CYO Project Report

Nikhil Venkatachalam

August 16, 2023

Introduction

The goal of this project was to make a car rating prediction software utilizing machine learning algorithms. Specifically, the car rating prediction software must have aspired to a certain level of quality and accuracy. The RMSE (that being the Root Mean Squared Error) was used to determine both quality and accuracy measured in this project, though a specific number was not aspired to, just the fact that the number had to be as low as possible. This was accomplished by initially engaging in data exploration (i.e. visualizing the data and outputting its summary statistics), then moving on to actually training the data by first accounting for various biases in the rating of various cars, then using machine learning techniques to further decrease the RMSE and make the car rating prediction software optimal.

Initial Setup

This project utilized a dataset made publicly available on a site called Hugging Face and provided by a user named Florent Gbelidji (user id florentgbelidji). This dataset contained around 37 thousand different and unique user reviews for various car makes and models, along with information about each specific review. In truth, for the purposes of this particular project, I didn't need all of the information provided, so I preserved what I needed, and cut what I didn't, as can be observed below:

```r
# Create edx and final_holdout_test sets

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(tidyverse)
library(caret)

# CYO car rating dataset:
# https://huggingface.co/datasets/florentgbelidji/car-reviews/resolve/main/train_car.csv

options(timeout = 120)

# Retrieve dataset
dl <- "train_car.csv"
if(!file.exists(dl))
  download.file(
    "https://huggingface.co/datasets/florentgbelidji/car-reviews/resolve/main/train_car.csv", dl)

# Clean and further modify dataset
dl_frame <- read.csv(dl)
dl_frame <- subset(dl_frame, select = -c(X, Unnamed..0, Author_Name,
                                         Review_Title, Review))
dl_frame <- dl_frame %>%
  separate(Vehicle_Title, c("Model_Year", "Make", "Model"),
           extra = "drop", fill = "right")
dl_frame$Model_Year = as.numeric(as.character(dl_frame$Model_Year))
dl_frame <- dl_frame[order(dl_frame$Model_Year),]
dl_frame$Review_Date = substr(dl_frame$Review_Date, 11, 12)
dl_frame$Review_Date = paste("20", dl_frame$Review_Date, sep = "")
colnames(dl_frame)[1] ="Review_Year"
dl_frame$Review_Year = as.numeric(as.character(dl_frame$Review_Year))

# Final hold-out test set will be 10% of the given data
set.seed(1, sample.kind="Rounding") # using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
test_index <- createDataPartition(y = dl_frame$Rating, times = 1,
                                  p = 0.1, list = FALSE)
edx <- dl_frame[-test_index,]
final_holdout_test <- dl_frame[test_index,]
```

Dividing the Algorithm

As can be observed, the Hugging Face Car Reviews dataset was divided into two separate datasets, those being edx and final_holdout_test. Then, the edx dataset was further split into two datasets in order to properly build and test the machine learning algorithm. However, the final test was conducted using the final_holdout_test dataset.

```
# Further division of edx into training and testing sets
set.seed(1, sample.kind = "Rounding") # using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = edx$Rating, times = 1,
                                  p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

Data Exploration

First, before engaging in any training of the data, I first had to take a closer look at the dataset that was provided. To do this, I outputted some summary statistics, and visualized the data so that I could better understand the data contained within the dataset.

Statistical Summary

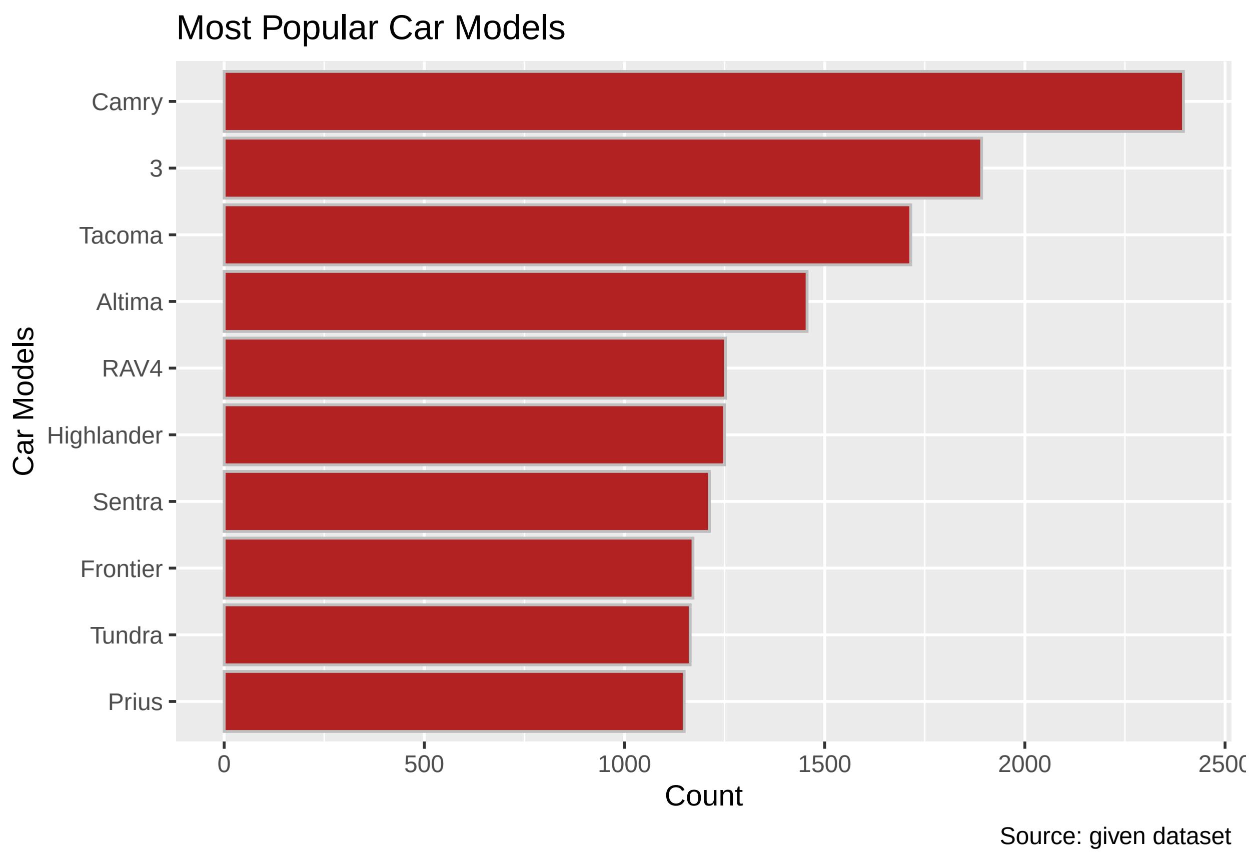
```
# Statistical summary of the dataset edx
summary(edx)
```

