Final Report

Nicholas Allen, Surya Maddali, Jake Adams

Introduction:

In recent decades, cell phones have become a hot commodity around the world. The idea of calling with the tips of your fingers was a revolutionary idea that continues to set the standard for telecommunications. With advancements to cell phones, one question that is always present is pricing There may be certain factors that affect cell phone pricing such as storage, camera capabilities, and battery power. The goal of this project is to assess that, seeing if certain features of phones affect pricing in a significant way. It is an interesting and important question to answer because it can inform others about what phone features matter the most to companies that make phones as well as inform us about what features matter the most to a phone's functionality when looking to buy one. Machine learning is a reasonable approach to tackle this question because it can give us insight into why or how phones are purchased. Moreover, it can help us predict phone prices in the future based on what features they possess, which would be informed by past data on this exact matter. In other words, it would help readers assess what features are continuing to affect the price of the phone the most in the present.

Illustration:

Background and Related Works:

We looked at an article from IEEE Xplore. This article was about predicting mobile phone prices using a data set from kaggle. This article differed from ours because they were predicting phone prices with classification. They had their y variable in as a factor with 4 levels. The levels were form "low cost" to "very high cost". Some examples of their x variables were battery power and clock speed. They used several different models to predict phone price such as a decision tree and SVM. Their most accurate model was SVM with an accuracy of 94.8%.

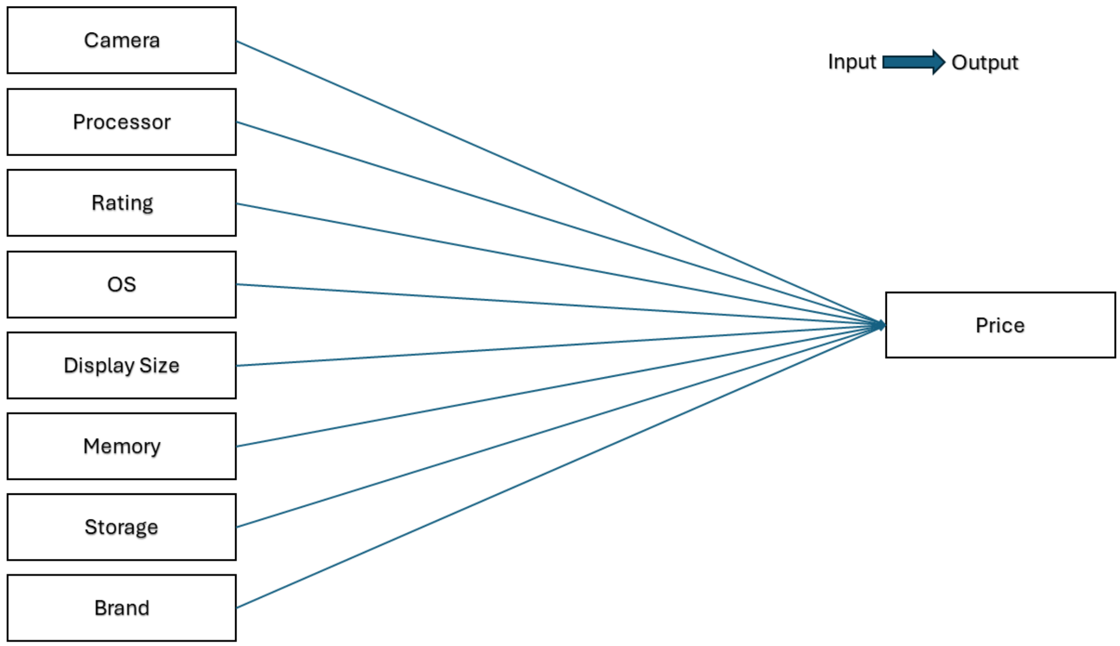


Figure 1: Elephant

Reference: N. Hu, "Classification of Mobile Phone Price Dataset Using Machine Learning Algorithms," 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), Chengdu, China, 2022, pp. 438-443, doi: 10.1109/PRML56267.2022.9882236. keywords: {Support vector machines;Machine learning algorithms;Random access memory;Machine learning;Feature extraction;Mobile handsets;Batteries;computer science;machine learning;classification;price prediction},

Data Processing:

We loaded in the data sets though the readxl package

```
Loading required package: Matrix
```

```
Loaded glmnet 4.1-8
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
```

```
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x tidyr::expand() masks Matrix::expand()
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x tidyr::pack()   masks Matrix::pack()
x tidyr::unpack() masks Matrix::unpack()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
corrplot 0.92 loaded

Warning: package 'caret' was built under R version 4.3.2

Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift

Rows: 3114 Columns: 12-- Column specification --------------------------------------------
Delimiter: ","
chr (7): Brands, Models, Colors, Memory, Storage, Camera, Mobile
dbl (5): Rating, Selling Price, Original Price, Discount, discount percentage
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

First Dataset

The first data set looked like this before processing.

```
# A tibble: 6 x 11
  model    price rating sim   processor ram   battery display camera card  os
  <chr>    <chr>  <dbl> <chr> <chr>     <chr> <chr>   <chr>   <chr>  <chr> <chr>
1 OnePlus~ 54,~      89 Dual~ Snapdrag~ 12 G~ 5000 m~ 6.7 in~ 50 MP~ Memo~ Andr~
2 OnePlus~ 19,~      81 Dual~ Snapdrag~ 6 GB~ 5000 m~ 6.59 i~ 64 MP~ Memo~ Andr~
3 Samsung~ 16,~      75 Dual~ Exynos 1~ 4 GB~ 5000 m~ 6.6 in~ 50 MP~ Memo~ Andr~
4 Motorol~ 14,~      81 Dual~ Snapdrag~ 6 GB~ 5000 m~ 6.55 i~ 50 MP~ Memo~ Andr~
5 Realme ~ 24,~      82 Dual~ Dimensit~ 6 GB~ 5000 m~ 6.7 in~ 108 M~ Memo~ Andr~
6 Samsung~ 16,~      80 Dual~ Snapdrag~ 6 GB~ 5000 m~ 6.6 in~ 50 MP~ Memo~ Andr~
```

It is a tabular data set on some mobile phones. Some examples of columns in the data set are mobile which represents the name of the phone and the price of the phone.

To start off we took out the model column because it represented the names of the phones which will not impact the price. We also took out the sim column.

