Predicting MSRP based on Car Features

Leslie Cervantes Rivera

2024-10-26

Figure 1: Car Price Tag

Introduction

The purpose of this project is to build a machine learning model to predict the Manufacturer's Suggested Retail Price (MSRP) of cars based on various features, such as make, year, and engine fuel type. Beyond prediction, this project aims to address two questions:

- What are the most important car features that influence the price of a car?
- Is there a significant price difference between different types of transmissions?

By examining the relationships between the predictors and MSRP, I aim to identify the factors that contribute to the price variations across models and quantify the impact of transmission types on MSRP.

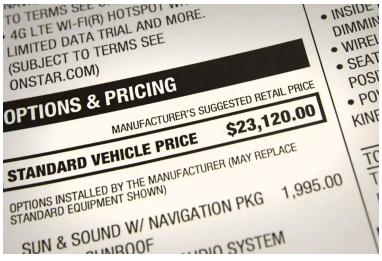


Figure 2: MSRP

What is MSRP?

As an adult, I believe it is important to understand how a car's features affect its MSRP. MSRP serves as the price the original producer of the car suggests the cost to be (Indeed Editorial Team).The main purpose

of MSRP is to give distributors and retailers guidelines to maintain prices consistent and affordable for the majority of customers across locations. This also serves as the starting point for negotiations at the dealership.

MSRP not only sets expectations around affordability, financing, and loan terms, but it also affects recurring expenses like insurance. In today's economy, MSRP has been affected by inflation, supply chain disruptions, and technological advancements. Understanding these factors is crucial for making wise financial decisions, especially for new buyers who want to get the best value for their investment.

Car Features and Pricing

What is something you look for when buying a car?

The car's features! Car features are important factors that influence the price of the car, as they offer functionality, performance, and appeal. Features like engine horsepower and transmission type affect how the car performs and driving experience, while other features such as vehicle style and market category offer aesthetic appeal. Luxury cars tend to have advanced features like engine horsepower, transmission type, and more, which significantly raise the MSRP. On the other hand, non-luxury cars focus more on affordability. This project will explore the relationship between the features and the MSRP, seeking to determine which features are most significant.

Transmission Types and Pricing

An important feature to consider when buying a new car is the transmission type, as it impacts your driving experience, affordability, and fuel efficiency. Direct drive transmissions are fuel efficient when driving on a flat ground highway and are commonly found in electric vehicles (Eaton). Manual transmissions provide a more engaging driving experience and can save you money on fuel and maintenance. Automatic transmissions are more popular because they eliminate the need to change gears manually, though they are typically less fuel efficient ("Which Transmission Type Is Right for You?" Dennis Dillon Mazda). Automated manual transmissions combine the fuel efficiency of automatics and the cost-effectiveness of manuals ("Automated Manual Transmissions (AMT): Pros and Cons.") This project will explore if there are significant price differences between transmissions.

Figure 3: Variety of Cars

Data

The data that will be used for this project contains detailed information about cars, including features, such as make, mode, year, engine specifications, MSRP, and more with approximately twelve thousand observations. This dataset comes from Kaggle, a platform for everyone to compete, collaborate, and learn, and was

provided by user Rupinder Singh Rana (Rana,https://www.kaggle.com/datasets/rupindersinghrana/car-features-and-prices-dataset/data). It includes different features of cars along with the price from 1990 to 2017, providing a historical overview on pricing trends as cars advanced in technology. The dataset consists of numerical values, such as engine hp, highway mpg, etc. and categorical values, such as engine fuel type, transmission type, etc.

Explanatory Data Analysis

In this section, I am conducting Explanatory Data Analysis (EDA) to better understand the data and the variables that may influence the MSRP of cars. This is the stage where we are given insights into the structure and characteristics of the data, for instance missing values and outliers.

I will provide visuals on missing values, correlation matrices, and boxplots to give us an idea of what the data is providing.

Loading Data

Firstly, I am going to load the dataset.

```
car_data <- read.csv("~/Desktop/data.csv")
car_data <- car_data %>%
  mutate(Category = ifelse(Make %in% c("BMW", "Audi", "Mercedes-Benz", "Volvo", "Ferrari",
          "Alfa Romero", "McLaren", "Maybach","Porsche", "Saab", "Cadillac", "Bentley", "Lamborghini",
          "Lincoln", "Rolls-Royce", "Buick", "Maserati","Lexus", "Aston Martin", "Land Rover",
          "Lotus", "Infiniti", "Genesis", "Bugatti"),
                         "Luxury",
                         "Non-Luxury"))
```

Now, I am going to get the dataset's dimensions. This tells us the number of rows and columns.

```
dim(car_data)
```

```
## [1] 11914    17
```

This dataset includes 11,914 rows and 17 columns. The 11,914 rows are different cars and 16 columns are the car's features and the remaining one is the `MSRP`. Here are the car's features and price:

- `Make`: The company that produces the vehicle
- `Model`: The product name of the vehicle used by the manufacturer
- `Year`: The year the vehicle was made
- `Engine.Fuel.Type`: The type of fuel the vehicle uses
- `Engine.HP`: The engine's horse power
- `Engine.Cylinders`: Number of cylinders
- `Transmission.Type`: The type of transmission the vehicle has
- `Driven_Wheels`: The type of wheels the vehicle has

- `Number.of.Doors`: The number of doors the vehicle has
- `Market.Category`: The classification of the vehicle
- `Vehicle.Size`: The size of the vehicle
- `Vehicle.Style`: Type of vehicle
- `highway.MPG`: How many miles a vehicle can travel on a gallon of gas while driving on the freeway
- `city.MPG`: How many miles a vehicle can travel on a gallon of gas while driving in the city
- `Popularity`: Popularity of the vehicle
- `MSRP`: Price of the vehicle in ($)
- `Category`: Whether a a car is luxury or non-luxury

After finding out how many observations we have, we want to know how much missing data there is. It is important to know how much data we are missing as it can lead to a reduced sample size, biased results, or misleading results.

Tidying Data

Missing Data

```
missing <- is.na(car_data)

total_missing <- sum(missing)
print(total_missing)
```

```
## [1] 105
```

The dataset is missing 105 observations. Let us see which specific variables are missing data.

```
colSums(is.na(car_data))
```

```
##              Make             Model              Year  Engine.Fuel.Type
##                 0                 0                 0                 0
##         Engine.HP  Engine.Cylinders Transmission.Type     Driven_Wheels
##                69                30                 0                 0
##   Number.of.Doors   Market.Category      Vehicle.Size     Vehicle.Style
##                 6                 0                 0                 0
##       highway.MPG          city.mpg        Popularity              MSRP
##                 0                 0                 0                 0
##          Category
##                 0
```

Looking at the table, we can see there are a few missing observations for `Engine.HP`, `Engine.Cylinders`, `Number.of.Doors`. Specifically, there are 69 missing observations for `Engine.HP`, 30 for `Engine.Cylinders`, and 6 for `Number.of.Doors`. While skimming through the dataset, I noticed the function `total_missing` did not catch other missing values due to how it was represented.

```
updated_car_data <- car_data %>%
  drop_na(Engine.HP, Engine.Cylinders, Number.of.Doors)

NA_car_data <- subset(updated_car_data, !( Engine.Fuel.Type == "" | Engine.Cylinders == 0
                                          | Transmission.Type == "UNKNOWN"))
```

Upon reviewing the dataset, I decided on removing the observations with missing data in `Engine.HP`, `Engine.Cylinders`, and `Number.of.Doors` since we could afford to remove the missing observations. After reviewing the updated set with the missing values removed, I noticed that some rows had a value of 0 in `Engine.Cylinders`, which seemed incorrect. Additionally, there were blank entries for `Engine.Fuel.Type` and "UNKNOWN" for `Transmission.Type`. As a result, I chose to remove the rows.

