Final Capstone Project:
Vehicle Price Prediction & Recommendation

Farouk Ibrahim

April 18, 2024

Contents

1 Introduction

Welcome to "Vehicle Price Prediction & Recommendation" Project! This project is the final project submission for the capstone course of the Harvard Online Data Science series[1]. The R markdown file for this report and the PDF are available on GitHub[2].

Many of the residents in Qatar and GCC in general, have a habit of changing their cars on a frequent basis. They like to experience different types of cars every once in a while. Residents including myself at some point of time used browse hundreds of pages and car listing in our free time to find a suitable car either for fun or for profit. The existing car listing sites have added many new features that allows you to filter and sort, however there is a limit to the analysis and search capability you can perform for you to find the best deals.

In search for a final project and to automated the manual car hunt process, I am excited that to have choose this project using real live market data which I am familiar with. This system will recommend the best car deals in an automated and scientific method leveraging on the knowledge and skills I have acquired during this data science series.

The goal for this project is develop a price prediction and recommendation system based on the up to date real dataset. The recommendation of best deals us based on the comparing of the prediction vs the actual car price along other parameter filtering.

If you find this project interesting or if you have any questions I will be more than happy to connect [3].

2 Dataset

For this project, the dataset used will be web scrapped from one of the most popular car selling websites[4]. I have scrapped data using a free chrome extension called Web Scrapper. This data set named "input04Apr202410pm.csv" is available in the root folder in the same github link. This dataset has over 17,000+ recent car listing and was scrapped on 9th April 2024.

Some data wrangling techniques will be used to tidy the data. The dataset will also be filtered having car price of less than 100,000 USD (365,000 QAR) and the car age to be at least 10 years old (where model year >= 2014). The filter selection is based on the availability of data in this years and price range to improve the prediction model accuracy.

The dataset will be is split to 10 % validation "validate_set" and 90% training dataset "train_set".

Training set is further split for training "train_set" and test "test_set".

As a practical test, additional more recent data have been scrapped on 9th April 2024 and stored in this file "pred09Apr202411am.csv" which contains around 350+ new car listing which will be used as an input to the system. The system will then intelligently recommend the best deals based on the model trained and the new unseen recent data scrapped.

3 Evaluation

The Root Mean Squared Error (RMSE) metric function was used to evaluate the difference between predicted price values and actual price values. RMSE was used to evaluate how close my predictions are to the true values in the "validate_set" set.

[1] https://www.harvardonline.harvard.edu/course/data-science-professional-certificate
[2] https://github.com/FaroukIbrahim/harvard_final_project
[3] https://www.linkedin.com/in/farouk-ibrahim-72394b7/
[4] https://qatarsale.com/

The goal of the RMSE value is based on a best effort approach using multiple advance data science algorithms.

The goal of this project is to recommend the best car deals by providing a list of cars where their actual price is far lower (20%) than predicted market value of this type of vehicle.

4 Analysis

4.1 Data Wrangling

In this section, the required packages will be loaded and installed. The source dataset will be loaded. Some data wrangling techniques will be used to tidy the data. The dataset will also be filtered having car price of less than 100,000 USD (365,000 QAR) and the car age to be at least 10 years old (where model year >= 2014).

4.1.1 Install Packages

Install and load the required packages and libraries.

```
if (!require(magrittr)) install.packages("magrittr")
if (!require(dplyr)) install.packages("dplyr")
library(magrittr)
library(dplyr)
library(caret)
library(stringr)
library(scales)
```

4.1.2 Import Data

Dataset will be read from a csv from the root folder. There are 17,748 records in this dataset.

```
# Read the csv file from relative path
data <- read.csv("input04Apr202410pm.csv")
nrow(data)
```

```
## [1] 17748
```

4.1.3 Tidy Data

First let's view the data and the 16 columns. "mileage", "price" and "cylinder" columns will be formatted to number removing any comma. Car Date sold and last posted "time" will be extracted from the time column. Finally, we "mileage_range" column will be created based on mileage tiers of every 10,000 km.

```
# lets first understand what columns we have
head(data)
```

```
##   web.scraper.order                                    web.scraper.start.url
## 1      1712250886-1 https://qatarsale.com/en/products/cars_for_sale?page=500
## 2      1712250886-2 https://qatarsale.com/en/products/cars_for_sale?page=500
## 3      1712250886-3 https://qatarsale.com/en/products/cars_for_sale?page=500
```

```
## 4      1712250886-4 https://qatarsale.com/en/products/cars_for_sale?page=500
## 5      1712250886-5 https://qatarsale.com/en/products/cars_for_sale?page=500
## 6      1712250886-6 https://qatarsale.com/en/products/cars_for_sale?page=500
##
## 1 https://qsbproduction.blob.core.windows.net/12756-productcovers/a0035e0f-aeaa-464d-b17e-d12e047d5c
## 2  https://qsbproduction.blob.core.windows.net/5841-productcovers/9b26aa14-04b5-4599-b76a-42b9ab0cbe
## 3  https://qsbproduction.blob.core.windows.net/6383-productcovers/098f67fb-af69-4ffa-814d-b9724a1187
## 4  https://qsbproduction.blob.core.windows.net/5545-productcovers/6d6b05d4-9e78-46e1-baa3-30636ee60d
## 5  https://qsbproduction.blob.core.windows.net/5145-productcovers/05e3a371-b448-4225-8d3e-9402a78004
## 6  https://qsbproduction.blob.core.windows.net/4322-productcovers/56cb6a96-0933-410d-be11-338b0318da
##     make        model               trim                   personal year
## 1 Toyota Land Cruiser                  G                            2017
## 2 Toyota Land Cruiser GXR- Grand Touring                  Al Bremai 2021
## 3 Nissan       Pickup           Standard Doha Meghna - Mawater City 2016
## 4 Toyota Land Cruiser                GXR                  Al Bremai 2021
## 5 Toyota Land Cruiser                GXR                  Al Bremai 2021
## 6 Toyota        Hilux           Standard                            2022
##   gear_type cylinder    mileage   price       time likes car
## 1 Automatic        6  22,000 Km 138,000 18/02/2024 10542  NA
## 2 Automatic        6       0 Km 275,000             2015  NA
## 3    Manual        4 225,000 Km  27,000             1099  NA
## 4 Automatic        8       0 Km 269,000             2596  NA
## 5 Automatic        8       0 Km 247,000             3978  NA
## 6    Manual        4       0 Km  87,000 11/12/2023  2264  NA
##                                                                                                car.hr
## 1               https://qatarsale.com/en/product/toyota_land_cruiser_g_white_suv_automatic_2017-127
## 2 https://qatarsale.com/en/product/toyota_land_cruiser_gxr_grand_touring_white_suv_automatic_2021-58
## 3                          https://qatarsale.com/en/product/nissan_pickup_2016_white_manual_suv-63
## 4              https://qatarsale.com/en/product/toyota_land_cruiser_gxr_white_suv_automatic_2021-55
## 5              https://qatarsale.com/en/product/toyota_land_cruiser_gxr_white_suv_automatic_2021-51
## 6                        https://qatarsale.com/en/product/toyota_hilux_white_pick_up_manual_2022-43
```

```
str(data)
```

```
## 'data.frame':    17748 obs. of  16 variables:
##  $ web.scraper.order    : chr  "1712250886-1" "1712250886-2" "1712250886-3" "1712250886-4" ...
##  $ web.scraper.start.url: chr  "https://qatarsale.com/en/products/cars_for_sale?page=500" "https://q
##  $ image.src            : chr  "https://qsbproduction.blob.core.windows.net/12756-productcovers/a003
##  $ make                 : chr  "Toyota" "Toyota" "Nissan" "Toyota" ...
##  $ model                : chr  "Land Cruiser" "Land Cruiser" "Pickup" "Land Cruiser" ...
##  $ trim                 : chr  "G" "GXR- Grand Touring" "Standard" "GXR" ...
##  $ personal             : chr  "" "Al Bremai" "Doha Meghna - Mawater City" "Al Bremai" ...
##  $ year                 : int  2017 2021 2016 2021 2021 2022 2021 2008 2021 2022 ...
##  $ gear_type            : chr  "Automatic" "Automatic" "Manual" "Automatic" ...
##  $ cylinder             : int  6 6 4 8 8 4 4 8 6 4 ...
##  $ mileage              : chr  "22,000 Km" "0 Km" "225,000 Km" "0 Km" ...
##  $ price                : chr  "138,000" "275,000" "27,000" "269,000" ...
##  $ time                 : chr  "18/02/2024" "" "" "" ...
##  $ likes                : int  10542 2015 1099 2596 3978 2264 7083 4299 5061 4733 ...
##  $ car                  : logi  NA NA NA NA NA NA ...
##  $ car.href             : chr  "https://qatarsale.com/en/product/toyota_land_cruiser_g_white_suv_aut
```

```r
# we have 16 columns listed above, some of which needs to be formatted and cleaned

# mileage column includes "," and "Km", so lets remove "Km" and comma
# lets also convert it to numeric
data$mileage <- (gsub(" Km", "", data$mileage))
data$mileage <- (gsub(",", "", data$mileage))
data$mileage <- as.numeric(data$mileage)
# price column includes ",",lets remove "," and concert to numeric
data$price <- (gsub(",", "", data$price))
data$price <- as.numeric(data$price)
# convert cylinder to a number
data$cylinder <- as.integer(data$cylinder)

# Time column includes either sold date or last posted time
# lets split it to date_sold and posted_time columns
data$date_sold <- NA
# Check if time column has "/" in it and transform the value to a date
data$date_sold <- ifelse(grepl("/", data$time), format(as.POSIXct(data$time, format = "%d/%m/%Y"), "%d-%
# Create a new column called posted_time
data$posted_time <- NA
# Extract the number of hours or minutes from the time column and convert to hours and decimals
data$posted_time <- ifelse(grepl("hour", data$time), as.numeric(gsub(" hours ago", "", data$time)), "")
data$posted_time <- ifelse(grepl("minute", data$time), as.numeric(gsub(" minutes ago", "", data$time))/6
# With regards to the mileage column, later When we split data for training, testing
# and validation we don't want to remove records if they it existing in train and
# don't exists in test or validate dataset, hence we have create tiers every 10,000
# to reduce chance of committing the record.
data <- data %>%
  mutate(mileage_range = case_when(
    mileage == 0 ~ 0,
    mileage > 0 ~ ceiling(mileage / 10000) * 10000
  ))
str(data)
```

```
## 'data.frame':    17748 obs. of  19 variables:
##  $ web.scraper.order    : chr  "1712250886-1" "1712250886-2" "1712250886-3" "1712250886-4" ...
##  $ web.scraper.start.url: chr  "https://qatarsale.com/en/products/cars_for_sale?page=500" "https://qa
##  $ image.src            : chr  "https://qsbproduction.blob.core.windows.net/12756-productcovers/a0035
##  $ make                 : chr  "Toyota" "Toyota" "Nissan" "Toyota" ...
##  $ model                : chr  "Land Cruiser" "Land Cruiser" "Pickup" "Land Cruiser" ...
##  $ trim                 : chr  "G" "GXR- Grand Touring" "Standard" "GXR" ...
##  $ personal             : chr  "" "Al Bremai" "Doha Meghna - Mawater City" "Al Bremai" ...
##  $ year                 : int  2017 2021 2016 2021 2021 2022 2021 2008 2021 2022 ...
##  $ gear_type            : chr  "Automatic" "Automatic" "Manual" "Automatic" ...
##  $ cylinder             : int  6 6 4 8 8 4 4 8 6 4 ...
##  $ mileage              : num  22000 0 225000 0 0 0 0 109000 0 0 ...
##  $ price                : num  138000 275000 27000 269000 247000 87000 102000 35000 176000 74000 ...
##  $ time                 : chr  "18/02/2024" "" "" "" ...
##  $ likes                : int  10542 2015 1099 2596 3978 2264 7083 4299 5061 4733 ...
##  $ car                  : logi  NA NA NA NA NA NA ...
##  $ car.href             : chr  "https://qatarsale.com/en/product/toyota_land_cruiser_g_white_suv_auto
##  $ date_sold            : chr  "18-Feb-2024" "" "" "" ...
##  $ posted_time          : chr  "" "" "" "" ...
```

```
##  $ mileage_range        : num  30000 0 230000 0 0 0 0 110000 0 0 ...
```

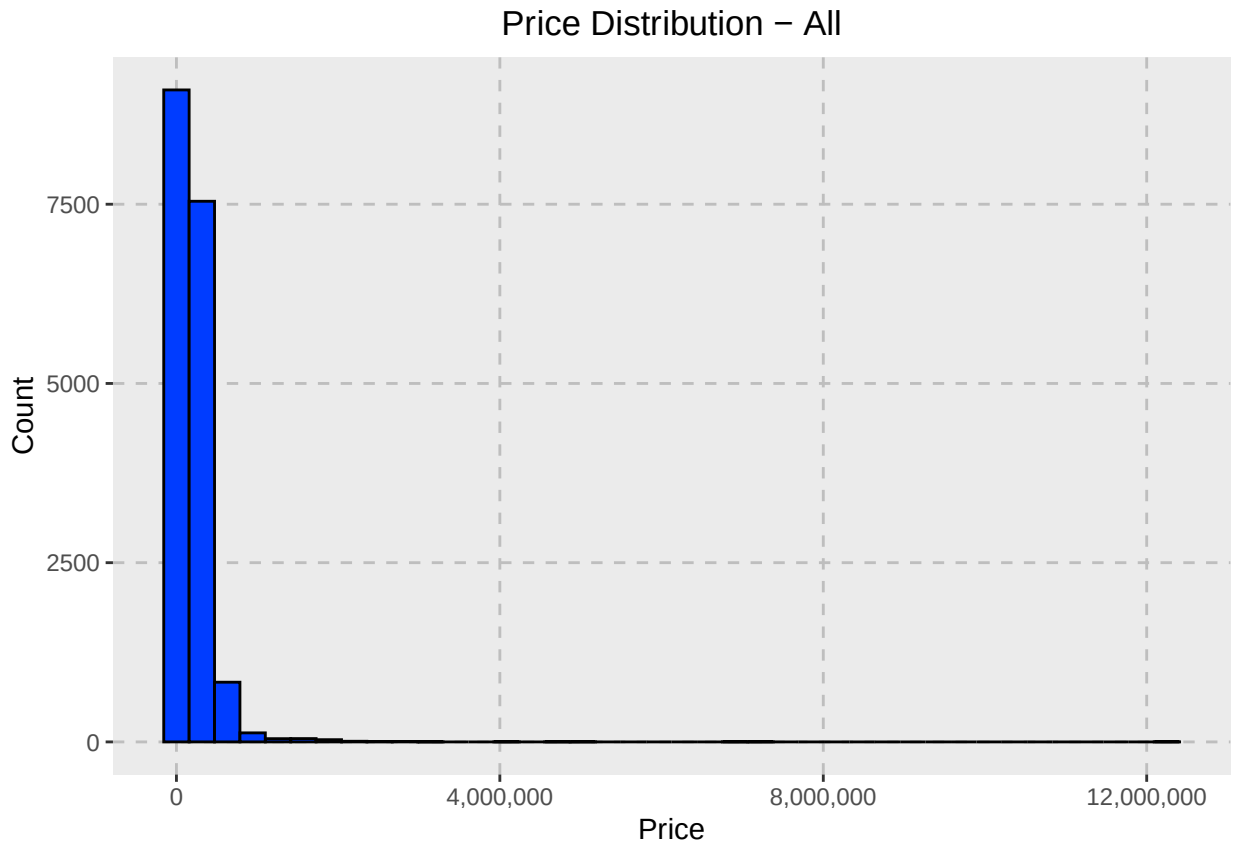
4.1.4 Filter Data

The dataset have been filtered having car price of less than 100,000 USD (365,000 QAR) and the car age to be at least 10 years old (where model year >= 2014).

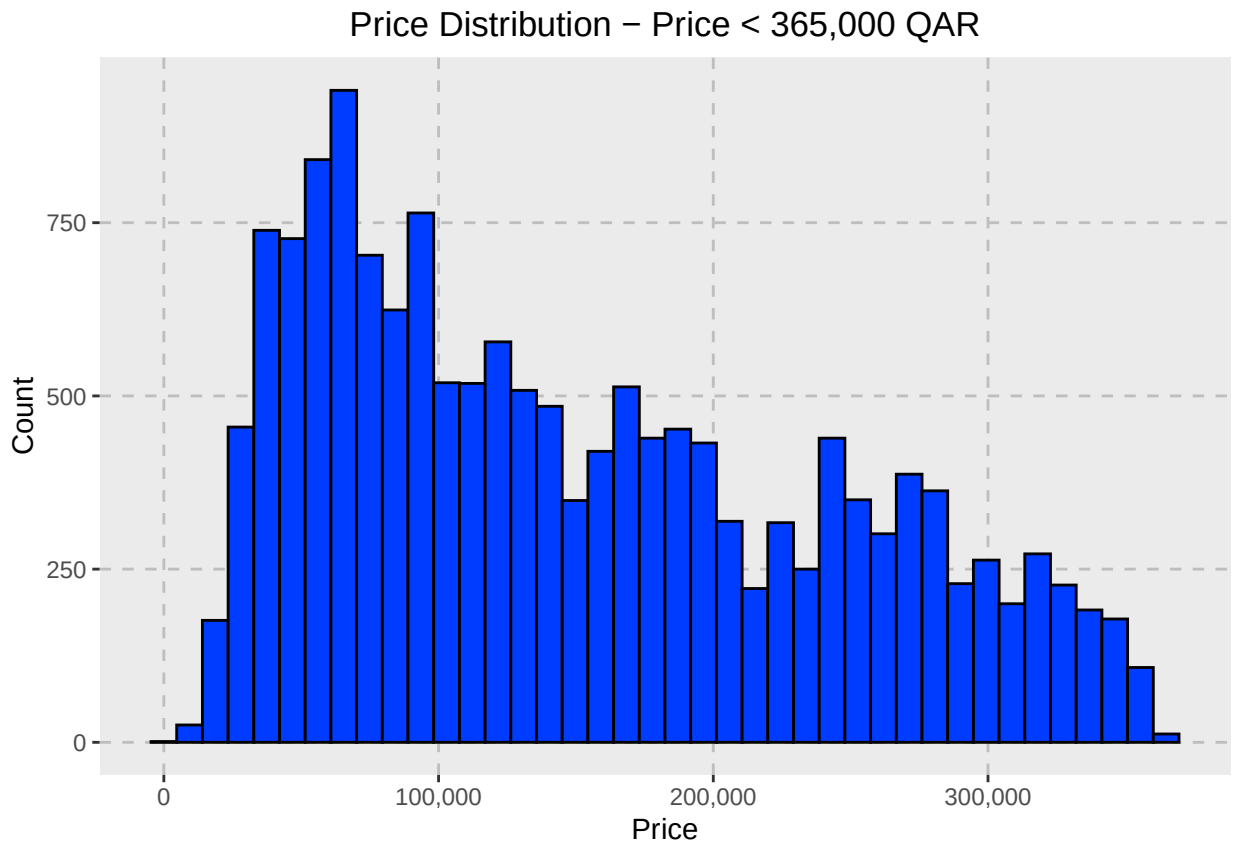
As a result the number of records have reduced to 13,500+ from 17,500+.

As you can see the number of car listing dramatically reduces after 100,000 USD or 365,000 QAR.

```
# Number of car on y axis are plotted against their price value on x axis, as you can see the listing d
# The number
data %>%
  filter(price > 0) %>%
  ggplot(aes(x = price)) +
  geom_histogram(bins = 40, color = "black", fill = "#003CFF") +
  labs(title = "Price Distribution - All", x = "Price", y = "Count") +
  scale_x_continuous(labels = comma_format())+
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
panel.grid.minor = element_blank() # Removing minor grid lines
)
```



```r
# I have filtered the price less than 365,000. I am more familiar with car of prices with this range.
data %>%
  filter(price < 365000) %>%
  ggplot(aes(x = price)) +
  geom_histogram(bins = 40, color = "black", fill = "#003CFF") +
  labs(title = "Price Distribution - Price < 365,000 QAR", x = "Price", y = "Count") +
  scale_x_continuous(labels = comma_format())+
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
panel.grid.minor = element_blank() # Removing minor grid lines
)
```




```r
# After filtering the number of records have reduced as per below.
data <- data[data$price < 365000, ]
nrow(data)
```

```
## [1] 15844
```

```r
# As you can see below if the number of car listing of older cars is much less,
# so i have limited it to the last 11 years.
top_years <- head(sort(table(data$year), decreasing = TRUE), 10)
top_years_table <- data.frame(Make = names(top_years),Count = as.numeric(top_years))
top_years_table
```

```
##    Make Count
## 1  2023  2730
## 2  2022  1706
## 3  2024  1357
## 4  2020  1238
## 5  2021  1146
## 6  2016  1044
## 7  2019   996
## 8  2015   951
## 9  2018   844
## 10 2017   838
```

```r
# After filtering the number of records are as per below
data <- data[data$year > 2013, ]
nrow(data)
```

```
## [1] 13567
```

```r
# we checked that the price has 7 nulls, so lets remove it
```

```r
summary(data$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     159   89000  155000  165768  239000  364999       7
```

```r
data <- data[!is.na(data$price) & data$price != "", ]
nrow(data)
```

```
## [1] 13560
```

```r
# Now we are checking that all data is tidy and filtered
str(data)
```

```
## 'data.frame':    13560 obs. of  19 variables:
##  $ web.scraper.order    : chr  "1712250886-1" "1712250886-2" "1712250886-3" "1712250886-4" ...
##  $ web.scraper.start.url: chr  "https://qatarsale.com/en/products/cars_for_sale?page=500" "https://qa
##  $ image.src            : chr  "https://qsbproduction.blob.core.windows.net/12756-productcovers/a003
##  $ make                 : chr  "Toyota" "Toyota" "Nissan" "Toyota" ...
##  $ model                : chr  "Land Cruiser" "Land Cruiser" "Pickup" "Land Cruiser" ...
##  $ trim                 : chr  "G" "GXR- Grand Touring" "Standard" "GXR" ...
##  $ personal             : chr  "" "Al Bremai" "Doha Meghna - Mawater City" "Al Bremai" ...
##  $ year                 : int  2017 2021 2016 2021 2021 2022 2021 2021 2022 2018 ...
##  $ gear_type            : chr  "Automatic" "Automatic" "Manual" "Automatic" ...
##  $ cylinder             : int  6 6 4 8 8 4 4 6 4 6 ...
##  $ mileage              : num  22000 0 225000 0 0 0 0 0 0 34000 ...
##  $ price                : num  138000 275000 27000 269000 247000 87000 102000 176000 74000 280000 ..
##  $ time                 : chr  "18/02/2024" "" "" "" ...
##  $ likes                : int  10542 2015 1099 2596 3978 2264 7083 5061 4733 5174 ...
##  $ car                  : logi  NA NA NA NA NA NA ...
##  $ car.href             : chr  "https://qatarsale.com/en/product/toyota_land_cruiser_g_white_suv_aut
##  $ date_sold            : chr  "18-Feb-2024" "" "" "" ...
##  $ posted_time          : chr  "" "" "" "" ...
##  $ mileage_range        : num  30000 0 230000 0 0 0 0 0 0 40000 ...
```

4.2 Data Visualization

In this section, we explore the different dataset parameters and visualize it.

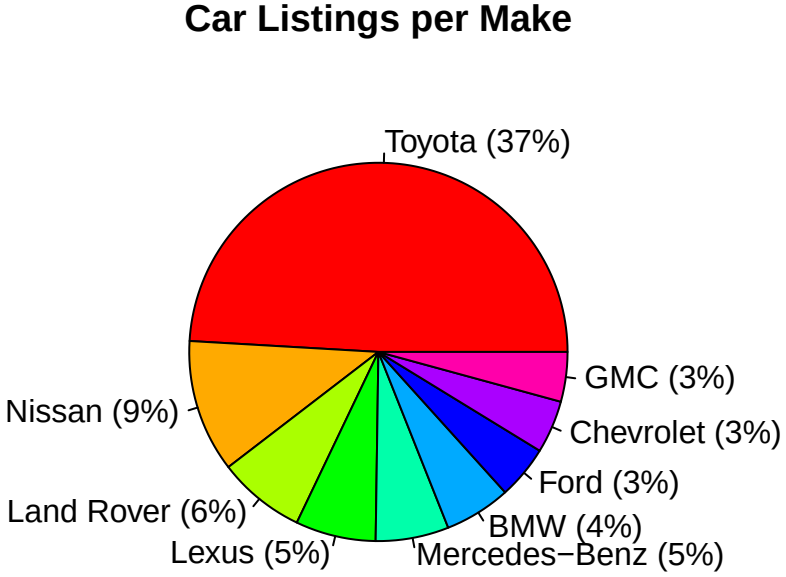
Before we start we just want to recap on some of the important columns.