```
# A tibble: 6 x 9
  price   rating processor              ram   battery display camera card  os
  <chr>    <dbl> <chr>                  <chr> <chr>   <chr>   <chr>  <chr> <chr>
1 54,999     89 Snapdragon 8 Gen2, Oc~ 12 G~ 5000 m~ 6.7 in~ 50 MP~ Memo~ Andr~
2 19,989     81 Snapdragon 695, Octa ~ 6 GB~ 5000 m~ 6.59 i~ 64 MP~ Memo~ Andr~
3 16,499     75 Exynos 1330, Octa Cor~ 4 GB~ 5000 m~ 6.6 in~ 50 MP~ Memo~ Andr~
4 14,999     81 Snapdragon  695, Octa~ 6 GB~ 5000 m~ 6.55 i~ 50 MP~ Memo~ Andr~
5 24,999     82 Dimensity 1080, Octa ~ 6 GB~ 5000 m~ 6.7 in~ 108 M~ Memo~ Andr~
6 16,999     80 Snapdragon  750G, Oct~ 6 GB~ 5000 m~ 6.6 in~ 50 MP~ Memo~ Andr~
```

Cleaning battery column

We extracted the battery life of each phone in mAH and made the column numeric

```
Warning: There was 1 warning in `mutate()`.
i In argument: `battery = .Primitive("as.double")(battery)`.
Caused by warning:
! NAs introduced by coercion
```

```
# A tibble: 6 x 9
  price   rating processor          ram   battery_mAh display camera card  os
  <chr>    <dbl> <chr>              <chr>       <dbl> <chr>   <chr>  <chr> <chr>
1 54,999     89 Snapdragon 8 Gen2~ 12 G~        5000 6.7 in~ 50 MP~ Memo~ Andr~
2 19,989     81 Snapdragon 695, O~ 6 GB~        5000 6.59 i~ 64 MP~ Memo~ Andr~
3 16,499     75 Exynos 1330, Octa~ 4 GB~        5000 6.6 in~ 50 MP~ Memo~ Andr~
4 14,999     81 Snapdragon  695, ~ 6 GB~        5000 6.55 i~ 50 MP~ Memo~ Andr~
5 24,999     82 Dimensity 1080, O~ 6 GB~        5000 6.7 in~ 108 M~ Memo~ Andr~
6 16,999     80 Snapdragon  750G,~ 6 GB~        5000 6.6 in~ 50 MP~ Memo~ Andr~
```

Cleaning processor variable

We extracted the power of the processor in GHz. We then made the column numeric

```
# A tibble: 6 x 9
  price   rating `processor GHz)` ram     battery_mAh display camera card  os
  <chr>    <dbl>            <dbl> <chr>         <dbl> <chr>   <chr>  <chr> <chr>
1 54,999     89               3.2 12 GB ~        5000 6.7 in~ 50 MP~ Memo~ Andr~
2 19,989     81               2.2 6 GB R~        5000 6.59 i~ 64 MP~ Memo~ Andr~
3 16,499     75               2.4 4 GB R~        5000 6.6 in~ 50 MP~ Memo~ Andr~
4 14,999     81               2.2 6 GB R~        5000 6.55 i~ 50 MP~ Memo~ Andr~
5 24,999     82               2.6 6 GB R~        5000 6.7 in~ 108 M~ Memo~ Andr~
6 16,999     80               2.2 6 GB R~        5000 6.6 in~ 50 MP~ Memo~ Andr~
```

Cleaning os column

We noticed that because the data was unclean, some of the values that should be in the os column were in the card column. We put these value in the os column and removed the card column after. We also made the os column a factor.

```
# A tibble: 6 x 8
  price   rating `processor GHz)` ram           battery_mAh display camera os
  <chr>    <dbl>            <dbl> <chr>               <dbl> <chr>   <chr>  <fct>
1 54,999     89               3.2 12 GB RAM, 2~        5000 6.7 in~ 50 MP~ Andr~
2 19,989     81               2.2 6 GB RAM, 12~        5000 6.59 i~ 64 MP~ Andr~
3 16,499     75               2.4 4 GB RAM, 64~        5000 6.6 in~ 50 MP~ Andr~
4 14,999     81               2.2 6 GB RAM, 12~        5000 6.55 i~ 50 MP~ Andr~
5 24,999     82               2.6 6 GB RAM, 12~        5000 6.7 in~ 108 M~ Andr~
6 16,999     80               2.2 6 GB RAM, 12~        5000 6.6 in~ 50 MP~ Andr~
```

Cleaning camera column

We extracted the amount of mega pixels in the front camera of each phone. We made this column numeric

```
# A tibble: 6 x 8
  price   rating `processor GHz)` ram      battery_mAh display f_camera_MP os
  <chr>    <dbl>            <dbl> <chr>          <dbl> <chr>         <dbl> <fct>
1 54,999     89               3.2 12 GB R~        5000 6.7 in~          16 Andr~
2 19,989     81               2.2 6 GB RA~        5000 6.59 i~          16 Andr~
3 16,499     75               2.4 4 GB RA~        5000 6.6 in~          13 Andr~
4 14,999     81               2.2 6 GB RA~        5000 6.55 i~          16 Andr~
5 24,999     82               2.6 6 GB RA~        5000 6.7 in~          16 Andr~
6 16,999     80               2.2 6 GB RA~        5000 6.6 in~           8 Andr~
```

Cleaning ram column

We extracted the ram of the phones in GB and made it a factor because phones only have a few preset values for their ram

```
# A tibble: 6 x 8
  price   rating `processor GHz)` ram   battery_mAh display    f_camera_MP os
  <chr>    <dbl>            <dbl> <fct>       <dbl> <chr>            <dbl> <fct>
1  54,999     89              3.2 12 GB        5000 6.7 inche~          16 Andr~
2  19,989     81              2.2 6 GB         5000 6.59 inch~          16 Andr~
3  16,499     75              2.4 4 GB         5000 6.6 inche~          13 Andr~
4  14,999     81              2.2 6 GB         5000 6.55 inch~          16 Andr~
5  24,999     82              2.6 6 GB         5000 6.7 inche~          16 Andr~
6  16,999     80              2.2 6 GB         5000 6.6 inche~           8 Andr~
```

Cleaning Display column

We extracted the display size and the Hz of the display and turned that into two new columns. We made these new columns numeric and removed the original

```
# A tibble: 6 x 9
  price  rating `processor GHz)` ram   battery_mAh f_camera_MP os    displaySize
  <chr>   <dbl>            <dbl> <fct>       <dbl>       <dbl> <fct>       <dbl>
1  54,9~     89              3.2 12 GB       5000           16 Andr~        6.7
2  19,9~     81              2.2 6 GB        5000           16 Andr~        6.59
3  16,4~     75              2.4 4 GB        5000           13 Andr~        6.6
4  14,9~     81              2.2 6 GB        5000           16 Andr~        6.55
5  24,9~     82              2.6 6 GB        5000           16 Andr~        6.7
6  16,9~     80              2.2 6 GB        5000            8 Andr~        6.6
# i 1 more variable: displayHz <dbl>
```

Cleaning Price column

We converted the value in rupees to dollars to make it easier to understand for our audience

```
# A tibble: 6 x 9
  price rating `processor GHz)` ram   battery_mAh f_camera_MP os     displaySize
  <dbl>  <dbl>            <dbl> <fct>       <dbl>       <dbl> <fct>        <dbl>
1  659.     89              3.2 12 GB        5000          16 Andro~         6.7
2  240.     81              2.2 6 GB         5000          16 Andro~         6.59
```

```
3  198.     75                 2.4 4 GB          5000            13 Andro~         6.6
4  180.     81                 2.2 6 GB          5000            16 Andro~         6.55
5  300.     82                 2.6 6 GB          5000            16 Andro~         6.7
6  204.     80                 2.2 6 GB          5000             8 Andro~         6.6
# i 1 more variable: displayHz <dbl>
```

General Analysis

After Cleaning:

```
# A tibble: 6 x 9
  price rating `processor GHz)` ram   battery_mAh f_camera_MP os     displaySize
  <dbl>  <dbl>            <dbl> <fct>       <dbl>       <dbl> <fct>        <dbl>
1  659.     89              3.2 12 GB        5000          16 Andro~         6.7
2  240.     81              2.2 6 GB         5000          16 Andro~         6.59
3  198.     75              2.4 4 GB         5000          13 Andro~         6.6
4  180.     81              2.2 6 GB         5000          16 Andro~         6.55
5  300.     82              2.6 6 GB         5000          16 Andro~         6.7
6  204.     80              2.2 6 GB         5000           8 Andro~         6.6
# i 1 more variable: displayHz <dbl>
```