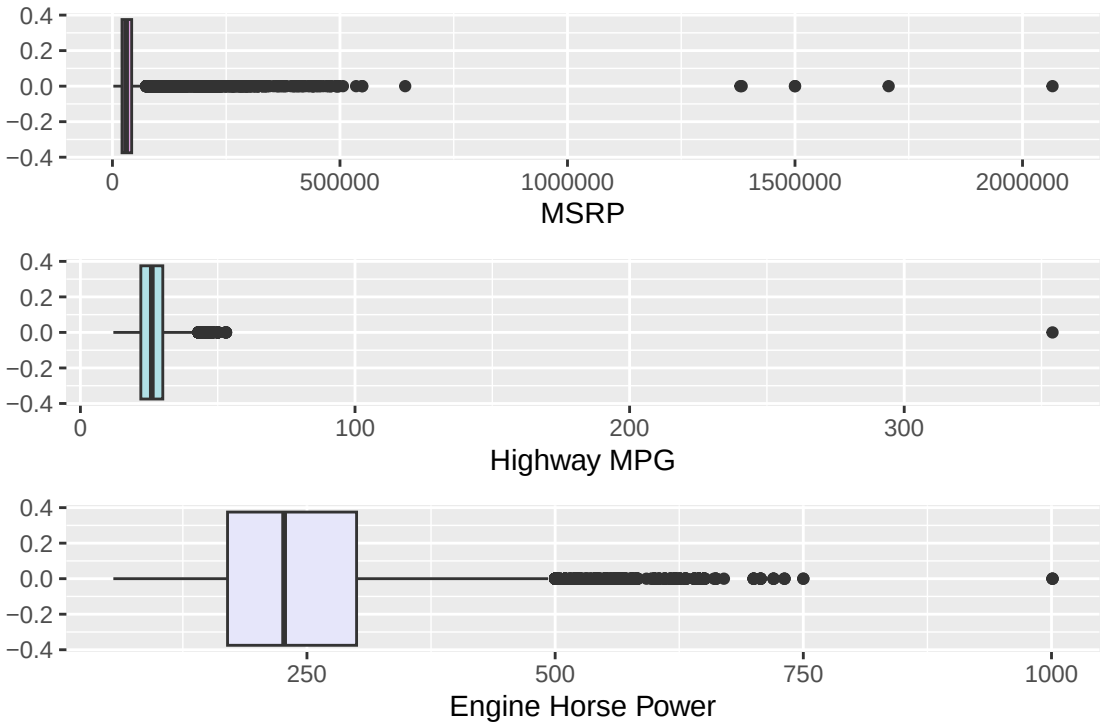
In the `Market.Category` variable there are approximately three thousand observations marked as "N/A" that the program did not detect. To evaluate the significance of the missing data points, I conducted two trials: one where missing observations were removed (No NA trial) and the other where missing observations were kept without imputation (NA trial).

After training and comparing the model results on both trials, the NA trial demonstrated better performance across RMSE. This tell us that keeping the missing values contributes to the predictive power of the models. I chose not to impute the missing data points because doing so will introduce bias which would lead to false conclusions and be misleading for our inference conclusions.

Outliers

To better understand the data, we're going to visualize predictors with a wide range of values. This is important as they create outliers in the data. Outliers are data values that significantly deviate from the majority of other values.

```
MSRP_outliers <- ggplot(NA_car_data, aes(MSRP)) +
  geom_boxplot(fill = "plum") + labs(x = "MSRP")
Highway_outliers <- ggplot(NA_car_data, aes(highway.MPG)) +
  geom_boxplot(fill = "powderblue") + labs(x = "Highway MPG")
EngineHP_outliers <- ggplot(NA_car_data, aes(Engine.HP)) +
  geom_boxplot(fill = "lavender")  + labs(x = "Engine Horse Power")
combined_plot <- MSRP_outliers + Highway_outliers + EngineHP_outliers + plot_layout(ncol = 1)
combined_plot
```



As you can see the `MSRP` boxplot reveals significant outliers, with values ranging from approximately $250,000 to $2,000,000. In the `highway.MPG` variable, there is a single extreme outlier approximately 350 miles per gallon (mpg), which is unrealistic. Both `MSRP` and `highway.MPG` have a small interquartile range (IQR), but their overall ranges cannot be described due to the many outliers. Lastly, `Engine.HP` has outliers ranging from 500 to 1,000 horse power. The IQR for `Engine.HP` is approximately 150 to 300 horsepower.

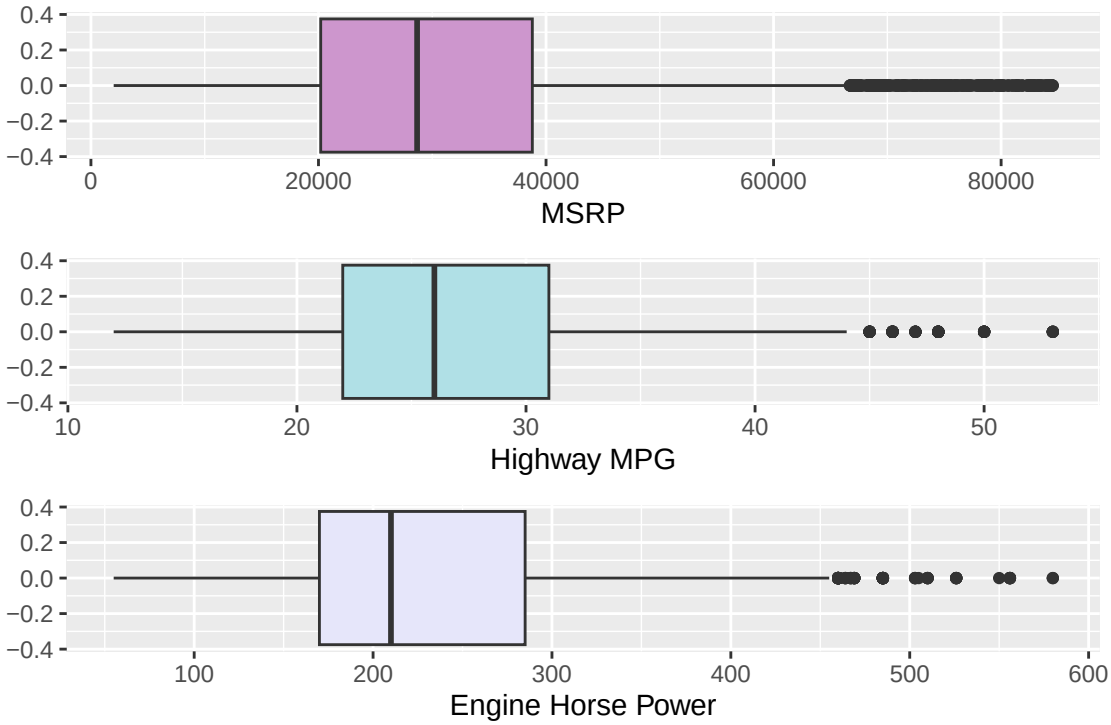
Let's transform the data to exclude the extreme outliers.

```
IQR_HP_NA <- IQR(updated_car_data$Engine.HP)
lower_HP_NA <- quantile(updated_car_data$Engine.HP, 0.25) - 2.0 * IQR_HP_NA
upper_HP_NA <- quantile(updated_car_data$Engine.HP, 0.75) + 2.5 * IQR_HP_NA

IQR_highway_NA <- IQR(updated_car_data$highway.MPG)
lower_highway_NA <- quantile(updated_car_data$highway.MPG, 0.25) - 3.0 * IQR_highway_NA
upper_highway_NA <- quantile(updated_car_data$highway.MPG, 0.75) + 4.0 * IQR_highway_NA

IQR_MSRP_NA <- IQR(updated_car_data$MSRP)
lower_MSRP_NA <- quantile(updated_car_data$MSRP, 0.25) - 3.0 * IQR_MSRP_NA
upper_MSRP_NA <- quantile(updated_car_data$MSRP, 0.75) + 2.0 * IQR_MSRP_NA

outliers_removed_car_NA <- NA_car_data %>%
  filter(Engine.HP >= lower_HP_NA & Engine.HP <= upper_HP_NA) %>%
  filter(highway.MPG >= lower_highway_NA & highway.MPG <= upper_highway_NA) %>%
  filter(MSRP >= lower_MSRP_NA & MSRP <= upper_MSRP_NA)
```

Let us see the boxplots without outliers.

```
MSRP_noOutliers <- ggplot(outliers_removed_car_NA, aes(MSRP)) +
  geom_boxplot(fill = "plum3") + labs(x = "MSRP")
Highway_noOutliers <- ggplot(outliers_removed_car_NA, aes(highway.MPG)) +
```

```
  geom_boxplot(fill = "powderblue") + labs(x = "Highway MPG")
EngineHP_noOutliers <- ggplot(outliers_removed_car_NA, aes(Engine.HP)) +
  geom_boxplot(fill = "lavender") + labs(x = "Engine Horse Power")
noOutliers_plot <- MSRP_noOutliers + Highway_noOutliers + EngineHP_noOutliers + plot_layout(ncol = 1)
noOutliers_plot
```



Above are the updated boxplots after applying the transformations to the data. Some outliers remain which are intentionally kept to preserve the high range of values. We can now see the IQRs more clearly. The IQR for `MSRP` is approximately $20,000 to approximately $40,000.. The IQR for `highway.MPG` is approximately 24 to 31 mpg, and the IQR for `Engine.HP` is approximately 160 to 280 horsepower.

Transformation

Let's transform the data before visualizing.

```
converted_car_data_NA <- outliers_removed_car_NA %>%
    mutate(
        Market.Category = fct_other(Market.Category,
            keep = c("Performance", "Hatchback", "Crossover"),
            other_level = "Other"),
        Vehicle.Style = fct_lump(Vehicle.Style, n = 11, other_level = "Other"),
        Make = fct_lump(Make, n = 32, other_level = "Other"),
        Model = fct_lump(Model, n = 32, other_level = "Other"),
        MSRP = MSRP / 1000,
        Transmission.Type = relevel(factor(Transmission.Type), ref = "AUTOMATIC"),
        Vehicle.Size = factor(Vehicle.Size),
        Vehicle.Style = factor(Vehicle.Style),
        Engine.Fuel.Type = factor(Engine.Fuel.Type),
```

```
        Driven_Wheels = factor(Driven_Wheels),
        Transmission.Type = factor(Transmission.Type),
        Make = factor(Make),
        Model = factor(Model),
        Category = factor(Category)
    )
```

Here, I am lumping categories into "Other" in `Market.Category` that are not "Performance", "Hatchback", or "Crossover. I picked these 3 as they were in the top 10 with the highest number of observations. I also lumped categories in `Vehicle.Style`, `Make`, and `Model` keeping the top 11 and 32 categories and grouping all other observations as"Other." I divided the `MSRP` by a thousand to improve the visual range in the y-axis for better interpretation of the data. Lastly, I factored only the categorical values to represent distinct categories.