Make column is like "Mercedes", "Audi", etc. Model column is like "200", "A3", etc, which is a child of Make, Trim column can vary like "standard", etc. which is also a child of model. cylinder column has how many cylinder the car has gear_type has is either automatic, manual, tiptronic, etc.

This pie chart shows the number of car listings per make As you can see Toyota is the most popular make or brand in Qatar making up more than 1/3. In the next few charts we will focus on Toyota and Nissan model distribution over years

Note that we have filtered out any make that is less than 3% that's why the Pie shows 37% as larger than it should.

```
car_counts <- table(data$make)
car_counts <- car_counts[order(car_counts, decreasing = TRUE)]
percent <- car_counts/sum(car_counts)*100
car_counts <- car_counts[percent > 3]
pie(car_counts, labels = paste0(names(car_counts), " (", round(percent[percent > 1]), "%)"), main = "Ca
```



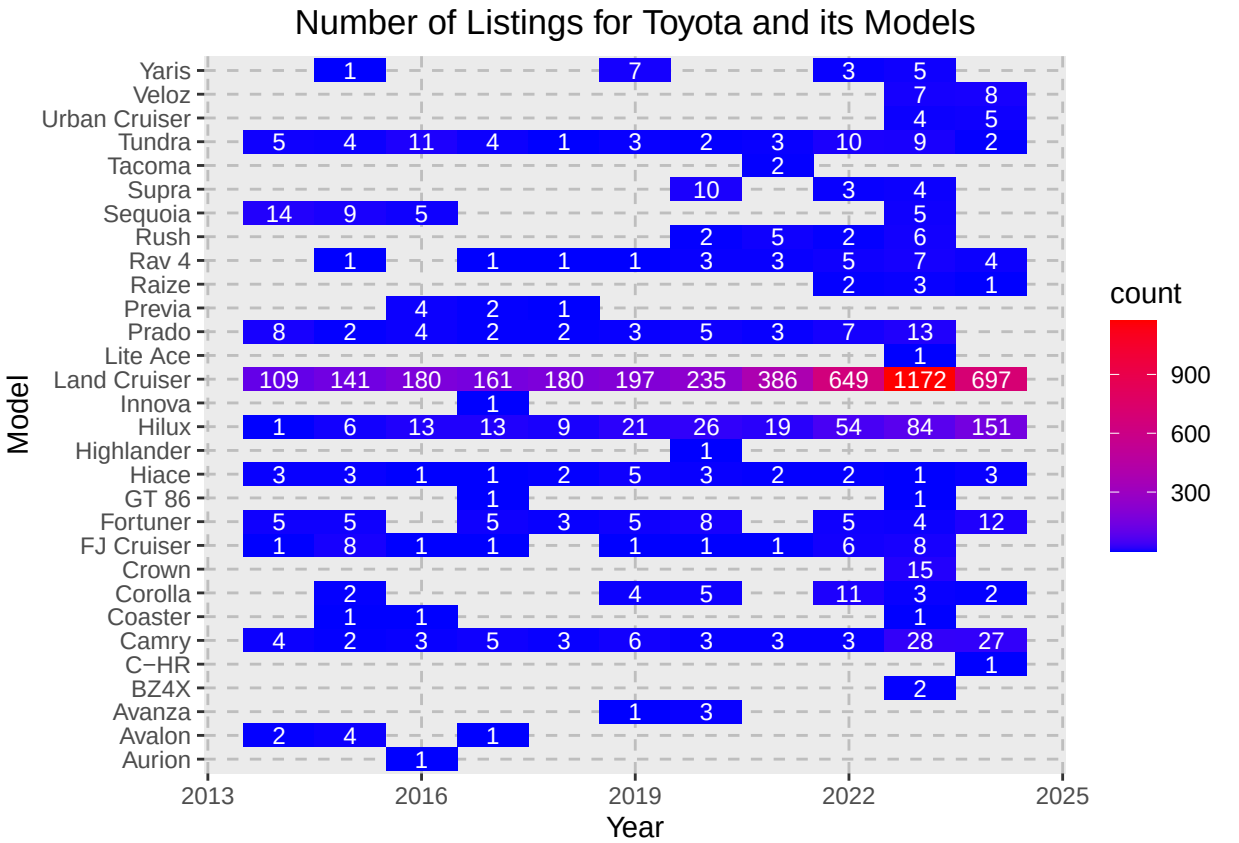
The below function returns the make based on the rank number this will be used in the next few charts.

```
get_make_rank <- function(rank_number) {
    car_counts <- table(data$make)
    car_counts <- car_counts[order(car_counts, decreasing = TRUE)]
```

```
    make_rank <- names(car_counts)[rank_number]
    return(make_rank)
}
```

This chart shows Toyota models distribution over years. The Y axis is the Toyota models and x Axis is the years, the numbers in each block indicate the number of car listings. As you can see the most popular listing is Land cruisers model for 2023

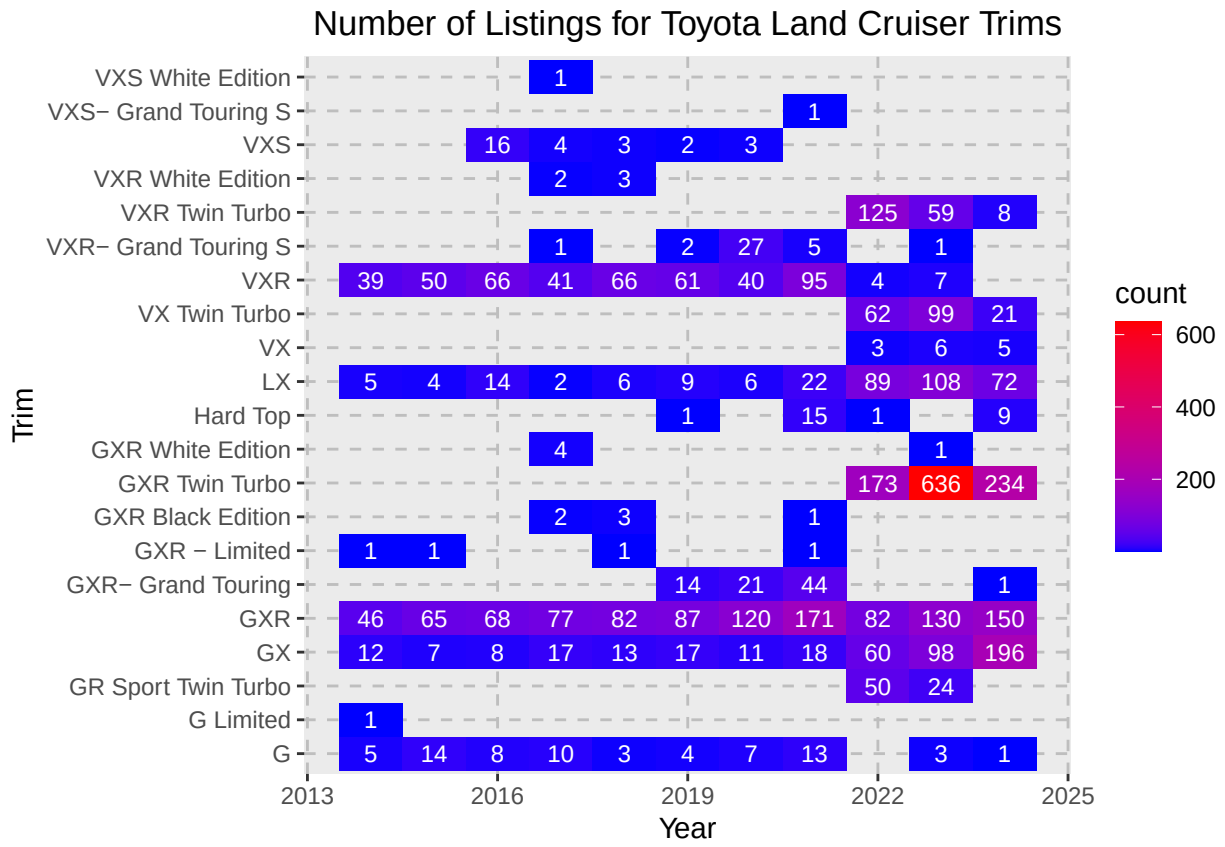
```
make_rank_1 <- get_make_rank(1)  # Returns "Toyota"
data %>%
    filter(make == make_rank_1) %>%
    group_by(make, model, year) %>%
    summarise(count = n()) %>%
    ggplot(aes(x = year, y = model, fill = count)) +
    geom_tile() +
    labs(title = paste("Number of Listings for", make_rank_1, "and its Models"), x = "Year", y = "Model
    theme(
        plot.title = element_text(hjust = 0.5), # center title
        panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add dashed lines
        panel.grid.minor = element_blank() # Removing minor grid lines
    ) +
    geom_text(aes(label = count), color = "white", size = 3, position = position_dodge(width = 1)) +
    scale_fill_gradient(low = "blue", high = "red") # Add color gradient
```



Since Land Cruiser model is the most popular in Toyota, The Y axis is the Toyota- Land Cruiser trims and

x axis is the years, the numbers in each block indicate the number of car listings. The below chart shows that GXR Twin Turbo is the most popular trim in 2023.

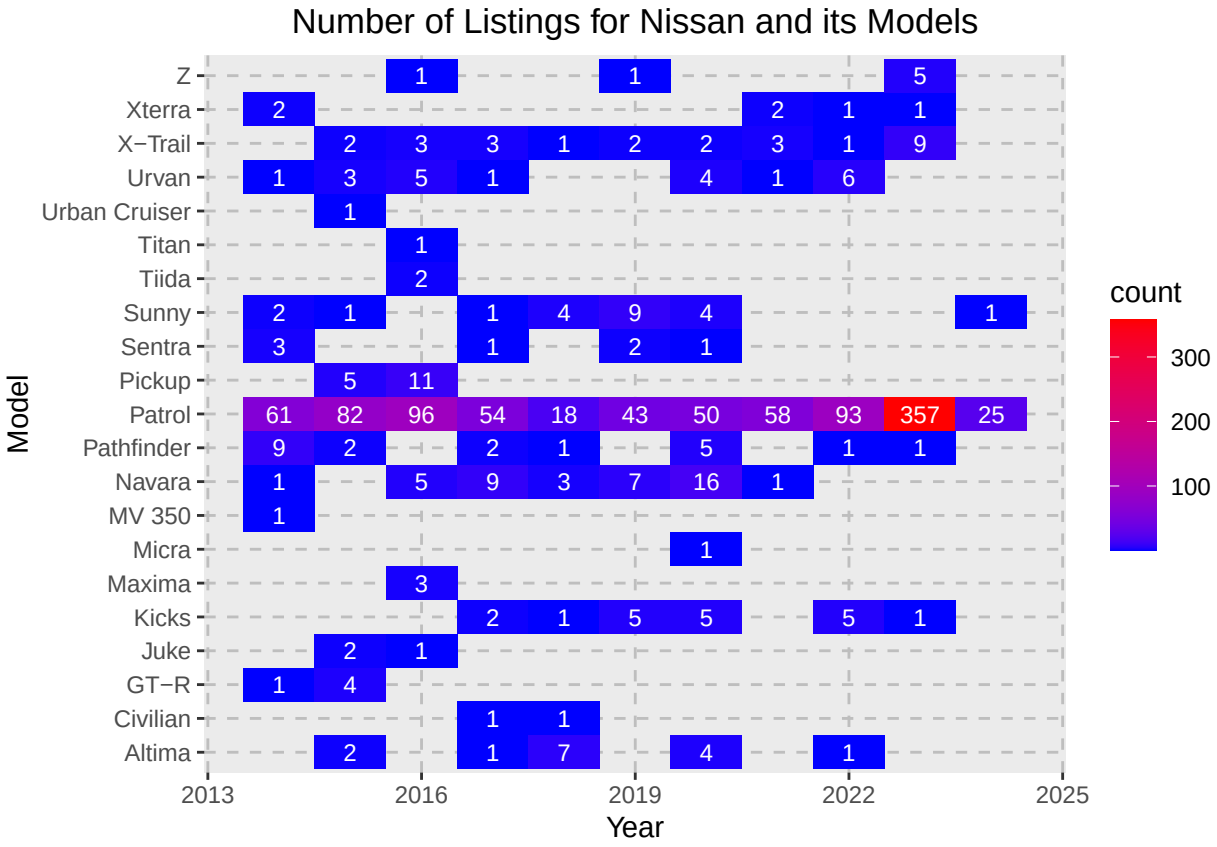
```
make_rank_1 <- get_make_rank(1)  # Returns "Toyota"
data %>%
    filter(make == make_rank_1 & model == "Land Cruiser") %>%
    group_by(make, model, year, trim) %>%
    summarise(count = n()) %>%
    ggplot(aes(x = year, y = trim, fill = count)) +
    geom_tile() +
    labs(title = paste("Number of Listings for", make_rank_1, "Land Cruiser Trims"), x = "Year", y =
    theme(
        plot.title = element_text(hjust = 0.5), # center title
        panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add dashed lines
        panel.grid.minor = element_blank() # Removing minor grid lines
    ) +
    geom_text(aes(label = count), color = "white", size = 3, position = position_dodge(width = 1)) +
    scale_fill_gradient(low = "blue", high = "red") # Add color gradient
```



This chart shows Nissan models distribution over years. The Y axis is the Nissan models and x Axis is the years, the numbers in each block indicate the number of car listings. As you can see the most popular listing is Patrol model for 2023

```
make_rank_2 <- get_make_rank(2)
data %>%
    filter(make == make_rank_2) %>%
```

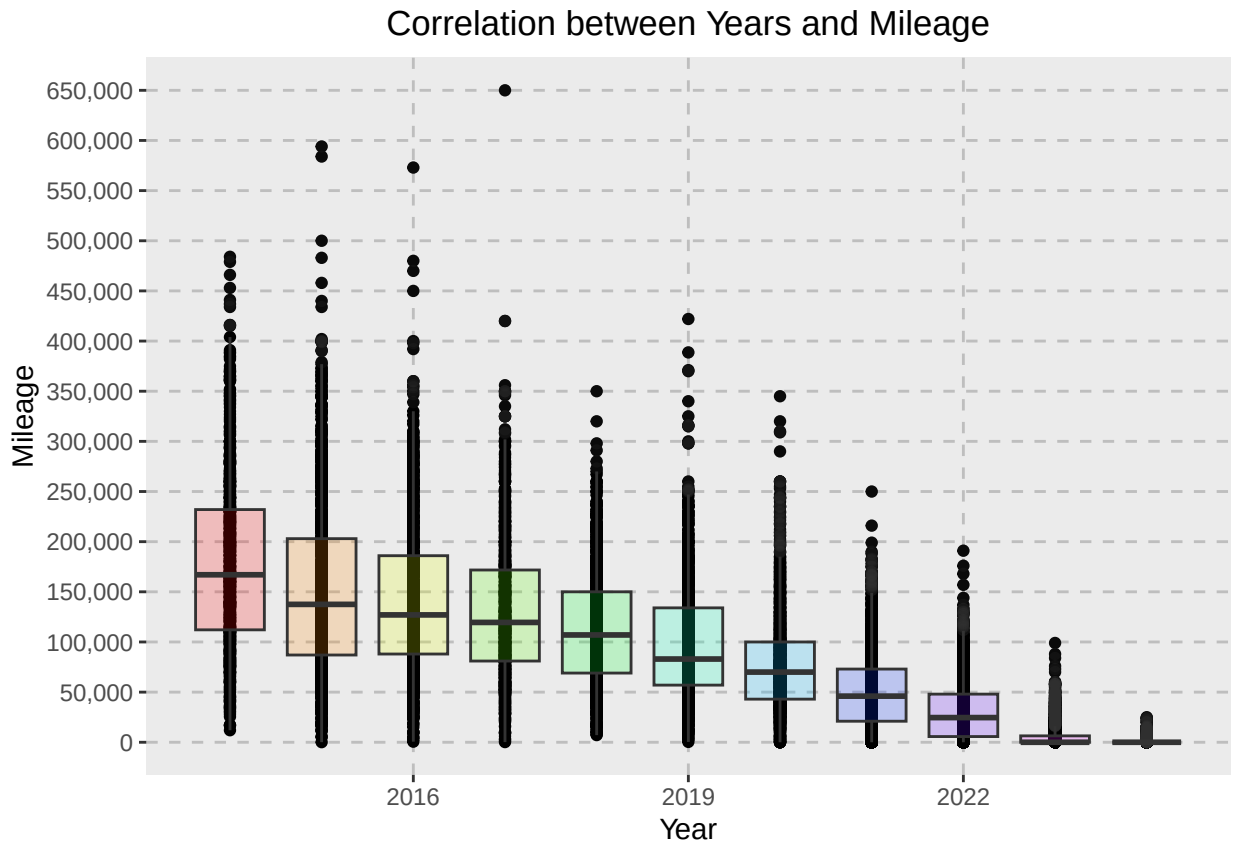
```
    group_by(make, model, year) %>%
    summarise(count = n()) %>%
    ggplot(aes(x = year, y = model, fill = count)) +
    geom_tile() +
    labs(title = paste("Number of Listings for", make_rank_2, "and its Models"), x = "Year", y = "Model
    theme(
        plot.title = element_text(hjust = 0.5), # center title
        panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add dashed lines
        panel.grid.minor = element_blank() # Removing minor grid lines
    ) +
    geom_text(aes(label = count), color = "white", size = 3, position = position_dodge(width = 1)) +
    scale_fill_gradient(low = "blue", high = "red") # Add color gradient
```



The below chart to correlate years and mileage, for example median of 2014 is around 175,000 which means around car usually run for 17,500 km average per year.

```
data %>%
    filter(mileage <= 650000) %>% # there was 1 anomoly manuall removed.
    ggplot(aes(x = year, y = mileage)) +
    geom_point() +
    labs(title = "Correlation between Years and Mileage", x = "Year", y = "Mileage") +
    theme(
        plot.title = element_text(hjust = 0.5), # center title
        panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add dashed lines
        panel.grid.minor = element_blank() # Removing minor grid lines
    ) +
```

```
    geom_boxplot(aes(group=year), fill = rainbow(length(unique(data$year))), alpha = 0.2) + # Add boxpl
    scale_y_continuous(breaks = seq(0, max(data$mileage), 50000), labels = comma_format()) # Set y-axis
```



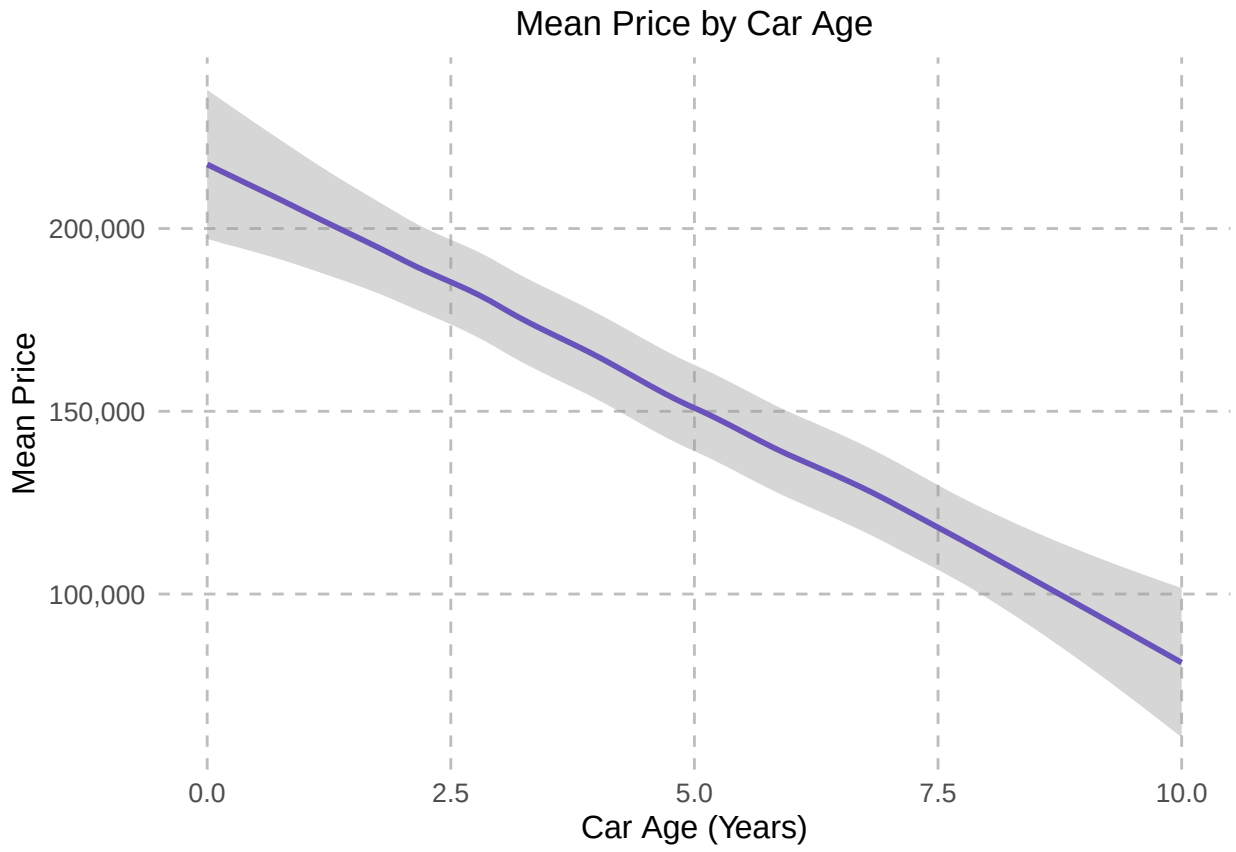
This plot shows the relationship between the car ages or years and their average price. The line was created utilizing the LOESS (Locally Estimated Scatterplot Smoothing) method. This provides an approximate relationship between a car's age and the mean price. The older the car the lower the price in this timeframe. From the chart we can see the mean price for all cars with 0 age is around 220,000 QAR (~ 63,000 USD) which depreciates to around 60,000 QAR (~ 16,000 USD) after 10 years.

```
# Calculate car age based on the year column
data_v <- data %>%
    mutate(car_age = 2024 - year)

mean_price_age <- data_v %>%
    group_by(car_age) %>%
    summarise(mean_price = mean(price, na.rm = TRUE))

# plotting mean_price_age
ggplot(mean_price_age, aes(x = car_age, y = mean_price)) +
        #geom_point(color = "#c4553a", size = 3) + # setting color and size of points
        geom_smooth(method = "loess", color = "#6952b9", method.args = list(span = 0.15, degree = 1)) +
        labs(x = "Car Age (Years)", y = "Mean Price", title = "Mean Price by Car Age") +
        theme_minimal() +
        theme(
                plot.title = element_text(hjust = 0.5), # set title
                axis.text = element_text(size = 10), # set axis text size
```

```
            axis.title = element_text(size = 12), # set axis title size
            panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add dashed grid
            panel.grid.minor = element_blank() # removing minor grid lines
        )+ # set custom fill colors
    scale_y_continuous(labels = comma_format())
```

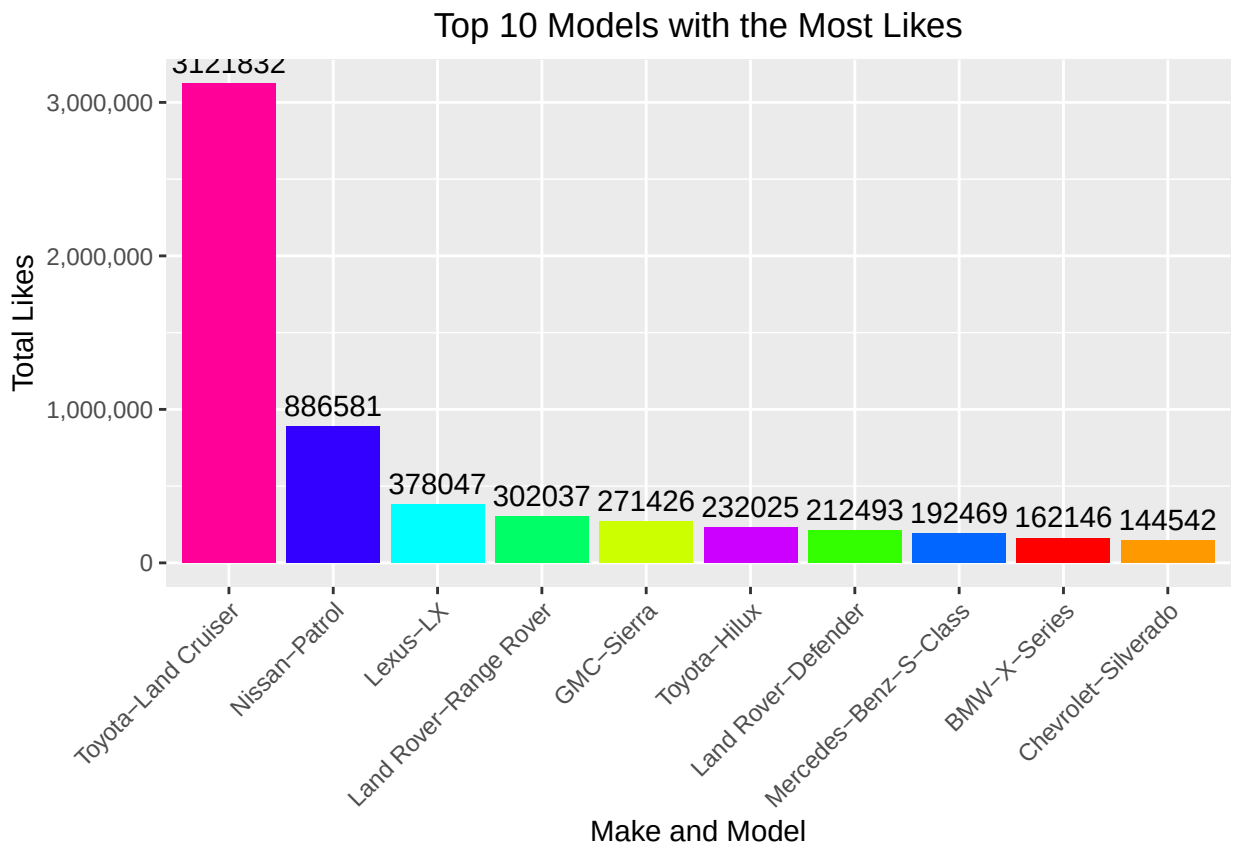


The below charts shows the top 10 make-models are the most liked, as this site allows users to like a listing.

```
# Get the top 10 models based on the likes column
top_models <- head(data %>% group_by(make, model) %>% summarise(total_likes = sum(likes)) %>% arrange(d

# Create a table with the top 10 models, their make, total likes, and number of likes
top_models_table <- data.frame(Make_Model = paste(top_models$make, top_models$model, sep = "-"), Total_

# Create a bar chart to visualize the total likes and number of likes for the top 10 models
ggplot(top_models_table, aes(x = reorder(Make_Model, -Total_Likes), y = Total_Likes, fill = Make_Model)
    geom_bar(stat = "identity") +
    labs(title = "Top 10 Models with the Most Likes", x = "Make and Model", y = "Total Likes") +
    theme(
        plot.title = element_text(hjust = 0.5), # center title
        axis.text.x = element_text(angle = 45, hjust = 1), # rotate x-axis labels
        legend.position = "none" # remove legend
    ) +
    scale_fill_manual(values = rainbow(nrow(top_models_table))) + # set custom fill colors
    scale_y_continuous(labels = comma_format()) + # format y-axis labels with comma every 3 digits
    geom_text(aes(label = Number_of_Likes), vjust = -0.5) # add labels for number of likes above each b
```



4.3 Feature Engineering

In this section, we create new columns and tables as new features that will be used to train our models (mostly for linear regression) and will be also used to predict the price outcome.