```
     price             rating      processor GHz)       ram
 Min.   :  88.71   Min.   :62.00   Min.   :1.600   8 GB   :231
 1st Qu.: 191.81   1st Qu.:78.00   1st Qu.:2.200   6 GB   :144
 Median : 275.73   Median :82.00   Median :2.400   4 GB   : 81
 Mean   : 345.50   Mean   :81.37   Mean   :2.526   12 GB  : 38
 3rd Qu.: 395.62   3rd Qu.:85.00   3rd Qu.:2.900   16 GB  :  7
 Max.   :2877.34   Max.   :89.00   Max.   :3.220   3 GB   :  3
                                                   (Other):  2
  battery_mAh     f_camera_MP                os        displaySize
 Min.   : 3095   Min.   : 1.00   Android v12  :255   Min.   :5.900
 1st Qu.: 4600   1st Qu.:13.00   Android v11  :150   1st Qu.:6.500
 Median : 5000   Median :16.00   Android v13  : 68   Median :6.600
 Mean   : 4934   Mean   :18.16   Android v10  : 17   Mean   :6.593
 3rd Qu.: 5000   3rd Qu.:20.00   Android v10.0:  4   3rd Qu.:6.670
 Max.   :22000   Max.   :60.00   iOS v15      :  3   Max.   :6.950
                                 (Other)      :  9
   displayHz
 Min.   : 90.0
 1st Qu.: 90.0
 Median :120.0
```

```
 Mean   :110.5
 3rd Qu.:120.0
 Max.   :240.0
```

Second Dataset

The first dataset looked like this before processing

```
# A tibble: 6 x 12
  Brands  Models      Colors        Memory Storage Camera Rating `Selling Price`
  <chr>   <chr>       <chr>         <chr>  <chr>   <chr>   <dbl>           <dbl>
1 SAMSUNG GALAXY M31S Mirage Black  8 GB   128 GB  Yes       4.3           19330
2 Nokia   3.2         Steel         2 GB   16 GB   Yes       3.8           10199
3 realme  C2          Diamond Black 2 GB   <NA>    Yes       4.4            6999
4 Infinix Note 5      Ice Blue      4 GB   64 GB   Yes       4.2           12999
5 Apple   iPhone 11   Black         4GB    64 GB   Yes       4.6           49900
6 GIONEE  L800        Black         8 MB   16 MB   Yes       4              2199
# i 4 more variables: `Original Price` <dbl>, Mobile <chr>, Discount <dbl>,
#   `discount percentage` <dbl>
```

It is also a tabular data set with information on mobile phones. This data set differs from the first because it has less columns that are useful for predicting price but it has more rows.

Cleaning P1

To start we removed unneeded columns. These were models, Camera, selling price, mobile, discount, and discount percentage. We then make all the column names lowercase. We then made all the data in the colors and brands columns lowercase. We then removed the underscore from the original_price column name. We then converted the price to dollars. We them made the memory, brands, and storage columns factors.

```
# A tibble: 6 x 5
  brands  memory storage rating original_price
  <fct>   <fct>  <fct>    <dbl>          <dbl>
1 samsung 8 GB   128 GB     4.3          252.
2 nokia   2 GB   16 GB      3.8          122.
3 infinix 4 GB   64 GB      4.2          156.
4 apple   4GB    64 GB      4.6          599.
5 gionee  8 MB   16 MB      4             26.4
6 apple   3 GB   64 GB      4.6          575.
```

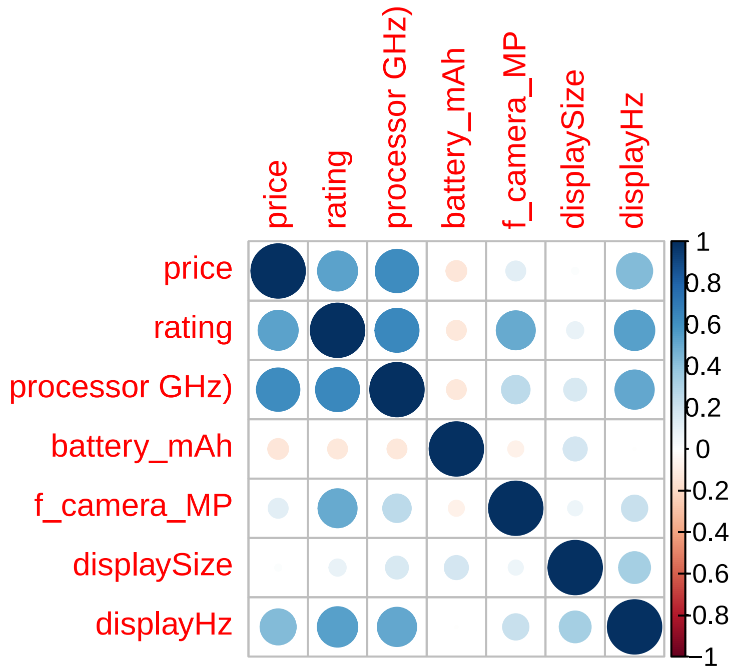
General Analysis

After cleaning:

```
# A tibble: 6 x 5
  brands  memory storage rating original_price
  <fct>   <fct>  <fct>    <dbl>          <dbl>
1 samsung 8 GB   128 GB     4.3          252.
2 nokia   2 GB   16 GB      3.8          122.
3 infinix 4 GB   64 GB      4.2          156.
4 apple   4GB    64 GB      4.6          599.
5 gionee  8 MB   16 MB      4             26.4
6 apple   3 GB   64 GB      4.6          575.
```

```
     brands        memory        storage         rating       original_price
 samsung:685   4 GB   :711   64 GB  :757   Min.   :2.300   Min.   :  12.0
 apple   :319   3 GB   :479   128 GB :720   1st Qu.:4.100   1st Qu.: 124.7
 realme :281   6 GB   :444   32 GB  :545   Median :4.300   Median : 195.6
 oppo   :251   2 GB   :376   16 GB  :312   Mean   :4.241   Mean   : 319.9
```

```
xiaomi :191   8 GB    :326   256 GB :216   3rd Qu.:4.400   3rd Qu.: 360.0
nokia  :184   1 GB    :193   8 GB   :133   Max.   :5.000   Max.   :2280.0
(Other):986   (Other):368   (Other):214
```



Model Creation: Predicting Price

To start with our introductory model, we have decided on a multi-linear regression model. This will set a foundation for the model complex models all predicting price. In the future, we plan to extend apon this will stepwise regression, and a neural network.

Split Test and Train Data:

======= ### Model Creation: Predicting Price

To start with our introductory model, we have decided on a multi-linear regression model. This will set a foundation for the more complex models all predicting price. In the future, we plan to extend apon this will stepwise regression and a neural network.

Fit Two Linear Regression Models:

Summary of Models:

```
Call:
lm(formula = price ~ ., data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-228.09  -55.93  -14.42   40.95  665.74