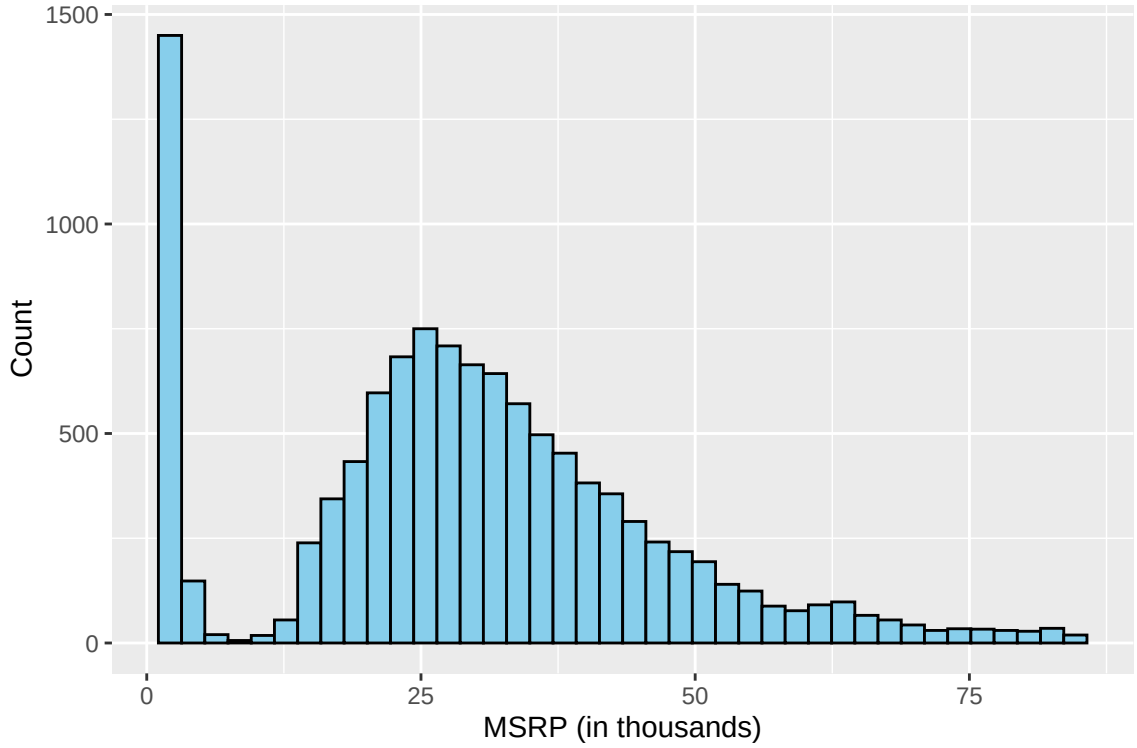
Visual Data

We will explore some relationships between the predictors and MSRP.

MSRP

Does the dataset have more data on cars with low or high MSRP?

```
ggplot(converted_car_data_NA, aes(x = MSRP)) +
  geom_histogram(bins = 40, fill = "skyblue", color = "black") +
  labs(x = "MSRP (in thousands)", y = "Count")
```



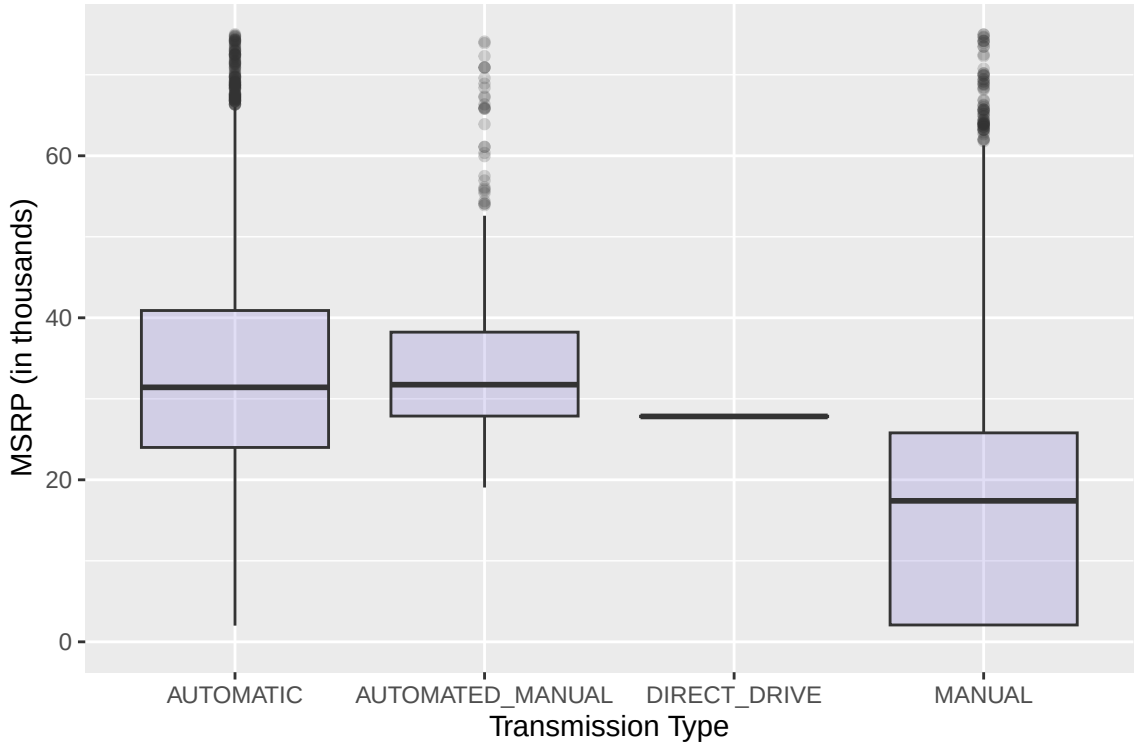
The histogram visualizes the distribution of `MSRP` with a range from close to $0 up to at least $85,000. The distribution is right skewed meaning most vehicles have a lower MSRP. The largest mode is near $0 to $5,000, this indicates there is a significant amount of cars with low MSRP and there is a another cluster around $20,000 to $30,000.

MSRP vs Transmission Type

How does MSRP vary by transmission type? Let us see with boxplots.

```
ggplot(converted_car_data_NA, aes(x = Transmission.Type, y = MSRP)) +
  geom_boxplot(fill = "slateblue", alpha = 0.2) + xlab("Transmission Type") +
  ylab("MSRP (in thousands)") + ylim(0,75)
```

```
## Warning: Removed 146 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



The boxplot visualizes the distribution of `MSRP` across different `Transmission.Type` categories.The median MSRP varies among transmission types, with "AUTOMATIC" and "AUTOMATED_MANUAL" having approximately equal highest median. The interquartile range (IQR), which is the spread of the data, reveals that "DIRECT_DRIVE" has the narrowest price range, indicating consistency in pricing for this transmission type.

Examining the whiskers,"MANUAL" has a longer spread toward the maximum, while "AUTOMATIC" displays an equal amount of spread toward the maximum and minimum values. Additionally, three out of the four transmissions have a good amount of outliers, highlighting the presence of cars with MSRP significantly above or below their boxplots.

MSRP vs Year

How does MSRP vary from 1990 to 2017?

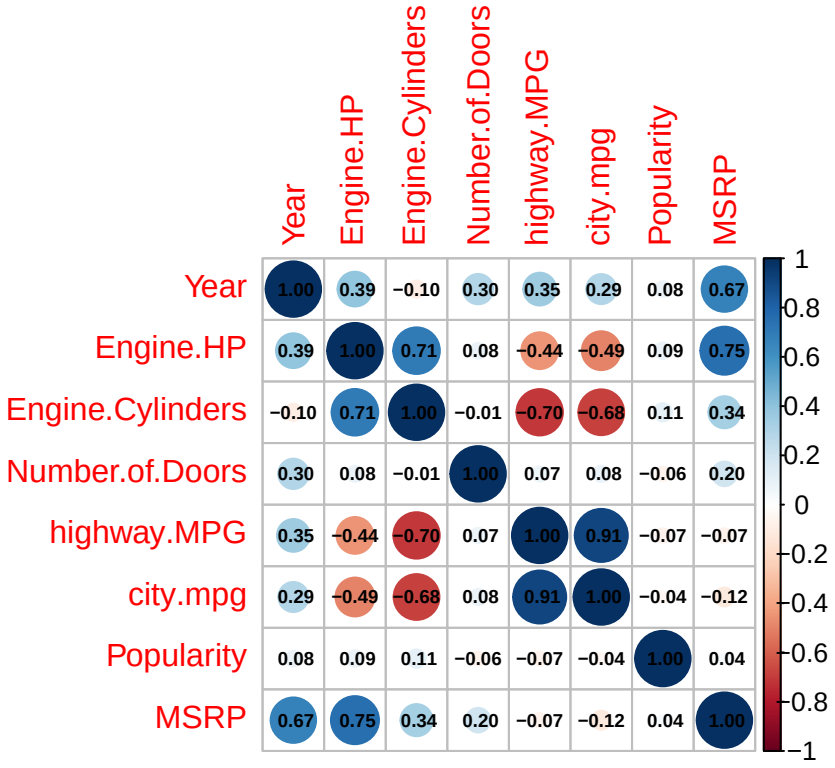
```
ggplot(converted_car_data_NA, aes(x = factor(Year), y = MSRP)) +
  geom_point(alpha = 0.7, color = "darkgreen")  +
  labs( x = "Year", y = "MSRP (in thousands)") + ylim(0,200) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