The below columns were created/updated based on a feedback loop during training to improve model accuracy.

4.3.1 Column Update - Model

```
# We are updating model column to be linked its parent column "Make", this will be
# used with the model bias effect
data$model <- paste(data$make, data$model, sep = "-")
head(data$model)
```

```
## [1] "Toyota-Land Cruiser" "Toyota-Land Cruiser" "Nissan-Pickup"
## [4] "Toyota-Land Cruiser" "Toyota-Land Cruiser" "Toyota-Hilux"
```

4.3.2 Column Update - Trim

```r
# We are updating "trim" to be linked to its parent column "model".
# We also have included "cylinder" and "gear_type"
# this will be used with the "trim" bias effect
data$trim <- paste(data$make,  data$model, data$trim, data$cylinder, data$gear_type,sep = "-")
head(data$trim)
```

```
## [1] "Toyota-Toyota-Land Cruiser-G-6-Automatic"
## [2] "Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic"
## [3] "Nissan-Nissan-Pickup-Standard-4-Manual"
## [4] "Toyota-Toyota-Land Cruiser-GXR-8-Automatic"
## [5] "Toyota-Toyota-Land Cruiser-GXR-8-Automatic"
## [6] "Toyota-Toyota-Hilux-Standard-4-Manual"
```

4.3.3 Column Creation - Avg Price, Year, Mileage

```r
# We are creating new columns by calculating per trim the average "year" and "price",
# this will be used for the price prediction
data <- data %>% group_by(trim) %>% mutate(avg_ymmt = mean(year))
data <- data %>% group_by(trim) %>% mutate(price_pmmt = mean(price))
# We are creating new columns by calculating per year the average mileage and price
# this will be used for the price prediction
data <- data %>% group_by(year) %>% mutate(avg_mileage = mean(mileage)+1) # to avoid division by zero
data <- data %>% group_by(year) %>% mutate(price_pmmty = mean(price))
head(data %>% select(make, model, year, price,avg_ymmt,price_pmmt,avg_mileage,price_pmmty))
```

```
## # A tibble: 6 x 8
## # Groups:   year [4]
##   make   model           year  price avg_ymmt price_pmmt avg_mileage price_pmmty
##   <chr>  <chr>          <int>  <dbl>    <dbl>      <dbl>       <dbl>       <dbl>
## 1 Toyota Toyota-Land C~  2017 138000    2018.    131385.     131321.     124860.
## 2 Toyota Toyota-Land C~  2021 275000    2021.    243548.      52391.     180289.
## 3 Nissan Nissan-Pickup   2016  27000    2016.     28000      142337.     108704.
## 4 Toyota Toyota-Land C~  2021 269000    2018.    164700.      52391.     180289.
## 5 Toyota Toyota-Land C~  2021 247000    2018.    164700.      52391.     180289.
## 6 Toyota Toyota-Hilux    2022  87000    2021.     84105.      31180.     201070.
```

4.3.4 Table Creation - Average Year per trim

```r
# We are creating separate table for average "year" per "trim".
# this will be used for the price prediction
data_avg_ymmt <- data %>%
    group_by(trim) %>%
    mutate(avg_ymmt = mean(year))%>%
    select(trim, avg_ymmt) %>%
    unique()
head(data_avg_ymmt)
```

```
## # A tibble: 6 x 2
## # Groups:   trim [6]
```

```
##   trim                                                      avg_ymmt
##   <chr>                                                        <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic                     2018.
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic    2021.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual                       2016.
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic                   2018.
## 5 Toyota-Toyota-Hilux-Standard-4-Manual                        2021.
## 6 Toyota-Toyota-Camry-GLE-4-Automatic                          2022.
```

4.3.5 Column Update - Year

```
# We are updating "year" column to include "trim"
# This will be used with the year bias effect
data$year <- paste(data$trim, data$year, sep = "-")
head(data$year)
```

```
## [1] "Toyota-Toyota-Land Cruiser-G-6-Automatic-2017"
## [2] "Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic-2021"
## [3] "Nissan-Nissan-Pickup-Standard-4-Manual-2016"
## [4] "Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021"
## [5] "Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021"
## [6] "Toyota-Toyota-Hilux-Standard-4-Manual-2022"
```

4.3.6 Column Creation - Average Mileage per Year

```
# We are creating a new column by calculating per "year" the average "mileage_range"
data <- data %>% group_by(year) %>% mutate(avg_mileage_range = mean(mileage_range))
head(data %>% select(make, model, year,avg_mileage_range))
```

```
## # A tibble: 6 x 4
## # Groups:   year [5]
##   make   model               year                                avg_mileage_range
##   <chr>  <chr>               <chr>                                           <dbl>
## 1 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-G-6-A~           180000
## 2 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR- ~            18696.
## 3 Nissan Nissan-Pickup       Nissan-Nissan-Pickup-Standard-4-~           261818.
## 4 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR-8~            71806.
## 5 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR-8~            71806.
## 6 Toyota Toyota-Hilux        Toyota-Toyota-Hilux-Standard-4-M~            14615.
```

4.3.7 Table Creation - Average Mileage per Year

```
# We are creating separate table for "average_mileage_range" per "year".
# this will be used for the price prediction
data_avg_mileage_range <- data %>% group_by(year) %>%
    mutate(avg_mileage_range = mean(mileage_range))%>%
    select(year, avg_mileage_range) %>%
    unique()
head(data_avg_mileage_range)
```

```
## # A tibble: 6 x 2
## # Groups:   year [6]
##   year                                                  avg_mileage_range
##   <chr>                                                             <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic-2017                    180000
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic-2~      18696.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual-2016                     261818.
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021                  71806.
## 5 Toyota-Toyota-Hilux-Standard-4-Manual-2022                       14615.
## 6 Toyota-Toyota-Camry-GLE-4-Automatic-2021                             0
```

4.3.8 Column Update - Mileage_range

```
# Mileage_range update
# We are updating mileage_range with trim, as this combination will be needed later
# for the mileage effect (linear regression - bias)
data$mileage_range <- paste( data$trim, data$mileage_range, sep = "-")
head(data %>% select(make, model, year,mileage_range))
```

```
## # A tibble: 6 x 4
## # Groups:   year [5]
##   make   model               year                                  mileage_range
##   <chr>  <chr>               <chr>                                 <chr>
## 1 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-G-6-Autom~ Toyota-Toyot~
## 2 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR- Gran~ Toyota-Toyot~
## 3 Nissan Nissan-Pickup       Nissan-Nissan-Pickup-Standard-4-Manu~ Nissan-Nissa~
## 4 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR-8-Aut~ Toyota-Toyot~
## 5 Toyota Toyota-Land Cruiser Toyota-Toyota-Land Cruiser-GXR-8-Aut~ Toyota-Toyot~
## 6 Toyota Toyota-Hilux        Toyota-Toyota-Hilux-Standard-4-Manua~ Toyota-Toyot~
```

4.3.9 Table Creation - Average price per trim

```
# Create Table Average price per trim
# We are creating separate table for average "price" per "trim".
# this will be used for the price prediction
data_price_pmmt <- data %>%
    group_by(trim) %>%
    mutate(price_pmmt = mean(price))  %>%
    select(trim, price_pmmt) %>%
    unique()
head(data_price_pmmt)
```

```
## # A tibble: 6 x 2
## # Groups:   trim [6]
##   trim                                                  price_pmmt
##   <chr>                                                      <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic                  131385.
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic 243548.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual                     28000
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic                164700.
```

```
## 5 Toyota-Toyota-Hilux-Standard-4-Manual                          84105.
## 6 Toyota-Toyota-Camry-GLE-4-Automatic                            91369.
```

4.3.10 Table Creation - Average price per year

```
# Create Table Average price per year
# We are creating separate table for average "price" per "year".
# this will be used for the price prediction
data_price_pmmty  <- data %>%
    group_by(year) %>%
    mutate(price_pmmty = mean(price)) %>%
    select(year, price_pmmty) %>%
    unique()
head(data_price_pmmty)
```

```
## # A tibble: 6 x 2
## # Groups:   year [6]
##   year                                                          price_pmmty
##   <chr>                                                               <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic-2017                      122200
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic-2021     255957.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual-2016                         30000
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021                    220194.
## 5 Toyota-Toyota-Hilux-Standard-4-Manual-2022                         100000
## 6 Toyota-Toyota-Camry-GLE-4-Automatic-2021                           102000
```

4.3.11 Table Creation - Average Mileage per year

```
# Create Table Average Mileage per year
# We are creating separate table for average "mileage" per "year".
# this will be used for the price prediction
data_avg_mileage <- data %>%
    group_by(year) %>%
    mutate(avg_mileage = mean(mileage)+1)%>%
    select(year, avg_mileage) %>%
    unique()
head(data_avg_mileage)
```

```
## # A tibble: 6 x 2
## # Groups:   year [6]
##   year                                                          avg_mileage
##   <chr>                                                               <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic-2017                      177001
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic-2021      17523.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual-2016                        259456.
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021                     68523.
## 5 Toyota-Toyota-Hilux-Standard-4-Manual-2022                          11487.
## 6 Toyota-Toyota-Camry-GLE-4-Automatic-2021                                1
```

4.4 Data Splitting

In this section, we split our data into train_set, test_set and validate_set for training, testing and validation respectively. We have split training to be 90 % of data (train_test), and the remaining 10 % for validation. (validation_set) We have split training data to be 80% training (train_set) and 20% for testing (test_set). We have also removed records from test and validation set to be included in train set where certain columns are missing in test and validation.

```
# We have already loaded the caret package to be able split the data use "createDataPartition"
# Set the random seed in order to be able to repeat the results.
set.seed(123)

validate_index <- createDataPartition(y = data$price, times = 1, p = 0.1, list = FALSE)
train_test <- data[-validate_index,]
temp <- data[validate_index,]

validate_set <- temp %>%
  semi_join(train_test, by = "make") %>%
  semi_join(train_test, by = "model") %>%
  semi_join(train_test, by = "trim") %>%
  semi_join(train_test, by = "year") %>%
  semi_join(train_test, by = "gear_type") %>%
  semi_join(train_test, by = "cylinder") %>%
  semi_join(train_test, by = "mileage_range")%>%
  semi_join(train_test, by = "personal")  # optional
# Add rows removed from validate_set test set back into train_test
removed <- anti_join(temp, validate_set)
train_test <- rbind(train_test, removed)

# Split the train_set data 1 time into 80% for training and 20% for testing.
test_index <- createDataPartition(y = train_test$price, times = 1, p = 0.2, list = FALSE)
train_set <- train_test[-test_index,]
temp <- train_test[test_index,]

test_set <- temp %>%
  semi_join(train_set, by = "make") %>%
  semi_join(train_set, by = "model") %>%
  semi_join(train_set, by = "trim") %>%
  semi_join(train_set, by = "year") %>%
  semi_join(train_set, by = "gear_type") %>%
  semi_join(train_set, by = "cylinder") %>%
  semi_join(train_set, by = "mileage_range") %>%
  semi_join(train_set, by = "personal")  # optional

# Add rows removed from test_set test set back into train_set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)

# remove rows with missing values in the price column
# show last row in test_set
#test_set[nrow(test_set),]
test_set <- test_set[!is.na(test_set$price) & test_set$price != "", ]
#test_set$price
#train_set[nrow(train_set),]
```

```r
train_set <- train_set[!is.na(train_set$price) & train_set$price != "", ]
#train_set$price
#validate_set[nrow(validate_set),]
validate_set <- validate_set[!is.na(validate_set$price) & validate_set$price != "", ]
#validate_set$price
nrow(train_set)
```

```
## [1] 10923
```

```r
nrow(test_set)
```

```
## [1] 1669
```

```r
nrow(validate_set)
```

```
## [1] 968
```

4.5 Model Evaluation

Here we define the function for Root Mean Squared Error used for evaluation model accuracy.

```r
# Here we define the function for Root Mean Squared Error.
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

5 Method 01 - Linear Regression

5.1 Model Training

My first method used is the Linear regression. I have used the bias effects of Make, Model, Trim where Trim includes Gear_type and Cylinder.

As explained in the Feature Engineering section, for Make, Model and Trim, they were updated and linked with each other to archive more meaningful bias and higher accuracy or lower RMSE.

I have not simply used bias effect on Year and mileage as above as the accuracy was not satisfactory. Instead, I have calculate their own linear regression arriving at with values of a and b for each year and trim as a function a + bx where x is the year of the car. This was particularly useful when we wanted to predict car price for years that data was not trained on.

Similar to the year effect, I have implemented the mileage effect.

5.1.1 Overall Mean Effect

We start by calculating the mean of all the pricing in the train_set. We then calculate the RMSE and include it in a table.

```
# Overall Mean Effect
# We start by calculating the mean of all the pricing in the train_set.
# We then calculate the RMSE and include it in a table.
mu <- mean(train_set$price)
mu
```

```
## [1] 160951.4
```

```
naive_rmse <- RMSE(train_set$price, mu)
rmse_results <- data_frame(Method_on_train_test = "Overall Average", RMSE = naive_rmse, Improvement = na
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.00

The RMSE value 89,000 QAR is roughly 25,000 USD for car values less than 100,000 USD. This is based on overall mean. The average price of a car is 160,951 QAR ~ 44k USD.

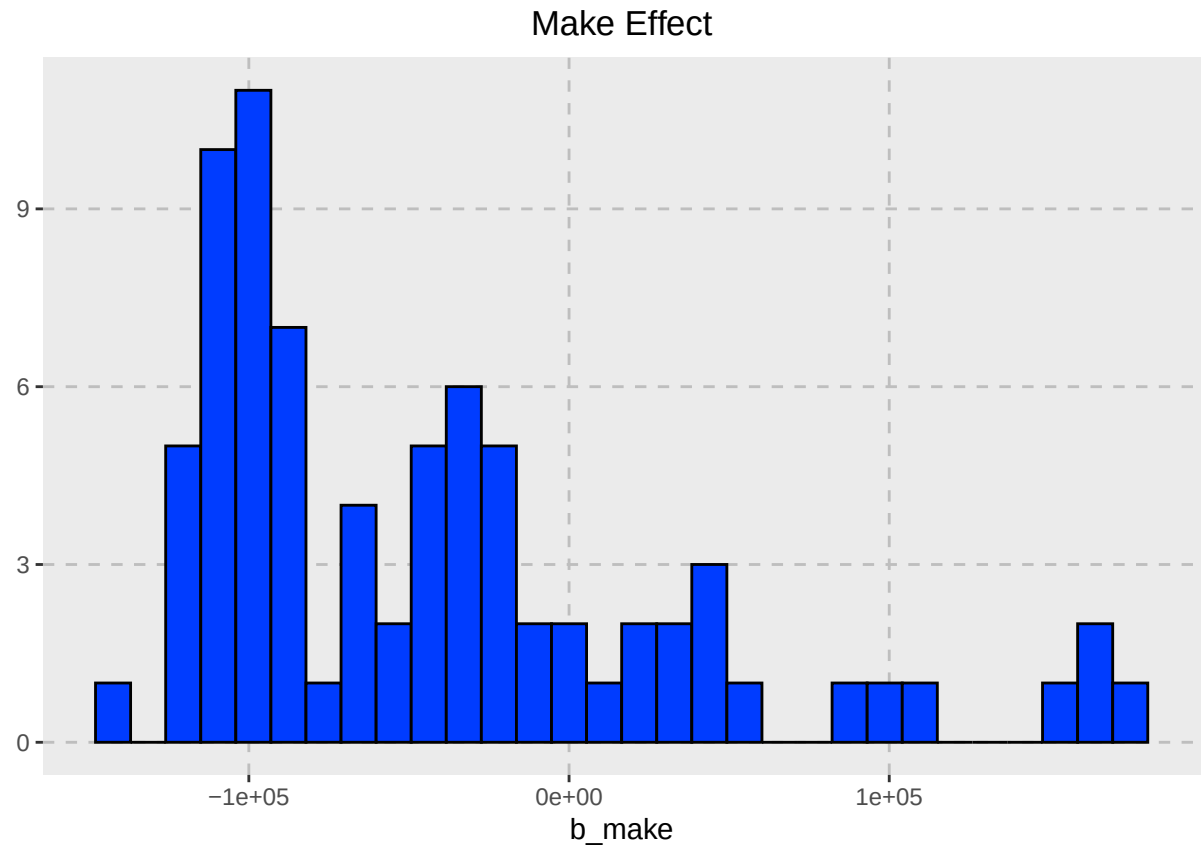
5.1.2 Make Effect

We used mu as a baseline and calculated the make bias b_make. We then calculate the RMSE and include it in a table.

```
# Create Make Effect table
make_avgs <- train_set %>%
  group_by(make) %>%
  summarize(b_make = mean(price - mu))
head(make_avgs)
```

```
## # A tibble: 6 x 2
##   make           b_make
##   <chr>           <dbl>
## 1 Alfa Romeo    -64951.
## 2 Aston Martin   28049.
## 3 Audi          -18680.
## 4 Avatr          83715.
## 5 BAIC         -121951.
## 6 BMW            1652.
```

```
# Plot Make Effect
make_avgs %>% qplot(b_make, geom ="histogram", bins = 30, data = ., color = I("black"), fill = I("#003C
labs(title = "Make Effect") +
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
panel.grid.minor = element_blank() # Removing minor grid lines
)
```




```
# Calculate the predicted price
# if the price is Na use the mean.
predicted_price <- mu + test_set %>%
  left_join(make_avgs, by='make') %>%
  pull(b_make)

summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   44298  131148  204375  173556  204375  261510
```

```
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   44298  131148  204375  173556  204375  261510
```

There was no NA in price predicted.

```
# Calculate the RMSE
model_1_rmse <- RMSE(predicted_price, test_set$price)
rmse_results <- bind_rows(
  rmse_results,
```

```
  data_frame(Method_on_train_test="Make Effect",
  RMSE = model_1_rmse ,
  Improvement = naive_rmse - model_1_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.00
Make Effect	75,115.39	14,350.23

As you can see RMSE has reduced significantly 14,350 QAR.

5.1.3 Model Effect

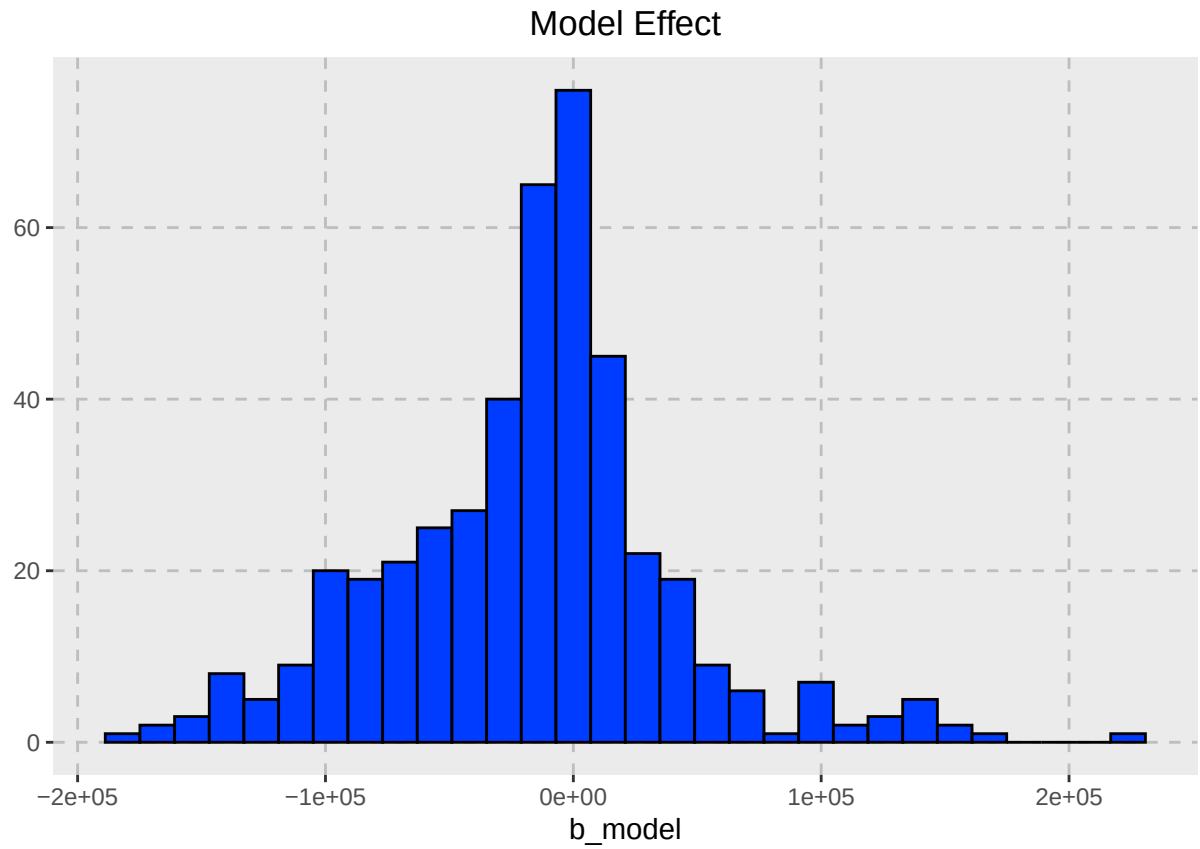
We used mu, Make as a baseline and calculated the Model bias b_model. We then calculate the RMSE and include it in a table. We have added a check to eliminate negative price values and limit to a lower bound of 20% value of average price for that trim.