```
##   Review_Year     Model_Year       Make               Model
##  Min.   :2002   Min.   :1997   Length:33284       Length:33284
##  1st Qu.:2006   1st Qu.:2003   Class :character   Class :character
##  Median :2009   Median :2006   Mode  :character   Mode  :character
##  Mean   :2010   Mean   :2007
##  3rd Qu.:2014   3rd Qu.:2010
##  Max.   :2018   Max.   :2018
##      Rating
##  Min.   :1.000
##  1st Qu.:4.000
##  Median :5.000
##  Mean   :4.366
##  3rd Qu.:5.000
##  Max.   :5.000
```

The summary() function provided a statistical summary of the data contained within the dataset, which helped me look at the data with a new perspective.

Top Car Models Graph

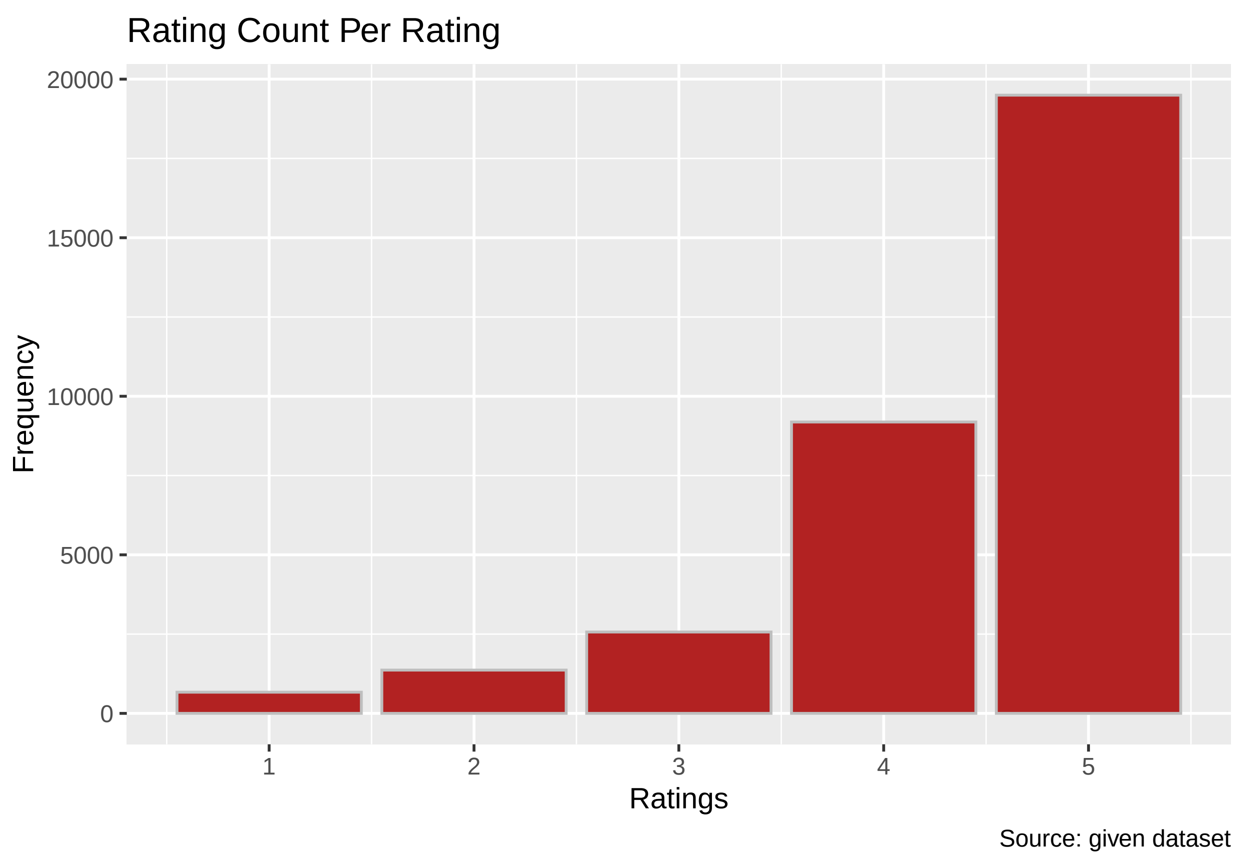
```r
# Graph top car models
edx %>%
  group_by(Model) %>%
  summarize(count = n()) %>%
  top_n(10, count) %>%
  arrange(-count) %>%
  ggplot(aes(count, reorder(Model, count))) +