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)        -5.080e+02  3.026e+02  -1.679 0.093801 .
rating              1.754e+01  1.923e+00   9.121  < 2e-16 ***
`processor GHz)`    1.423e+02  1.607e+01   8.855  < 2e-16 ***
ram16 GB           4.169e+01  4.240e+01   0.983 0.326042
ram18 GB           3.340e+02  7.465e+01   4.474 9.58e-06 ***
ram3 GB            7.775e+01  7.126e+01   1.091 0.275799
ram4 GB           -5.905e+01  2.889e+01  -2.044 0.041487 *
ram6 GB           -1.195e+02  2.246e+01  -5.320 1.59e-07 ***
ram8 GB           -9.567e+01  1.894e+01  -5.052 6.22e-07 ***
battery_mAh        -4.114e-04  5.388e-03  -0.076 0.939174
f_camera_MP        -2.566e+00  5.697e-01  -4.505 8.34e-06 ***
osAndroid v10.0     2.238e+01  5.705e+01   0.392 0.695058
osAndroid v11      -8.211e+01  2.645e+01  -3.104 0.002020 **
osAndroid v12      -6.965e+01  2.570e+01  -2.710 0.006966 **
osAndroid v13      -5.858e+01  2.830e+01  -2.070 0.039017 *
osEMUI v12         -8.082e+01  1.050e+02  -0.770 0.441879
osHarmony v2.0     -1.405e+02  1.058e+02  -1.328 0.184904
osHarmonyOS         8.734e+01  1.074e+02   0.813 0.416422
osHarmonyOS v2.0   -1.582e+01  1.046e+02  -0.151 0.879849
osHongmeng OS v3.0  2.303e+03  1.064e+02  21.642  < 2e-16 ***
osHongmeng OS v4.0  6.622e+02  1.049e+02   6.311 6.29e-10 ***
osiOS v15           1.266e+03  6.576e+01  19.258  < 2e-16 ***
osiOS v15.0         9.982e+02  6.767e+01  14.750  < 2e-16 ***
displaySize        -1.389e+02  3.901e+01  -3.562 0.000405 ***
displayHz           1.487e+00  3.293e-01   4.517 7.89e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 101.1 on 481 degrees of freedom
Multiple R-squared:  0.8629,    Adjusted R-squared:  0.856
F-statistic: 126.1 on 24 and 481 DF,  p-value: < 2.2e-16
```

```
Call:
lm(formula = original_price ~ ., data = df2)

Residuals:
    Min      1Q  Median      3Q     Max 
-639.24  -77.98   -8.29   46.11 1462.30 

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   1488.684    109.378  13.610  < 2e-16 ***
brandsasus                    -491.833     24.253 -20.279  < 2e-16 ***
brandsgionee                  -517.992     25.129 -20.613  < 2e-16 ***
brandsgoogle pixel              -2.467     35.188  -0.070 0.944121    
brandshtc                     -281.096     29.623  -9.489  < 2e-16 ***
brandsinfinix                 -582.695     21.739 -26.804  < 2e-16 ***
brandsiqoo                    -604.930     77.309  -7.825 7.12e-15 ***
brandslenovo                  -504.803     24.749 -20.397  < 2e-16 ***
brandslg                      -398.468     25.380 -15.700  < 2e-16 ***
brandsmotorola                -468.750     24.288 -19.300  < 2e-16 ***
brandsnokia                   -482.104     23.818 -20.241  < 2e-16 ***
brandsoppo                    -552.058     20.076 -27.498  < 2e-16 ***
brandspoco                    -606.829     26.194 -23.167  < 2e-16 ***
brandsrealme                  -612.254     19.535 -31.341  < 2e-16 ***
brandssamsung                 -426.849     18.813 -22.689  < 2e-16 ***
brandsvivo                    -566.371     23.302 -24.305  < 2e-16 ***
brandsxiaomi                  -563.306     20.867 -26.995  < 2e-16 ***
memory1.5 GB                    -7.670     34.218  -0.224 0.822665    
memory10 MB                    -15.231    141.636  -0.108 0.914373    
memory100 MB                   -61.859    288.988  -0.214 0.830520    
memory12 GB                    469.002     37.802  12.407  < 2e-16 ***
memory128 MB                   -18.251    196.796  -0.093 0.926115    
memory153 MB                 -1309.630    183.502  -7.137 1.21e-12 ***
memory16 GB                    731.421    172.211   4.247 2.23e-05 ***
memory16 MB                     -9.389    113.022  -0.083 0.933801    
memory2 GB                      -4.512     20.620  -0.219 0.826795    
memory2 MB                      -1.560    157.219  -0.010 0.992083    
memory3 GB                      64.776     22.653   2.859 0.004275 ** 
memory30 MB                     49.614    213.551   0.232 0.816301    
memory32 MB                      3.628    116.866   0.031 0.975234    
memory4 GB                     143.757     25.813   5.569 2.80e-08 ***
memory4 MB                       5.756    120.911   0.048 0.962037    
memory46 MB                    -11.058    167.606  -0.066 0.947403    
memory4GB                      259.285     27.958   9.274  < 2e-16 ***
```