```
price_lower_boundary<- 0.2

# Create Model Effect table
model_avgs <- train_set %>%
left_join(make_avgs, by='make') %>%
group_by(model) %>%
summarize(b_model = mean(price - mu - b_make))
head(model_avgs)
```

```
## # A tibble: 6 x 2
##   model                b_model
##   <chr>                  <dbl>
## 1 Alfa Romeo-4 C       104000
## 2 Alfa Romeo-GIULIA    -17000
## 3 Alfa Romeo-Giulietta -39200
## 4 Alfa Romeo-Stelvio    63000
## 5 Aston Martin-Vantage      0
## 6 Audi-A3              -57071.
```

```
# Plot Model Effect
model_avgs %>% qplot(b_model, geom ="histogram", bins = 30, data = ., color = I("black"), fill = I("#003
labs(title = "Model Effect") +
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
panel.grid.minor = element_blank() # Removing minor grid lines
)
```




```
# Calculate the predicted price
# if the price is Na use the mean.
predicted_price <- test_set %>%
  left_join(make_avgs, by='make') %>%
  left_join(model_avgs, by='model') %>%
  mutate(pred = mu + b_make + b_model) %>%
  mutate(pred2 = ifelse((pred) >= (price_pmmt *price_lower_boundary),
                        pred,
                        price_pmmt *price_lower_boundary)) %>%
  pull(pred2)

summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20714  150095  228727  182658  229659  364999
```

```
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20714  150095  228727  182658  229659  364999
```

There was no NA in price predicted.

```r
# Calculate the RMSE
model_2_rmse <- RMSE(predicted_price, test_set$price)
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_train_test="Make + Model Effect",
  RMSE = model_2_rmse ,
  Improvement = model_1_rmse - model_2_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.00
Make Effect	75,115.39	14,350.23
Make + Model Effect	62,988.79	12,126.59

As you can see RMSE has reduced significantly by 12,126 QAR.

5.1.4 Trim Effect

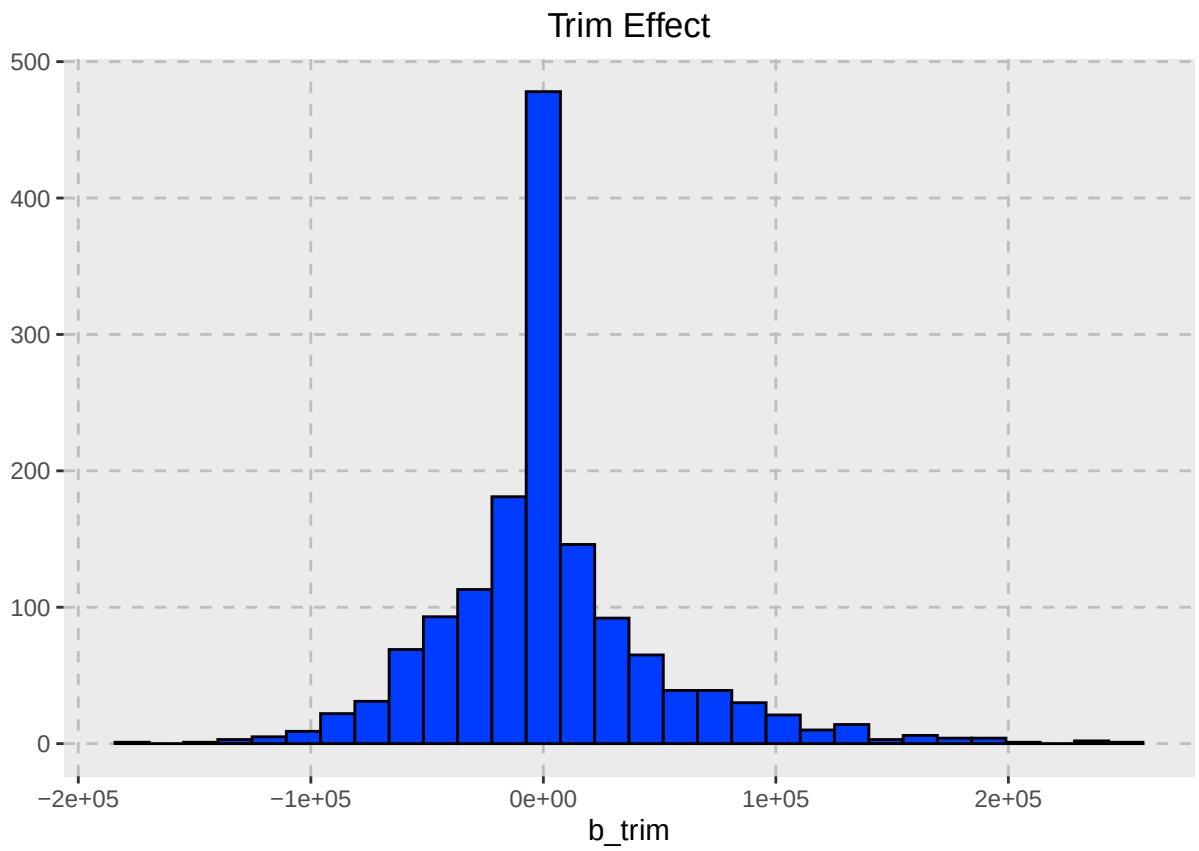
We used mu, Make, Model as a baseline and calculated the Trim bias b_trim. We then calculate the RMSE and include it in a table. We have added a check to eliminate negative price values and limit to a lower bound of 20% value of average price for that trim.

```r
# Create Trim Effect table
trim_avgs <- train_set %>%
left_join(make_avgs, by='make') %>%
left_join(model_avgs, by='model') %>%
group_by(trim) %>%
summarize(b_trim = mean(price - mu - b_make - b_model))
head(trim_avgs)
```

```
## # A tibble: 6 x 2
##   trim                                                b_trim
##   <chr>                                                <dbl>
## 1 Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic           0
## 2 Alfa Romeo-Alfa Romeo-GIULIA-Standard-4-Automatic   -20000
## 3 Alfa Romeo-Alfa Romeo-GIULIA-Veloce-4-Automatic      20000
## 4 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic     0
## 5 Alfa Romeo-Alfa Romeo-Stelvio-Quadrifoglio-6-Automatic  50000
## 6 Alfa Romeo-Alfa Romeo-Stelvio-Standard-4-Automatic    -50000
```

```r
# Plot Trim Effect
trim_avgs %>% qplot(b_trim, geom ="histogram", bins = 30, data = ., color = I("black"), fill = I("#003C
labs(title = "Trim Effect") +
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
```

```
panel.grid.minor = element_blank() # Removing minor grid lines
)
```




```
# Calculate the predicted price
# if the price is Na use the mean.
predicted_price <- test_set %>%
  left_join(make_avgs, by='make') %>%
  left_join(model_avgs, by='model') %>%
  left_join(trim_avgs, by='trim') %>%
  mutate(pred = mu + b_make + b_model + b_trim) %>%
  mutate(pred2 = ifelse((pred) >= (price_pmmt *price_lower_boundary),
                        pred,
                        price_pmmt *price_lower_boundary)) %>%
  pull(pred2)

summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20714  128577  185614  185045  229059  364999
```

```
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   20714  128577  185614  185045  229059  364999
```

There was no NA in price predicted.

```
# Calculate the RMSE

model_3_rmse <- RMSE(predicted_price, test_set$price)
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_train_test="Make + Model + Trim Effect",
  RMSE = model_3_rmse ,
  Improvement = model_2_rmse - model_3_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.00
Make Effect	75,115.39	14,350.23
Make + Model Effect	62,988.79	12,126.59
Make + Model + Trim Effect	39,819.28	23,169.51

As you can see RMSE has reduced significantly 23,170 QAR.

5.1.5 Year Effect

We used mu, Make, Model, Trim as a baseline and calculated the Year bias b_year. We then used linear regression to create a linear function where given any year we can calculate the bias effect of the year even if the year doesn't exist in the training data. We have used this link[5] for the linear regression formula. We then calculate the RMSE and include it in a table. We have added a check to eliminate negative price values and limit to a lower bound of 20% value of average price for that trim.

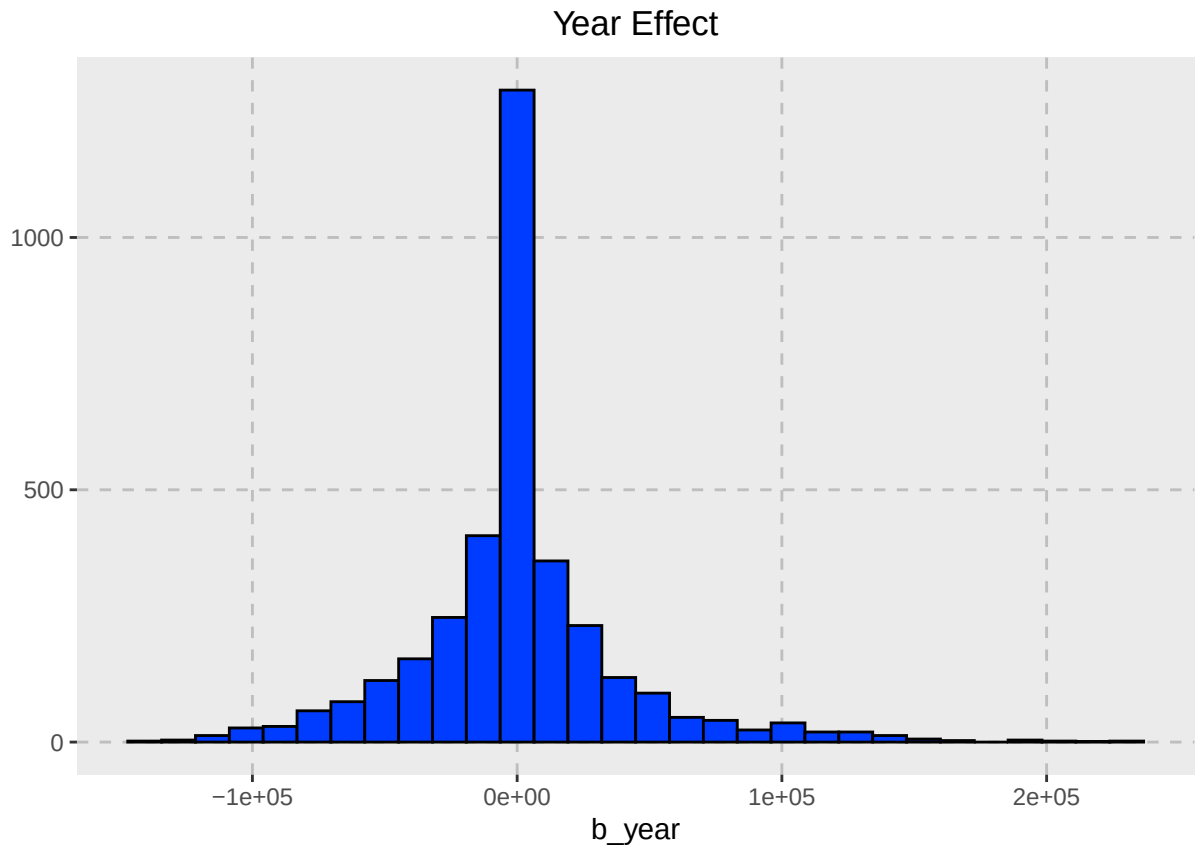
```
# Create year Effect table
year_avgs <- train_set %>%
left_join(make_avgs, by='make') %>%
left_join(model_avgs, by='model') %>%
left_join(trim_avgs, by='trim') %>%
#left_join(personal_avgs, by='personal') %>%
group_by(year) %>%
summarize(b_year = mean(price - mu - b_make - b_model - b_trim ))
head(year_avgs)
```

```
## # A tibble: 6 x 2
##   year                                                    b_year
##   <chr>                                                    <dbl>
## 1 Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic-2015          0
## 2 Alfa Romeo-Alfa Romeo-GIULIA-Standard-4-Automatic-2019       0
## 3 Alfa Romeo-Alfa Romeo-GIULIA-Veloce-4-Automatic-2020         0
```

[5] https://www.vedantu.com/formula/linear-regression-formula

```
## 4 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2014 -17800
## 5 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2015 -14800
## 6 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2020  12200
```

```
# Plot Year Effect
year_avgs %>% qplot(b_year, geom ="histogram", bins = 30, data = ., color = I("black"), fill = I("#003C
labs(title = "Year Effect") +
theme(
plot.title = element_text(hjust = 0.5), # center title
panel.grid.major = element_line(color = "gray", linetype = "dashed"), # add
#dashed lines
panel.grid.minor = element_blank() # Removing minor grid lines
)
```



5.1.5.1 Linear Regression First we extracted year and trim from year as they were combined earlier. We then calculated a and b, where b_year = a +bx and x is year2.

Since you need more than 2 points to calculate a and b, in case of 1 record We used the same b_year bias effect for a and b is zero.

In case we are predicting for a year that was not trained we are using the average price of the car for that particular trim with a factor of year_v=0.1 which is how much the car will devalue per year deviated away from the average price per trim.

```
year_avgs_updated <- year_avgs
year_avgs_updated <- year_avgs_updated %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))
year_avgs_updated <- year_avgs_updated %>% mutate (year2 = as.numeric(year2))
year_avgs_updated <- year_avgs_updated %>% mutate (trim = substr(year,0, nchar(year) - 5))
head(year_avgs_updated)
```

```
## # A tibble: 6 x 4
##   year                                                    b_year year2 trim
##   <chr>                                                    <dbl> <dbl> <chr>
## 1 Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic-2015          0  2015 Alfa R~
## 2 Alfa Romeo-Alfa Romeo-GIULIA-Standard-4-Automatic-2019       0  2019 Alfa R~
## 3 Alfa Romeo-Alfa Romeo-GIULIA-Veloce-4-Automatic-2020         0  2020 Alfa R~
## 4 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2014 -17800  2014 Alfa R~
## 5 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2015 -14800  2015 Alfa R~
## 6 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic-2020  12200  2020 Alfa R~
```

```
# Calculate a and b
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(n= n())
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(x =  year2)
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(y =  b_year)
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(Ex =  sum(x))
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(Ey =  sum(y))
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(Exx =  sum(x*x))
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(Exy =  sum(x*y))
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(b = (n*Exy - Ex*Ey) / (n*Exx - (Ex
year_avgs_updated <- year_avgs_updated %>% group_by(trim) %>% mutate(a = (Ey - b*Ex) / n)
year_avgs_updated <- year_avgs_updated %>%
mutate(a = ifelse(n == 1, b_year, a),
b = ifelse(n == 1, 0, b))
# the below is used to verify the results however for prediction we calculate again.
year_avgs_updated <- year_avgs_updated %>% mutate (b_year_updated = a + b*year2)
# below is used for prediction where you need a and b
year_avgs_uu <- year_avgs_updated %>%
    select(trim, a, b) %>%
    distinct()
head(year_avgs_uu)
```

```
## # A tibble: 6 x 3
## # Groups:   trim [6]
##   trim                                                      a     b
##   <chr>                                                 <dbl> <dbl>
## 1 Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic            0     0
## 2 Alfa Romeo-Alfa Romeo-GIULIA-Standard-4-Automatic         0     0
## 3 Alfa Romeo-Alfa Romeo-GIULIA-Veloce-4-Automatic           0     0
## 4 Alfa Romeo-Alfa Romeo-Giulietta-Standard-4-Automatic -8645584. 4284.
## 5 Alfa Romeo-Alfa Romeo-Stelvio-Quadrifoglio-6-Automatic    0     0
## 6 Alfa Romeo-Alfa Romeo-Stelvio-Standard-4-Automatic        0     0
```

```
year_v <- 0.1

# Calculate the predicted price
# if the price is Na use the mean.
```

```
predicted_price <- test_set %>%
  left_join(make_avgs, by='make') %>%
  left_join(model_avgs, by='model') %>%
  left_join(trim_avgs, by='trim') %>%
  left_join(year_avgs_uu, by='trim') %>% # check if trim is correct
  mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
  mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(yea
  mutate(pred = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_updated
  + price_year_value
  ) %>%
  mutate(pred2 = ifelse((pred) >= (price_pmmt *price_lower_boundary),
                        pred,
                        price_pmmt *price_lower_boundary))  %>%
  pull(pred2)
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19000  113544  188701  185686  256900  364999
```

```
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19000  113544  188701  185686  256900  364999
```

There was no NA in price predicted.

```
# Calculate the RMSE
model_4_rmse <- RMSE(predicted_price, test_set$price)
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_train_test="Make + Model + Trim + Year Effect",
  RMSE = model_4_rmse ,
  Improvement = model_3_rmse - model_4_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.00
Make Effect	75,115.39	14,350.23
Make + Model Effect	62,988.79	12,126.59
Make + Model + Trim Effect	39,819.28	23,169.51
Make + Model + Trim + Year Effect	22,194.05	17,625.23

As you can see RMSE has reduced significantly 17,625 QAR.

5.1.6 Mileage Effect

We used Average_mileage_range table which was created in feature engineering. We then used linear regression to create a linear function where given any mileage_range we can calculate the bias effect of the mileage even if the mileage range doesn't # exist in the training data. We have used the same link[6] for the linear regression formula. We then calculate the RMSE and include it in a table. We have added a check to eliminate negative price values and limit to a lower bound of 20% value of average price for that trim.

5.1.6.1 Linear Regression First we extracted year as it was linked to trim. We then calculated aa and bb. b_mileage = aa +bbx where x is mileage_range. Since you need more than 2 points to calculate aa and bb, in case of 1 record. We used the same avg_mileage_range bias effect for aa and bb is zero. We are using a factor of 0.16 as weightage of the mileage effect as it was tested. to have the highest effect by trial and error. We could have used sapply function. After calculating the average mileage_range for a particular trim and year, we then compared how far it is from the avg_milage_range for that trim, divided by 100,000 mileage to create this mileage effect. This was also done via trial and error to arrive to a good RMSE.

```
damru <- data_avg_mileage_range
damru <- damru %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))
damru <- damru %>% mutate (year2 = as.numeric(year2))
damru <- damru %>% mutate (trim = substr(year,0, nchar(year) - 5))
head(data_avg_mileage_range)
```

```
## # A tibble: 6 x 2
## # Groups:   year [6]
##   year                                                  avg_mileage_range
##   <chr>                                                             <dbl>
## 1 Toyota-Toyota-Land Cruiser-G-6-Automatic-2017                    180000
## 2 Toyota-Toyota-Land Cruiser-GXR- Grand Touring-6-Automatic-2~      18696.
## 3 Nissan-Nissan-Pickup-Standard-4-Manual-2016                     261818.
## 4 Toyota-Toyota-Land Cruiser-GXR-8-Automatic-2021                  71806.
## 5 Toyota-Toyota-Hilux-Standard-4-Manual-2022                       14615.
## 6 Toyota-Toyota-Camry-GLE-4-Automatic-2021                             0
```

```
# Calculate aa and bb
damru <- damru %>% group_by(trim) %>% mutate(n= n())
damru <- damru %>% group_by(trim) %>% mutate(x =  year2)
damru <- damru %>% group_by(trim) %>% mutate(y =  avg_mileage_range)
damru <- damru %>% group_by(trim) %>% mutate(Ex =  sum(x))
damru <- damru %>% group_by(trim) %>% mutate(Ey =  sum(y))
damru <- damru %>% group_by(trim) %>% mutate(Exx =  sum(x*x))
damru <- damru %>% group_by(trim) %>% mutate(Exy =  sum(x*y))
damru <- damru %>% group_by(trim) %>% mutate(bb = (n*Exy - Ex*Ey) / (n*Exx - (Ex*Ex))   )
damru <- damru %>% group_by(trim) %>% mutate(aa = (Ey - bb*Ex) / n)

damru <- damru %>%
mutate(aa = ifelse(n == 1, avg_mileage_range, aa),
bb = ifelse(n == 1, 0, bb))

# the below is used to verify the results however for prediction we calculate again.
damru <- damru %>% mutate (avg_mileage_range_updated = ifelse ((aa + bb*year2)>0,aa + bb*year2,0 ))
# below is used for prediction where you need aa and bb
```

[6] https://www.vedantu.com/formula/linear-regression-formula

```r
damruu <- damru %>%
     select(trim, aa, bb) %>%
     distinct()
head(damru)
```

```
## # A tibble: 6 x 14
## # Groups:   trim [6]
##   year      avg_mileage_range year2 trim      n     x      y    Ex     Ey    Exx
##   <chr>                 <dbl> <dbl> <chr> <int> <dbl>  <dbl> <dbl>  <dbl>  <dbl>
## 1 Toyota-T~           180000   2017 Toyo~    10  2017 1.80e5 20187 1.51e6 4.08e7
## 2 Toyota-T~            18696.  2021 Toyo~     4  2021 1.87e4  8084 2.03e5 1.63e7
## 3 Nissan-N~           261818.  2016 Niss~     2  2016 2.62e5  4031 5.44e5 8.12e6
## 4 Toyota-T~            71806.  2021 Toyo~     8  2021 7.18e4 16140 1.42e6 3.26e7
## 5 Toyota-T~            14615.  2022 Toyo~    11  2022 1.46e4 22209 1.19e6 4.48e7
## 6 Toyota-T~                0   2021 Toyo~     5  2021 0      10104 3.52e5 2.04e7
## # i 4 more variables: Exy <dbl>, bb <dbl>, aa <dbl>,
## #   avg_mileage_range_updated <dbl>
```

```r
mileage_v=0.16
mileage_v2= 0 # not used
mileage_v3= 10 # 10 * 10,000 = 100,000 km

# Calculate the predicted price
# if the price is Na use the mean.
predicted_price <- test_set %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>%
    left_join(damruu, by='trim') %>%
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                        pred2,
                        price_pmmt *price_lower_boundary))  %>%
    pull(pred3)
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20368  113300  185470  186907  262479  364999
```

```r
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20368  113300  185470  186907  262479  364999
```

There was no NA in price predicted.

```
# Calculate the RMSE
model_5_rmse <- RMSE(predicted_price, test_set$price)
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_train_test="Make + Model + Trim + Year + Mileage Effect",
  RMSE = model_5_rmse ,
  Improvement = model_4_rmse - model_5_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement
Overall Average	89,465.62	0.000
Make Effect	75,115.39	14,350.230
Make + Model Effect	62,988.79	12,126.591
Make + Model + Trim Effect	39,819.28	23,169.513
Make + Model + Trim + Year Effect	22,194.05	17,625.233
Make + Model + Trim + Year + Mileage Effect	19,763.28	2,430.771

As you can see RMSE has reduced significantly 2,431 QAR.

5.2 Model Testing

5.2.1 Validation Results

We have previously tested against our test_set test dataset. We will now test against the unseen validate_set validation dataset.

```
# Predict price
predicted_price <- validate_set %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>% # check if trim is correct
    left_join(damruu, by='trim') %>%
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                          pred2,
                          price_pmmt *price_lower_boundary))  %>%
    pull(pred3)
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20747  112726  180747  184399  262932  364610
```

```
mean_predicted_rating <- mean(predicted_price, na.rm = TRUE)
predicted_price[is.na(predicted_price)] <- mean_predicted_rating
summary(predicted_price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20747  112726  180747  184399  262932  364610
```

There was no NA in price predicted.

```
# Calculate the RMSE
model_6_rmse <- RMSE(predicted_price, validate_set$price)
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_validation="Method 01 - Linear Regression",
  RMSE = model_6_rmse ,
  Improvement = model_5_rmse - model_6_rmse))
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement	Method_on_validation
Overall Average	89,465.62	0.0000	NA
Make Effect	75,115.39	14,350.2301	NA
Make + Model Effect	62,988.79	12,126.5907	NA
Make + Model + Trim Effect	39,819.28	23,169.5130	NA
Make + Model + Trim + Year Effect	22,194.05	17,625.2332	NA
Make + Model + Trim + Year + Mileage Effect	19,763.28	2,430.7708	NA
NA	20,225.45	-462.1678	Method 01 - Linear Regression

As you can see Final RMSE on validation is 20,225 vs 19,763 on test dataset.

The RMSE has dropped down from 89,000 to around 20,000.