  geom_bar(color = "gray", fill = "firebrick", stat = "identity") +
  labs(x = "Count", y = "Car Models", caption = "Source: given dataset") +
  ggtitle("Most Popular Car Models")
```



This graph is of the top 10 most car models within the dataset.

Number of Ratings Per Rating Graph

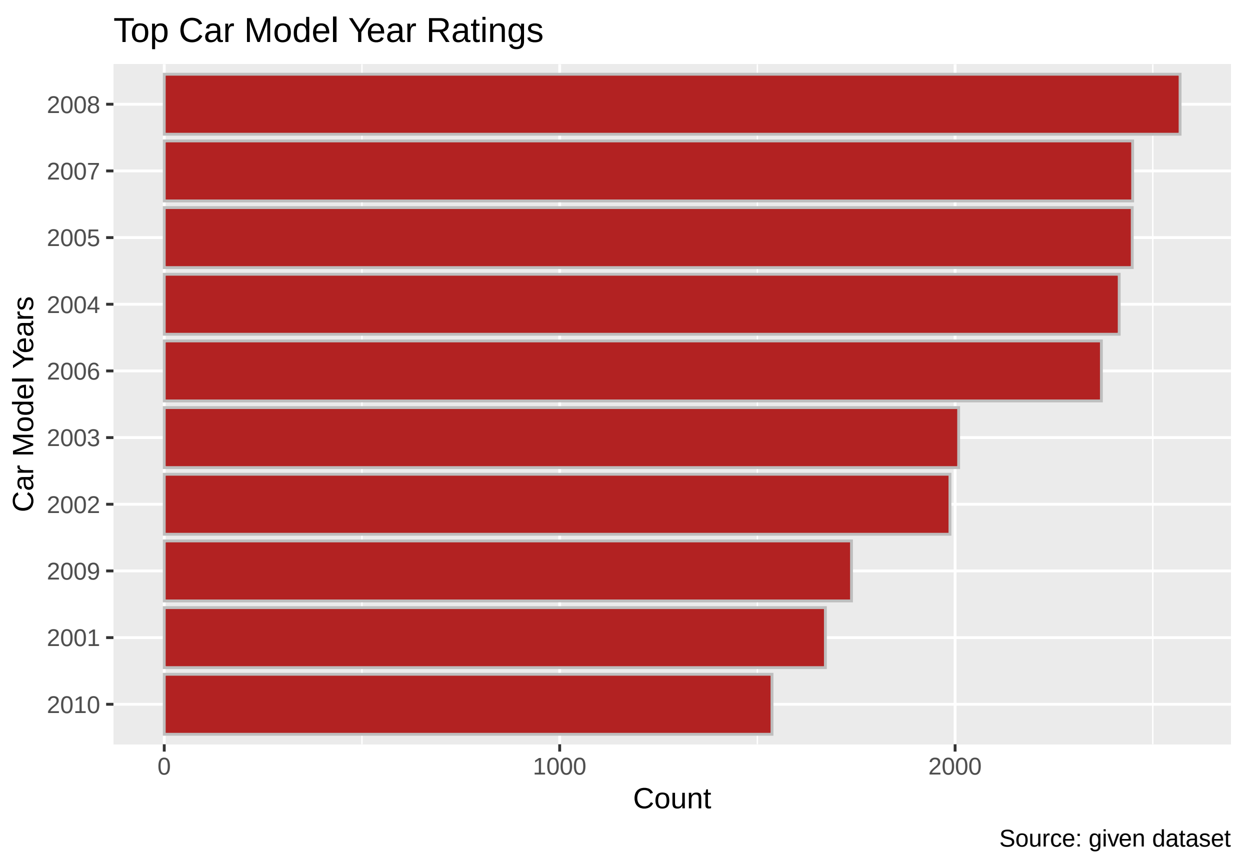
```r
# Graph number of ratings per rating
edx %>%
  ggplot(aes(Rating)) +
  geom_bar(color = "gray", fill = "firebrick") +
  labs(x = "Ratings", y = "Frequency", caption = "Source: given dataset") +
  scale_x_continuous(breaks = seq(0, 5, by = 1)) +
  ggtitle("Rating Count Per Rating")
```



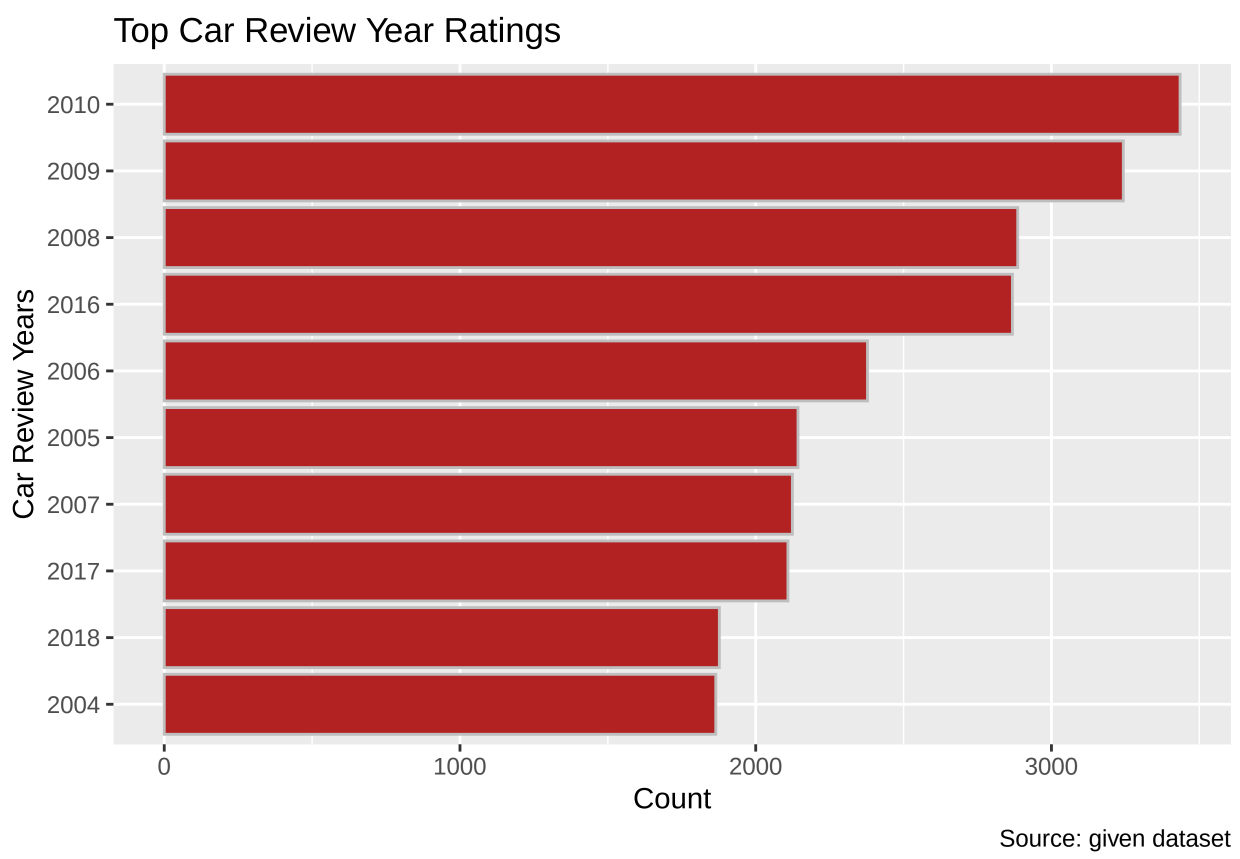
Within this graph, one can observe that users give ratings in whole-number increments, and that this particular graph is left-skewed, with much more users giving higher ratings than lower ratings.

Top Car Model Years Versus Top Car Review Years Graphed

