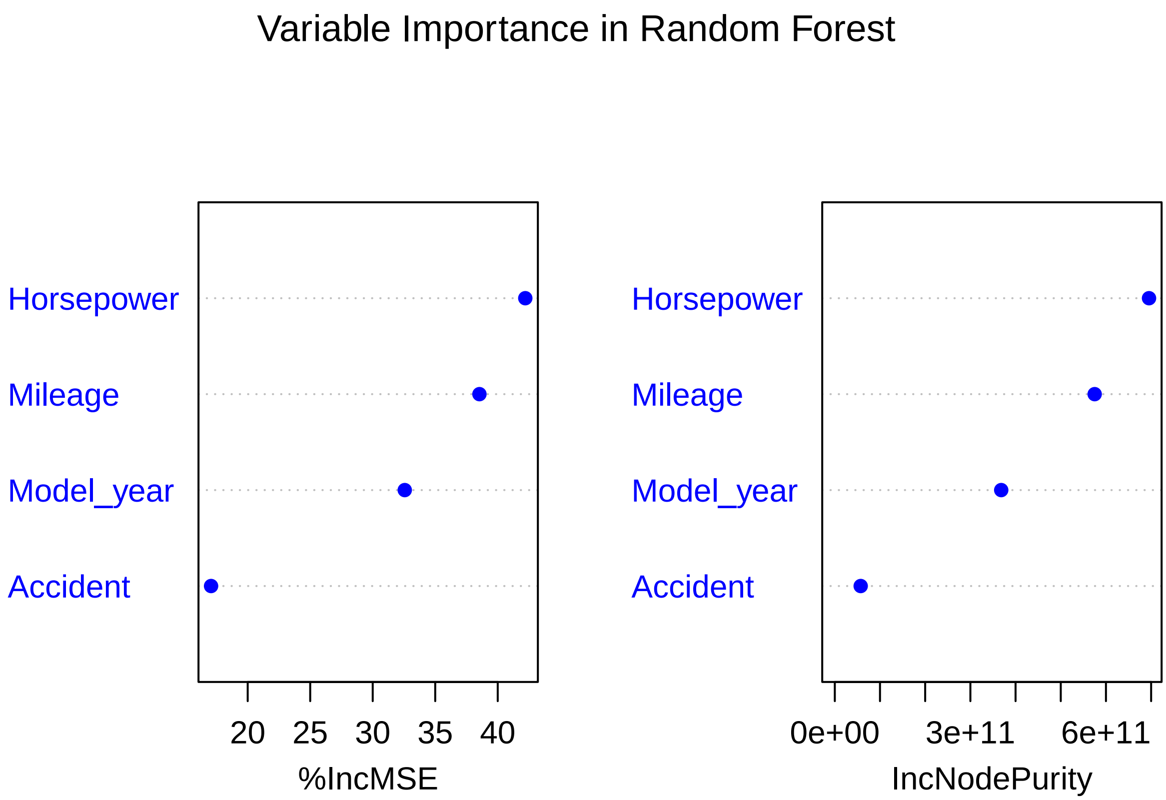

```
## Bootstrap CI for Intercept :
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-2181097, -1610499 )
## Calculations and Intervals on Original Scale
## Bootstrap CI for Mileage :
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-0.1993, -0.1697 )
## Calculations and Intervals on Original Scale
## Bootstrap CI for Horsepower :
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (120.1, 138.6 )
## Calculations and Intervals on Original Scale
## Bootstrap CI for Accident1 :
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   (-4398, -1848 )
## Calculations and Intervals on Original Scale
## Bootstrap CI for Model_year :
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.res, type = "perc", index = i)
##
## Intervals :
## Level     Percentile
## 95%   ( 802.7, 1087.1 )
## Calculations and Intervals on Original Scale
```

The bootstrap analysis confirms the stability of our coefficient estimates, particularly for Mileage and Horsepower.