```
memory512 MB                    -11.839     46.844  -0.253 0.800492    
memory6 GB                      203.957     27.609   7.387 1.96e-13 ***
memory64 MB                       8.718    165.345   0.053 0.957955    
memory768 MB                    -22.954     80.677  -0.285 0.776034    
memory8 GB                      271.173     29.204   9.285  < 2e-16 ***
memory8 MB                       13.266    126.205   0.105 0.916295    
storage10 MB                  -1211.900    214.648  -5.646 1.81e-08 ***
storage100 MB                 -1301.445    246.001  -5.290 1.31e-07 ***
storage128 GB                 -1133.096     70.270 -16.125  < 2e-16 ***
storage128 MB                 -1238.420    186.657  -6.635 3.88e-11 ***
storage129 GB                 -1180.067    120.111  -9.825  < 2e-16 ***
storage130 GB                 -1168.477    138.156  -8.458  < 2e-16 ***
storage140 MB                 -1172.078    189.277  -6.192 6.78e-10 ***
storage153 MB                        NA         NA      NA       NA    
storage16 GB                  -1160.439     73.034 -15.889  < 2e-16 ***
storage16 MB                  -1227.396    146.969  -8.351  < 2e-16 ***
storage2 MB                   -1299.064    125.149 -10.380  < 2e-16 ***
storage256 GB                  -892.080     70.121 -12.722  < 2e-16 ***
storage256 MB                 -1144.196    155.237  -7.371 2.22e-13 ***
storage32 GB                  -1158.027     71.743 -16.141  < 2e-16 ***
storage4 GB                   -1196.870     85.441 -14.008  < 2e-16 ***
storage4 MB                   -1260.412    144.187  -8.742  < 2e-16 ***
storage512 GB                  -392.645     73.653  -5.331 1.05e-07 ***
storage512 MB                 -1224.462    147.505  -8.301  < 2e-16 ***
storage64 GB                  -1176.786     70.468 -16.700  < 2e-16 ***
storage64 MB                  -1225.131    170.344  -7.192 8.13e-13 ***
storage8 GB                   -1186.759     76.455 -15.522  < 2e-16 ***
storage8 MB                   -1265.865    183.724  -6.890 6.85e-12 ***
storageExpandable Upto 16 GB -1285.055    149.474  -8.597  < 2e-16 ***
storageExpandable Upto 32 GB -1208.655    136.399  -8.861  < 2e-16 ***
rating                           64.971     17.707   3.669 0.000248 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 167.4 on 2833 degrees of freedom
Multiple R-squared:  0.7616,    Adjusted R-squared:  0.7563
F-statistic: 143.6 on 63 and 2833 DF,  p-value: < 2.2e-16
```

Initial Thoughts:

We would expect to see price increase as the newer specs for phones are released and are put onto the market. Thus, we would expect things like memory, storage, ram, and other specs to be significant predictors for both of our data sets. As we can see this is the true.

One noteworthy observation we noticed in both models in the high level of Adjusted R-Squared(.85 and .75). This can indicate over fitting, or it can indicator an accurate model. This will need to be verified on the test data with some prediction tests after our interpretations.

Interpreting our coefficients:

For the first model,

Intercept: When all other predictor variables are zero, the estimated price of the product is approximately -$5.080e+02.

Rating: For every one-unit increase in the rating, the price is estimated to increase by approximately $1.754e+01, holding all other variables constant.

Processor(GHz): For every one-unit increase in the processor GHz, the price is estimated to increase by approximately $1.423e+02, holding all other variables constant.

RAM(16GB): Phones with 16GB of RAM are estimated to have a price increase of approximately $4.169e+01 compared to the reference category, holding all other variables constant.

RAM(18GB): Phones with 18GB of RAM are estimated to have a price increase of approximately $3.340e+02 compared to the reference category, holding all other variables constant.

RAM(3GB): Phones with 3GB of RAM are estimated to have a price increase of approximately $7.775e+01 compared to the reference category, holding all other variables constant.

RAM(4GB): Phones with 4GB of RAM are estimated to have a price decrease of approximately $5.905e+01 compared to the reference category, holding all other variables constant.

RAM(6GB): Phones with 6GB of RAM are estimated to have a price decrease of approximately $1.195e+02 compared to the reference category, holding all other variables constant.

Ram(8GB): Phones with 6GB of RAM are estimated to have a price decrease of approximately $9.567e+01 compared to the reference category, holding all other variables constant.

Battery(mAH): For every one-unit increase in battery mAh, the price is estimated to decrease by approximately $4.114e-04, holding all other variables constant.

Camera(MP): For every one-unit increase in front camera megapixels, the price is estimated to decrease by approximately $2.566, holding all other variables constant.

Operating System: Devices with Android v10.0: On average, devices with Android v10.0 have prices that are $22.38 higher than the reference category, holding all other variables constant. Android v11: On average, devices with Android v11 have prices that are $82.11 lower than the reference category, holding all other variables constant. Android v12: On average, devices with Android v12 have prices that are $69.65 lower than the reference category, holding all other variables constant. Android v13: On average, devices with Android v13 have prices that are $58.58 lower than the reference category, holding all other variables constant. EMUI v12: On average, devices with EMUI v12 have prices that are $80.82 lower than the reference category, holding all other variables constant. Harmony v2.0: On average, devices with Harmony v2.0

have prices that are \$140.50 lower than the reference category, holding all other variables constant. HarmonyOS: On average, devices with HarmonyOS have prices that are \$87.34 higher than the reference category, holding all other variables constant. HarmonyOS v2.0: On average, devices with HarmonyOS v2.0 have prices that are \$15.82 lower than the reference category, holding all other variables constant. Hongmeng OS v3.0: On average, devices with Hongmeng OS v3.0 have prices that are \$2303 higher than the reference category, holding all other variables constant. Hongmeng OS v4.0: On average, devices with Hongmeng OS v4.0 have prices that are \$662.20 higher than the reference category, holding all other variables constant. iOS v15 and iOS v15.0: On average, devices with iOS v15 have prices that are \$1266 higher than the reference category, and devices with iOS v15.0 have prices that are \$998.20 higher than the reference category, holding all other variables constant.

Display size: For every one-unit increase in display size, the price is estimated to decrease by approximately \$1.389e+02, holding all other variables constant.

Display Hz: For every one-unit increase in display size, the price is estimated to decrease by approximately \$1.487, holding all other variables constant.

For the second model,

Operating System: Phones of the brands ASUS, Gionee, Google Pixel, HTC, Infinix, IQOO, Lenovo, LG, Motorola, Nokia, OPPO, POCO, Realme, Samsung, Vivo, and Xiaomi have an estimated price change of -491.83, -517.99, -2.47, -281.1, -582.7, -604.93, -504.80, -398.47, -468.75, -482.1, -552.06, -606.83, -612.25, -426.85, -566.37, and -563.31 respectively in dollars compared to the reference category holding all other variables constant.

Memory: Phones with RAM memory capacity of 1.5GB, 10MB, 100 MB, 12GB, 128MB, 153MB, 16GB, 16 MB, 2GB, 2MB, 3GB, 30MB, 32MB, 4GB, 46MB, 4G, 512MB, 6GB, 64MB, 768MB, 8GB, and 8MB have an estimated price change of -7.67,-15.23,-61.86, 469.00, -18.25, -1309.63, 731.42, -9.39, -4.51, -1.56, 64.78, 49.61, 3.63, 143.76, 5.76, -11.06, 259.28, -11.84, 203.96, 8.72, -22.95, 271.17, and 13.27 respectively in dollars compared to the reference category holding all other variables constant.

Storage: Phones with storage capacity of 10MB, 100MB, 128GB, 128MB, 129GB, 130GB, 140MB, 153MB, 16GB, 16MB, 2MB, 256GB, 256MB, 32GB, 4GB, 4Mb, 512GB, 512MB, 64GB, 64MB, 8 GB, and 8 MB have an estimated price change of -1211.90, -1301.45, 1133.10,-1238.42, -1180.07, -1168.48, -1172.08, NULL(due to multicollinearity), 1160.44, -1227.40, -1299.06, 892.08, -1144.20, 1158.03, 1196.87, -1260.41, 392.65, -1224.56, 1176.79, -1225.13, 1186.76, and -1265.87 respectively in dollars when compared to the reference category holding all other predictors constant. Rating: For every one unit increase in rating we see a price change of \$64.97 when holding all other predictors constant.

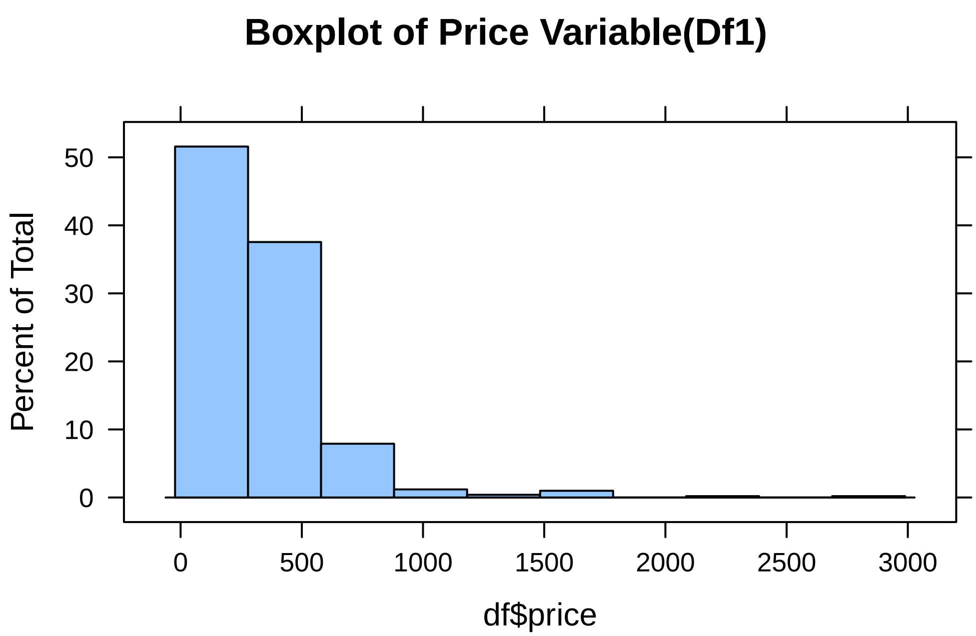
Rating: For every one-unit increase in rating, the price is estimated to increase by approximately \$119.62, holding all other variables constant.

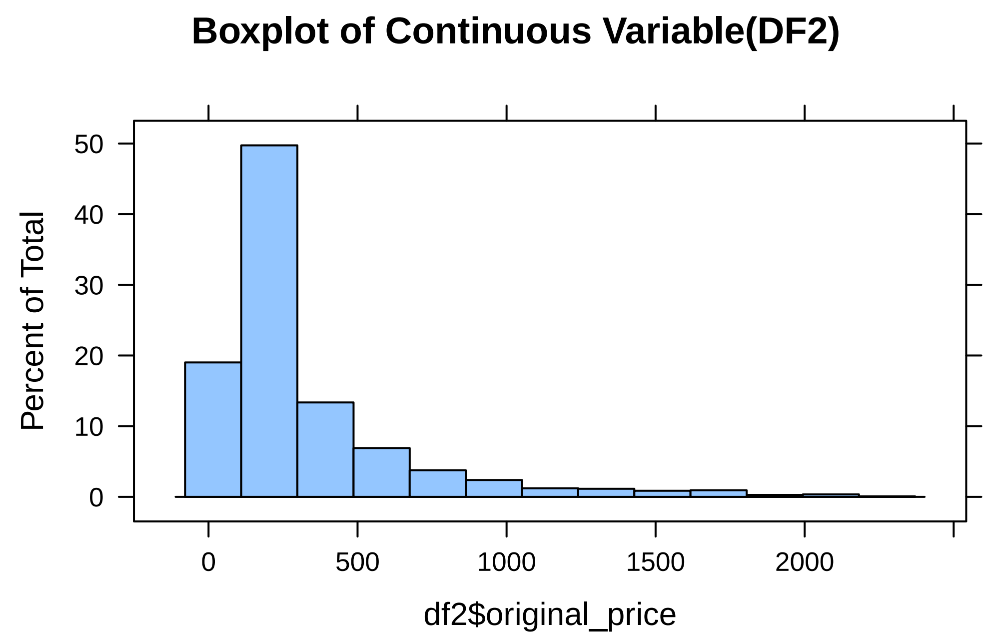
=======