Looking at the scatterplot of the distribution of `MSRP` and `Year`, we see an upward trend in the MSRP values over the years. After the year 2000, the prices dramatically increased suggesting vehicle prices increase every year. The significant increase could indicate factors such economic changes, advancements in technology, inflation, or an increase in demand.

Correlation Matrix

Let's see the correlation matrix on the numerical values.

```
converted_car_data_NA %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(addCoef.col = 1, , number.cex = 0.6)
```



Now, we are going to explore the relationships between pairs of variables in the dataset. As a reminder, correlation coefficients range from -1 to 1, where values close to -1 or 1 indicating a stronger relationship.

- City MPG and Highway MPG: A very strong positive correlation (r = 0.91) shows that cars with better fuel efficiency in city conditions will also be efficient in highway conditions.
- Engine Cylinders and Engine HP: There is a strong positive correlation (r = 0.71), indicating vehicles with more engine cylinders tend to have higher horsepower.
- Engine Horsepower and MSRP: The strong positive correlation (r = 0.75) indicates vehicles with higher horsepower tend to have higher prices.
- Engine Cylinders and Fuel Efficiency: There is a strong negative correlation between engine cylinders and both city mpg (r = -0.68) and highway mpg (r = -0.70). This implies vehicles with more cylinders tend to have worse fuel efficiency in both city and highway conditions.
- Engine Horsepower and Fuel Efficiency: There is moderate negative correlation between engine horsepower and both city mpg (r = -0.49) and highway mpg (r = -0.44). This implies vehicles with more horsepower results in lower fuel efficiency.

Setting up Models

It is time to set up our models. This is where split our data and fit the training data into different models to see which one is the best to predict the MSRP. Before fitting, we have to create the recipe and the folds for k-fold cross validation.

Data Split

The first step would be to split the data into training and testing data sets. The training data is used to train the model to estimate `MSRP` based on observed data (James, pg.21). The testing data, which contains

the unseen observations, is used to test the model's accuracy (James, pg.30). I have chosen a 70/30 split, assigning 70% to the training set and 30% to the testing set. This split will help the model train while reserving a portion to test the accuracy on the remaining observations, helping to prevent overfitting. The split is stratified on the `MSRP` variable to ensure the same distribution on both the training and testing sets, and a random seed to reproduce my results consistently.

```
set.seed(314)

NA_car_split <- initial_split(converted_car_data_NA, prop = 0.7,
                              strata = MSRP)
NA_car_training <- training(NA_car_split)
NA_car_testing <- testing(NA_car_split)
```

To make sure the data is split evenly, we need to check the dimensions.

```
dim(NA_car_training)
```

```
## [1] 7664   17
```

```
dim(NA_car_testing)
```

```
## [1] 3288   17
```

The observations for the training and testing data sets are adequate.

Recipe

A recipe is a set of procedures to prepare your data for modeling, such as centering, scaling, and transforming categorical values into numerical values. Throughout our analysis, different recipes will be used for each model to accommodate their specific requirements. For stepwise linear regression, 12 predictors will be used: `Make`, `Model`, `Year`, `Engine.Fuel.Type`, `Engine.HP`, `Engine.Cylinders`, `Transmission.Type`, `Driven_Wheels`, `Market.Category`, `Vehicle.Size`, `Vehicle.Style`, and `city.mpg`.

For lasso regression, random forest, and Principal Component Analysis (PCA) regression, all available predictors will be used. In addition to the 11 mentioned above, the other predictors are: `Number.of.Doors`, `highway.MPG`, `Popularity`, and `Category`.

```
#Stepwise Regression
NA_stepwise_car_recipe <- recipe(MSRP ~ Make + Model + Year + Engine.Fuel.Type +
          Engine.HP + Engine.Cylinders + Transmission.Type + Driven_Wheels +
            Market.Category + Vehicle.Size +
    Vehicle.Style + city.mpg, data = NA_car_training) %>%
  step_dummy(Make, Model, Engine.Fuel.Type, Transmission.Type,
             Driven_Wheels, Market.Category,
             Vehicle.Style, Vehicle.Size)

#Lasso Regression
lasso_recipe_NA <- recipe(MSRP ~ ., data = NA_car_training) %>%
  step_dummy(Make, Model, Engine.Fuel.Type, Transmission.Type,
             Driven_Wheels, Market.Category, Vehicle.Style, Vehicle.Size, Category) %>%
  step_zv(all_predictors()) %>%
```

```
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

#Random Forest
rf_recipe_NA <- recipe(MSRP ~ ., data = NA_car_training) %>%
  step_unknown(Make, Model, Market.Category, Vehicle.Style) %>%
  step_dummy(Make, Model, Engine.Fuel.Type, Transmission.Type,
             Driven_Wheels, Market.Category, Vehicle.Style, Vehicle.Size, Category) %>%
  step_zv(all_predictors())

#PCA Regression
pca_recipe_NA <- recipe(MSRP ~ ., data = NA_car_training) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_pca(all_predictors(), num_comp = 2)
```

Stepwise Regression

As you can see in the Stepwise Regression recipe, it doesn't have all the variables like the rest of the models. The variables were picked using backward selection. Backward selection is a variable selection method which starts with all the variables and removes the least significant variable and gives the AIC in each step. The AIC is the Akaike Information Criterion, it evaluates how well the model fits the data.

```
lumped_full_model_NA <- lm(MSRP ~ ., data = converted_car_data_NA) %>%
  stats::step(direction = "backward")
```

```
## Start:  AIC=40430.66
## MSRP ~ Make + Model + Year + Engine.Fuel.Type + Engine.HP + Engine.Cylinders +
##     Transmission.Type + Driven_Wheels + Number.of.Doors + Market.Category +
##     Vehicle.Size + Vehicle.Style + highway.MPG + city.mpg + Popularity +
##     Category
##
##                       Df Sum of Sq    RSS   AIC
## - Number.of.Doors      1        69 431245 40430
## <none>                             431177 40431
## - Engine.Cylinders     1       899 432076 40451
## - highway.MPG          1      1207 432384 40459
## - Transmission.Type    3      1558 432735 40464
## - Popularity           1      3271 434448 40511
## - city.mpg             1      4145 435322 40533
## - Category             1      4241 435418 40536
## - Market.Category      3      4403 435580 40536
## - Driven_Wheels        3      8968 440144 40650
## - Vehicle.Size         2     13965 445142 40776
## - Vehicle.Style       11     29618 460795 41136
## - Model               32     37747 468924 41286
## - Engine.Fuel.Type     7     45338 476515 41512
## - Make                32     69048 500225 41993
## - Engine.HP            1     68830 500007 42051
## - Year                 1    133643 564819 43386
```

```
##
## Step:  AIC=40430.4
## MSRP ~ Make + Model + Year + Engine.Fuel.Type + Engine.HP + Engine.Cylinders +
##     Transmission.Type + Driven_Wheels + Market.Category + Vehicle.Size +
##     Vehicle.Style + highway.MPG + city.mpg + Popularity + Category
##
##                     Df Sum of Sq    RSS   AIC
## <none>                           431245 40430
## - Engine.Cylinders   1       911 432157 40452
## - highway.MPG        1      1256 432502 40460
## - Transmission.Type  3      1653 432899 40466
## - Popularity         1      3265 434511 40511
## - city.mpg           1      4184 435429 40534
## - Category           1      4223 435469 40535
## - Market.Category    3      4430 435676 40536
## - Driven_Wheels      3      8899 440144 40648
## - Vehicle.Size       2     13921 445167 40774
## - Vehicle.Style     11     29612 460857 41136
## - Model             32     37755 469001 41286
## - Engine.Fuel.Type   7     45283 476529 41510
## - Make              32     69004 500249 41992
## - Engine.HP          1     68781 500026 42049
## - Year               1    139119 570365 43491
```