```r
# Graph top car model years
edx %>%
  group_by(Model_Year) %>%
  summarize(count = n()) %>%
  top_n(10, count) %>%
  arrange(-count) %>%
  ggplot(aes(count, reorder(Model_Year, count))) +
  geom_bar(color = "gray", fill = "firebrick", stat = "identity") +
  labs(x = "Count", y = "Car Model Years", caption = "Source: given dataset") +
  ggtitle("Top Car Model Year Ratings")
```



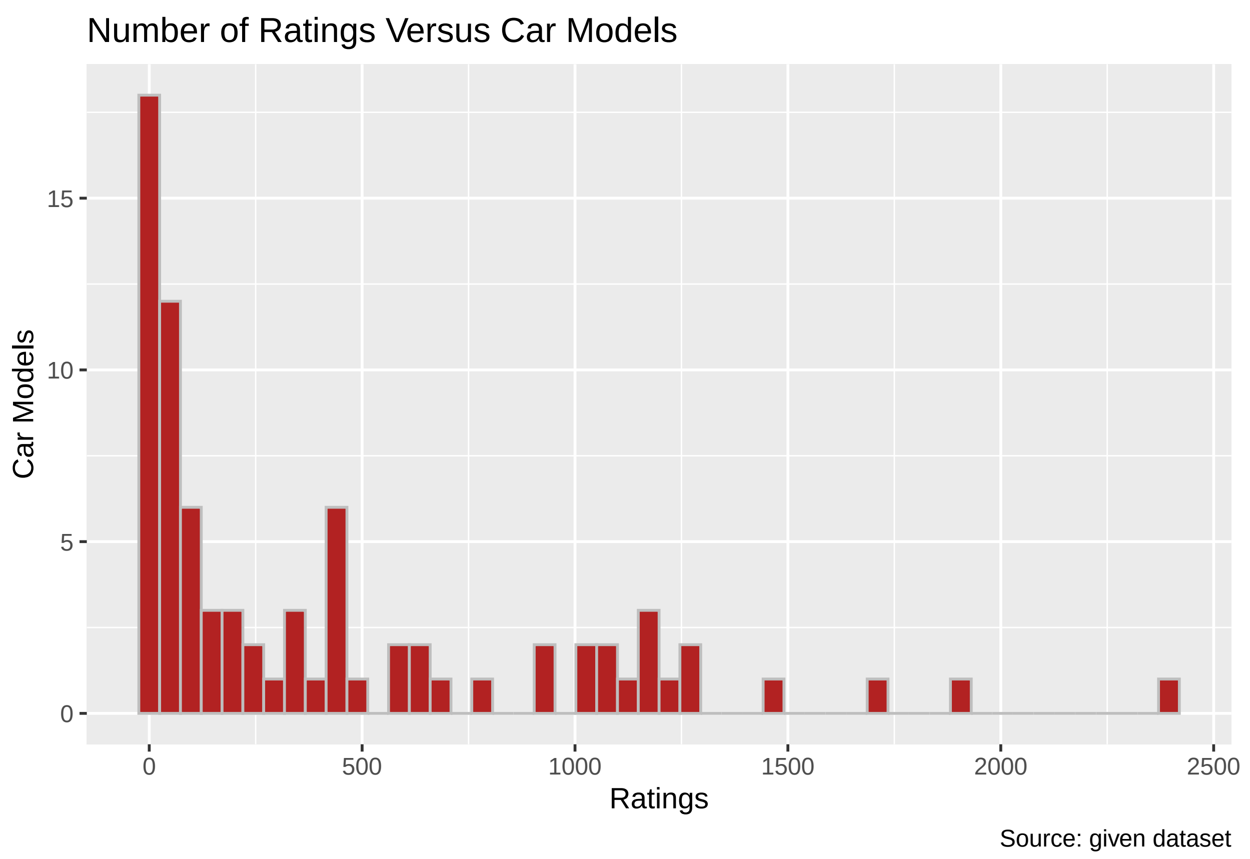
```
# Graph top car review years
edx %>%
  group_by(Review_Year) %>%
  summarize(count = n()) %>%
  top_n(10, count) %>%
  arrange(-count) %>%
  ggplot(aes(count, reorder(Review_Year, count))) +
  geom_bar(color = "gray", fill = "firebrick", stat = "identity") +
  labs(x = "Count", y = "Car Review Years", caption = "Source: given dataset") +
  ggtitle("Top Car Review Year Ratings")
```



There are slight variations that can be observed between these two graphs, such as different years placed in different rankings throughout. This can be attributed to the fact that the reviews for the cars are often newer than the cars themselves, which results in newer review years as opposed to older model years.

Number of Ratings Versus Car Models Graph