```
[1] 98.61083

[1] 165.5091

[1] 345.4993

[1] 319.8801
```

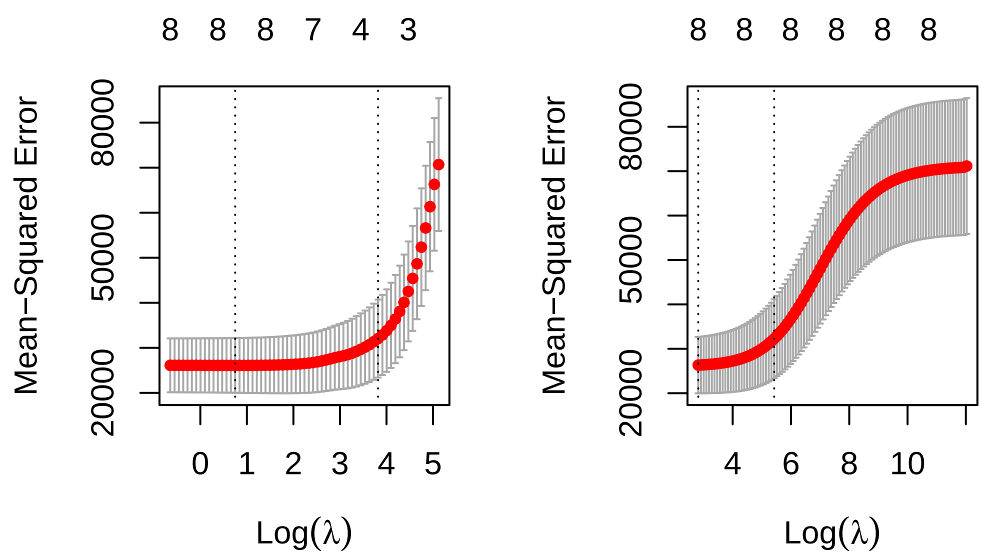
As you see, we have a RMSE of 98.6 and 165.5 for both data sets respectively. On average, this means our model is off by about $98.6 and $165.5. Since the average price of the phones is around $345.50 and $319.88, our model is not too inaccurate, but it can be improved.

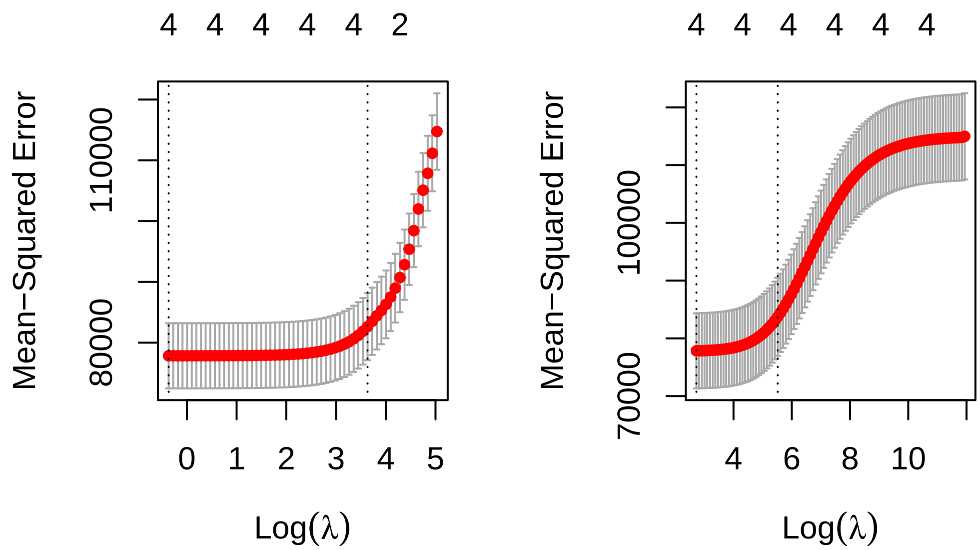
The RMSE variation is likely caused by the heavy right-skewness in the price variable of more expensive phones. More expensive prices leads to an exponential decay of more expensive components in a thinner market; thus, leading to an exponentially distribution price vector column.





Lasso and Ridge:





Stepwise:

```
Start:  AIC=5653.61
price ~ 1

                    Df Sum of Sq      RSS    AIC
+ os                12  20405396 15476192 5252.1
+ `processor GHz)`   1  14201208 21680381 5400.7
+ rating             1  10260354 25621234 5485.2
+ ram                6   8080244 27801345 5536.5
+ displayHz          1   6707698 29173891 5550.9
+ battery_mAh        1    677355 35204234 5646.0
+ f_camera_MP        1    570140 35311449 5647.5
<none>                           35881589 5653.6
+ displaySize        1      5945 35875644 5655.5

Step:  AIC=5252.11
price ~ os

                    Df Sum of Sq      RSS    AIC
+ rating             1   7422062  8054130 4923.6
+ `processor GHz)`   1   7269263  8206930 4933.1
+ ram                6   6360285  9115908 4996.3
```

```
+ displayHz         1   4136106 11340086 5096.8
+ f_camera_MP       1    746681 14729511 5229.1
+ displaySize       1     63509 15412684 5252.0
<none>                          15476192 5252.1
+ battery_mAh       1     59302 15416890 5252.2

Step:  AIC=4923.63
price ~ os + rating

                   Df Sum of Sq     RSS    AIC
+ `processor GHz)`  1   1646077 6408053 4809.9
+ ram               6   1586293 6467838 4824.6
+ displayHz         1    561247 7492883 4889.1
+ f_camera_MP       1    338351 7715779 4903.9
<none>                          8054130 4923.6
+ battery_mAh       1      5888 8048242 4925.3
+ displaySize       1      4518 8049613 4925.3

Step:  AIC=4809.94
price ~ os + rating + `processor GHz)`

              Df Sum of Sq     RSS    AIC
+ ram          6    979346 5428708 4738.0
+ displayHz    1    240046 6168008 4792.6
+ f_camera_MP  1    225907 6182146 4793.8
<none>                     6408053 4809.9
+ displaySize  1     19254 6388799 4810.4
+ battery_mAh  1     16034 6392019 4810.7

Step:  AIC=4738.02
price ~ os + rating + `processor GHz)` + ram

              Df Sum of Sq     RSS    AIC
+ f_camera_MP  1    249163 5179544 4716.2
+ displayHz    1    155134 5273574 4725.4
+ displaySize  1     51875 5376833 4735.2
<none>                     5428708 4738.0
+ battery_mAh  1       319 5428389 4740.0

Step:  AIC=4716.25
price ~ os + rating + `processor GHz)` + ram + f_camera_MP

              Df Sum of Sq     RSS    AIC
```

```
+ displayHz    1    125900 5053644 4705.8
+ displaySize  1     50356 5129189 4713.3
<none>                     5179544 4716.2
+ battery_mAh  1       872 5178672 4718.2

Step:  AIC=4705.8
price ~ os + rating + `processor GHz)` + ram + f_camera_MP +
    displayHz

              Df Sum of Sq     RSS    AIC
+ displaySize  1    133193 4920452 4694.3
<none>                     5053644 4705.8
+ battery_mAh  1      3476 5050168 4707.4

Step:  AIC=4694.28
price ~ os + rating + `processor GHz)` + ram + f_camera_MP +
    displayHz + displaySize

              Df Sum of Sq     RSS    AIC
<none>                     4920452 4694.3
+ battery_mAh  1    59.627 4920392 4696.3

Start:  AIC=33757.1
original_price ~ 1

          Df Sum of Sq       RSS   AIC
+ storage 24 178005343 154834932 31588
+ brands  16 152690410 180149866 32011
+ memory  23 136380092 196460183 32276
+ rating   1  67635920 265204355 33101
<none>                 332840275 33757

Step:  AIC=31588.01
original_price ~ storage

         Df Sum of Sq       RSS   AIC
+ brands 16  66089397  88745535 30008
+ memory 22  27970150 126864782 31055
+ rating  1   7926316 146908616 31438
<none>                154834932 31588

Step:  AIC=30007.58
```

```
original_price ~ storage + brands

         Df Sum of Sq      RSS   AIC
+ memory 22   9010140 79735395 29741
+ rating  1    602183 88143352 29990
<none>                88745535 30008

Step:  AIC=29741.43
original_price ~ storage + brands + memory

         Df Sum of Sq      RSS   AIC
+ rating  1    377153 79358242 29730
<none>                79735395 29741

Step:  AIC=29729.7
original_price ~ storage + brands + memory + rating

Start:  AIC=4696.28
price ~ rating + `processor GHz)` + ram + battery_mAh + f_camera_MP +
    os + displaySize + displayHz

                   Df Sum of Sq      RSS    AIC
- battery_mAh       1        60  4920452 4694.3
<none>                           4920392 4696.3
- displaySize       1    129776  5050168 4707.4
- f_camera_MP       1    207604  5127996 4715.2
- displayHz         1    208756  5129148 4715.3
- `processor GHz)`  1    802116  5722508 4770.7
- ram               6    946025  5866417 4773.3
- rating            1    850993  5771385 4775.0
- os               12  13374882 18295274 5336.8

Step:  AIC=4694.28
price ~ rating + `processor GHz)` + ram + f_camera_MP + os +
    displaySize + displayHz

                   Df Sum of Sq      RSS    AIC
<none>                           4920452 4694.3
- displaySize       1    133193  5053644 4705.8
- f_camera_MP       1    207549  5128001 4713.2
- displayHz         1    208737  5129189 4713.3
- `processor GHz)`  1    810063  5730514 4769.4
```

```
- ram               6    960551  5881003 4772.5
- rating            1    852479  5772930 4773.1
- os               12  13530876 18451328 5339.1

Start:  AIC=29729.7
original_price ~ brands + memory + storage + rating

          Df Sum of Sq       RSS   AIC
<none>                  79358242 29730
- rating   1    377153  79735395 29741
- memory  22   8785110  88143352 29990
- storage 23  28630931 107989173 30576
- brands  16  44457006 123815248 30986
```

Neural Network:

```
Epoch 1/50
Train metrics: Loss: 112506.5469 - Acc: 0
Epoch 2/50
Train metrics: Loss: 85766.7578 - Acc: 0
Epoch 3/50
Train metrics: Loss: 79610.3672 - Acc: 0
Epoch 4/50
Train metrics: Loss: 81697.3203 - Acc: 0
Epoch 5/50
Train metrics: Loss: 80163.8359 - Acc: 0
Epoch 6/50
Train metrics: Loss: 79525.0781 - Acc: 0
Epoch 7/50
Train metrics: Loss: 65035.8555 - Acc: 0
Epoch 8/50
Train metrics: Loss: 80852.9844 - Acc: 0
Epoch 9/50
Train metrics: Loss: 75625.9766 - Acc: 0
Epoch 10/50
Train metrics: Loss: 78820.7578 - Acc: 0
Epoch 11/50
Train metrics: Loss: 74856.2812 - Acc: 0
Epoch 12/50
Train metrics: Loss: 78154.7969 - Acc: 0
Epoch 13/50
Train metrics: Loss: 72478.1719 - Acc: 0
```