Now, we have the significant predictors based on the backward selection. It is time to check the VIF, Variance Inflation Factor. The VIF function measures the amount of multicollinearity in the regression equation. Multicollinearity is when two independent variables are correlated with each other. As you can see, there are some variables missing from the backward selection variables and the `model_NA` variables. I removed `Popularity` and `Category` for their high collinearity. Additionally, I removed `highway.MPG` because it was correlated with `city.mpg` between the two `highway.MPG` was more strongly connected to other variables.

```
model_NA <- lm(MSRP ~ Make + Model + Year + Engine.Fuel.Type + Engine.HP + Engine.Cylinders +
    Transmission.Type + Driven_Wheels + Market.Category + Vehicle.Size +
    Vehicle.Style + city.mpg, data = converted_car_data_NA)
vif_values_NA <- vif(model_NA)
print(vif_values_NA)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## Make               1483.243587 32        1.120857
## Model              1751.181461 32        1.123769
## Year                  3.779453  1        1.944082
## Engine.Fuel.Type      9.089676  7        1.170760
## Engine.HP             7.402325  1        2.720721
## Engine.Cylinders      5.561377  1        2.358257
## Transmission.Type     2.837066  3        1.189812
## Driven_Wheels         5.976073  3        1.347109
## Market.Category       7.563376  3        1.401047
## Vehicle.Size          4.691182  2        1.471704
## Vehicle.Style       180.855333 11        1.266502
## city.mpg              3.723436  1        1.929621
```

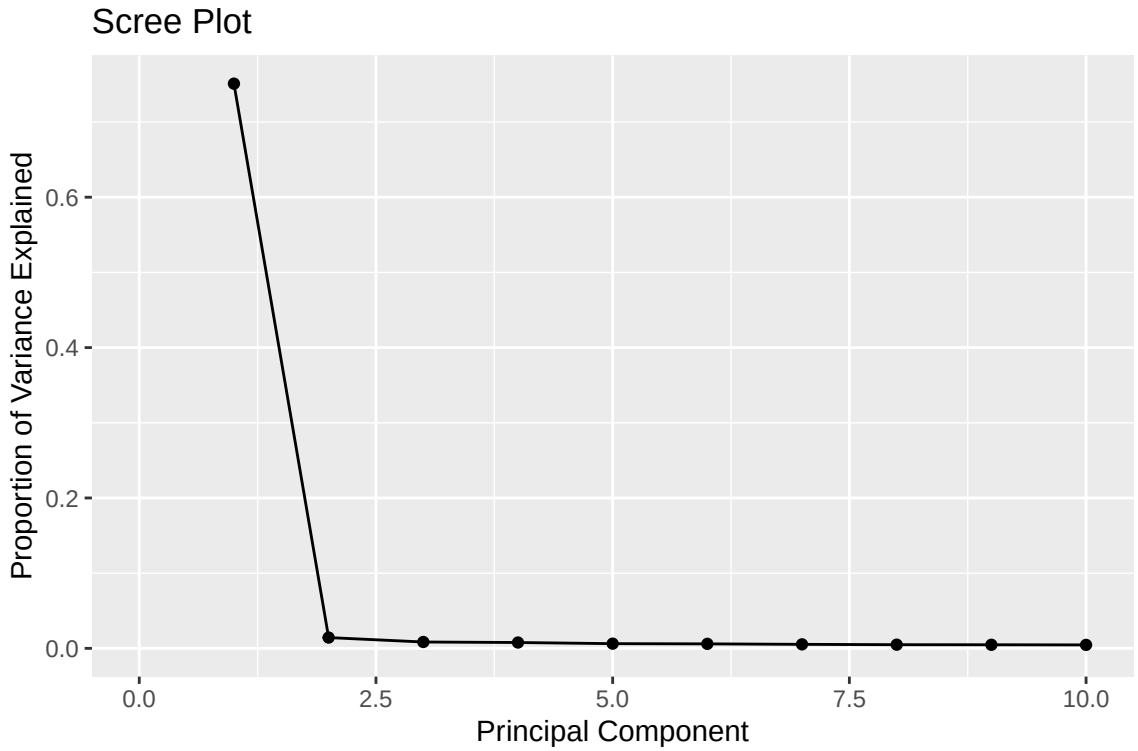
Principal Component Analysis (PCA)

To determine the number of principal components to keep in our principal component analysis, we converted the categorical variables into numerical values, as PCA requires numeric data. Principal components are linear combinations of the initial variables, constructed into new variables that capture the most variance (Jaadi). After preparing the data, we built a scree plot to visualize the proportion of variance explained by each component, helping us decided how many components to keep. The recipe above is updated on the number of components to be kept.

```
pca_recipe_1 <- recipe(MSRP ~ ., data = NA_car_training) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_center(all_numeric_predictors()) %>%
  step_scale(all_numeric_predictors())

prepped_recipe_1 <- prep(pca_recipe_1, training = NA_car_training)
pca_data_1 <- bake(prepped_recipe_1, new_data = NULL)

pca_results_1 <- prcomp(pca_data_1, scale. = FALSE)

var_explained_1 <- pca_results_1$sdev^2 / sum(pca_results_1$sdev^2)

scree_data_1 <- data.frame(
    PC = 1:length(var_explained_1),
    Variance_Explained = var_explained_1
)

ggplot(scree_data_1, aes(x = PC, y = Variance_Explained)) +
   geom_line() +
    geom_point() +
    labs(title = "Scree Plot", x = "Principal Component", y = "Proportion of Variance Explained") +
     xlim(0,10)
```

```
## Warning: Removed 91 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 91 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The Scree Plot demonstrates an elbow shape graph. This indicates the point at which additional components contribute little to the proportion of variance. The proportion of variance is the amount of total variation explained by each principal component (James, 503). by Anything before the elbow point explains a significant proportion of the variance and should be kept, while components after contribute less and should not be kept. Based on this, we should only keep two principal components.

K-Fold Cross Validation

We are using K-Fold Cross Validation, which randomly divides the set of observations into 5 folds that are approximately equal size (James, pg.203). The first fold is treated as a validation set and the model is trained on the remaining k-1 folds. A performance metric, such as RMSE or R^2, is then computed from the remaining observations used to evaluate the model. This process is repeated 5 times with a different group of observations used as a validation set. Finally, it then averages the test error across all folds.

```
set.seed(314)
NA_car_folds <- vfold_cv(NA_car_training, v = 5, strata = MSRP)
```

Model Building

In this section, we implement several models to predict `MSRP` and to learn more about the variables influencing car prices. We will build the model using the tidied data from earlier, prepare the training data, select appropriate models, train the models on the data, fine tune the parameters, and evaluate each model's performance. We then choose the model with the best metric. I will be using Root Mean Square Error (RMSE) as my metric for all of the models. RMSE is used to quantify how far predictions are from the true values. The lower the RMSE, the closer the predictions are from the true values and the higher the RMSE, the farther away the predictions are from the true values.

As seen above in the recipe building, I will be using Stepwise Linear Regression, Lasso Regression, Random Forest, and PCA Regression. Each model contributes its own unique strengths: stepwise regression examines the significance of each variable (Hayes), lasso eliminates the less important predictors in the model by shrinking their coefficients towards zero ("What Is Lasso Regression?"), random forest provides information about the significance of the predictors in the data ("What Are the Advantages and Disadvantages of Random Forest?"), and PCA regression reduces dimensionality (James, pg.253).

Fitting the Models

The models used are going to have the same procedure. Here are the steps:

1. Set up the model by specifying what type of model, engine, and mode. For Lasso and Random Forest, add in hyperparameters.

```
#Stepwise Regression
NA_stepwise_model <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

#Lasso Regression
lasso_model_NA <- linear_reg(
  mixture = 1,
  penalty = tune()
) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

#Random Forest
rf_model_NA <- rand_forest(
  mtry = tune(),
  trees = tune(),
  min_n = tune()
) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

#PCA Regression
pca_model_NA <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

2. Set up the workflow, add in the model and recipe.

```
#Stepwise Regression
NA_stepwise_workflow <- workflow() %>%
  add_model(NA_stepwise_model) %>%
  add_recipe(NA_stepwise_car_recipe)

#Lasso Regression
lasso_workflow_NA <- workflow() %>%
  add_model(lasso_model_NA) %>%
  add_recipe(lasso_recipe_NA)
```

```
#Random Forest
rf_workflow_NA <- workflow() %>%
  add_model(rf_model_NA) %>%
  add_recipe(rf_recipe_NA)

#PCA Regression
pca_workflow_NA <- workflow() %>%
  add_model(pca_model_NA) %>%
  add_recipe(pca_recipe_NA)
```

3. Create a grid and tune for those with hyperparameters in step 1.

```
#No tuning for Stepwise Regression

#Lasso Regression
lasso_grid_NA <- grid_regular(penalty(), levels = 30)

#Random Forest
#I commented out the rf code to avoid rerunning the grid_regular
#rf_grid_NA <- grid_regular(mtry(range = c(1,5)),
                           #trees(range = c(200, 650)),
                           #min_n(range = c(5, 15)),
                           #levels = 10)

#No tuning for PCA Regression
```