For those who are familiar with the Netflix project, if we compare with the Netflix target goal RMSE of 0.865, the 20,000 RMSE is roughly 3 times better. (20,000*5/365000)

5.2.2 Predict Car Price - User input

In this section, we can predict the price of a car based on the below user input.

This has been used to test the model manually.

Create a list of a few car and its features, you can change this as you like.

```r
new_data <- data.frame(make = "Toyota", model = "Land Cruiser", trim = "GXR",
personal = "", year = 2015,  gear_type = "Automatic", cylinder = 8, mileage = 10000)
new_row <- data.frame(make = "Genesis", model = "G70", trim = "Standard",
personal = "", year = 2019, gear_type = "Automatic", cylinder = 6, mileage = 90000)
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Nissan", model = "Patrol", trim = "SE",
personal = "", year = 2014, gear_type = "Automatic", cylinder = 8, mileage = 125000)
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Toyota", model = "Land Cruiser", trim = "GXR",
personal = "", year = 2013, gear_type = "Automatic", cylinder = 6, mileage = 175000)
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Chevrolet", model = "Camaro", trim = "ZL1",
personal = "", year = 2014, gear_type = "Automatic", cylinder = 8, mileage = 10000)
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Alfa Romeo", model = "4 C", trim = "Standard",
personal = "", year = 2019, gear_type = "Automatic", cylinder = 4, mileage = 10000)
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Audi", model = "A8", trim = "4.0",
personal = "", year = 2019, gear_type = "Automatic", cylinder = 8, mileage = 100000)
new_data <- rbind(new_data, new_row)
new_data
```

```
##         make        model     trim personal year gear_type cylinder mileage
## 1     Toyota Land Cruiser      GXR          2015 Automatic        8   10000
## 2    Genesis          G70 Standard          2019 Automatic        6   90000
## 3     Nissan       Patrol       SE          2014 Automatic        8  125000
## 4     Toyota Land Cruiser      GXR          2013 Automatic        6  175000
## 5  Chevrolet       Camaro      ZL1          2014 Automatic        8   10000
## 6 Alfa Romeo          4 C Standard          2019 Automatic        4   10000
## 7       Audi           A8      4.0          2019 Automatic        8  100000
```

```r
# Data wrangling by updating mileage_range, year, model and trim
new_data$mileage_range <- ifelse(new_data$mileage == 0, 0, ceiling(new_data$mileage / 10000) * 10000)
new_data$year <- paste(new_data$make, new_data$make, new_data$model, new_data$trim, new_data$year, sep =
new_data$model <- paste(new_data$make, new_data$model, sep = "-")
new_data$trim <- paste( new_data$make, new_data$model, new_data$trim,new_data$cylinder, new_data$gear_ty
new_data
```

```
##         make               model                                        trim
## 1     Toyota Toyota-Land Cruiser     Toyota-Toyota-Land Cruiser-GXR-8-Automatic
## 2    Genesis         Genesis-G70       Genesis-Genesis-G70-Standard-6-Automatic
## 3     Nissan       Nissan-Patrol            Nissan-Nissan-Patrol-SE-8-Automatic
## 4     Toyota Toyota-Land Cruiser     Toyota-Toyota-Land Cruiser-GXR-6-Automatic
## 5  Chevrolet    Chevrolet-Camaro     Chevrolet-Chevrolet-Camaro-ZL1-8-Automatic
## 6 Alfa Romeo      Alfa Romeo-4 C Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic
## 7       Audi             Audi-A8                   Audi-Audi-A8-4.0-8-Automatic
##   personal                                    year gear_type cylinder mileage
## 1             Toyota-Toyota-Land Cruiser-GXR-2015 Automatic        8   10000
## 2               Genesis-Genesis-G70-Standard-2019 Automatic        6   90000
## 3                    Nissan-Nissan-Patrol-SE-2014 Automatic        8  125000
## 4             Toyota-Toyota-Land Cruiser-GXR-2013 Automatic        6  175000
## 5             Chevrolet-Chevrolet-Camaro-ZL1-2014 Automatic        8   10000
## 6          Alfa Romeo-Alfa Romeo-4 C-Standard-2019 Automatic        4   10000
```

```
## 7                            Audi-Audi-A8-4.0-2019 Automatic        8  100000
##   mileage_range
## 1         10000
## 2         90000
## 3        130000
## 4        180000
## 5         10000
## 6         10000
## 7        100000
```

```r
# predict using above linear regression model
predicted_price <- new_data %>%
  left_join(make_avgs, by='make') %>%
  left_join(model_avgs, by='model') %>%
  left_join(trim_avgs, by='trim') %>%
  left_join(year_avgs_uu, by='trim') %>%
  left_join(data_price_pmmt, by='trim') %>%
  left_join(data_avg_ymmt, by='trim') %>%
  left_join(data_avg_mileage_range, by='year') %>%
  left_join(damruu, by='trim') %>%
  mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
  mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000) *
  mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(yea
  mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeric
  mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_update
         + price_year_value
  ) %>%
  mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mil
  mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                        pred2,
                        price_pmmt *price_lower_boundary))

# add Year and mileage effects to the table
predicted_price<- predicted_price %>% mutate(Year_Effect = pred1- price_pmmt)
predicted_price<- predicted_price %>% mutate(Mileage_Effect = pred2- pred1)
# add some other parameters used for debugging
predicted_price <- predicted_price %>%
  group_by(trim) %>%
  mutate(pred3_pmmt = mean(pred3))
predicted_price <- predicted_price %>%
  group_by(trim) %>%
  mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
predicted_price <- predicted_price %>%
  group_by(trim) %>%
  mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)

# print predicted price
predicted_price %>% select(trim, year, mileage, L_Pred_Price = pred3) %>%
mutate(L_Pred_Price = format(L_Pred_Price, nsmall = 2, big.mark = ",")) %>%
mutate(mileage = format(mileage, nsmall = 2, big.mark = ",")) %>%
 knitr::kable()
```

trim	year	mileage	L_Pred_Price
Toyota-Toyota-Land Cruiser-GXR-8-Automatic	Toyota-Toyota-Land Cruiser-GXR-2015	10,000.00	164,967.47
Genesis-Genesis-G70-Standard-6-Automatic	Genesis-Genesis-G70-Standard-2019	90,000.00	73,737.78
Nissan-Nissan-Patrol-SE-8-Automatic	Nissan-Nissan-Patrol-SE-2014	125,000.00	62,466.69
Toyota-Toyota-Land Cruiser-GXR-6-Automatic	Toyota-Toyota-Land Cruiser-GXR-2013	175,000.00	89,810.60
Chevrolet-Chevrolet-Camaro-ZL1-8-Automatic	Chevrolet-Chevrolet-Camaro-ZL1-2014	10,000.00	94,773.15
Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic	Alfa Romeo-Alfa Romeo-4 C-Standard-2019	10,000.00	280,000.00
Audi-Audi-A8-4.0-8-Automatic	Audi-Audi-A8-4.0-2019	1e+05	187,106.40

5.2.3 Export Recommendation - Full Data

In this section, we will use the model above and predict the prices of the full dataset (train, test, validate). We will then export files to provide the best recommended car deals.

A Linear Regression Full data export folder will be generated and a subfolder will be created based on run time time stamp. A few files will be created used for debugging, however there are 2 export files that are of interest.

_Export_Formated.csv contains full data along with predicted price and comparison against actual price. You can filter by column Price_Gap_Percentage with the the highest positive value to get the best deals.

_Export_Formated_Filtered.csv is the same as the earlier file, however has been sorted and filtered with the best deals per trim, with Price_Gap_Percentage > 20 having mileage less than 80,000 km and which is still available for sale (Not sold).

```
# Predict Price
data2 <- data %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>%
    left_join(damruu, by='trim') %>%
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                          pred2,
                          price_pmmt *price_lower_boundary))
# Calculate the difference between the predicted and actual price
data2<- data2 %>% mutate(price_difference = pred3 - price)
# Calculate the percentage difference based on the price
data2<- data2 %>% mutate(percentage_difference = (price_difference / price) * 100)
data2<- data2 %>% mutate(percentage_difference_updated = ifelse (percentage_difference<=0.01 & percentag
# add Year and mileage effects to the table
```

```
data2<- data2 %>% mutate(absolut_difference = abs(price_difference))
data2<- data2 %>% mutate(Year_Effect = pred1- price_pmmt)
data2<- data2 %>% mutate(Mileage_Effect = pred2- pred1)
# add some other parameters used for debugging
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pred3_pmmt = mean(pred3))
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)
```

```
# create folder for linear regression and automatic timestamp subfolder
version_folder <- "01. Linear Regression Model - Full Data Export"

# Check if the folder already exists
if (!dir.exists(version_folder)) {
    # Create the folder
    dir.create(version_folder)
}
# Get the current date and time
current_datetime <- Sys.time()
# Create a folder name based on date and time
folder_name <- format(current_datetime, "%Y%b%d_%H-%M-%S")
# Create a folder with current_datetime inside the version folder
folder_path <- paste0(version_folder, "/", folder_name)
dir.create(folder_path)
folder_path
```

```
## [1] "01. Linear Regression Model - Full Data Export/2024Apr18_11-54-03"
```

```
L_Full_path<-folder_path
```

```
# exporting full data with predictions
write.csv(data2, file = paste0(folder_path, "/_Export_full.csv"), row.names = FALSE)

# Renaming and selecting Columns to be more user friendly
data3<- data2 %>% select(
Id = web.scraper.order,
Link = car.href,
Make = make,
Model = model,
Trim= trim,
Year = year_updated,
Price= price,
Mileage =mileage,
Date_sold= date_sold,
Average_Year_per_Trim=avg_ymmt,
#Average_Milerange_per_Trim_Year= avg_mileage_range,
Average_Milerange_per_Trim_Year_LR= avg_mileage_range_updated,
```

```
Average_Price_per_Trim =price_pmmt,
Year_Effect,
Mileage_Effect,
L_Predicted_Price = pred3,
L_Price_Gap_Percentage =percentage_difference_updated,
)

# Formatting number with 2 digits and comma
data3 <- data3 %>%
   mutate(Price = format(Price, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage = format(Mileage, nsmall = 0, big.mark = ",")) %>%
   mutate(Average_Year_per_Trim = format(Average_Year_per_Trim, nsmall = 2, big.mark = ",")) %>%
   mutate(Average_Price_per_Trim = format(Average_Price_per_Trim, nsmall = 2, big.mark = ",")) %>%
   #mutate(Average_Milerange_per_Trim_Year = format(Average_Milerange_per_Trim_Year, nsmall = 2, big.ma
   mutate(Average_Milerange_per_Trim_Year_LR = format(Average_Milerange_per_Trim_Year_LR, nsmall = 2, bi
   mutate(Year_Effect = format(Year_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage_Effect = format(Mileage_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ","))

# Sequencing data by trim then by Price_Gap_Percentage
data3 <- data3 %>%
    arrange(Trim, desc(L_Price_Gap_Percentage)) %>%
    group_by(Trim) %>%
    mutate(Count = n())

# export full recommendation file for best car deals.
write.csv(data3, file = paste0(folder_path, "/_Export_Formated.csv"), row.names = FALSE)

# Filtering by high Price_Gap_Percentage > 20, where there are many training data
# count > 10, not sold, and mileage < 80,000 km
data4 <- data3 %>%
    arrange(Trim, desc(L_Price_Gap_Percentage)) %>%
    filter(Count > 10) %>%
    filter(L_Price_Gap_Percentage > 20) %>%
    filter(Date_sold == "") %>%
    mutate( mileage2 = (gsub(",", "", Mileage))) %>%
    mutate( mileage2 = as.numeric(mileage2)) %>%
    filter((mileage2) < 80000)

# Reducing the number of results by top 2 per trim.
data4 <- data4 %>%
    group_by(Trim) %>%
    #arrange(Trim, desc(Price_Gap_Pecentage)) %>%
    #arrange(desc(Price_Gap_Pecentage)) %>%
    top_n(2, L_Price_Gap_Percentage)
    #top_n(if (Count/10<1,1,Count/10), Price_Gap_Pecentage)

# Rearranging Column order.
data4 <- data4 %>%
select(
Id,
Link,
Make,
```

```
Model,
Trim,
Year,
Price,
Mileage,
L_Predicted_Price,
L_Price_Gap_Percentage,
Average_Year_per_Trim,
#Average_Milerange_per_Trim_Year,
Average_Milerange_per_Trim_Year_LR,
Average_Price_per_Trim,
Year_Effect,
Mileage_Effect
)
# export full recommendation file for best car deals which is sorted and filtered.
write.csv(data4, file = paste0(folder_path, "/_Export_Formated_Filtered.csv"), row.names = FALSE)
nrow(data4)
```

```
## [1] 90
```

```
# Below we can save the model for future use. model 48 have been saved.
# save.image(file = "Model_48.RData")
```

Navigate to "01. Linear Regression Model - Full Data Export" and select the most recent folder, you will find _Export_Formated_Filtered.csv which will provide you with the best car deals.

You can open the file _Export_Formated_Filtered.csv and view the car deals, all of which have much lower actual price compared to our prediction. Upon verification, we realize many of those of are great deals.

If you open via excel Columns A to H are the original data, where G is the actual price and I is the predicted price. The 13560 listing have been shortlisted to around 90 deals where their actual price is way lower than predicted. This is being recommended by the system indicating a possible good deal.

The last 5 columns also explain how this prediction came about. This offers interpretability. These columns show the average year, mileage and price for this trim, then year and mileage price effects. This explains the prediction and gives us reassurance.

For wider selection, you can refer to the _Export_Formated.csv where you have the full analysis.

This has been a very exciting journey for me and an important milestone so far.

The below table has a sample list of best deals. As you can see the actual price is way lower (at least 20%) than the predicted. These are a great deals. If you check one by one like I did, when you compare the same or similar trim and year you'll notice that these are the lowest prices.

```
data4<-data4 %>%
  mutate(L_Predicted_Price = round(as.numeric((gsub(",", "", L_Predicted_Price))), digits = 0)) %>%
  mutate(L_Price_Gap_Percentage = round(as.numeric((gsub(",", "", L_Price_Gap_Percentage))), digits = 0
  mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(L_Price_Gap_Percentage = format(L_Price_Gap_Percentage, nsmall = 0, big.mark = ","))

L_Full_Table<-data4 %>%
    group_by(Trim) %>%
    slice(1) %>%
    select(Trim, Year, Mileage, Price = Price,Pred_Price= L_Predicted_Price,Gap_Perc=L_Price_Gap_Percen
    #mutate(Mileage = format(Mileage, nsmall = 2, big.mark = ",")) %>%
```

```
    #head(20) %>%
    knitr::kable()
L_Full_Table
```

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
BMW-BMW-7-Series-730 Li-6-Automatic	2019	73,700	120,000	159,320	33
BMW-BMW-7-Series-740 Li-6-Automatic	2020	24,000	185,000	229,251	24
BMW-BMW-X-Series-X6-6-Automatic	2016	70,000	80,000	99,988	25
BMW-BMW-X-Series-X6-8-Automatic	2016	62,000	89,000	116,843	31
Cadillac-Cadillac-Escalade-Platinum-8-Automatic	2019	79,000	149,000	204,423	37
Cadillac-Cadillac-Escalade-Standard-8-Automatic	2020	77,000	159,000	225,965	42
Chevrolet-Chevrolet-Silverado-LT-8-Automatic	2021	64,000	112,000	138,801	24
Chevrolet-Chevrolet-Silverado-RST-8-Automatic	2020	77,000	94,000	125,492	34
Chevrolet-Chevrolet-Silverado-Standard-8-Automatic	2021	50,000	89,000	116,040	30
Chevrolet-Chevrolet-Silverado-Trail Boss-8-Automatic	2021	27,000	116,000	139,637	20
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	35
Dodge-Dodge-Charger-RT-8-Automatic	2021	42,000	90,000	119,800	33
Dodge-Dodge-Ram-Standard-8-Automatic	2021	30,000	101,000	154,459	53
Fiat-Fiat-595-Abarth-4-Automatic	2022	18,000	70,000	88,865	27
Ford-Ford-F-150-6-Automatic	2023	1,000	195,000	269,775	38
Ford-Ford-Mustang-GT-8-Automatic	2021	28,000	105,000	134,911	28
GMC-GMC-Sierra-1500-8-Automatic	2022	24,000	106,750	145,834	37
GMC-GMC-Sierra-2500 HD-8-Automatic	2021	28,000	149,000	181,072	22
GMC-GMC-Sierra-AT4-8-Automatic	2021	43,000	119,000	154,424	30
GMC-GMC-Sierra-Elevation-8-Automatic	2021	18,000	98,000	158,350	62
GMC-GMC-Yukon-Standard-8-Automatic	2019	64,000	123,000	159,331	30
Geely-Geely-Coolray-Standard-3-Automatic	2022	18,000	47,000	56,511	20
Hyundai-Hyundai-Staria-Standard-6-Automatic	2023	24,517	91,000	130,702	44
Infiniti-Infiniti-QX-80-8-Automatic	2020	55,000	145,000	198,102	37
Jeep-Jeep-Wrangler-Sahara-6-Automatic	2018	75,000	89,000	110,093	24
Kia-Kia-Picanto-Standard-4-Automatic	2022	36,000	33,000	39,843	21
Kia-Kia-Sportage-Standard-4-Automatic	2020	68,000	58,000	70,548	22
Land Rover-Land Rover-Evoque-Standard-4-Automatic	2020	42,000	144,000	181,170	26
Land Rover-Land Rover-Range Rover-Sport Super charged-8-Automatic	2016	68,000	117,000	146,029	25
Land Rover-Land Rover-Range Rover-Vogue SE Super charged-8-Automatic	2017	79,300	135,000	202,081	50
Land Rover-Land Rover-Range Rover-Vogue SE-8-Automatic	2020	54,000	185,000	277,907	50
Lexus-Lexus-LX-570-8-Automatic	2017	33,000	197,000	265,688	35
Lexus-Lexus-RX-350-6-Automatic	2020	33,000	130,000	168,147	29
Mercedes-Benz-Mercedes-Benz-E-Class-200-4-Automatic	2016	78,000	59,000	73,818	25
Mercedes-Benz-Mercedes-Benz-E-Class-400-6-Automatic	2016	34,000	82,000	101,554	24
Mercedes-Benz-Mercedes-Benz-S-Class-450-6-Automatic	2020	55,000	195,000	268,259	38
Mini-Mini-Cooper-JCW-4-Automatic	2023	73,000	100,000	135,227	35
Mini-Mini-Cooper-S-4-Automatic	2020	34,000	65,000	93,814	44
Mini-Mini-Cooper-Standard-4-Automatic	2020	54,000	73,000	87,795	20
Mitsubishi-Mitsubishi-Eclipse-Cross Highline-4-Automatic	2022	0	69,500	88,664	28
Mitsubishi-Mitsubishi-Fuso Canter-Standard-4-Manual	2020	0	83,000	102,035	23
Nissan-Nissan-Navara-Standard-4-Automatic	2017	36,751	35,995	49,329	37
Nissan-Nissan-Patrol-Super Safari-6-Automatic	2022	28,000	154,000	187,007	21
Nissan-Nissan-Sunny-Standard-4-Automatic	2020	78,000	27,500	34,854	27

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
Suzuki-Suzuki-Jimny-Standard-4-Automatic	2018	16,000	39,000	50,227	29
Suzuki-Suzuki-Jimny-Standard-4-Manual	2023	0	65,000	82,035	26
Toyota-Toyota-Fortuner-Standard-4-Automatic	2015	13,000	55,000	67,465	23
Toyota-Toyota-Hilux-SR5-4-Automatic	2019	54,000	65,000	83,343	28
Toyota-Toyota-Hilux-Standard-4-Automatic	2020	37,000	68,000	88,409	30
Toyota-Toyota-Hilux-Standard-4-Manual	2023	0	69,000	94,529	37
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2021	51,000	149,000	194,185	30
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	249,000	312,285	25
Toyota-Toyota-Land Cruiser-GXR-6-Automatic	2021	25,000	179,000	234,238	31
Toyota-Toyota-Land Cruiser-Hard Top-6-Manual	2021	0	103,000	134,732	31
Toyota-Toyota-Land Cruiser-LX-6-Manual	2021	0	110,000	137,700	25
Toyota-Toyota-Land Cruiser-LX-8-Manual	2023	0	136,000	164,240	21
Toyota-Toyota-Land Cruiser-VX Twin Turbo-6-Automatic	2023	39,000	255,000	314,842	23

5.2.4 Export Recommendation - Recent Data

In this section we will run the same model above on fresh new data web scrapped. The system will then automatically provide recommendation of which cars are the best deals. This is a powerful and useful tool.

Below can be used to avoid rerunning the entire code above.

```
# load("Model_48.RData")
# nrow(data4)
```

The below can be run again in case we loaded the model 48 and started a new session.

```
# Install the packages if not already installed
if (!require(magrittr)) install.packages("magrittr")
if (!require(dplyr)) install.packages("dplyr")
# Load the packages
library(magrittr)
library(dplyr)
library(caret)
library(stringr)
```

```
# Read the csv file from relative path
data_p <- read.csv("pred09Apr202411am.csv")
nrow(data_p)
```

```
## [1] 360
```

```
# same data cleaning and filtering process used earlier ---

# Clean mileage column from "Km" and comma and convert to numeric
data_p$mileage <- (gsub(" Km", "", data_p$mileage))
data_p$mileage <- (gsub(",", "", data_p$mileage))
data_p$mileage <- as.numeric(data_p$mileage)
# Convert cylinder to number
data_p$cylinder <- as.integer(data_p$cylinder)
```

```r
# Clean price column from comma and convert to numeric
data_p$price <- (gsub(",", "", data_p$price))
data_p$price <- as.numeric(data_p$price)
# Extract 2 columns from time, sold_date and posted time
data_p$date_sold <- NA
data_p$date_sold <- ifelse(grepl("/", data_p$time), format(as.POSIXct(data_p$time, format = "%d/%m/%Y")
data_p$posted_time <- NA
data_p$posted_time <- ifelse(grepl("hour", data_p$time), as.numeric(gsub(" hours ago", "", data_p$time)
data_p$posted_time <- ifelse(grepl("minute", data_p$time), as.numeric(gsub(" minutes ago", "", data_p$t
# Create new column mileage_range from mileage
data_p <- data_p %>%
  mutate(mileage_range = case_when(
    mileage == 0 ~ 0,
    mileage > 0 ~ ceiling(mileage / 10000) * 10000
  ))
# Filter data where price < 365,000 QAR and year > 2013.
data_p <- data_p[data_p$price < 365000, ]
data_p <- data_p[data_p$year > 2013, ]
# set set value and lower boundary threshold
price_lower_boundary<- 0.2
set.seed(123)
# remove rows with missing values in the price column
data_p <- data_p[!is.na(data_p$price) & data_p$price != "", ]
nrow(data_p)
```

```
## [1] 289
```

```r
# feature engineering by update model ,trim, year and mileage range.
data_p$model <- paste(data_p$make, data_p$model, sep = "-")
data_p$trim <- paste(data_p$make,  data_p$model, data_p$trim, data_p$cylinder, data_p$gear_type,sep = "
data_p$year <- paste(data_p$trim, data_p$year, sep = "-")
data_p$mileage_range <- paste( data_p$trim, data_p$mileage_range, sep = "-")

# Check new data received is less than the full data earlier
nrow(data)
```

```
## [1] 13560
```

```r
nrow(data_p)
```

```
## [1] 289
```

```r
# Predict Price
data2 <- data_p %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>%
    left_join(damruu, by='trim') %>%
    left_join(data_price_pmmt, by='trim') %>%
    left_join(data_avg_ymmt, by='trim') %>%
```

```
    left_join(data_avg_mileage_range, by='year') %>%
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                          pred2,
                          price_pmmt *price_lower_boundary))
# Calculate the difference between the predicted and actual price
data2<- data2 %>% mutate(price_difference = pred3 - price)
# Calculate the percentage difference based on the price
data2<- data2 %>% mutate(percentage_difference = (price_difference / price) * 100)
data2<- data2 %>% mutate(percentage_difference_updated = ifelse (percentage_difference<=0.01 & percentag
# add Year and mileage effects to the table
data2<- data2 %>% mutate(absolut_difference = abs(price_difference))
data2<- data2 %>% mutate(Year_Effect = pred1- price_pmmt)
data2<- data2 %>% mutate(Mileage_Effect = pred2- pred1)
# add some other parameters used for debugging
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pred3_pmmt = mean(pred3))
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
data2 <- data2 %>%
    group_by(trim) %>%
    mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)
```

```
# create output folder
version_folder <- "02. Linear Regression Model - Recent Data Export"

# Check if the folder already exists
if (!dir.exists(version_folder)) {
    # Create the folder
    dir.create(version_folder)
}

# Get the current date and time
current_datetime <- Sys.time()
# Create a folder name based on date and time
folder_name <- format(current_datetime, "%Y%b%d_%H-%M-%S")

# Create a folder with current_datetime inside the version folder
folder_path <- paste0(version_folder, "/", folder_name)
dir.create(folder_path)
folder_path
```

```
## [1] "02. Linear Regression Model - Recent Data Export/2024Apr18_11-54-32"
```

```
L_Recent_path<-folder_path
```

```
# export files for debugging
write.csv(data2, file = paste0(folder_path, "/_Export_full.csv"), row.names = FALSE)
# Renaming and selecting Columns to be more user friendly
data3<- data2 %>% select(
Id = web.scraper.order,
Link = car.href,
Make = make,
Model = model,
Trim= trim,
Year = year_updated,
Price= price,
Mileage =mileage,
Date_sold= date_sold,
Average_Year_per_Trim=avg_ymmt,
#Average_Milerange_per_Trim_Year= avg_mileage_range,
Average_Milerange_per_Trim_Year_LR= avg_mileage_range_updated,
Average_Price_per_Trim =price_pmmt,
Year_Effect,
Mileage_Effect,
L_Predicted_Price = pred3,
L_Price_Gap_Percentage =percentage_difference_updated,
)
# Formatting number with 2 digits and comma
data3 <- data3 %>%
   mutate(Price = format(Price, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage = format(Mileage, nsmall = 0, big.mark = ",")) %>%
   mutate(Average_Year_per_Trim = format(Average_Year_per_Trim, nsmall = 2, big.mark = ",")) %>%
   mutate(Average_Price_per_Trim = format(Average_Price_per_Trim, nsmall = 2, big.mark = ",")) %>%
   #mutate(Average_Milerange_per_Trim_Year = format(Average_Milerange_per_Trim_Year, nsmall = 2, big.ma
   mutate(Average_Milerange_per_Trim_Year_LR = format(Average_Milerange_per_Trim_Year_LR, nsmall = 2, bi
   mutate(Year_Effect = format(Year_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage_Effect = format(Mileage_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ","))
# Sequencing data by trim then by Price_Gap_Percentage
data3 <- data3 %>%
    arrange(Trim, desc(L_Price_Gap_Percentage)) %>%
    group_by(Trim) %>%
    mutate(Count = n())
# export full recommendation file for best car deals.
write.csv(data3, file = paste0(folder_path, "/_Export_Formated.csv"), row.names = FALSE)
# Filtering by high Price_Gap_Percentage > 10
data4 <- data3 %>%
    arrange(Trim, desc(L_Price_Gap_Percentage)) %>%
    filter(L_Price_Gap_Percentage > 10) #%>%
# Rearranging Column order.
data4 <- data4 %>%
select(
Id,
Link,
Make,
Model,
```

```
Trim,
Year,
Price,
Mileage,
L_Predicted_Price,
L_Price_Gap_Percentage,
Average_Year_per_Trim,
#Average_Milerange_per_Trim_Year,
Average_Milerange_per_Trim_Year_LR,
Average_Price_per_Trim,
Year_Effect,
Mileage_Effect
)
# export full recommendation file for best car deals which is sorted and filtered.
write.csv(data4, file = paste0(folder_path, "/_Export_Formated_Filtered.csv"), row.names = FALSE)
nrow(data4)
```

```
## [1] 46
```