```
Epoch 14/50
Train metrics: Loss: 80209.3281 - Acc: 0
Epoch 15/50
Train metrics: Loss: 76235.0625 - Acc: 0
Epoch 16/50
Train metrics: Loss: 78544.0078 - Acc: 0
Epoch 17/50
Train metrics: Loss: 77271.5938 - Acc: 0
Epoch 18/50
Train metrics: Loss: 76881.2109 - Acc: 0
Epoch 19/50
Train metrics: Loss: 71989.1406 - Acc: 0
Epoch 20/50
Train metrics: Loss: 75301.0859 - Acc: 0
Epoch 21/50
Train metrics: Loss: 65565.375 - Acc: 0
Epoch 22/50
Train metrics: Loss: 66891.6641 - Acc: 0
Epoch 23/50
Train metrics: Loss: 69964.75 - Acc: 0
Epoch 24/50
Train metrics: Loss: 68333.7266 - Acc: 0
Epoch 25/50
Train metrics: Loss: 63634.2578 - Acc: 0
Epoch 26/50
Train metrics: Loss: 51410.9688 - Acc: 0
Epoch 27/50
Train metrics: Loss: 61619.1523 - Acc: 0
Epoch 28/50
Train metrics: Loss: 63912.418 - Acc: 0
Epoch 29/50
Train metrics: Loss: 54266.207 - Acc: 0
Epoch 30/50
Train metrics: Loss: 42864.8438 - Acc: 0
Epoch 31/50
Train metrics: Loss: 52724.9883 - Acc: 0
Epoch 32/50
Train metrics: Loss: 53360.2852 - Acc: 0
Epoch 33/50
Train metrics: Loss: 55283.2695 - Acc: 0
Epoch 34/50
Train metrics: Loss: 56338.1562 - Acc: 0
Epoch 35/50
```

```
Train metrics: Loss: 51082.9453 - Acc: 0
Epoch 36/50
Train metrics: Loss: 51757.2344 - Acc: 0
Epoch 37/50
Train metrics: Loss: 64937.6875 - Acc: 0
Epoch 38/50
Train metrics: Loss: 65497.8164 - Acc: 0
Epoch 39/50
Train metrics: Loss: 56118.0156 - Acc: 0
Epoch 40/50
Train metrics: Loss: 43047.8828 - Acc: 0
Epoch 41/50
Train metrics: Loss: 54793.7383 - Acc: 0
Epoch 42/50
Train metrics: Loss: 52634.1758 - Acc: 0
Epoch 43/50
Train metrics: Loss: 49707.7617 - Acc: 0
Epoch 44/50
Train metrics: Loss: 50425.582 - Acc: 0
Epoch 45/50
Train metrics: Loss: 57680.4844 - Acc: 0
Epoch 46/50
Train metrics: Loss: 61455.3828 - Acc: 0
Epoch 47/50
Train metrics: Loss: 50908.5195 - Acc: 0
Epoch 48/50
Train metrics: Loss: 52184.6172 - Acc: 0
Epoch 49/50
Train metrics: Loss: 47321.5547 - Acc: 0
Epoch 50/50
Train metrics: Loss: 50804.043 - Acc: 0
```

```
Epoch 1/50
Train metrics: Loss: 165096.5938 - Acc: 0
Epoch 2/50
Train metrics: Loss: 69674.6094 - Acc: 0
Epoch 3/50
Train metrics: Loss: 35680.7695 - Acc: 0
Epoch 4/50
Train metrics: Loss: 31565.9258 - Acc: 0
Epoch 5/50
Train metrics: Loss: 30040.6953 - Acc: 0
```

```
Epoch 6/50
Train metrics: Loss: 28820.0645 - Acc: 0
Epoch 7/50
Train metrics: Loss: 27476.5332 - Acc: 0
Epoch 8/50
Train metrics: Loss: 26603.627 - Acc: 0
Epoch 9/50
Train metrics: Loss: 25764.3359 - Acc: 0
Epoch 10/50
Train metrics: Loss: 25892.1738 - Acc: 0
Epoch 11/50
Train metrics: Loss: 25045.9062 - Acc: 0
Epoch 12/50
Train metrics: Loss: 24550.5312 - Acc: 0
Epoch 13/50
Train metrics: Loss: 23697.8906 - Acc: 0
Epoch 14/50
Train metrics: Loss: 23902.9414 - Acc: 0
Epoch 15/50
Train metrics: Loss: 23255.5156 - Acc: 0
Epoch 16/50
Train metrics: Loss: 23440.5918 - Acc: 0
Epoch 17/50
Train metrics: Loss: 22738.5293 - Acc: 0
Epoch 18/50
Train metrics: Loss: 23325.0371 - Acc: 0
Epoch 19/50
Train metrics: Loss: 22682.6406 - Acc: 0
Epoch 20/50
Train metrics: Loss: 22552.627 - Acc: 0
Epoch 21/50
Train metrics: Loss: 21867 - Acc: 0
Epoch 22/50
Train metrics: Loss: 22269.502 - Acc: 0
Epoch 23/50
Train metrics: Loss: 22073.7207 - Acc: 0
Epoch 24/50
Train metrics: Loss: 21878.0957 - Acc: 0
Epoch 25/50
Train metrics: Loss: 21743.793 - Acc: 0
Epoch 26/50
Train metrics: Loss: 21527.5625 - Acc: 0
Epoch 27/50
```

```
Train metrics: Loss: 21715.8223 - Acc: 0
Epoch 28/50
Train metrics: Loss: 21873.252 - Acc: 0
Epoch 29/50
Train metrics: Loss: 21451.8223 - Acc: 0
Epoch 30/50
Train metrics: Loss: 21179.2285 - Acc: 0
Epoch 31/50
Train metrics: Loss: 21086.6211 - Acc: 0
Epoch 32/50
Train metrics: Loss: 21401.8555 - Acc: 0
Epoch 33/50
Train metrics: Loss: 20948.5293 - Acc: 0
Epoch 34/50
Train metrics: Loss: 21148.4258 - Acc: 0
Epoch 35/50
Train metrics: Loss: 21381.3418 - Acc: 0
Epoch 36/50
Train metrics: Loss: 20817.4121 - Acc: 0
Epoch 37/50
Train metrics: Loss: 21304.4316 - Acc: 0
Epoch 38/50
Train metrics: Loss: 21120.2734 - Acc: 0
Epoch 39/50
Train metrics: Loss: 21004.4629 - Acc: 0
Epoch 40/50
Train metrics: Loss: 20604.1328 - Acc: 0
Epoch 41/50
Train metrics: Loss: 20290.3477 - Acc: 0
Epoch 42/50
Train metrics: Loss: 20742.4863 - Acc: 0
Epoch 43/50
Train metrics: Loss: 20452.8125 - Acc: 0
Epoch 44/50
Train metrics: Loss: 20365.6172 - Acc: 0
Epoch 45/50
Train metrics: Loss: 20576.9746 - Acc: 0
Epoch 46/50
Train metrics: Loss: 20645.0684 - Acc: 0
Epoch 47/50
Train metrics: Loss: 20185.3242 - Acc: 0
Epoch 48/50
Train metrics: Loss: 20478.6211 - Acc: 0
```

```
Epoch 49/50
Train metrics: Loss: 20762.3398 - Acc: 0
Epoch 50/50
Train metrics: Loss: 20324.2051 - Acc: 0
```

Summary of results

```
       Model Dataset      RMSE
1     Linear       1  98.61083
2      Lasso       1 173.73677
3      Ridge       1 170.54350
4   Forwards       1  98.61142
5  Backwards       1  98.61142
6   NNetwork       1 301.61023
7     Linear       2 165.50906
8      Lasso       2 286.82097
9      Ridge       2 288.88781
10  Forwards       2 165.50906
11 Backwards       2 165.50906
12  NNetwork       2 452.43591
```