```
# Graph number of ratings versus car models
edx %>%
  group_by(Model) %>%
  summarize(count = n()) %>%
  ggplot(aes(count)) +
  geom_histogram(color = "gray", fill = "firebrick", bins = 50) +
  labs(x = "Ratings", y = "Car Models", caption = "Source: given dataset") +
  ggtitle("Number of Ratings Versus Car Models")
```



One can observe that this graph is very right-skewed. This shows the fact that some car models get rated much more often than others, though most car models are not rated very much.

Data Analysis

With the data exploration finished, it was time to start on the actual data analysis and training. During this time, I trained 7 models in total, each one varying in quality and accuracy, with some being closer to an ideal "lowest" RMSE than others.

RMSE and Mean

To start the data analysis, I made a function that would calculate the RMSE so that I would not have to calculate it manually every time I created a new model.

```
# Function to calculate RMSE
rmse <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

This is the function mentioned above. It takes in the actual car ratings and the car ratings predicted by the model, and outputs the RMSE of the model.

```r
# Mean of all ratings
mean_rating <- mean(train_set$Rating)
mean_rating
```

```
## [1] 4.366049
```

As a baseline, I calculated the mean of all the ratings in the training set. This was then used to train the first model.

```r
# RMSE calculated with just the mean
mean_rmse <- rmse(test_set$Rating, mean_rating)
mean_rmse
```

```
## [1] 0.9232887
```

This first model was trained using simply the mean of the training set. Despite the fact that the RMSE is below 1, this is not a very good model, and needs to be improved upon in later models.

Adding Bias to Calculations

Once I established a starting point for my future models to improve on, I could implement various changes and fixes that would decrease the RMSE further in order to bring myself closer to the goal set previously. Therefore, I decided to account for the various biases that can exist in the ratings of cars.

Adding Car Model Bias

As can be seen through experience, some types of car models are more popular than others. Therefore, I added this bias into the next model trained.

```r
# Add car model bias to calculation
bi <- train_set %>%
  group_by(Model) %>%
  summarize(b_i = mean(Rating - mean_rating))
predicted_ratings <- mean_rating + test_set %>%
  left_join(bi, by = "Model") %>%
  pull(b_i)
```

```r
# RMSE calculated with mean and car model bias
car_model_bias_rmse <- rmse(predicted_ratings, test_set$Rating)
car_model_bias_rmse
```

```
## [1] 0.9002732
```

As can be observed, adding car model bias to the model decreased the RMSE further. This was built upon in future models.

Adding Car Make Bias

Next, some types of car makes are more popular than others. People will have “brand loyalty”, and be more likely to buy one car make and rate that make higher than others. Thus, I took this into account with the next model.

```r
# Add car make bias to calculation
bu <- train_set %>%
  left_join(bi, by = "Model") %>%
  group_by(Make) %>%
  summarize(b_u = mean(Rating - mean_rating - b_i))
predicted_ratings <- test_set %>%
  left_join(bi, by = "Model") %>%
  left_join(bu, by = "Make") %>%
  mutate(pred = mean_rating + b_i + b_u) %>%
  pull(pred)
```

```r
# RMSE calculated with mean, car model, and car make bias
car_make_bias_rmse <- rmse(predicted_ratings, test_set$Rating)
car_make_bias_rmse
```

```
## [1] 0.9002732
```

Adding car make bias to this next model interestingly did not reduce the RMSE further. In fact, all accounts seem to show that the RMSE was functionally the same. Therefore, further models strived to improve on this number.