Navigate to "02. Linear Regression Model - Recent Data Export" and find most recent folder, you will find _Export_Formated_Filtered.csv which will provide you with the best car deals. You can open the file _Export_Formated_Filtered.csv and view the car deals, all of which have much lower actual price compared to our prediction. Upon verification, we realize many of those are great deals. For wider selection, you can refer to the _Export_Formated.csv where you can finetune your search results.

The below table has a sample list of best deals. As you can see the actually price is way lower (at least 10%) than the predicted. These are a great deals. If you check one by one like I did, when you compare the same or similar trim and year you'll notice that these are the lowest prices.

```
data4<-data4 %>%
  mutate(L_Predicted_Price = round(as.numeric((gsub(",", "", L_Predicted_Price))), digits = 0)) %>%
  mutate(L_Price_Gap_Percentage = round(as.numeric((gsub(",", "", L_Price_Gap_Percentage))), digits = 0
  mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(L_Price_Gap_Percentage = format(L_Price_Gap_Percentage, nsmall = 0, big.mark = ","))

L_Recent_Table<-data4 %>%
    group_by(Trim) %>%
    slice(1) %>%
    select(Trim, Year, Mileage, Price = Price,Pred_Price= L_Predicted_Price,Gap_Perc=L_Price_Gap_Percen
    #mutate(Mileage = format(Mileage, nsmall = 2, big.mark = ",")) %>%
    #head(20) %>%
    knitr::kable()
L_Recent_Table
```

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
Audi-Audi-Q8-S-Line-6-Automatic	2023	26,000	310,000	349,234	13
Avatr-Avatr-11-Standard-0-Automatic	2024	1,200	235,000	264,696	13
BMW-BMW-6-Series-650i-8-Automatic	2015	79,000	70,000	102,649	47
BMW-BMW-8-Series-850i-8-Automatic	2019	31,000	205,000	265,174	29
BMW-BMW-X-Series-X5-6-Automatic	2023	29,000	244,000	283,174	16
Dodge-Dodge-Challenger-GT-6-Automatic	2021	39,573	99,000	111,795	13
Dodge-Dodge-Ram-Big Horn-8-Automatic	2017	108,000	72,000	82,952	15

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
Ford-Ford-F-150-6-Automatic	2016	87,000	73,000	87,322	20
Ford-Ford-Raptor-SVT-8-Automatic	2014	286,000	48,000	71,715	49
Land Rover-Land Rover-Defender-90 HSE-6-Automatic	2022	40,000	265,000	297,826	12
Land Rover-Land Rover-Evoque-Dynamic-4-Automatic	2016	152,000	55,000	65,935	20
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2015	180,000	120,000	152,171	27
Land Rover-Land Rover-Range Rover-Sport Super charged-6-Automatic	2016	145,000	95,000	108,987	15
Land Rover-Land Rover-Range Rover-Sport-6-Automatic	2017	98,000	109,000	134,270	23
Land Rover-Land Rover-Range Rover-Velar SE-4-Automatic	2019	22,000	150,000	166,124	11
Land Rover-Land Rover-Range Rover-Vogue Autobiography-8-Automatic	2014	169,000	90,000	103,776	15
Land Rover-Land Rover-Range Rover-Vogue-6-Automatic	2021	120,000	210,000	254,907	21
Land Rover-Land Rover-Range Rover-Vogue-8-Automatic	2016	64,000	139,000	160,163	15
Lexus-Lexus-LX-570 S-8-Automatic	2015	300,000	115,000	133,544	16
Lexus-Lexus-LX-570-8-Automatic	2020	85,000	315	317,458	100,680
Mercedes-Benz-Mercedes-Benz-CLA-250-4-Automatic	2023	22,000	143,000	183,669	28
Mercedes-Benz-Mercedes-Benz-GLS-500-8-Automatic	2017	119,000	119,000	133,344	12
Mercedes-Benz-Mercedes-Benz-S-Class-400-6-Automatic	2015	66,000	100,000	127,592	28
Mercedes-Benz-Mercedes-Benz-S-Class-500 AMG-8-Automatic	2015	136,000	129,999	168,144	29
Mini-Mini-Cooper-GP-4-Automatic	2021	42,000	149,000	195,778	31
Mini-Mini-Cooper-JCW-4-Automatic	2022	49,891	119,000	131,951	11
Mini-Mini-Cooper-Standard-4-Automatic	2020	67,000	62,000	86,354	39
Nissan-Nissan-Patrol-Platinum-8-Automatic	2016	165,000	79,000	105,805	34
Nissan-Nissan-Patrol-SE-8-Automatic	2016	165,000	59,000	87,134	48
Porsche-Porsche-Cayenne-GTS-8-Automatic	2014	162,000	73,000	80,364	10
Porsche-Porsche-Cayenne-S-6-Automatic	2016	140,000	90,000	132,927	48
Suzuki-Suzuki-Vitara-Standard-4-Automatic	2020	1e+05	36,500	41,076	13
Toyota-Toyota-Fortuner-Standard-4-Automatic	2020	42,000	88,000	99,343	13
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2019	130,000	118,000	158,293	34
Toyota-Toyota-Land Cruiser-Hard Top-6-Automatic	2024	3,000	174,999	196,099	12
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2015	227,000	100,000	132,651	33
Toyota-Toyota-Land Cruiser-VXS-8-Automatic	2016	157,000	157,000	184,042	17

6 Method 02 - Random Forest

6.1 Model Training

Based on the content and knowledge gained from chapter 31.11 in this link[7] I have selected this algorithm to be the most suitable with the best accuracy for my project.

I have first tried knn, however I faced errors (too many ties in knn) as there were many near by neighbor even when increasing value of k.

I have also experimented with many parameters for random forest like mtry, ntree, etc. However the results were not good documented in Train Trials.xlsx.

Finally I have used the default parameters for random forest and kept the model training for around 20 hours and the results were great, an RMSE of 13,500 vs 20,000 using my custom linear regression.

[7] https://rafalab.dfci.harvard.edu/dsbook/examples-of-algorithms.html#random-forests

I also used cross validation k =3, I could have used a higher value but that will take longer to train.

Below are optional code, just saving previous work and can be ignored.

```
# save.image(file = "Data_Load.RData") # was used to save the loaded data.
# load("Data_Load.RData")
# if (!require(dplyr)) install.packages("dplyr")
# library(dplyr)
# library(caret)
```

6.1.1 Data Preparation

```
# We are excluding car column as its empty
set.seed(123)
data_rfr <- data
data_rfr <- data_rfr[, -which(names(data_rfr) == "car")]

# Creating year2 column
data_rfr<- data_rfr %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))

# checking if there is any NA in the data
colSums(is.na(data_rfr))
```

```
##     web.scraper.order web.scraper.start.url            image.src
##                     0                     0                    0
##                  make                 model                 trim
##                     0                     0                    0
##              personal                  year            gear_type
##                     0                     0                    0
##              cylinder               mileage                price
##                     0                     0                    0
##                  time                 likes             car.href
##                     0                     0                    0
##             date_sold           posted_time        mileage_range
##                     0                     0                    0
##              avg_ymmt            price_pmmt          avg_mileage
##                     0                     0                    0
##           price_pmmty     avg_mileage_range                year2
##                     0                     0                    0
```

```
# I had first started with subset data to test the random forest, but then used full data
#data_small <- data_rfr[sample(nrow(data_rfr), 1000), ]
data_small <- data_rfr

# Selecting the columns we need to optimize the data needed for training,
data_small <- data_small%>% ungroup() %>% select(price, make, model, trim, year2, mileage)

# Used factor for the columns to be used by random forest
data_small$make <- as.factor(data_small$make)
data_small$model <- as.factor(data_small$model)
data_small$trim <- as.factor(data_small$trim)
str(data_small)
```

```
## tibble [13,560 x 6] (S3: tbl_df/tbl/data.frame)
##  $ price  : num [1:13560] 138000 275000 27000 269000 247000 87000 102000 176000 74000 280000 ...
##  $ make   : Factor w/ 77 levels "Alfa Romeo","Aston Martin",..: 72 72 58 72 72 72 72 72 72 53 ...
##  $ model  : Factor w/ 444 levels "Alfa Romeo-4 C",..: 414 414 343 414 414 412 403 416 420 300 ...
##  $ trim   : Factor w/ 1483 levels "Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic",..: 1350 1356 1185
##  $ year2  : chr [1:13560] "2017" "2021" "2016" "2021" ...
##  $ mileage: num [1:13560] 22000 0 225000 0 0 0 0 0 0 34000 ...
```

6.1.2 Hyper Parameter

We tried different combination of knn and random parameters but results were not satisfactory as documented in Train Trials.xlsx

```
#mt = "knn"
#tg = expand.grid(.k = seq(from = 1, to = 5, by = 1))
mt ="rf"
#tg = data.frame(.mtry = 5)
#tg <- expand.grid(.mtry = 5)
#tg <- expand.grid(.mtry = seq(from = 2, to = 5, by = 1), .ntree = c(500, 1000, 1500))
ctrl <- trainControl(method = "repeatedcv", number = 3, verboseIter = TRUE)  # x-fold cross-validation
```

6.1.3 Training

The below random forest training will take 20 hours on i7 laptop, alternatively you can load the saved trained model rf01.RData in the next command.

```
# rf_model <- train(price ~ make + model + trim + year2 + mileage, data = data_small, method = mt , trC
# print(rf_model$bestTune)
# predictions <- predict(rf_model, data_small)
# accuracy <- postResample(predictions, data_small$price)
# print(accuracy)
```

We have saved the Random Forest Model, so you can load it here.

```
#save.image(file = "rf01.RData")
load("rf01.RData")
```

6.2 Model Testing

6.2.1 Validation Results

Benchmarking the results and comparing with previous linear regression. As you can see the results have vastly improved from an RMSE 20,000 to 13,500.

```
rmse_results <- bind_rows(
  rmse_results,
  data_frame(Method_on_validation="Method 02 - Random Forest",
  RMSE = accuracy[[1]] ,
  Improvement = model_6_rmse - accuracy[[1]]))
rmse_results %>%
```

```
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
mutate(RMSE = round(as.numeric((gsub(",", "", RMSE))), digits = 0)) %>%
mutate(Improvement = round(as.numeric((gsub(",", "", Improvement))), digits = 0))  %>%
mutate(RMSE = format(RMSE, nsmall = 0, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 0, big.mark = ","))%>%
knitr::kable()
```

Method_on_train_test	RMSE	Improvement	Method_on_validation
Overall Average	89,466	0	NA
Make Effect	75,115	14,350	NA
Make + Model Effect	62,989	12,127	NA
Make + Model + Trim Effect	39,819	23,170	NA
Make + Model + Trim + Year Effect	22,194	17,625	NA
Make + Model + Trim + Year + Mileage Effect	19,763	2,431	NA
NA	20,225	-462	Method 01 - Linear Regression
NA	13,417	6,808	Method 02 - Random Forest

6.2.2 Predict Car Price - User input

In this section, we can predict the price of a car based on the below user input.

This has been used to test the model manually.

Create a list of a few car and its features, you can change this as you like.

```
new_data <- data.frame(make = "Toyota", model = "Land Cruiser", trim = "GXR",
                       personal = "", year = 2015,  gear_type = "Automatic", cylinder = 8, mileage = 1000
new_row <- data.frame(make = "Genesis", model = "G70", trim = "Standard",
                       personal = "", year = 2019, gear_type = "Automatic", cylinder = 6, mileage = 90000
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Nissan", model = "Patrol", trim = "SE",
                       personal = "", year = 2014, gear_type = "Automatic", cylinder = 8, mileage = 1250
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Toyota", model = "Land Cruiser", trim = "GXR",
                       personal = "", year = 2013, gear_type = "Automatic", cylinder = 6, mileage = 1750
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Chevrolet", model = "Camaro", trim = "ZL1",
                       personal = "", year = 2014, gear_type = "Automatic", cylinder = 8, mileage = 1000
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Alfa Romeo", model = "4 C", trim = "Standard",
                       personal = "", year = 2019, gear_type = "Automatic", cylinder = 4, mileage = 1000
new_data <- rbind(new_data, new_row)
new_row <- data.frame(make = "Audi", model = "A8", trim = "4.0",
                       personal = "", year = 2019, gear_type = "Automatic", cylinder = 8, mileage = 1000
new_data <- rbind(new_data, new_row)
new_data
```

```
##         make        model     trim personal year gear_type cylinder mileage
## 1     Toyota Land Cruiser      GXR          2015 Automatic        8   10000
## 2    Genesis          G70 Standard          2019 Automatic        6   90000
```

```
## 3     Nissan       Patrol       SE          2014 Automatic        8  125000
## 4     Toyota Land Cruiser      GXR          2013 Automatic        6  175000
## 5  Chevrolet       Camaro      ZL1          2014 Automatic        8   10000
## 6 Alfa Romeo          4 C Standard          2019 Automatic        4   10000
## 7       Audi           A8      4.0          2019 Automatic        8  100000
```

```
# Data wrangling by updating mileage_range, year, model and trim
new_data$mileage_range <- ifelse(new_data$mileage == 0, 0, ceiling(new_data$mileage / 10000) * 10000)
new_data$year <- paste(new_data$make, new_data$make, new_data$model, new_data$trim, new_data$year, sep =
new_data$model <- paste(new_data$make, new_data$model, sep = "-")
new_data$trim <- paste( new_data$make, new_data$model, new_data$trim,new_data$cylinder, new_data$gear_ty
new_data
```

```
##         make               model                                          trim
## 1     Toyota Toyota-Land Cruiser     Toyota-Toyota-Land Cruiser-GXR-8-Automatic
## 2    Genesis         Genesis-G70       Genesis-Genesis-G70-Standard-6-Automatic
## 3     Nissan       Nissan-Patrol            Nissan-Nissan-Patrol-SE-8-Automatic
## 4     Toyota Toyota-Land Cruiser     Toyota-Toyota-Land Cruiser-GXR-6-Automatic
## 5  Chevrolet    Chevrolet-Camaro     Chevrolet-Chevrolet-Camaro-ZL1-8-Automatic
## 6 Alfa Romeo      Alfa Romeo-4 C Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic
## 7       Audi             Audi-A8                   Audi-Audi-A8-4.0-8-Automatic
##   personal                                    year gear_type cylinder mileage
## 1              Toyota-Toyota-Land Cruiser-GXR-2015 Automatic        8   10000
## 2                Genesis-Genesis-G70-Standard-2019 Automatic        6   90000
## 3                     Nissan-Nissan-Patrol-SE-2014 Automatic        8  125000
## 4              Toyota-Toyota-Land Cruiser-GXR-2013 Automatic        6  175000
## 5              Chevrolet-Chevrolet-Camaro-ZL1-2014 Automatic        8   10000
## 6          Alfa Romeo-Alfa Romeo-4 C-Standard-2019 Automatic        4   10000
## 7                           Audi-Audi-A8-4.0-2019 Automatic        8  100000
##   mileage_range
## 1         10000
## 2         90000
## 3        130000
## 4        180000
## 5         10000
## 6         10000
## 7        100000
```

```
# predict using above linear regression model first then random forest for comparison
predicted_price_rf <- new_data %>%
  left_join(make_avgs, by='make') %>%
  left_join(model_avgs, by='model') %>%
  left_join(trim_avgs, by='trim') %>%
  left_join(year_avgs_uu, by='trim') %>%
  left_join(data_price_pmmt, by='trim') %>%
  left_join(data_avg_ymmt, by='trim') %>%
  left_join(data_avg_mileage_range, by='year') %>%
  left_join(damruu, by='trim') %>%
  mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
  mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000) =
  mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(yea
  mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeric
  mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_update
```

```
        + price_year_value
  ) %>%
  mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mil
  mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                        pred2,
                        price_pmmt *price_lower_boundary))


# add Year and mileage effects to the table
predicted_price_rf<- predicted_price_rf %>% mutate(Year_Effect = pred1- price_pmmt)
predicted_price_rf<- predicted_price_rf %>% mutate(Mileage_Effect = pred2- pred1)
# add some other parameters used for debugging
predicted_price_rf <- predicted_price_rf %>%
  group_by(trim) %>%
  mutate(pred3_pmmt = mean(pred3))
predicted_price_rf <- predicted_price_rf %>%
  group_by(trim) %>%
  mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
predicted_price_rf <- predicted_price_rf %>%
  group_by(trim) %>%
  mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)


# Create and update Columns ( years, make, model, trim)
predicted_price_rf_2<- predicted_price_rf %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)
predicted_price_rf_2$make <- as.factor(predicted_price_rf$make)
predicted_price_rf_2$model <- as.factor(predicted_price_rf$model)
predicted_price_rf_2$trim <- as.factor(predicted_price_rf$trim)


# Checking which record we need to remove,
# as random forest need to have been trained on the data
data_temp <- data
data_temp<- data_temp %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year))) %>%
  mutate (year2 = as.numeric(year2))

# check trims not trained
missing_trims_rf <- setdiff(predicted_price_rf_2$trim, data_temp$trim)
head(missing_trims_rf)
```

```
## character(0)
```

```
# remove years not trained
missing_years <- setdiff(predicted_price_rf_2$year2, data_temp$year2)
head(missing_years)
```

```
## [1] "2013"
```

```
# Remove the new untrained data
nrow(predicted_price_rf_2)
```

```
## [1] 7
```

```
predicted_price_rf_2 <- predicted_price_rf_2 %>%
  filter(trim %in% data$trim)
predicted_price_rf_2 <- predicted_price_rf_2 %>%
  filter(year2 %in% data_temp$year2)
nrow(predicted_price_rf_2)
```

```
## [1] 6
```

```
# Prepare the data that needs to be predicted
new_rf_data <- predicted_price_rf_2%>% ungroup() %>% select( make, model, trim, year2, mileage)

# Run Random forest model on new small data.
predicted_price_rf_2 <- predicted_price_rf_2 %>%
  ungroup() %>%
  mutate(predicted_price_rf_2 = predict(rf_model, newdata = new_rf_data))
```

```
# Print out the prediction
predicted_price_rf_2 %>% select(trim, year, mileage,
                                L_Pred_Price = pred3,
                                R_Pred_Price = predicted_price_rf_2
                                ) %>%
mutate(L_Pred_Price = format(L_Pred_Price, nsmall = 2, big.mark = ",")) %>%
mutate(R_Pred_Price = format(R_Pred_Price, nsmall = 2, big.mark = ",")) %>%
mutate(mileage = format(mileage, nsmall = 2, big.mark = ","))%>%
knitr::kable()
```

trim	year	mileage	L_Pred_Price	R_Pred_Price
Toyota-Toyota-Land Cruiser-GXR-8-Automatic	Toyota-Toyota-Land Cruiser-GXR-2015	10,000.00	164,967.47	268,040.16
Genesis-Genesis-G70-Standard-6-Automatic	Genesis-Genesis-G70-Standard-2019	90,000.00	73,737.78	100,003.34
Nissan-Nissan-Patrol-SE-8-Automatic	Nissan-Nissan-Patrol-SE-2014	125,000.00	62,466.69	94,102.25
Chevrolet-Chevrolet-Camaro-ZL1-8-Automatic	Chevrolet-Chevrolet-Camaro-ZL1-2014	10,000.00	94,773.15	211,693.10
Alfa Romeo-Alfa Romeo-4 C-Standard-4-Automatic	Alfa Romeo-Alfa Romeo-4 C-Standard-2019	10,000.00	280,000.00	73,578.88
Audi-Audi-A8-4.0-8-Automatic	Audi-Audi-A8-4.0-2019	100,000.00	187,106.40	77,817.38

6.2.3 Export Recommendation - Full Data

In this section, we will use the model above to predict all the prices of the full dataset (train, test, validate) and export files that provide the best recommendation car deals.

A Random Forest Full data export folder will be generated and a subfolder will be created based on run time time stamp. A few files will be created used for debugging, however there are 2 export files that are of interest.

_Export_Formated_rf.csv contains full data along with predicted price and comparison againsed actual price. You can filter by column Price_Gap_Percentage with the the highest positive value to get the best deals.

_Export_Formated_Filtered_rf.csv is the same as the earlier file, however has been sorted and filtered with the best deals per trim, with Price_Gap_Percentage_RF & Price_Gap_Percentage_RF > 20. Also having mileage less than 80,000 km and is still available for sale (Not sold).