4. Tune the models with grids by adding in the workflow, k-fold cross validation, and grid. For the other models, fit the workflow and k-folds.

```
#Stepwise Regression
NA_stepwise_fit <- fit_resamples(NA_stepwise_workflow, NA_car_folds)
```

```
## > A | warning: prediction from rank-deficient fit; consider predict(., rankdeficient="NA")

## There were issues with some computations   A: x1There were issues with some computations   A: x2There
```

```
#Lasso Regression
lasso_tunegrid_NA <- tune_grid(
  lasso_workflow_NA,
  resamples = NA_car_folds,
  grid = lasso_grid_NA
)

#Random Forest
#I commented out this code since rf_grid_NA is commented out
#rf_tune_NA <- tune_grid(
  #rf_workflow_NA,
  #resamples = NA_car_folds,
  #grid = rf_grid_NA,
  #metrics = metric_set(rmse)
#)
```

```
#PCA Regression
pca_fit_NA <- fit_resamples(pca_workflow_NA, NA_car_folds)
```

5. Save the tuned model.

```
#No tuning for Stepwise Regression

#Lasso
save(lasso_tunegrid_NA, file = "lasso_tunegrid_NA.rda")

#Random Forest
#I commented out this since rf_tune_NA is commented out
#save(rf_tune_NA, file = "~/Desktop/rf_tune_NA.rda")

#No tuning for PCA Regression
```

6. Load the saved files

```
#Lasso
load("lasso_tunegrid_NA.rda")

#Random Forest
load("~/Desktop/rf_tune_NA.rda")
```

7. Collect the metrics from each model, filter rmse, arrange from ascending order, and slice to show the first result.

```
#Stepwise Regression
NA_stepwise_metrics <- collect_metrics(NA_stepwise_fit) %>%
  filter(.metric == "rmse") %>%
  arrange(mean) %>%
  slice_head(n=1)
NA_stepwise_metrics
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    6.34     5   0.126 Preprocessor1_Model1
```

```
#Lasso Regression
lasso_metrics_NA <- collect_metrics(lasso_tunegrid_NA )%>%
  filter(.metric == "rmse") %>%
  arrange(mean) %>%
  slice_head(n=1)

lasso_metrics_NA
```

```
## # A tibble: 1 x 7
##   penalty .metric .estimator  mean     n std_err .config
##     <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.00174 rmse    standard    6.32     5   0.128 Preprocessor1_Model22
```

```r
#Random Forest
rf_metrics_NA <- rf_tune_NA %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  arrange(mean) %>%
  slice_head(n=1)

rf_metrics_NA
```

```
## # A tibble: 1 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5   300     6 rmse    standard    5.79     5   0.157 Preprocessor1_Model0~
```

```r
#PCA Regression
pca_metrics_NA <- collect_metrics(pca_fit_NA) %>%
  filter(.metric == "rmse") %>%
  arrange(mean) %>%
  slice_head(n=1)

pca_metrics_NA
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard    12.1     5   0.105 Preprocessor1_Model1
```

These values indicate the hyperparameters that produced the smallest RMSE.

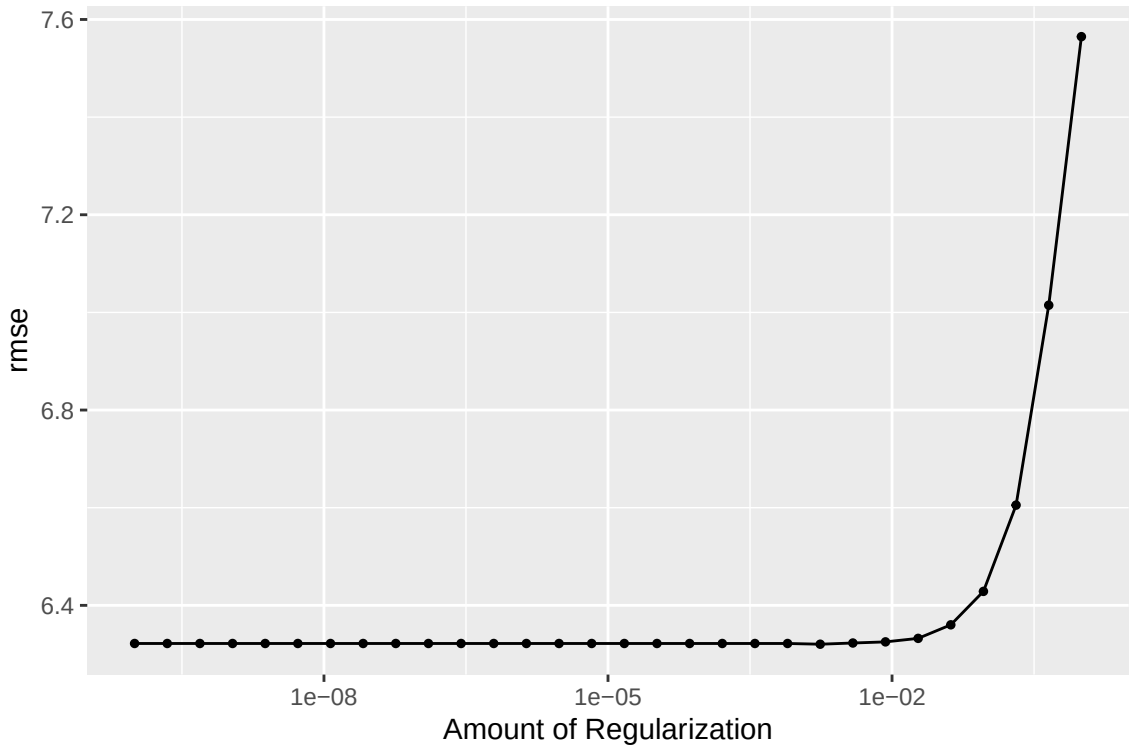
Model Autoplot

Stepwise Regression

Stepwise Regression: Stepwise Regression does not have model autoplot but was used as an inferential model to identify significant predictors of `MSRP`. The model began with all predictors and removed non-significant predictors using backward selection. The final model included `Make`, `Year`, `Engine.Fuel.Type`, `Engine.HP`,`Transmission.Type`, `Driven_Wheels`,`Market.Category`,`Vehicle.Size`,`Vehicle.Style`, and `city.mpg`. The model was tuned using 5-fold cross validation and achieved an RMSE of 6.34.

Lasso

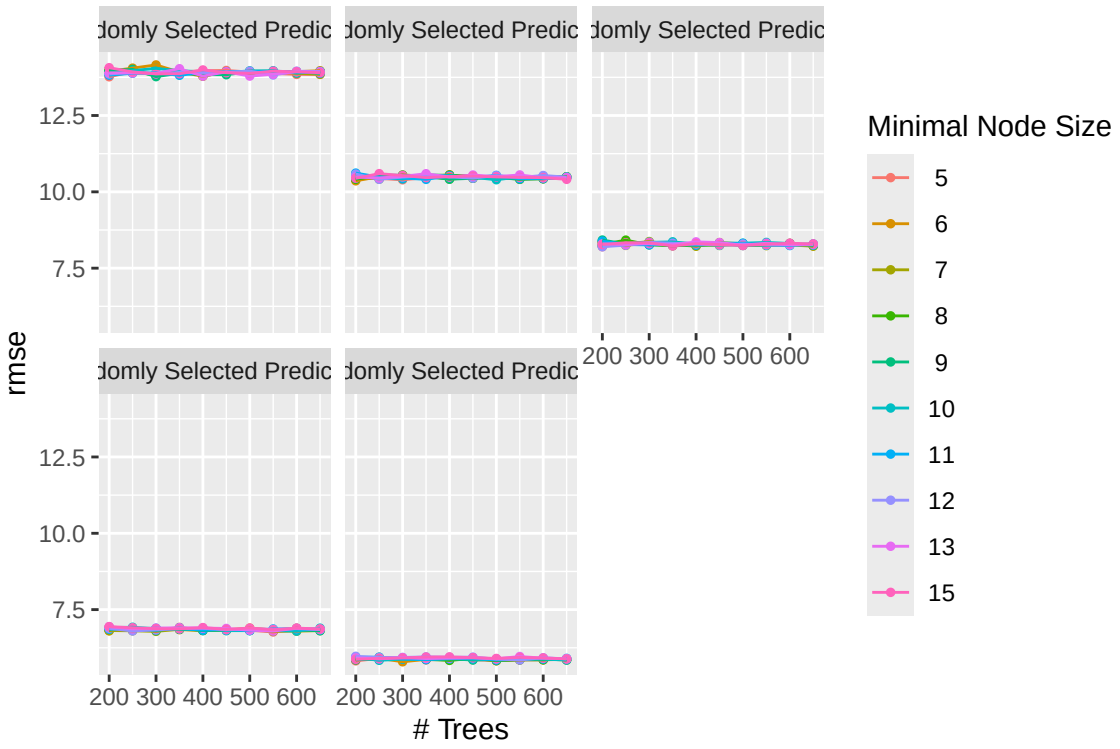
```r
#Lasso
autoplot(lasso_tunegrid_NA, metric = 'rmse')
```



Lasso Regression: Lasso Regression was used as a predictive model. The hyperparameter `penalty()` was tuned using 5-fold cross validation over a grid of 30 levels. The `penalty()` function shrinks the coefficients towards zero (James, 241). The RMSE is low and stable when the amount of regularization, how much penalty is applied, is around 1e-08 to 1e-02. After 1e-02, the RMSE increases significantly. The best penalty was 0.00174 with an RMSE of 6.32.