Adding Review Year Bias

Cars are also rated differently depending on when they were reviewed. Some cars may be looked at in a different light depending on how new they are, so they may be rated differently. Thus, I accounted for this with the next model by adding review year bias into the calculations.

```r
# Add review year bias to calculation
bt <- train_set %>%
  left_join(bi, by = "Model") %>%
  left_join(bu, by = "Make") %>%
  group_by(Review_Year) %>%
  summarize(b_t = mean(Rating - mean_rating - b_i - b_u))
predicted_ratings <- test_set %>%
  left_join(bi, by = "Model") %>%
  left_join(bu, by = "Make") %>%
  left_join(bt, by = "Review_Year") %>%
  mutate(pred = mean_rating + b_i + b_u + b_t) %>%
  pull(pred)
```

```r
# RMSE calculated with mean, car model, car make, and review year bias
review_year_bias_rmse <- rmse(predicted_ratings, test_set$Rating)
review_year_bias_rmse
```

```
## [1] 0.8876623
```

Adding review year bias to the model made it perform better, but still not by a significant amount. Therefore, to decrease the RMSE further, I had to employ a new strategy.

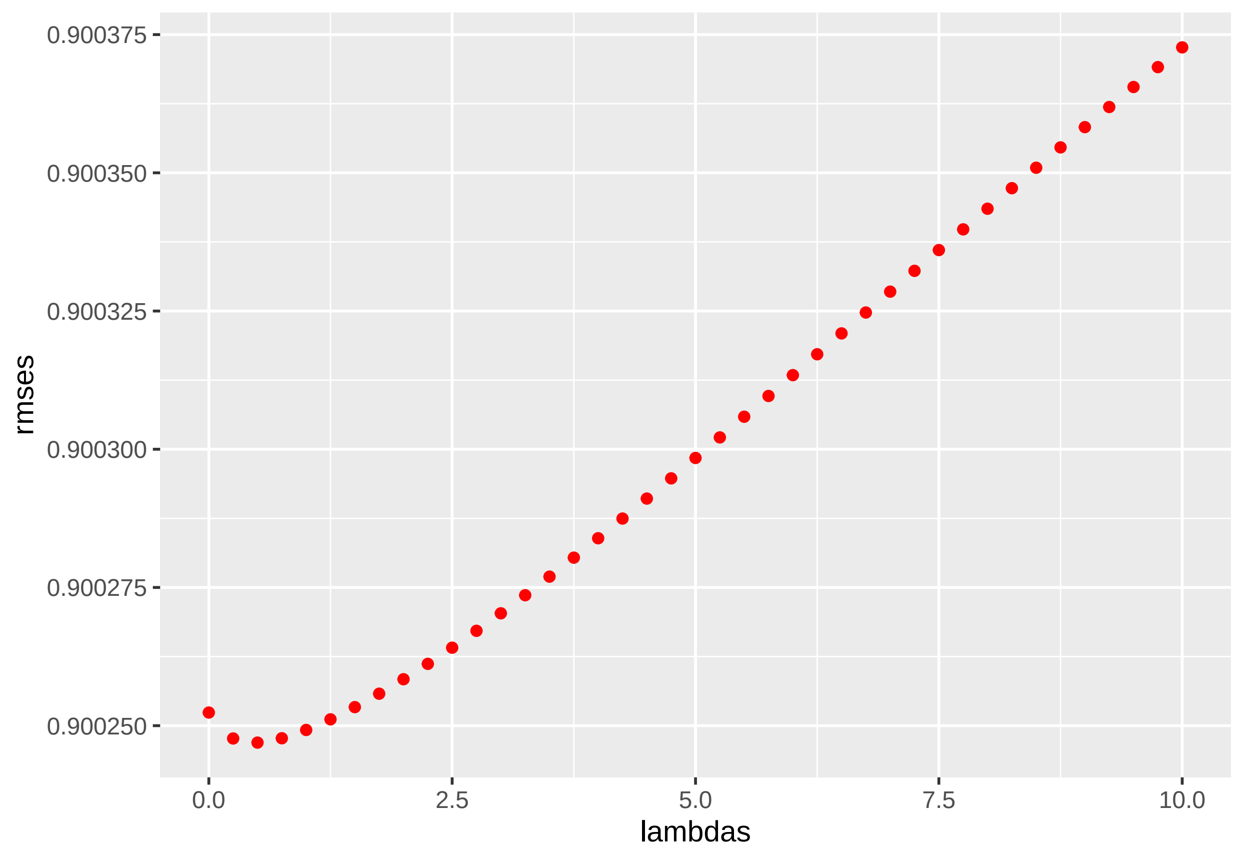
Data Regularization

Some of the data in any dataset can be "noisy", where large estimates come from small sample sizes. In the case of this dataset, it can mean that there are ratings of obscure cars by only a few users within the dataset. Therefore, I used regularization to penalize these large estimates from small sample sizes. In addition to this, I used cross-validation to find the optimal penalty to use (denoted by lambda) and applied it to the next model generated.

```r
# Applying data regularization
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(x){
  b_i <- train_set %>%
    group_by(Model) %>%
    summarize(b_i = sum(Rating - mean_rating)/(n() + x)) # adding car model bias
  b_u <- train_set %>%
    left_join(b_i, by = "Model") %>%
    group_by(Make) %>%
    summarize(b_u = sum(Rating - b_i -
                          mean_rating)/(n() + x)) # adding car make bias
  b_t <- train_set %>%
    left_join(b_i, by = "Model") %>%
    left_join(b_u, by = "Make") %>%
    group_by(Review_Year) %>%
    summarize(b_t = mean(Rating - b_i - b_u -
                           mean_rating)/(n() + x)) # adding review year bias
  predicted_ratings <- test_set %>%
    left_join(b_i, by = "Model") %>%
    left_join(b_u, by = "Make") %>%
    left_join(b_t, by = "Review_Year") %>%
    mutate(pred = mean_rating + b_i + b_u + b_t) %>%
    pull(pred)
  return(rmse(predicted_ratings, test_set$Rating))
})
```

```r
# Plotting lambdas versus RMSEs
qplot(lambdas, rmses, color = I("red"))
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```r
# Finding which lambda has the lowest RMSE
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 0.5
```

After calculating and plotting a range of lambdas versus RMSEs, I found the optimal lambda to use on the model. This optimal lambda provides the lowest error (i.e. it is the lowest point on the graph).

```r
# Selecting the lambda with the lowest RMSE
regularized_rmse <- min(rmses)
regularized_rmse
```

```
## [1] 0.9002469
```

Interestingly, the RMSE shows no improvement. In fact, it has increased from what it was before. To make the RMSE sufficiently low, I had to switch tactics once again.