Random Forest Model

```r
# Random forest model
set.seed(3180)
rf.fit <- randomForest(Price ~ Mileage + Horsepower + Accident + Model_year,
                       data=car_data_clean,
                       importance=TRUE,
                       ntree=500)

# Display model performance
print(rf.fit)

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

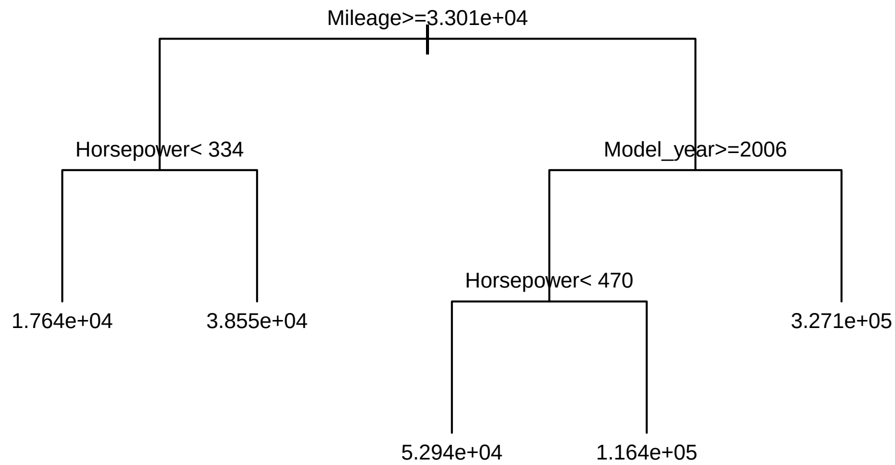



```r
# Get predictions
rf.pred <- predict(rf.fit, car_data_clean)
```

```
##
## Call:
##  randomForest(formula = Price ~ Mileage + Horsepower + Accident +      Model_year, data = car_data_cl
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 306853923
##                     % Var explained: 65.6
```

Decision Tree Model

```r
# Decision tree model. May or may not be useful to put in the paper. If needed text me I can find a way
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)
```



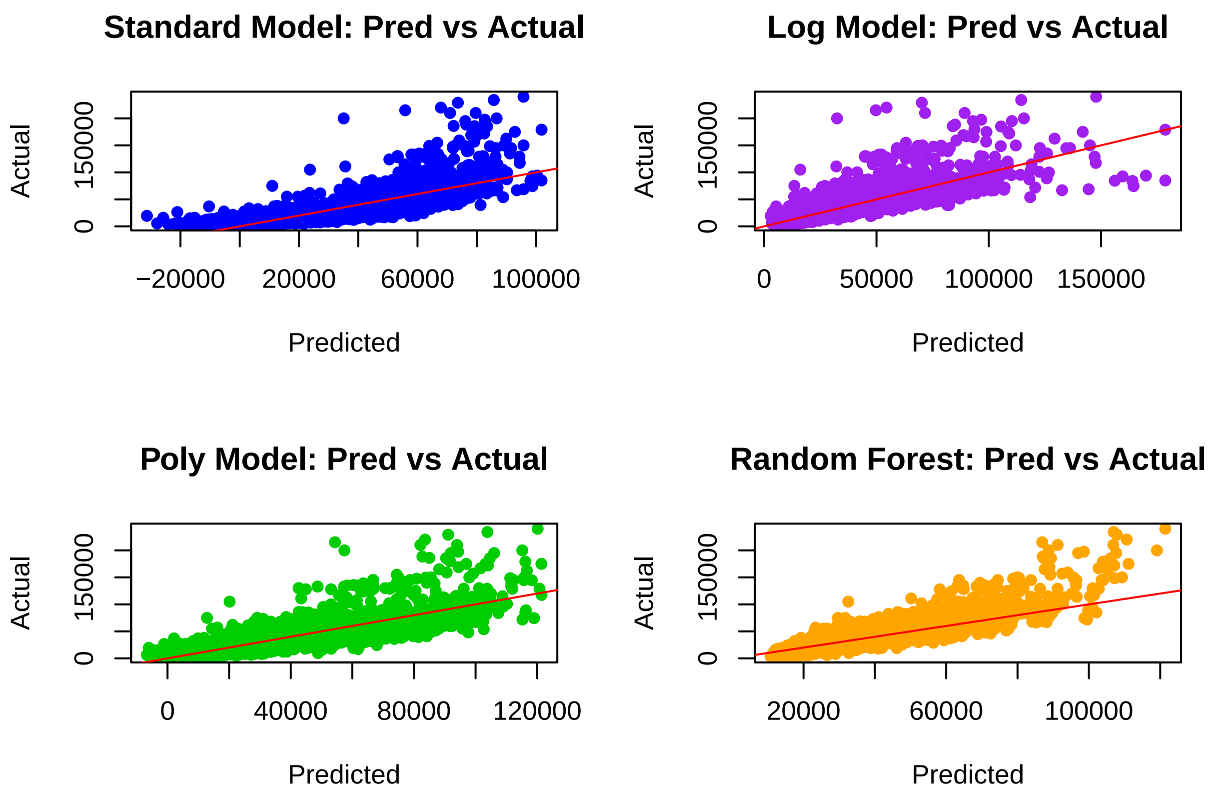
The decision tree provides an model showing the key factors in price determination. Each node shows decision rules and average prices.

Model Comparison

Now we'll compare all models using appropriate metrics.