Creating an output folder to save the random forest prediction on all data.

```
# create folder for Random Forest Folder and automatic timestamp subfolder
version_folder <- "03. Random Forest Model - Full Data Export"

# Check if the folder already exists
if (!dir.exists(version_folder)) {
    # Create the folder
    dir.create(version_folder)
}
# Get the current date and time
current_datetime <- Sys.time()
# Create a folder name based on date and time
folder_name <- format(current_datetime, "%Y%b%d_%H-%M-%S")
# Create a folder with current_datetime inside the version folder
folder_path <- paste0(version_folder, "/", folder_name)
dir.create(folder_path)
folder_path
```

```
## [1] "03. Random Forest Model - Full Data Export/2024Apr18_11-54-37"
```

```
R_Full_path<-folder_path
```

First run Linear regression prediction which will be used for comparison It is also used for analysis and interpretability.

```
data_rf_2 <- data %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>%
    left_join(damruu, by='trim') %>%
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                          pred2,
                          price_pmmt *price_lower_boundary))
#str(data_rf_2)

# Create and update Columns ( years, make, model, trim)
data_rf_2<- data_rf_2 %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))
data_rf_2$make <- as.factor(data_rf_2$make)
data_rf_2$model <- as.factor(data_rf_2$model)
```

```
data_rf_2$trim <- as.factor(data_rf_2$trim)
new_rf_data <- data_rf_2%>% ungroup() %>% select(price, make, model, trim, year2, mileage)

# Run Random forest model on all the data.
data_rf_2 <- data_rf_2 %>%
  ungroup() %>%
  mutate(predicted_price_rf = predict(rf_model, newdata = new_rf_data))

# Add columns to compare actual price vs linear regression vs random forest
data_rf_2<- data_rf_2 %>% mutate(price_difference = pred3 - price)
# Calculate the percentage difference based on the price for linear regression
data_rf_2<- data_rf_2 %>% mutate(percentage_difference = (price_difference / price) * 100)
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_updated = ifelse (percentage_difference<=0.01 & p
data_rf_2<- data_rf_2 %>% mutate(price_difference_rf = predicted_price_rf - price)
# Calculate the percentage difference based on the price for random forest
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_rf = (price_difference_rf / price) * 100)
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_updated_rf = ifelse (percentage_difference_rf<=0
# Add Year and Mileage effects based on Linear Regression model for interpretability
data_rf_2<- data_rf_2 %>% mutate(absolut_difference = abs(price_difference))
data_rf_2<- data_rf_2 %>% mutate(Year_Effect = pred1- price_pmmt)
data_rf_2<- data_rf_2 %>% mutate(Mileage_Effect = pred2- pred1)
# Other paramaters used for debugging
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pred3_pmmt = mean(pred3))
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)
# Export the full data with prediction which is still not formatted
write.csv(data_rf_2, file = paste0(folder_path, "/_Export_full_rf.csv"), row.names = FALSE)
nrow(data_rf_2)
```

```
## [1] 13560
```

```
# Rename and format the columns and export file
data_rf_3<- data_rf_2 %>% select(
Id = web.scraper.order,
Link = car.href,
Make = make,
Model = model,
Trim= trim,
Year = year_updated,
Price= price,
Mileage =mileage,
Date_sold= date_sold,
Average_Year_per_Trim=avg_ymmt,
#Average_Milerange_per_Trim_Year= avg_mileage_range,
Average_Milerange_per_Trim_Year_LR= avg_mileage_range_updated,
Average_Price_per_Trim =price_pmmt,
Year_Effect,
```

```
Mileage_Effect,
L_Predicted_Price = pred3,
L_Price_Gap_Percentage =percentage_difference_updated,
R_Predicted_Price = predicted_price_rf ,
R_Price_Gap_Percentage =percentage_difference_updated_rf
)
data_rf_3 <- data_rf_3 %>%
   mutate(Price = format(Price, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage = format(Mileage, nsmall = 0, big.mark = ",")) %>%
   mutate(Average_Year_per_Trim = format(Average_Year_per_Trim, nsmall = 2, big.mark = ",")) %>%
   mutate(Average_Price_per_Trim = format(Average_Price_per_Trim, nsmall = 2, big.mark = ",")) %>%
   #mutate(Average_Milerange_per_Trim_Year = format(Average_Milerange_per_Trim_Year, nsmall = 2, big.ma
   mutate(Average_Milerange_per_Trim_Year_LR = format(Average_Milerange_per_Trim_Year_LR, nsmall = 2, bi
   mutate(Year_Effect = format(Year_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage_Effect = format(Mileage_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
   mutate(R_Predicted_Price  = format(R_Predicted_Price , nsmall = 0, big.mark = ","))
data_rf_3 <- data_rf_3 %>%
    arrange(Trim, desc(R_Price_Gap_Percentage)) %>%
    group_by(Trim) %>%
    mutate(Count = n())
write.csv(data_rf_3, file = paste0(folder_path, "/_Export_Formated_rf.csv"), row.names = FALSE)
nrow(data_rf_3)
```

```
## [1] 13560
```

```
# Export recommendation of best car deals
data_rf_4 <- data_rf_3 %>%
    arrange(Trim, desc(R_Price_Gap_Percentage)) %>%
    filter(Count > 10) %>%
    filter(R_Price_Gap_Percentage > 20) %>%
    filter(L_Price_Gap_Percentage > 20) %>% # added
    filter(Date_sold == "") %>%
    mutate( mileage2 = (gsub(",", "", Mileage))) %>%
    mutate( mileage2 = as.numeric(mileage2)) %>%
    filter((mileage2) < 80000)

data_rf_4 <- data_rf_4 %>%
    group_by(Trim) %>%
    #arrange(Trim, desc(Price_Gap_Pecentage)) %>%
    #arrange(desc(Price_Gap_Pecentage)) %>%
    top_n(2, R_Price_Gap_Percentage)
    #top_n(if (Count/10<1,1,Count/10), Price_Gap_Pecentage)
data_rf_4 <- data_rf_4 %>%
select(
Id,
Link,
Make,
Model,
Trim,
Year,
Price,
Mileage,
```

```
L_Predicted_Price,
L_Price_Gap_Percentage,
R_Predicted_Price,
R_Price_Gap_Percentage,
Average_Year_per_Trim,
#Average_Milerange_per_Trim_Year,
Average_Milerange_per_Trim_Year_LR,
Average_Price_per_Trim,
Year_Effect,
Mileage_Effect
)
write.csv(data_rf_4, file = paste0(folder_path, "/_Export_Formated_Filtered_rf.csv"), row.names = FALSE)
nrow(data_rf_4)
```

```
## [1] 20
```

Navigate to "03. Random Forest Model - Full Data Export" and select the most recent folder, you will find _Export_Formated_Filtered.csv which will provide you with the best car deals.

You can open the file _Export_Formated_Filtered.csv and view the car deals, all of which have much lower actual price compared to our prediction. Upon verification, we realize many of those of are great deals.

If you open via excel Columns A to H are the original data, where G is the actual price and I is the linear predicted price and k is the Random forest predicted price.

The 13560 listing have been shortlisted to around 20 deals where their actual price is way lower than predicted (Random forest). This is being recommended by the system indicating a possible good deal.

The last 5 columns provides some interpretation from linear regression side. These columns show the average year, mileage and price for this trim, then year and mileage price effects. This helps us check if randmon forest is not suitable in certain cases.

For wider selection, you can refer to the _Export_Formated.csv where you have the full analysis.

The below table has a sample list of best deals. As you can see the actually price is way lower (at least 20 %) than the predicted. These are a great deals. If you check one by one like I did, when you compare the same or similar trim and year you'll notice that these are the lowest prices.

```
data_rf_4<-data_rf_4 %>%
  mutate(L_Predicted_Price = round(as.numeric((gsub(",", "", L_Predicted_Price))), digits = 0)) %>%
  mutate(R_Predicted_Price = round(as.numeric((gsub(",", "", R_Predicted_Price))), digits = 0)) %>%
  mutate(R_Price_Gap_Percentage = round(as.numeric((gsub(",", "", R_Price_Gap_Percentage))), digits = 0
  mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(R_Predicted_Price = format(R_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(R_Price_Gap_Percentage = format(R_Price_Gap_Percentage, nsmall = 0, big.mark = ","))

R_Full_Table<-data_rf_4 %>%
    #group_by(Trim) %>%
    #slice(1) %>%
    select(Trim, Year, Mileage, Price = Price,L_Price= L_Predicted_Price,R_Price= R_Predicted_Price,R_Ga
    #mutate(Mileage = format(Mileage, nsmall = 2, big.mark = ",")) %>%
    #head(20) %>%
    knitr::kable()
R_Full_Table
```

Trim	Year	Mileage	Price	L_Price	R_Price	R_Gap
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Ram-Standard-8-Automatic	2023	2,700	149,000	187,936	187,317	26
Dodge-Dodge-Ram-Standard-8-Automatic	2021	30,000	101,000	154,459	124,257	23
GMC-GMC-Sierra-1500-8-Automatic	2022	24,000	106,750	145,834	132,538	24
GMC-GMC-Sierra-AT4-8-Automatic	2021	43,000	119,000	154,424	144,495	21
GMC-GMC-Sierra-Elevation-8-Automatic	2021	18,000	98,000	158,350	118,173	21
Infiniti-Infiniti-QX-80-8-Automatic	2017	36,000	110,000	138,144	134,897	23
Mercedes-Benz-Mercedes-Benz-E-Class-400-6-Automatic	2016	34,000	82,000	101,554	111,703	36
Mini-Mini-Cooper-S-4-Automatic	2020	34,000	65,000	93,814	80,338	24
Nissan-Nissan-Navara-Standard-4-Automatic	2017	36,751	35,995	49,329	48,792	36
Suzuki-Suzuki-Jimny-Standard-4-Automatic	2018	16,000	39,000	50,227	52,182	34
Toyota-Toyota-Fortuner-Standard-4-Automatic	2015	13,000	55,000	67,465	81,206	48
Toyota-Toyota-Hilux-Standard-4-Automatic	2024	0	92,000	110,918	112,606	22
Toyota-Toyota-Hilux-Standard-4-Manual	2023	0	69,000	94,529	95,297	38
Toyota-Toyota-Hilux-Standard-4-Manual	2024	0	80,000	98,909	102,755	28
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	249,000	312,285	312,716	26
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	259,000	312,285	312,716	21

6.2.4 Export Recommendation - Recent Data

In this section we will run the same model above on fresh new data web scrapped. The system will then automatically provide recommendation of which cars are the best deals. This is a powerful and useful tool.

Optional to save old data in case needed

```
data_rf_2_old<-data_rf_2
data_p_old<- data_p
data_p<- data_p %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))
```

We are using the same dataset data_p extracted earlier from "pred09Apr202411am.csv"

```
nrow(data_p)
```

```
## [1] 289
```

Checking which record we need to remove, as random forest need to have been trained on the data You might need to remove more columns if rf is failing, like year in future. Here missing was new types of trims which are exported so that use is aware.

```
missing_trims <- setdiff(data_p$trim, data$trim)
head(missing_trims)
```

```
## [1] "Bentley-Bentley-Mulsanne-Standard-12-Automatic"
## [2] "Hyundai-Hyundai-Azera-Standard-4-Automatic"
```

```
## [3] "Rox Motor-Rox Motor-Jishi-01-6-Automatic"
## [4] "Land Rover-Land Rover-Defender-90 X Dynamic-4-Automatic"
## [5] "Toyota-Toyota-Crown-Hybrid-0-Automatic"
## [6] "Lexus-Lexus-RX-350 F Sport-6-Automatic"
```

Remove the new untrained data

```
data_p <- data_p %>%
    filter(trim %in% data$trim)
nrow(data_p)
```

```
## [1] 281
```

```
# Optional to remove un acceptable data like 2013 although filtered
# data_temp <- data
# data_temp<- data_temp %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year))) %>%
#   mutate (year2 = as.numeric(year2))
# data_p <- data_p %>%
#   filter(year2 %in% data_temp$year2)
# nrow(data_p)
```

Predict using Linear Regression & Random for comparison

```
data_rf_2 <- data_p %>%
    left_join(make_avgs, by='make') %>%
    left_join(model_avgs, by='model') %>%
    left_join(trim_avgs, by='trim') %>%
    left_join(year_avgs_uu, by='trim') %>%
    left_join(damruu, by='trim') %>%
    left_join(data_price_pmmt, by='trim') %>% # added
    left_join(data_avg_ymmt, by='trim') %>% # added
    left_join(data_avg_mileage_range, by='year') %>% # added
    mutate(year_updated = substr(year, nchar(year) - 3, nchar(year))) %>%
    mutate(mileage_range_updated = case_when( mileage == 0 ~ 0,   mileage > 0 ~ ceiling(mileage / 10000)
    mutate(price_year_value = ifelse (b==0, (as.numeric(price_pmmt)* as.numeric(year_v) * (as.numeric(y
    mutate(avg_mileage_range_updated = (ifelse ((aa + bb* as.numeric(year_updated))>0,aa + bb* as.numeri
    mutate(pred1 = mu + b_make + b_model + b_trim + (as.numeric(a) + as.numeric(b)*as.numeric(year_upda
    + price_year_value
    ) %>%
    mutate(pred2 = pred1 + (mileage_v) * pred1 * (as.numeric(avg_mileage_range_updated ) - as.numeric(mi
    mutate(pred3 = ifelse((pred2) >= (price_pmmt *price_lower_boundary),
                          pred2,
                          price_pmmt *price_lower_boundary))

# Update columns year2, make, model, and trim
data_rf_2<- data_rf_2 %>% mutate (year2 = substr(year, nchar(year) - 3, nchar(year)))
data_rf_2$make <- as.factor(data_rf_2$make)
data_rf_2$model <- as.factor(data_rf_2$model)
data_rf_2$trim <- as.factor(data_rf_2$trim)
new_rf_data <- data_rf_2%>% ungroup() %>% select(price, make, model, trim, year2, mileage)
#head(new_rf_data )
```

```r
# Run Random Forest Prediction
data_rf_2 <- data_rf_2 %>%
  ungroup() %>%
  mutate(predicted_price_rf = predict(rf_model, newdata = new_rf_data))

# Calculate gap between actual price, linear regression and random forest price prediction

data_rf_2<- data_rf_2 %>% mutate(price_difference = pred3 - price)
# Calculate the percentage difference based on the price Linear Regression
data_rf_2<- data_rf_2 %>% mutate(percentage_difference = (price_difference / price) * 100)
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_updated = ifelse (percentage_difference<=0.01 & p
data_rf_2<- data_rf_2 %>% mutate(price_difference_rf = predicted_price_rf - price)
# Calculate the percentage difference based on the price Random Forest
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_rf = (price_difference_rf / price) * 100)
data_rf_2<- data_rf_2 %>% mutate(percentage_difference_updated_rf = ifelse (percentage_difference_rf<=0
# Add Year and Mileage effect for interpretability
data_rf_2<- data_rf_2 %>% mutate(absolut_difference = abs(price_difference))
data_rf_2<- data_rf_2 %>% mutate(Year_Effect = pred1- price_pmmt)
data_rf_2<- data_rf_2 %>% mutate(Mileage_Effect = pred2- pred1)
# Other parameters for debugging
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pred3_pmmt = mean(pred3))
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pmmt_difference = (price_pmmt - pred3_pmmt))
data_rf_2 <- data_rf_2 %>%
    group_by(trim) %>%
    mutate(pmmt_absolute = (price_pmmt - pred3_pmmt)/price_pmmt*100)
```

Creating output folder

```r
version_folder <- "04. Random Forest Model - Recent Data Export"
# Check if the folder already exists
if (!dir.exists(version_folder)) {
    # Create the folder
    dir.create(version_folder)
}
# Get the current date and time
current_datetime <- Sys.time()
# Create a folder name based on date and time
folder_name <- format(current_datetime, "%Y%b%d_%H-%M-%S")
# Create a folder with current_datetime inside the version folder
folder_path <- paste0(version_folder, "/", folder_name)
dir.create(folder_path)
folder_path
```

```
## [1] "04. Random Forest Model - Recent Data Export/2024Apr18_11-55-19"
```

```r
R_Recent_path<-folder_path
```

```r
# Export full data un formatted with random forest prediction
write.csv(data_rf_2, file = paste0(folder_path, "/_Export_full_rf.csv"), row.names = FALSE)
nrow(data_rf_2)
```

```
## [1] 281
```

```r
# Export Car listing trims that were not inferred as the model was not trained on it.
write.csv(missing_trims, file = paste0(folder_path, "/_Export_missing.csv"), row.names = FALSE)
length(missing_trims)
```

```
## [1] 7
```

Format and Select Columns for the full data predicted

```r
data_rf_3<- data_rf_2 %>% select(
Id = web.scraper.order,
Link = car.href,
Make = make,
Model = model,
Trim= trim,
Year = year_updated,
Price= price,
Mileage =mileage,
Date_sold= date_sold,
Average_Year_per_Trim=avg_ymmt,
#Average_Milerange_per_Trim_Year= avg_mileage_range,
Average_Milerange_per_Trim_Year_LR= avg_mileage_range_updated,
Average_Price_per_Trim =price_pmmt,
Year_Effect,
Mileage_Effect,
L_Predicted_Price = pred3,
L_Price_Gap_Percentage =percentage_difference_updated,
R_Predicted_Price = predicted_price_rf ,
R_Price_Gap_Percentage =percentage_difference_updated_rf
)
data_rf_3 <- data_rf_3 %>%
   mutate(Price = format(Price, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage = format(Mileage, nsmall = 0, big.mark = ",")) %>%
   mutate(Average_Year_per_Trim = format(Average_Year_per_Trim, nsmall = 2, big.mark = ",")) %>%
   mutate(Average_Price_per_Trim = format(Average_Price_per_Trim, nsmall = 2, big.mark = ",")) %>%
   #mutate(Average_Milerange_per_Trim_Year = format(Average_Milerange_per_Trim_Year, nsmall = 2, big.ma
   mutate(Average_Milerange_per_Trim_Year_LR = format(Average_Milerange_per_Trim_Year_LR, nsmall = 2, bi
   mutate(Year_Effect = format(Year_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(Mileage_Effect = format(Mileage_Effect, nsmall = 0, big.mark = ",")) %>%
   mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
   mutate(R_Predicted_Price  = format(R_Predicted_Price , nsmall = 0, big.mark = ","))
data_rf_3 <- data_rf_3 %>%
    arrange(Trim, desc(R_Price_Gap_Percentage)) %>%
    group_by(Trim) %>%
    mutate(Count = n())
write.csv(data_rf_3, file = paste0(folder_path, "/_Export_Formated_rf.csv"), row.names = FALSE)
nrow(data_rf_3)
```

```
## [1] 281
```

Export the best deal recommendations, where random forest prediction is 10% higher than actual and linear regression is 10% higher than actual price.

```
data_rf_4 <- data_rf_3 %>%
    arrange(Trim, desc(R_Price_Gap_Percentage)) %>%
    #filter(Count > 10) %>%
    filter(R_Price_Gap_Percentage > 10) %>%
    filter(L_Price_Gap_Percentage > 10) #%>%
    # filter(Date_sold == "") %>%
    # mutate( mileage2 = (gsub(",", "", Mileage))) %>%
    # mutate( mileage2 = as.numeric(mileage2)) %>%
    # filter((mileage2) < 80000)

# data_rf_4 <- data_rf_4 %>%
#     group_by(Trim) %>%
#     #arrange(Trim, desc(Price_Gap_Pecentage)) %>%
#     #arrange(desc(Price_Gap_Pecentage)) %>%
#     top_n(2, Price_Gap_Percentage_RF)
#     #top_n(if (Count/10<1,1,Count/10), Price_Gap_Pecentage)
data_rf_4 <- data_rf_4 %>%
select(
Id,
Link,
Make,
Model,
Trim,
Year,
Price,
Mileage,
L_Predicted_Price,
L_Price_Gap_Percentage,
R_Predicted_Price,
R_Price_Gap_Percentage,
Average_Year_per_Trim,
#Average_Milerange_per_Trim_Year,
Average_Milerange_per_Trim_Year_LR,
Average_Price_per_Trim,
Year_Effect,
Mileage_Effect
)
write.csv(data_rf_4, file = paste0(folder_path, "/_Export_Formated_Filtered_rf.csv"), row.names = FALSE)
nrow(data_rf_4)
```

```
## [1] 20
```

Navigate to "04. Random Forest Model - Recent Data Export" and find most recent folder, you will find _Export_Formated_Filtered.csv which will provide you with the best car deals. You can open the file _Export_Formated_Filtered.csv and view the car deals, all of which have much lower actual price compared to our prediction. Upon verification, we realize many of those are great deals. For wider selection, you can refer to the _Export_Formated.csv where you can finetune your search results.

The below table has a sample list of best deals. As you can see the actually price is way lower (at least 10%) than the predicted. These are a great deals. If you check one by one like I did, when you compare the same or similar trim and year you'll notice that these are the lowest prices.

```
data_rf_4<-data_rf_4 %>%
  mutate(L_Predicted_Price = round(as.numeric((gsub(",", "", L_Predicted_Price))), digits = 0)) %>%
  mutate(R_Predicted_Price = round(as.numeric((gsub(",", "", R_Predicted_Price))), digits = 0)) %>%
  mutate(R_Price_Gap_Percentage = round(as.numeric((gsub(",", "", R_Price_Gap_Percentage))), digits = 0
  mutate(L_Predicted_Price = format(L_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(R_Predicted_Price = format(R_Predicted_Price, nsmall = 0, big.mark = ",")) %>%
  mutate(R_Price_Gap_Percentage = format(R_Price_Gap_Percentage, nsmall = 0, big.mark = ","))

R_Recent_Table<-data_rf_4 %>%
    #group_by(Trim) %>%
    #slice(1) %>%
    select(Trim, Year, Mileage, Price = Price,L_Price= L_Predicted_Price,R_Price= R_Predicted_Price,R_Ga
    #mutate(Mileage = format(Mileage, nsmall = 2, big.mark = ",")) %>%
    #head(20) %>%
    knitr::kable()
R_Recent_Table
```

Trim	Year	Mileage	Price	L_Price	R_Price	R_Gap
BMW-BMW-8-Series-850i-8-Automatic	2019	31,000	205,000	265,174	231,118	13
Ford-Ford-F-150-6-Automatic	2016	87,000	73,000	87,322	111,152	52
Ford-Ford-Raptor-SVT-8-Automatic	2014	286,000	48,000	71,715	65,119	36
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2016	89,000	165,000	207,684	187,717	14
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2015	180,000	120,000	152,171	132,613	11
Land Rover-Land Rover-Range Rover-Velar SE-4-Automatic	2019	22,000	150,000	166,124	290,988	94
Land Rover-Land Rover-Range Rover-Vogue-8-Automatic	2016	64,000	139,000	160,163	157,608	13
Land Rover-Land Rover-Range Rover-Vogue Autobiography-8-Automatic	2014	169,000	90,000	103,776	117,699	31
Lexus-Lexus-LX-570-8-Automatic	2020	85,000	315	317,458	303,516	96,254
Lexus-Lexus-LX-570 S-8-Automatic	2018	92,000	235,000	261,192	259,071	10
Mercedes-Benz-Mercedes-Benz-CLA-250-4-Automatic	2023	22,000	143,000	183,669	182,790	28
Mercedes-Benz-Mercedes-Benz-S-Class-400-6-Automatic	2015	66,000	100,000	127,592	135,098	35
Mini-Mini-Cooper-Standard-4-Automatic	2020	67,000	62,000	86,354	71,304	15
Nissan-Nissan-Patrol-Platinum-8-Automatic	2016	165,000	79,000	105,805	101,961	29
Nissan-Nissan-Patrol-SE-8-Automatic	2016	165,000	59,000	87,134	78,818	34
Porsche-Porsche-Cayenne-GTS-8-Automatic	2014	162,000	73,000	80,364	80,464	10
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2019	130,000	118,000	158,293	132,214	12
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2016	225,000	132,000	148,258	159,185	21
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2015	227,000	100,000	132,651	113,760	14
Toyota-Toyota-Land Cruiser-VXS-8-Automatic	2016	157,000	157,000	184,042	184,262	17

7 Summary Results

In this section, the results of both methods linear and random will be summarized and compared.

7.1 Price Prediction

Let's first start with results of the price prediction using RMSE as a means to evaluate. In the next section, we will summaries the results of the recommendations.

The RMSE using the overall mean is around 89,466 QAR (~ 25,000 USD) for cars up to 365,000 QAR (~ 100,000 USD). This was based on the filter selection process.

7.1.1 Method 01 - Linear Regression

Using the first Method, Linear Regression, we derived bias effects for car make, model and trim including the gear_type and cylinder similar to what we have learned from our Data Science chapter "Recommendation Systems". This has reduced the RMSE more than half at 39,819 QAR (~ 11,000 USD).

In attempt to further improve the RMSE, we utilized different methods within the area of linear regression specifically for the year and mileage parameters. This became a necessity as there aren't enough records for all years and mileage for each car trim. We used then researched and used linear regression formula from this site[8]. For more details please refer to section 5.1.5.1

This allowed us to create the bias for any year for a specific trim. This is particularly useful when predicting price of trims that dont have a year they were trained on. For mileage, we used a similar technique but we introduced a weightage component for the mileage effect. We adjusted this component till we couldn't not improve RMSE. This allowed us to further reduced RMSE to 20,225 QAR (~ 5,500 USD).

7.1.2 Method 02 - Random Forest

For the second method, we reviewed our course material and referred to this chapter[9]. We first explored knn but it was not suitable due to an error of many similar neighbors, although it did work on 3000 records and was giving impressive results. We couldn't train it on the full dataset.