Matrix Factorization (using the recosystem package)

To make the RMSE lower, I turned to matrix factorization, which is a process by which the data is processed as a large and sparse matrix, then decomposed into two smaller dimensional matrices with less sparsity and latent features. This was accomplished by using the recosystem package. The data was converted into the recosystem format, tuned to select the best tuning parameters along a set of candidate values, trained, and finally tested by computing the predicted values. To this extent, I elected to use the review year and model year for the cars as factors instead of what was used before, as this would make it easier for the recosystem algorithm.

```
# Applying matrix factorization using the recosystem package
if(!require(recosystem)) install.packages(
  "recosystem", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: recosystem
```

```
library(recosystem)
set.seed(1, sample.kind = "Rounding") # using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
reco_train <- with(train_set, data_memory(user_index = Review_Year,
                                          item_index = Model_Year,
                                          rating = Rating))
reco_test <- with(test_set, data_memory(user_index = Review_Year,
                                        item_index = Model_Year,
                                        rating = Rating))
reco <- Reco()

reco_para <- reco$tune(reco_train, opts = list(dim = c(20, 30),
                                               costp_l2 = c(0.01, 0.1),
                                               costq_l2 = c(0.01, 0.1),
                                               lrate = c(0.01, 0.1),
                                               nthread = 4, niter = 10))

reco$train(reco_train, opts = c(reco_para$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       2.1576   1.5220e+05
##    1       1.0022   4.2904e+04
##    2       0.9141   3.7638e+04
##    3       0.9090   3.7100e+04
##    4       0.9071   3.6775e+04
##    5       0.9062   3.6638e+04
##    6       0.9057   3.6409e+04
##    7       0.9049   3.6186e+04
##    8       0.9052   3.6112e+04
##    9       0.9047   3.5996e+04
##   10       0.9043   3.5857e+04
##   11       0.9044   3.5810e+04
##   12       0.9039   3.5697e+04
##   13       0.9037   3.5600e+04
##   14       0.9036   3.5545e+04
##   15       0.9033   3.5454e+04
##   16       0.9035   3.5414e+04
##   17       0.9029   3.5320e+04
##   18       0.9027   3.5226e+04
##   19        0.9029   3.5221e+04
##   20        0.9025   3.5132e+04
##   21        0.9023   3.5061e+04
##   22        0.9026   3.5045e+04
##   23        0.9023   3.4985e+04
##   24        0.9022   3.4942e+04
##   25        0.9021   3.4897e+04
##   26        0.9021   3.4851e+04
##   27        0.9019   3.4807e+04
##   28        0.9017   3.4760e+04
##   29        0.9016   3.4725e+04
```

```
reco_first <- reco$predict(reco_test, out_memory())
```

```
# RMSE calculated with matrix factorization
factorization_rmse <- RMSE(reco_first, test_set$Rating)
factorization_rmse
```

```
## [1] 0.8931866
```

This RMSE is almost the best one yet, and the improvement is significant enough that it can be the method by which I proceed. Therefore, this is the process by which I conducted the final holdout test.

Final Holdout Test

Because I determined that this method would be the method to conduct the final holdout test with, I trained the model using the edx set and tested it using the final_holdout_test set.

```
# Using matrix factorization on final holdout test
set.seed(1, sample.kind = "Rounding") # using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
reco_edx <- with(edx, data_memory(user_index = Review_Year,
                                  item_index = Model_Year,
                                  rating = Rating))
reco_final_holdout <- with(final_holdout_test, data_memory(user_index =
                                                             Review_Year,
                                                           item_index =
                                                             Model_Year,
                                                           rating = Rating))
reco <- Reco()

reco_para <- reco$tune(reco_edx, opts = list(dim = c(20, 30),
                                             costp_l2 = c(0.01, 0.1),
                                             costq_l2 = c(0.01, 0.1),
                                             lrate = c(0.01, 0.1),
                                             nthread = 4, niter = 10))

reco$train(reco_edx, opts = c(reco_para$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       0.9105   3.0660e+04
##    1       0.9041   3.0248e+04
##    2       0.9008   3.0046e+04
##    3       0.8992   2.9953e+04
##    4       0.8984   2.9904e+04
##    5       0.8980   2.9911e+04
##    6       0.8974   2.9843e+04
##    7       0.8972   2.9870e+04
##    8       0.8968   2.9816e+04
##    9       0.8967   2.9808e+04
##   10       0.8966   2.9817e+04
##   11       0.8963   2.9805e+04
##   12       0.8962   2.9781e+04
##   13       0.8959   2.9766e+04
##   14       0.8958   2.9763e+04
##   15       0.8957   2.9757e+04
##   16       0.8956   2.9767e+04
##   17       0.8956   2.9743e+04
##   18       0.8955   2.9737e+04
##   19       0.8954   2.9754e+04
##   20       0.8954   2.9749e+04
##   21       0.8953   2.9749e+04
##   22       0.8951   2.9743e+04
##   23       0.8951   2.9719e+04
##   24       0.8950   2.9721e+04
##   25       0.8950   2.9722e+04
##   26       0.8949   2.9718e+04
##   27       0.8949   2.9712e+04
##   28       0.8948   2.9714e+04
##   29       0.8947   2.9713e+04
```

```r
reco_final <- reco$predict(reco_final_holdout, out_memory())
```

```r
# Generating final RMSE
final_rmse <- RMSE(reco_final, final_holdout_test$Rating)
final_rmse
```

```
## [1] 0.9007888
```

This is the final RMSE, generated using the final_holdout_test dataset. No more testing occured after this.