```r
# Function to calculate metrics
calc_metrics <- function(actual, predicted) {
  rmse <- sqrt(mean((actual - predicted)^2))
  mae <- mean(abs(actual - predicted))
  r2 <- 1 - sum((actual - predicted)^2) / sum((actual - mean(actual))^2)
  return(c(RMSE=rmse, MAE=mae, R2=r2))
}

# Calculate metrics for each model
metrics <- rbind(
  Standard = calc_metrics(car_data_clean$Price, predict(mod_std, car_data_clean)),
  Log_Transformed = calc_metrics(car_data_clean$Price, log_pred),
  Polynomial = calc_metrics(car_data_clean$Price, predict(mod_poly, car_data_clean)),
  Random_Forest = calc_metrics(car_data_clean$Price, rf.pred)
)

# Print comparison table
print(metrics)
```

```r
# Plot comparison with actual values
par(mfrow=c(2,2))
plot(predict(mod_std, car_data_clean), car_data_clean$Price, pch=16, col="blue",
     main="Standard Model: Pred vs Actual", xlab="Predicted", ylab="Actual")
abline(0,1, col="red")

plot(log_pred, car_data_clean$Price, pch=16, col="purple",
     main="Log Model: Pred vs Actual", xlab="Predicted", ylab="Actual")
abline(0,1, col="red")