We knew from the beginning we will be using random forest as an advance method branching from decision trees. We initially tried different combinations parameters documented in "Train Trials.xlsx" available in the root folder however we could not improve RMSE till we finally used the default setting. This however took 20 hours on an i7 13 generation laptop.

We only used 3 fold cross validation, again with default settings and achieved an improved RMSE of 13,417 QAR (~ 3,500 USD). At this stage, I was very satisfied.

7.1.3 Overall

Please find the summary results of both methods shown below.

```
rmse_results %>%
mutate(RMSE = format(RMSE, nsmall = 2, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 2, big.mark = ",")) %>%
mutate(RMSE = round(as.numeric((gsub(",", "", RMSE))), digits = 0)) %>%
mutate(Improvement = round(as.numeric((gsub(",", "", Improvement))), digits = 0))  %>%
mutate(RMSE = format(RMSE, nsmall = 0, big.mark = ",")) %>%
mutate(Improvement = format(Improvement, nsmall = 0, big.mark = ","))%>%
knitr::kable()
```

[8] https://www.vedantu.com/formula/linear-regression-formula
[9] https://rafalab.dfci.harvard.edu/dsbook/examples-of-algorithms.html

Method_on_train_test	RMSE	Improvement	Method_on_validation
Overall Average	89,466	0	NA
Make Effect	75,115	14,350	NA
Make + Model Effect	62,989	12,127	NA
Make + Model + Trim Effect	39,819	23,170	NA
Make + Model + Trim + Year Effect	22,194	17,625	NA
Make + Model + Trim + Year + Mileage Effect	19,763	2,431	NA
NA	20,225	-462	Method 01 - Linear Regression
NA	13,417	6,808	Method 02 - Random Forest

The left hand side RMSE is trained and tested on the train and test sets, where as the right hand side RMSE is based on the validation test for linear and k fold cross validation on Random Forest.

Although the second method improved the RMSE, but it lacks interpretability, hence I have kept both predictions in the recommendation for analysis and comparison as you will see in the recommendation section. This way we can filter out any prediction that the Random forest that seems unusual.

7.2 Car Deal Recommendation

Both methods (Linear and Random) were used to predict the prices of vehicles and using filters and sorting we can extract the best car deals that are way lower than their predicted price. The comparison between the actual price and predicted was with both Linear predicted price and random forest predicted price. This is to ensure we have better deals and also that is interpretable. As the linear allows you to view the make, model, trim, year and mileage effect, thus understanding how this car is placed in the price range.

The system will provide recommendations by exporting csv files into the below directories which contains links to the car webpage for further viewing.

Each directory has its own purpose, explained in the following sections.

1. Linear Regression Model - Full Data Export
2. Linear Regression Model - Recent Data Export
3. Random Forest Model - Full Data Export
4. Random Forest Model - Recent Data Export

7.2.1 01. Linear Regression Model - Full Data Export

The first folder contains Recommendations based on the first Model "Linear Regression" and the data used was the full data 13,500 Records. This provides the end user with suggested car deals that he should consider that are much lower than market price. In this use case, 90 car deals have been recommended.

You can open the excel file "_Export_Formated_Filtered.csv" from this folder below. Please note everytime you run the model, you will have a new subfolder with a time stamp for that run.

Folder path:

```
L_Full_path
```

```
## [1] "01. Linear Regression Model - Full Data Export/2024Apr18_11-54-03"
```

The below table show the same list in the excel file however it has been trimmed in the columns and the rows to fit. You can still access the full file in the above location. Note: for any column that has "L_" it denotes linear.

```
L_Full_Table
```

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
BMW-BMW-7-Series-730 Li-6-Automatic	2019	73,700	120,000	159,320	33
BMW-BMW-7-Series-740 Li-6-Automatic	2020	24,000	185,000	229,251	24
BMW-BMW-X-Series-X6-6-Automatic	2016	70,000	80,000	99,988	25
BMW-BMW-X-Series-X6-8-Automatic	2016	62,000	89,000	116,843	31
Cadillac-Cadillac-Escalade-Platinum-8-Automatic	2019	79,000	149,000	204,423	37
Cadillac-Cadillac-Escalade-Standard-8-Automatic	2020	77,000	159,000	225,965	42
Chevrolet-Chevrolet-Silverado-LT-8-Automatic	2021	64,000	112,000	138,801	24
Chevrolet-Chevrolet-Silverado-RST-8-Automatic	2020	77,000	94,000	125,492	34
Chevrolet-Chevrolet-Silverado-Standard-8-Automatic	2021	50,000	89,000	116,040	30
Chevrolet-Chevrolet-Silverado-Trail Boss-8-Automatic	2021	27,000	116,000	139,637	20
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	35
Dodge-Dodge-Charger-RT-8-Automatic	2021	42,000	90,000	119,800	33
Dodge-Dodge-Ram-Standard-8-Automatic	2021	30,000	101,000	154,459	53
Fiat-Fiat-595-Abarth-4-Automatic	2022	18,000	70,000	88,865	27
Ford-Ford-F-150-6-Automatic	2023	1,000	195,000	269,775	38
Ford-Ford-Mustang-GT-8-Automatic	2021	28,000	105,000	134,911	28
GMC-GMC-Sierra-1500-8-Automatic	2022	24,000	106,750	145,834	37
GMC-GMC-Sierra-2500 HD-8-Automatic	2021	28,000	149,000	181,072	22
GMC-GMC-Sierra-AT4-8-Automatic	2021	43,000	119,000	154,424	30
GMC-GMC-Sierra-Elevation-8-Automatic	2021	18,000	98,000	158,350	62
GMC-GMC-Yukon-Standard-8-Automatic	2019	64,000	123,000	159,331	30
Geely-Geely-Coolray-Standard-3-Automatic	2022	18,000	47,000	56,511	20
Hyundai-Hyundai-Staria-Standard-6-Automatic	2023	24,517	91,000	130,702	44
Infiniti-Infiniti-QX-80-8-Automatic	2020	55,000	145,000	198,102	37
Jeep-Jeep-Wrangler-Sahara-6-Automatic	2018	75,000	89,000	110,093	24
Kia-Kia-Picanto-Standard-4-Automatic	2022	36,000	33,000	39,843	21
Kia-Kia-Sportage-Standard-4-Automatic	2020	68,000	58,000	70,548	22
Land Rover-Land Rover-Evoque-Standard-4-Automatic	2020	42,000	144,000	181,170	26
Land Rover-Land Rover-Range Rover-Sport Super charged-8-Automatic	2016	68,000	117,000	146,029	25
Land Rover-Land Rover-Range Rover-Vogue SE Super charged-8-Automatic	2017	79,300	135,000	202,081	50
Land Rover-Land Rover-Range Rover-Vogue SE-8-Automatic	2020	54,000	185,000	277,907	50
Lexus-Lexus-LX-570-8-Automatic	2017	33,000	197,000	265,688	35
Lexus-Lexus-RX-350-6-Automatic	2020	33,000	130,000	168,147	29
Mercedes-Benz-Mercedes-Benz-E-Class-200-4-Automatic	2016	78,000	59,000	73,818	25
Mercedes-Benz-Mercedes-Benz-E-Class-400-6-Automatic	2016	34,000	82,000	101,554	24
Mercedes-Benz-Mercedes-Benz-S-Class-450-6-Automatic	2020	55,000	195,000	268,259	38
Mini-Mini-Cooper-JCW-4-Automatic	2023	73,000	100,000	135,227	35
Mini-Mini-Cooper-S-4-Automatic	2020	34,000	65,000	93,814	44
Mini-Mini-Cooper-Standard-4-Automatic	2020	54,000	73,000	87,795	20
Mitsubishi-Mitsubishi-Eclipse-Cross Highline-4-Automatic	2022	0	69,500	88,664	28
Mitsubishi-Mitsubishi-Fuso Canter-Standard-4-Manual	2020	0	83,000	102,035	23
Nissan-Nissan-Navara-Standard-4-Automatic	2017	36,751	35,995	49,329	37

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
Nissan-Nissan-Patrol-Super Safari-6-Automatic	2022	28,000	154,000	187,007	21
Nissan-Nissan-Sunny-Standard-4-Automatic	2020	78,000	27,500	34,854	27
Suzuki-Suzuki-Jimny-Standard-4-Automatic	2018	16,000	39,000	50,227	29
Suzuki-Suzuki-Jimny-Standard-4-Manual	2023	0	65,000	82,035	26
Toyota-Toyota-Fortuner-Standard-4-Automatic	2015	13,000	55,000	67,465	23
Toyota-Toyota-Hilux-SR5-4-Automatic	2019	54,000	65,000	83,343	28
Toyota-Toyota-Hilux-Standard-4-Automatic	2020	37,000	68,000	88,409	30
Toyota-Toyota-Hilux-Standard-4-Manual	2023	0	69,000	94,529	37
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2021	51,000	149,000	194,185	30
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	249,000	312,285	25
Toyota-Toyota-Land Cruiser-GXR-6-Automatic	2021	25,000	179,000	234,238	31
Toyota-Toyota-Land Cruiser-Hard Top-6-Manual	2021	0	103,000	134,732	31
Toyota-Toyota-Land Cruiser-LX-6-Manual	2021	0	110,000	137,700	25
Toyota-Toyota-Land Cruiser-LX-8-Manual	2023	0	136,000	164,240	21
Toyota-Toyota-Land Cruiser-VX Twin Turbo-6-Automatic	2023	39,000	255,000	314,842	23

In this same folder, you also find: "_Export_Formated.csv", this contains the priced prediction in column "L_Predicted_Price" along with how far is the prediction from the actual in column "L_Price_Gap_Percentage". This file provides a wider selection where user can filter based on trim, year to find the best deals by filtering the highest value of "L_Price_Gap_Percentage". The higher value of "L_Price_Gap_Percentage" means the predicted value is that many percent higher than actual price, indicating actual price is quite low.

The third file in this folder is more for debugging and contains other parameters.

7.2.2 02. Linear Regression Model - Recent Data Export

The second folder contains Recommendations based on the first Model "Linear Regression" and the data used was a recent data of 360 Records (scrapped on 9th Apr 2024). This provides the end user with suggested car deals that he should consider that are much lower than market price. In this use case, 46 car deals have been recommended.

You can open the excel file "_Export_Formated_Filtered.csv" from this folder below. Please note everytime you run the model, you will have a new subfolder with a time stamp for that run.

Folder path:

```
L_Recent_path
```

```
## [1] "02. Linear Regression Model - Recent Data Export/2024Apr18_11-54-32"
```

The below table show the same list in the excel file however it has been trimmed in the columns and the rows to fit. You can still access the full file in the above location. Note: for any column that has "L_" it denotes linear.

```
L_Recent_Table
```

Trim	Year	Mileage	Price	Pred_Price	Gap_Perc
Audi-Audi-Q8-S-Line-6-Automatic	2023	26,000	310,000	349,234	13
Avatr-Avatr-11-Standard-0-Automatic	2024	1,200	235,000	264,696	13
BMW-BMW-6-Series-650i-8-Automatic	2015	79,000	70,000	102,649	47
BMW-BMW-8-Series-850i-8-Automatic	2019	31,000	205,000	265,174	29
BMW-BMW-X-Series-X5-6-Automatic	2023	29,000	244,000	283,174	16
Dodge-Dodge-Challenger-GT-6-Automatic	2021	39,573	99,000	111,795	13
Dodge-Dodge-Ram-Big Horn-8-Automatic	2017	108,000	72,000	82,952	15
Ford-Ford-F-150-6-Automatic	2016	87,000	73,000	87,322	20
Ford-Ford-Raptor-SVT-8-Automatic	2014	286,000	48,000	71,715	49
Land Rover-Land Rover-Defender-90 HSE-6-Automatic	2022	40,000	265,000	297,826	12
Land Rover-Land Rover-Evoque-Dynamic-4-Automatic	2016	152,000	55,000	65,935	20
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2015	180,000	120,000	152,171	27
Land Rover-Land Rover-Range Rover-Sport Super charged-6-Automatic	2016	145,000	95,000	108,987	15
Land Rover-Land Rover-Range Rover-Sport-6-Automatic	2017	98,000	109,000	134,270	23
Land Rover-Land Rover-Range Rover-Velar SE-4-Automatic	2019	22,000	150,000	166,124	11
Land Rover-Land Rover-Range Rover-Vogue Autobiography-8-Automatic	2014	169,000	90,000	103,776	15
Land Rover-Land Rover-Range Rover-Vogue-6-Automatic	2021	120,000	210,000	254,907	21
Land Rover-Land Rover-Range Rover-Vogue-8-Automatic	2016	64,000	139,000	160,163	15
Lexus-Lexus-LX-570 S-8-Automatic	2015	300,000	115,000	133,544	16
Lexus-Lexus-LX-570-8-Automatic	2020	85,000	315	317,458	100,680
Mercedes-Benz-Mercedes-Benz-CLA-250-4-Automatic	2023	22,000	143,000	183,669	28
Mercedes-Benz-Mercedes-Benz-GLS-500-8-Automatic	2017	119,000	119,000	133,344	12
Mercedes-Benz-Mercedes-Benz-S-Class-400-6-Automatic	2015	66,000	100,000	127,592	28
Mercedes-Benz-Mercedes-Benz-S-Class-500 AMG-8-Automatic	2015	136,000	129,999	168,144	29
Mini-Mini-Cooper-GP-4-Automatic	2021	42,000	149,000	195,778	31
Mini-Mini-Cooper-JCW-4-Automatic	2022	49,891	119,000	131,951	11
Mini-Mini-Cooper-Standard-4-Automatic	2020	67,000	62,000	86,354	39
Nissan-Nissan-Patrol-Platinum-8-Automatic	2016	165,000	79,000	105,805	34
Nissan-Nissan-Patrol-SE-8-Automatic	2016	165,000	59,000	87,134	48
Porsche-Porsche-Cayenne-GTS-8-Automatic	2014	162,000	73,000	80,364	10
Porsche-Porsche-Cayenne-S-6-Automatic	2016	140,000	90,000	132,927	48
Suzuki-Suzuki-Vitara-Standard-4-Automatic	2020	1e+05	36,500	41,076	13
Toyota-Toyota-Fortuner-Standard-4-Automatic	2020	42,000	88,000	99,343	13
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2019	130,000	118,000	158,293	34
Toyota-Toyota-Land Cruiser-Hard Top-6-Automatic	2024	3,000	174,999	196,099	12
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2015	227,000	100,000	132,651	33
Toyota-Toyota-Land Cruiser-VXS-8-Automatic	2016	157,000	157,000	184,042	17

In this same folder, you also find: "_Export_Formated.csv", this contains the priced prediction in column "L_Predicted_Price" along with how far is the prediction from the actual in column "L_Price_Gap_Percentage". This file provides a wider selection where user can filter based on trim, year to find the best deals by filtering the highest value of "L_Price_Gap_Percentage". The higher value of "L_Price_Gap_Percentage" means the predicted value is that many percent higher than actual price, indicating actual price is quite low.

The third file in this folder is more for debugging and contains other parameters.

7.2.3 03. Random Forest Model - Full Data Export

The third folder contains Recommendations based on the second Model "Random Forest" but it also includes "Linear Regression" for comparison. The data that was used is the full data 13,500 Records. This provides the end user with suggested car deals that he should consider that are much lower than market price. In this use case, 20 car deals have been recommended.

You can open the excel file "_Export_Formated_Filtered_rf.csv" from this folder below. Please note everytime you run the model, you will have a new subfolder with a time stamp for that run.

Folder path:

```
R_Full_path
```

```
## [1] "03. Random Forest Model - Full Data Export/2024Apr18_11-54-37"
```

The below table show the same list in the excel file however it has been trimmed in the columns and the rows to fit. You can still access the full file in the above location. Note: for any column that has "L_" it denotes linear and "R_" denotes Random Forest.

```
R_Full_Table
```

Trim	Year	Mileage	Price	L_Price	R_Price	R_Gap
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Charger-R/T Plus-8-Automatic	2023	0	159,999	216,110	216,500	35
Dodge-Dodge-Ram-Standard-8-Automatic	2023	2,700	149,000	187,936	187,317	26
Dodge-Dodge-Ram-Standard-8-Automatic	2021	30,000	101,000	154,459	124,257	23
GMC-GMC-Sierra-1500-8-Automatic	2022	24,000	106,750	145,834	132,538	24
GMC-GMC-Sierra-AT4-8-Automatic	2021	43,000	119,000	154,424	144,495	21
GMC-GMC-Sierra-Elevation-8-Automatic	2021	18,000	98,000	158,350	118,173	21
Infiniti-Infiniti-QX-80-8-Automatic	2017	36,000	110,000	138,144	134,897	23
Mercedes-Benz-Mercedes-Benz-E-Class-400-6-Automatic	2016	34,000	82,000	101,554	111,703	36
Mini-Mini-Cooper-S-4-Automatic	2020	34,000	65,000	93,814	80,338	24
Nissan-Nissan-Navara-Standard-4-Automatic	2017	36,751	35,995	49,329	48,792	36
Suzuki-Suzuki-Jimny-Standard-4-Automatic	2018	16,000	39,000	50,227	52,182	34
Toyota-Toyota-Fortuner-Standard-4-Automatic	2015	13,000	55,000	67,465	81,206	48
Toyota-Toyota-Hilux-Standard-4-Automatic	2024	0	92,000	110,918	112,606	22
Toyota-Toyota-Hilux-Standard-4-Manual	2023	0	69,000	94,529	95,297	38
Toyota-Toyota-Hilux-Standard-4-Manual	2024	0	80,000	98,909	102,755	28
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	249,000	312,285	312,716	26
Toyota-Toyota-Land Cruiser-GXR Twin Turbo-6-Automatic	2024	0	259,000	312,285	312,716	21

In this same folder, you also find: "_Export_Formated_rf.csv", this contains the priced prediction in column "L_Predicted_Price" & "R_Predicted_Price" along with how far is the prediction from the actual in column "L_Price_Gap_Percentage" & "R_Price_Gap_Percentage". This file provides a wider selection where user can filter based on trim, year to find the best deals by filtering the highest value of

"L_Price_Gap_Percentage". The higher value of "L_Price_Gap_Percentage" means the predicted value is that many percent higher than actual price, indicating actual price is quite low.

The "_Export_full_rf.csv" in this folder is more for debugging and contains other parameters. The "_Export_missing.csv" in this folder contains the cars which the model could not predict as these are new trims that the model have never seen.

7.2.4 04. Random Forest Model - Recent Data Export

The forth folder contains Recommendations based on the second Model "Random Forest" but it also includes "Linear Regression" for comparison. The data that used was a recent containing 360 Records (scrapped on 9th Apr 2024). This provides the end user with suggested car deals that he should consider that are much lower than market price. In this use case, 20 car deals have been recommended.

You can open the excel file "_Export_Formated_Filtered_rf.csv" from this folder below. Please note everytime you run the model, you will have a new subfolder with a time stamp for that run.

Folder path:

```
R_Recent_path
```

```
## [1] "04. Random Forest Model - Recent Data Export/2024Apr18_11-55-19"
```

The below table show the same list in the excel file however it has been trimmed in the columns and the rows to fit. You can still access the full file in the above location. Note: for any column that has "L_" it denotes linear and "R_" denotes Random Forest.

```
R_Recent_Table
```

Trim	Year	Mileage	Price	L_Price	R_Price	R_Gap
BMW-BMW-8-Series-850i-8-Automatic	2019	31,000	205,000	265,174	231,118	13
Ford-Ford-F-150-6-Automatic	2016	87,000	73,000	87,322	111,152	52
Ford-Ford-Raptor-SVT-8-Automatic	2014	286,000	48,000	71,715	65,119	36
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2016	89,000	165,000	207,684	187,717	14
Land Rover-Land Rover-Range Rover-Sport SVR-8-Automatic	2015	180,000	120,000	152,171	132,613	11
Land Rover-Land Rover-Range Rover-Velar SE-4-Automatic	2019	22,000	150,000	166,124	290,988	94
Land Rover-Land Rover-Range Rover-Vogue-8-Automatic	2016	64,000	139,000	160,163	157,608	13
Land Rover-Land Rover-Range Rover-Vogue Autobiography-8-Automatic	2014	169,000	90,000	103,776	117,699	31
Lexus-Lexus-LX-570-8-Automatic	2020	85,000	315	317,458	303,516	96,254
Lexus-Lexus-LX-570 S-8-Automatic	2018	92,000	235,000	261,192	259,071	10
Mercedes-Benz-Mercedes-Benz-CLA-250-4-Automatic	2023	22,000	143,000	183,669	182,790	28
Mercedes-Benz-Mercedes-Benz-S-Class-400-6-Automatic	2015	66,000	100,000	127,592	135,098	35
Mini-Mini-Cooper-Standard-4-Automatic	2020	67,000	62,000	86,354	71,304	15
Nissan-Nissan-Patrol-Platinum-8-Automatic	2016	165,000	79,000	105,805	101,961	29
Nissan-Nissan-Patrol-SE-8-Automatic	2016	165,000	59,000	87,134	78,818	34
Porsche-Porsche-Cayenne-GTS-8-Automatic	2014	162,000	73,000	80,364	80,464	10

Trim	Year	Mileage	Price	L_Price	R_Price	R_Gap
Toyota-Toyota-Land Cruiser-GX-6-Automatic	2019	130,000	118,000	158,293	132,214	12
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2016	225,000	132,000	148,258	159,185	21
Toyota-Toyota-Land Cruiser-VXR-8-Automatic	2015	227,000	100,000	132,651	113,760	14
Toyota-Toyota-Land Cruiser-VXS-8-Automatic	2016	157,000	157,000	184,042	184,262	17

In this same folder, you also find: "_Export_Formated_rf.csv", this contains the priced prediction in column "L_Predicted_Price" & "R_Predicted_Price" along with how far is the prediction from the actual in column "L_Price_Gap_Percentage" & "R_Price_Gap_Percentage". This file provides a wider selection where user can filter based on trim, year to find the best deals by filtering the highest value of "L_Price_Gap_Percentage". The higher value of "L_Price_Gap_Percentage" means the predicted value is that many percent higher than actual price, indicating actual price is quite low.

The "_Export_full_rf.csv" in this folder is more for debugging and contains other parameters. The "_Export_missing.csv" in this folder contains the cars which the model could not predict as these are new trims that the model have never seen.

8 Conclusion

This project took quite some time to be developed and I am very pleased with the final results. From the fourth folder, we can review and inspect the 20 car that were recommended by the system.

We notice one of them is wrongly priced at 315 QAR, obviously a site listing error. We are not left with 19 cars.

I checked them one by one on the 16th April, as they were extracted 1 week earlier on the 9th Apr 2024, and to my surprise already 4 of the cars have already been sold. That is more than 25% of the cars listed.

As you know, it takes some time for buyers to see the cars, inspect them, prepare the cash and then finally update the site that the car has been sold.

I have saved my analysis in this "_Export_Formated_Filtered_rf_check sold.csv" file in the root folder, where I have added 2 columns indicating which cars where sold and by which date.

Below are the list of cars already sold that were suggested by the model that are good deals.

https://qatarsale.com/en/product/ford_f_150_silver_pick_up_automatic_2016-215357

https://qatarsale.com/en/product/mercedes_benz_cla_250_black_sedan_automatic_2023-215163

https://qatarsale.com/en/product/toyota_land_cruiser_vxr_white_suv_automatic_2015-215175

https://qatarsale.com/en/product/lexus_lx_570_s_white_suv_automatic_2018-215331

8.1 Limitation

The first limitation I faced is that the size of the dataset is relatively small. This was challenging but I was able to maneuver using linear regression over the years and mileage parameters.

Another limitation was that I didn't get chance to extract the color form the site listing, as I would then need to scrap data at a car listing page and that would take quite some time.

Using my personal laptop limited resources posed another challenge. I was considering to use some cloud provider to speed my work, but I knew it would be better for me to learn how to optimize my model rather than just relying on large resources.

8.2 Future Work

I would like to extend this project to different types of datasets like residence property for sale or rent in a larger country. I would also probably leverage on cloud compute to speed up the training and automate the system to share the best deals that fits my range and preference so that I do not need to manually search in the data and websites. Another possible enhancement could be training the model to be able to predict prices of trims that were never trained, by analyzing similar trims with similar trim features such as cylinder, year, mileage and car type(sedan, SUV, convertible, production country etc.). Lastly we can also enhance the prediction by training the model on a much large car dataset such as Kelly Blue Book[10] than fine tune the model to adapt to Qatar market.

[10]https://www.kbb.com/