Final Results

The final results of the data analysis can be seen in the table generated below:

```r
# Table made using the reactable package
if(!require(reactable)) install.packages("reactable",
                                        repos = "http://cran.us.r-project.org")
```

```
## Loading required package: reactable
```

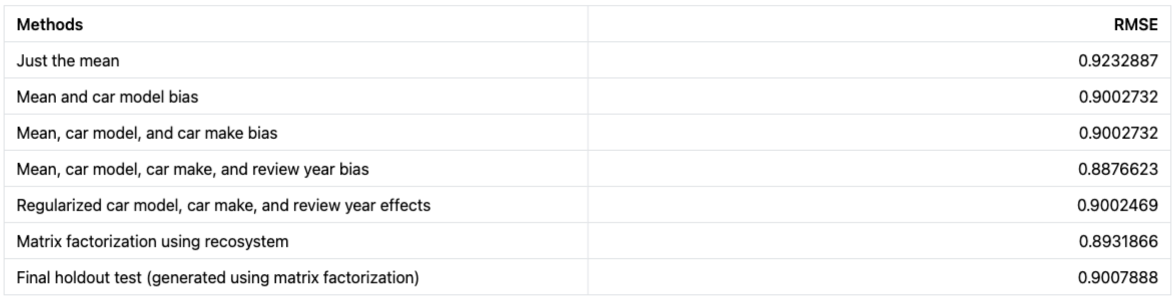
```r
library(reactable)
if(!require(webshot2)) install.packages("webshot2",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: webshot2
```

```r
library(webshot2)
if(!require(htmlwidgets)) install.packages("htmlwidgets",
                                        repos = "http://cran.us.r-project.org")
```

```
## Loading required package: htmlwidgets
```

```r
library(htmlwidgets)
Methods <- c("Just the mean", "Mean and car model bias",
             "Mean, car model, and car make bias", "Mean, car model,
             car make, and review year bias",
             "Regularized car model, car make, and review year effects",
             "Matrix factorization using recosystem",
             "Final holdout test
             (generated using matrix factorization)") # first column
RMSE <- c(round(mean_rmse, 7), round(car_model_bias_rmse, 7),
          round(car_make_bias_rmse, 7), round(review_year_bias_rmse, 7),
          round(regularized_rmse, 7), round(factorization_rmse, 7),
          round(final_rmse, 7)) # second column
final_results <- data.frame(Methods, RMSE)
table <- reactable(final_results,
  highlight = TRUE,
  bordered = TRUE,
  theme = reactableTheme(
    borderColor = "#dfe2e5",
    highlightColor = "#f0f5f9",
    cellPadding = "8px 12px",
    style = list(fontFamily = "-apple-system, BlinkMacSystemFont,
                 Segoe UI, Helvetica, Arial, sans-serif"),
    )
  )
saveWidget(widget = table, file = "table_html.html", selfcontained = TRUE)
webshot(url = "table_html.html", file = "final_table.png", delay = 0.1,
        vwidth = 1245)
```



Methods	RMSE
Just the mean	0.9232887
Mean and car model bias	0.9002732
Mean, car model, and car make bias	0.9002732
Mean, car model, car make, and review year bias	0.8876623
Regularized car model, car make, and review year effects	0.9002469
Matrix factorization using recosystem	0.8931866
Final holdout test (generated using matrix factorization)	0.9007888

As can be observed, the final RMSE generated was low, but not as low as I expected. I tested several models, but achieved the best results just accounting for car model, car make, and review year biases. Therefore, just accounting for various biases that can occur, at least concerning this model, was the best option which provided an optimal car rating prediction system by the metrics tested by this project, that being the RMSE.

Conclusion

To conclude this project, I set out to make a car rating prediction software using machine learning algorithms. This was accomplished by using data exploration to better understand the data I was working with, and then data analysis in order to eliminate various types of biases in car rating, regularize the data, and finally utilize matrix factorization in order to make the best car rating prediction system that I could. Though this project had no specific goal RMSE, I feel that the RMSE I acheived was as ideal as I could achieve.

However, some limitations of the project include the size of the dataset that was used. This dataset only included 37 thousand data points, which is not ideal for generating predictions like what was accomplished in this project. This must have accounted for the fact that the RMSEs of each test/algorithm were varying in ways that I personally did not expect. In the future, I will certainly strive to utilize datasets that have more data points. In addition to this, to state the obvious, this project is simply the first step in making software that can actually predict the ratings of cars before they even come out.

In the future, this project can be used as a shell or stepping stone for other machine learning algorithms to predict other forms of ratings. However, it could go even further than that, and be used for other predictions for any topic imaginable. This can all be accomplished by using the methods employed in this project as a base. The sky's the limit when it comes to machine learning algorithms such as these.

References

The site from which I procured the dataset can be found here: https://huggingface.co/datasets/florentgbelidji/car-reviews/viewer/florentgbelidji--car-reviews/train?p=0

The actual raw dataset can be found here: https://huggingface.co/datasets/florentgbelidji/car-reviews/resolve/main/train_car.csv