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

plot(rf.pred, car_data_clean$Price, pch=16, col="orange",
     main="Random Forest: Pred vs Actual", xlab="Predicted", ylab="Actual")
abline(0,1, col="red")
```



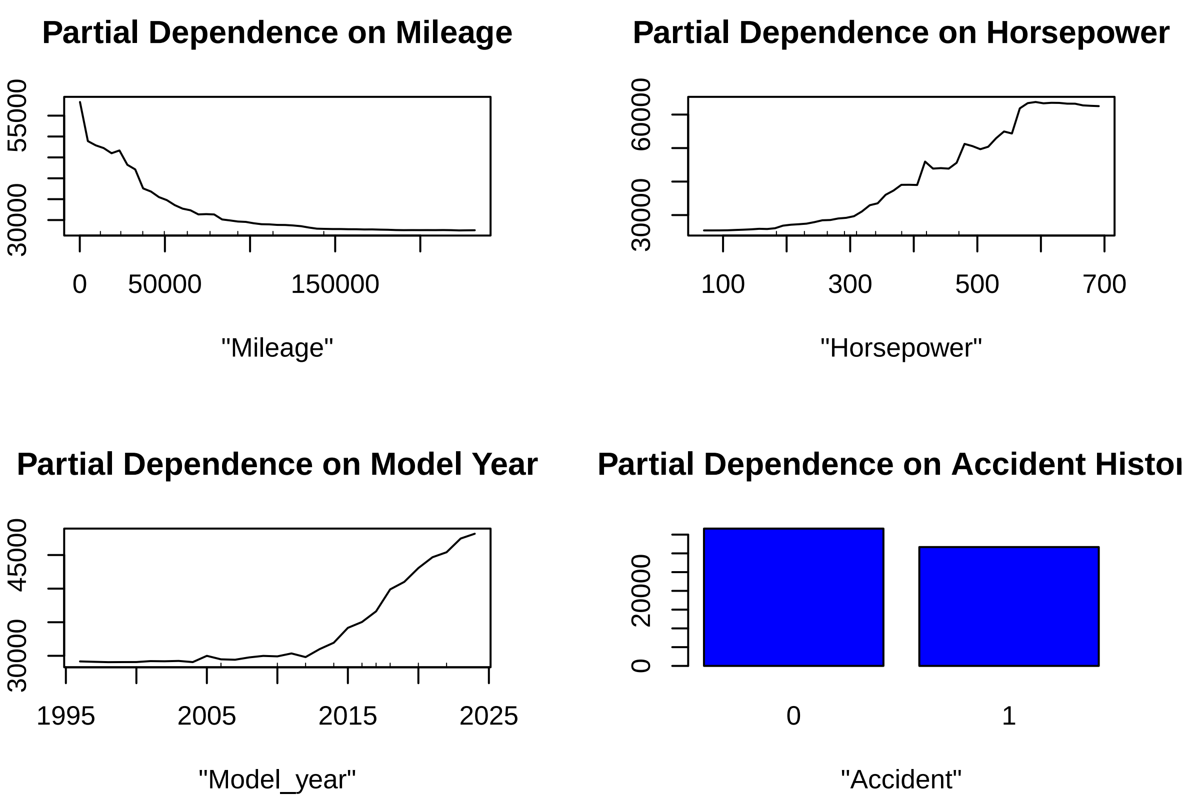

```r
par(mfrow=c(1,1))
```

```
##                     RMSE       MAE        R2
## Standard        19146.58 12227.833 0.5890499
## Log_Transformed 17688.82  9599.890 0.6492444
## Polynomial      17527.20 10611.591 0.6556248
## Random_Forest   15279.53  9251.913 0.7382862
```

Partial Dependence Plots

Let's examine how price depends on each predictor in the random forest model.

```r
# Generate partial dependence plots
par(mfrow=c(2,2))
partialPlot(rf.fit, car_data_clean, x.var="Mileage",
            main="Partial Dependence on Mileage")
partialPlot(rf.fit, car_data_clean, x.var="Horsepower",
            main="Partial Dependence on Horsepower")
partialPlot(rf.fit, car_data_clean, x.var="Model_year",
            main="Partial Dependence on Model Year")
partialPlot(rf.fit, car_data_clean, x.var="Accident",
            main="Partial Dependence on Accident History")
```




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

Conclusions

Based on our analysis, we can draw several important conclusions:

1. **Model Performance**: The Random Forest model outperforms all other models with the highest R-squared and lowest error metrics. The log-transformed linear model also performs well, indicating non-linear relationships between predictors and car prices. Even though our polynomial model has a

higher Rˆ2 than than our log transformed model, looking at the residual plots we have much a much better fit with log transformed so we ruled polynomial out.

2. **Key Price Determinants**: Model year and horsepower have the strongest positive associations with price, while mileage has a negative relationship. Accident history negatively impacts price but with smaller effect size compared to other variables.
3. **Price-Mileage Relationship**: The relationship between mileage and price is non-linear, with a steeper decline in value at lower mileages that gradually levels off.
4. **Horsepower Effect**: Higher horsepower consistently correlates with higher prices, showing an almost linear relationship up to around 400 HP, after which the price increase is more dramatic.
5. **Outlier Impact**: After removing outliers, our models showed substantial improvement in fit and prediction accuracy, indicating that extreme values were influencing our earlier analyses.
6. **Prediction Accuracy**: Our best model (Random Forest) can explain approximately 73.83% of the variation in used car prices using just four variables.