Random Forest

```
#Random Forest
autoplot(rf_tune_NA)
```



Random Forest: Random Forest was used to optimize prediction accuracy. The hyperparameters `mtry()`, `trees()`, and `min_n()` were tuned using grid regular: `mtry()` tested values from 1 to 5, `trees()` tested values from 200 to 650, and `min_n()` tested values from 5 to 15. `mtry` specifies the number of randomly selected variables. `trees` specifies the number of individual decision trees generated in the random forest. `min_n` specifies the minimum number of data points required to keep splitting the nodes. The hyperparameters were tuned over a grid of 10 levels and 5-fold cross validation. The best `mtry` was 5, `trees` was 300, and `min_n` was 6 with an RMSE of 5.79. As we can see, almost all of the minimal node sizes tend to have the same RMSE in each graph and the number of trees seems to not have an effect either.

PCA Regression

PCA Regression: PCA Regression does not have model autoplot, but was used to reduce dimensionality while keeping the important predictors. The model had to begin with creating a scree plot, which told us how many principal components to keep. This was important because it showed us how many principal components contribute to the proportion of variance. The recipe was then updated and the model was created and tuned with 5-fold cross validation. It had a RMSE of 12.10.

Model Results

We are going to compare the results of the models and see which one has the best RMSE.

```
stepwise_results <- tibble(
  Model = "Stepwise Regression",
  RMSE = 6.339756
)
lasso_results <- tibble(
  Model = "Lasso Regression",
```

```
  RMSE = 6.320401
)
randomf_results <- tibble(
  Model = "Random Forest",
  RMSE = 5.790814
)
pca_results <- tibble(
  Model = "PCA Regression",
  RMSE = 12.10727
)
all_metrics <- bind_rows(stepwise_results, lasso_results, randomf_results, pca_results)

print(all_metrics)
```

```
## # A tibble: 4 x 2
##   Model                RMSE
##   <chr>               <dbl>
## 1 Stepwise Regression  6.34
## 2 Lasso Regression     6.32
## 3 Random Forest        5.79
## 4 PCA Regression      12.1
```

As we can see Random Forest performed the best with the lowest RMSE. Our second best was Lasso Regression, third was Stepwise Regression, and last was PCA Regression. Something we can take from this is that Random Forest has the best prediction accuracy and can handle large number of features ("Random Forest Regression.")

As mentioned above our best parameters were 300 trees and 6 minimal nodes with 5 predictors and an RMSE of 5.79.

Fitting Best Model

Now that we have identified the model with the best RMSE, it will be used to make predictions on the testing data. The selected model will be trained on the full training dataset using the best parameters obtained during tuning.

```
set.seed(314)
best_rf1 <- select_best(rf_tune_NA, metric = "rmse")

rforest_final <- finalize_workflow(rf_workflow_NA, best_rf1)

rforest_final_fit <- fit(rforest_final, data = NA_car_training)
```

Testing Model

It is now time to evaluate our best model on unseen observations using the testing data. This step will demonstrate how well the model can predict unseen observations.

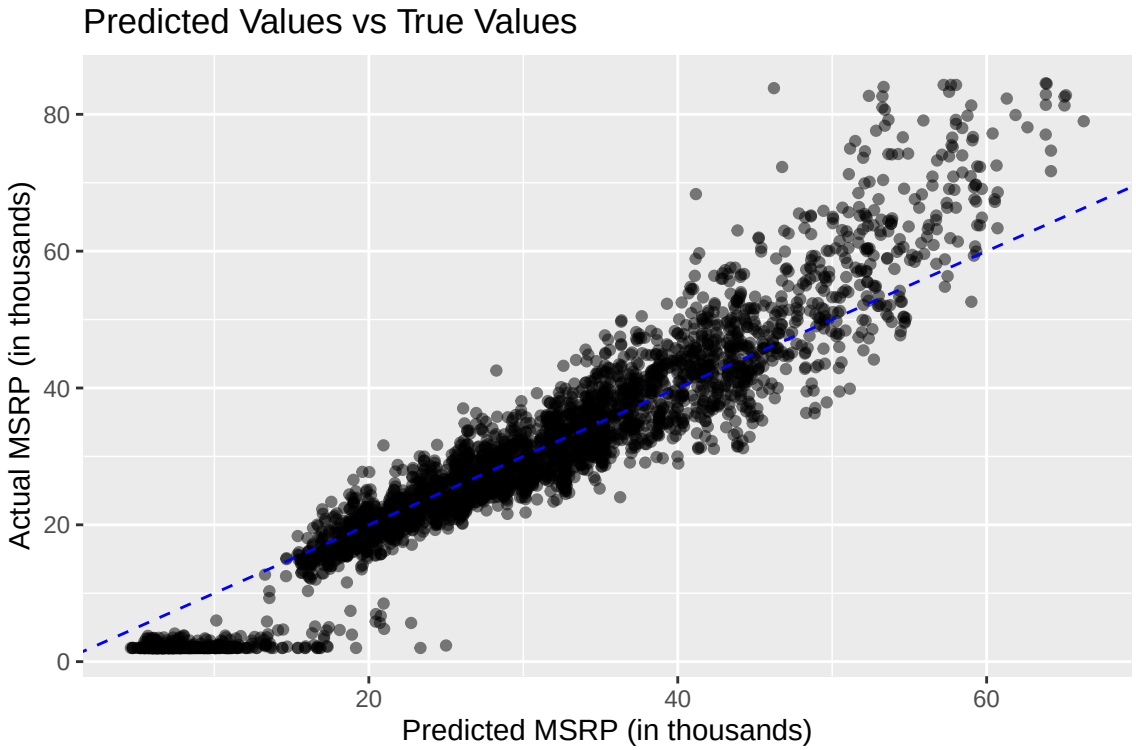
```
augment_rf <- augment(rforest_final_fit, new_data = NA_car_testing)

metric_rf <- augment_rf %>%
```

```
  metrics(truth = MSRP, estimate = .pred)

metric_rf %>% filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        5.73
```

Woohoo! Our testing RMSE is lower than our training RMSE. This means the prediction was on average 5.73 away from the true value.

```
ggplot(augment_rf, aes(x=.pred, y = MSRP)) + geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "blue", linetype = "dashed") +
  labs(x = "Predicted MSRP (in thousands)",
       y = "Actual MSRP (in thousands)",
       title = "Predicted Values vs True Values")
```

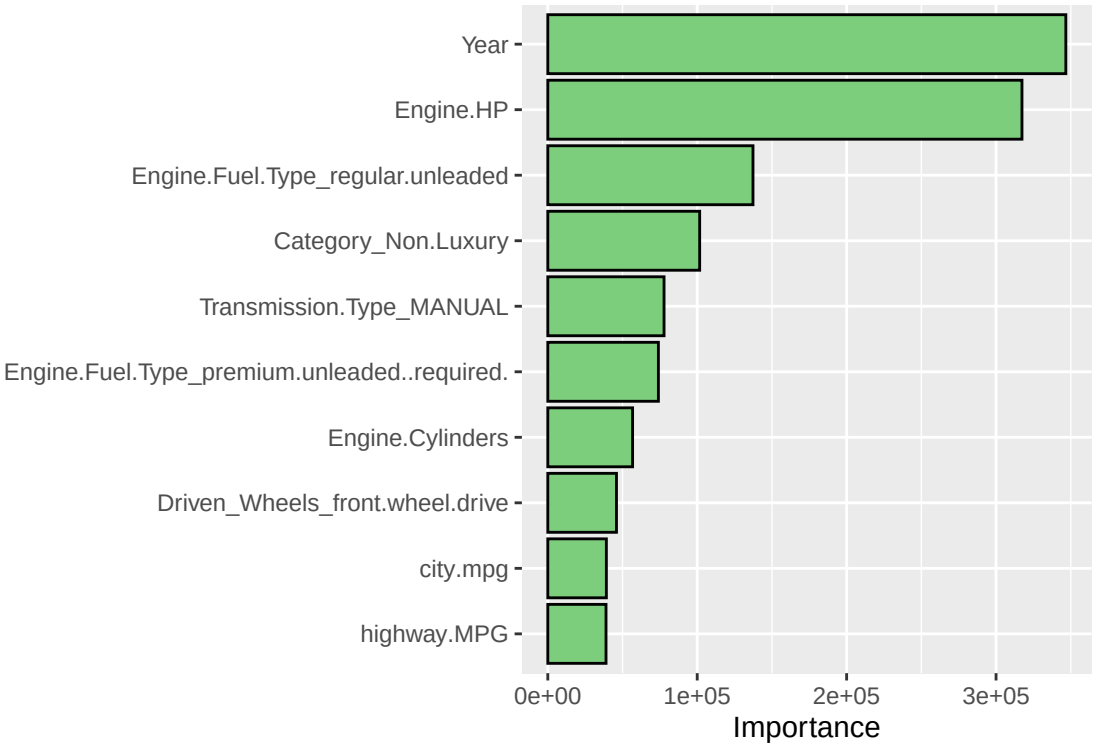


The testing model performed well in predicting MSRP values, as most of the predicted values lie close to the diagonal slope. However, the predictions on the extreme ends of the MSRP range are farther away from the line. This indicates the model was well-trained for cars priced between $18,000 to $42,000, but struggles with predicting MSRP values for cars priced significantly lower or higher. To get a more accurate predictions, we might need to focus more on either the lower or upper tails.

Variable Importance Plot

This plot demonstrates which variables were the most important on the best-performing random forest model.

```
rforest_final_fit %>%
  extract_fit_parsnip() %>%
  vip(geom = "col", aesthetics = list(fill = "palegreen3", color = "black"))
```



The two most significant features were `Year` and `Engine.HP`. This suggests that newer cars, which often have more advanced technology, tend to have higher MSRPs. Similarly, cars with better performance give a better driving experience which demand higher MSRPs. Additionally other significant features include `Engine.Fuel.Type_regular.unleaded`, and `Category_Non.Luxury`, which demonstrate the influence of fuel type and affordability. These features indicate the importance of technological features, driving experience, fuel availability, and affordability when it comes to MSRP.

Price Difference between Transmission Types

It is time to find out if there is a significant price difference between transmission types.

```
tidy_model <- tidy(model_NA)
transmission_summary <- tidy_model %>%
  filter(str_detect(term, "Transmission.Type"))
transmission_summary
```

```
## # A tibble: 3 x 5
##   term                              estimate std.error statistic  p.value
##   <chr>                                <dbl>     <dbl>     <dbl>    <dbl>
## 1 Transmission.TypeAUTOMATED_MANUAL   -0.791     0.395    -2.00  4.54e- 2
## 2 Transmission.TypeDIRECT_DRIVE       -3.47      4.52     -0.766 4.43e- 1
## 3 Transmission.TypeMANUAL             -1.33      0.191    -6.97  3.27e-12
```

Note: 4.54e- 2 corresponds to 0.045, 4.43e- 1 corresponds to 0.44, and 3.27e-12 corresponds to 3.27e-12.

The table represents a subset of the summary table focusing specifically on transmission type. I did this because the full summary table includes all the variables that are statistically significant to MSRP. As a reminder, a p-value less than 0.05 implies statistical significance. Earlier in the data transformation, I set `Transmission.TypeAUTOMATIC` as the reference category because it is the most common type of transmission in the market. Based on this reference, `Transmission.TypeAUTOMATED_MANUAL` and `Transmission.TypeMANUAL` are significant in relation to MSRP.

Direct Drive: -3.467

The estimate of -3.467 means, on average, cars with Direct Drive transmissions are $3,467 lower in MSRP than cars with Automatic transmissions. The price difference is not significant, therefore not reliable.

Manual: -1.333

The estimate of -1.333 means, on average, cars with Manual transmissions are $1,333 lower in MSRP than cars with Automatic transmissions. The price difference between Manual and Automatic transmissions is significant.

Automated Manual: -0.7911

The estimate of -0.7911 means, on average, cars with Automated Manual transmissions are $791.10 lower in MSRP than cars with Automatic transmissions. The price difference between Automated Manual and Automatic transmissions is significant.

Conclusion

At the beginning of this project, we set out to predict the MSRP based on car features. After cleaning and visualizing the data, we learned more about the range of the values. We continued on with creating our training and testing sets with the four models: Stepwise Regression, Lasso Regression, Random Forest, and PCA Regression. After building and fitting the models, we compared them by RMSE. The PCA Regression performed poorly, with the highest RMSE of 12.10. Stepwise Regression followed with a value of 6.34, then Lasso Regression with 6.32. Finally, Random Forest performed the best, with the lowest RMSE of 5.79.

I was not surprised about Random Forest's performance due to its resilience to noise and high accuracy for predictions. However, I was surprised by Stepwise Regression as it wasn't that far off from the RMSE value of Random Forest.

We concluded Random Forest was the best model and used it on our testing set. In addition to performing better than the other models, it provided the significant features, year, engine horsepower, regular unleaded engine fuel type, and non-luxury category. Overall, these insights can possibly help adults understand what features influence car prices and keep it in mind when buying a new car.

In addition to being the second-best model in this project, Stepwise Regression offers a summary table demonstrating the significance of each variable in relation to MSRP. We found that cars with Automated Manual and Manual transmissions are statistically significant when compared to the reference category, Automatic transmissions. This makes their price differences meaningful. Cars with Automated Manual, on average, are $791.10 lower in price than cars with Automatic transmissions. Similarly, cars with Manual transmissions are on average $1,333 lower in price than cars with Automatic transmissions. In contrast, cars with Direct Drive transmissions, on average, are $3,467 lower in price than cars with Automatic transmissions, but the difference is not significant. The price difference is not reliable. These results demonstrate how a car's transmission type can influence the price of the car.

One limitation of this study was the lack of data on recent cars with technology. For future research, I would like to implement more predictors into the dataset such as, whether a car has Bluetooth, blind-spot monitoring, a back-up camera, sunroof, and more. These variables could provide insight whether technological advancements impact MSRP.

Sources

"Automated Manual Transmissions (AMT): Pros and Cons." Automatic Transmissions R Us, 30 Oct. 2024, www.autotransrus.com.au/blog/amt/#:~:text=AMT%20offers%20a%20compromise%20between,ease%20of%20use%20is%20compromised. Accessed 06 Dec. 2024.

Conners, Angela. Monroney Sticker. Cars.com, https://www.cars.com/articles/what-does-msrp-mean-1420690419467/

Eaton. "Direct Drive or Overdrive?" Trucking Info, 7 Aug. 2014, www.truckinginfo.com/155143/direct-drive-or-overdrive#:~:text=It%20all%20depends%20on%20the,overdrive%20becomes%20a%20better%20fit. Accessed 06 Dec. 2024.

Figure 1."Car Price Tag." Adobe Stock, stock.adobe.com/search?k=car%2Bprice%2Btag%29. Accessed 07 Dec. 2024.

Figure 2. Conners, Angela. Monroney Sticker. Cars.com, https://www.cars.com/articles/what-does-msrp-mean-1420690419467/

Figure 3. "7 Reasons LED Lighting Is the Right Choice for Automotive Dealerships." Cree Lighting, 8 July 2022, www.creelighting.com/insights/article/7-reasons-led-lighting-is-the-right-choice-for-automotive-dealerships/. Accessed 07 Dec. 2024.

Hayes, Adam. "Stepwise Regression: Definition, Uses, Example, and Limitations." Investopedia, 10 Jan. 2022, www.investopedia.com/terms/s/stepwise-regression.asp. Accessed 06 Dec. 2024.

Ibm. "What Is Lasso Regression?" IBM, 7 Aug. 2024, www.ibm.com/topics/lasso-regression#:~:text=Lasso%20regression%20can%20help%20to,important%20features%20from%20the%20model.& text=The%20bias%20introduced%2 Accessed 06 Dec. 2024.

Indeed Editorial Team. "Understanding MSRP: A Guide to Manufacturer Pricing ." Indeed.Com, www.indeed.com/career-advice/career-development/msrp. Accessed 06 Dec. 2024.

Jaadi, Zakaria. "Principal Component Analysis (PCA): A Step-by-Step Explanation." Built In, builtin.com/data-science/step-step-explanation-principal-component-analysis#:~:text= Principal%20components%20are%20 Accessed 06 Dec. 2024.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2021) An Introduction to Statistical Learning: With Applications in R. 2nd Edition, Springer, Berlin. https://www.statlearning.com

Rana, Rupinder Singh. Kaggle. Car Features and Prices Dataset, Version 1.0, 2024, Kaggle, https://www.kaggle.com/datasets/rupindersinghrana/car-features-and-prices-dataset/data.

"Random Forest Regression." HEAVY.AI Docs, docs.heavy.ai/heavyml-beta/regression-algorithms/random-forest-regression. Accessed 06 Dec. 2024.

"7 Reasons LED Lighting Is the Right Choice for Automotive Dealerships." Cree Lighting, 8 July 2022, www.creelighting.com/insights/article/7-reasons-led-lighting-is-the-right-choice-for-automotive-dealerships/. Accessed 07 Dec. 2024.

"What Are the Advantages and Disadvantages of Random Forest?" AIML.Com, 22 May 2024, aiml.com/what-are-the-advantages-and-disadvantages-of-random-forest/. Accessed 06 Dec. 2024.

"Which Transmission Type Is Right for You?" Dennis Dillon Mazda, www.dennisdillonmazda.com/. Accessed 03 Dec. 2024.

"Why Choose Led Car Dealership Lighting - Cree Lighting." Why Choose LED Car Dealership Lighting - Cree Lighting, www.creelighting.com/insights/article/7-reasons-led-lighting-is-the-right-choice-for-automotive-dealerships/. Accessed 03 Dec. 2024.