STAT 701 Final Project - Laptop Pricing Analysis

Mike Guibas, Ashim Datta

4/7/2022

Contents

Executive Summary

Key Findings

We developed predictive models with the goal of predicting laptop price based on a bevvy of provided features. We applied clustering technique to learn more about different types of laptops within our data set. These clusters were later used as a feature along with various continuous and categorical variables to predict laptop price. We realized that the target variable (laptop price) was right-skewed so we applied a log transformation to the values

before starting our analysis and modeling. We also grouped multiple levels within some categorical variables that contained only a few records to aid in model development. Model performance was primarily assessed based on R^2 and RMSE in a validation data set which wasn't used during training across different predictive modelling approaches.

While we leveraged multiple predictive models, the Gradient Bossted Decision Tree (GBDT) model ultimately proved to be the most accurate in predicting laptop price. Our R^2'S were fairly high for all models. However, our RMSE on the actual prices was also fairly high whereas the RMSE on the log scale was fairly low. We believe we can further improve our models by applying various tuning techniques and introducing additional data for training.

Summary of Various R^2's and RMSE of the best models are given below:

Best Linear Model:

- The R^2 value of stepAIC is 0.720237665739132.
- The RMSE value of stepAIC is 298.724753420143.
- The R^2 value of stepAIC in log scale is 0.826822166731771.
- The RMSE value of stepAIC in log scale is 0.206641333801683.

Best Ensemble Model:

- The R^2 value for XGboost is 0.846570354015489.
- The RMSE value of XGboost is 218.695470595025.
- The R^2 value for XGboost in log scale is 0.886013879258513.
- The RMSE value of XGboost in log scale is 0.165913472726158.

Interpretations

Based on our validation metrics (R^2 and RMSE), we observed the following across different modeling techniques :

1. **Linear models** : StepAIC was a slightly better stepwise approach if we were to use a linear model to predict laptop prices. We also found that though regularization using lasso could give us a fairly good model (high R^2 and low RMSE), it wasn't better than our stepwise regression approach.
2. **Decision trees** : A basic tree and a pruned tree gave us worse predictive models than linear models.
3. **Ensemble methods** : GBDT model trained using XGboost gave us the best predictive model. Analysis of important variables across the ensemble approaches such as GBDT and random process revealed that our feature creation by clustering and our treatment of missing values in display size were useful. We also noticed that three of the top five variables in Random Forest and GBDT were different.
4. **Lastly**, our analysis using partial dependence suggested that higher performance laptops (i.e laptops with higher RAM, SSD, Graphic card) were priced higher. Intuitively, this makes sense.

Implications

Pricing products is a difficult problem for any e-commerce platform. The approach and findings here suggest that we can build a fairly accurate model to predict prices for a laptop based on common features. This type of model can be used by platforms such as Flipkart to recommend prices to their sellers to minimize friction on uploading their catalog to Flipkart. This is also a great prototype to prove that we can use similar techniques to price any other category of products which can further help smaller sellers when they try to list products on an e-commerce platform.

Introduction to the Data Set

Data source: https://www.kaggle.com/datasets/kuchhbhi/latest-laptop-price-list

Using data scraped from Indian e-commerce company Flipkart on 3/29/22, we developed models that predict laptop price based on various available attributes.

We know that that laptop price can often be driven by brand recognition (on top of actual performance attributes). For example, a basic macbook may command a higher price than a superior performing, custom-built computer due to Apple's brand recognition and amazing customer support. We also know that customers often care about features such has RAM, Graphic Card, Solid State Drive (ssd) etc. Additionally, online ratings and reviews can sometimes drive up a customer's willingness to pay for a laptop. Our data set has strong representative coverage across all these important features.

Our master data set, **laptop.data** has 896 observations and 23 variables. Below is a description of the columns:

```
##  [1] "brand"           "model"           "processor_brand" "processor_name"
##  [5] "processor_gnrtn" "ram_gb"          "ram_type"        "ssd"
##  [9] "hdd"             "os"              "os_bit"          "graphic_card_gb"
## [13] "weight"          "display_size"    "warranty"        "Touchscreen"
## [17] "msoffice"        "latest_price"    "old_price"       "discount"
## [21] "star_rating"     "ratings"         "reviews"
```

All the column names are self-explanatory. For example "touchscreen" column indicates if the laptop has a touch screen. Similarly "weight" tells us about the weight of the laptop.

Next, let us look at our data set a bit more closely to understand type of values each column has:

```
##     brand              model           processor_brand    processor_name
##  Length:896         Length:896         Length:896         Length:896
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  processor_gnrtn       ram_gb            ram_type             ssd
##  Length:896         Length:896         Length:896         Length:896
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##      hdd                 os               os_bit          graphic_card_gb
##  Length:896         Length:896         Length:896         Min.   :0.000
##  Class :character   Class :character   Class :character   1st Qu.:0.000
##  Mode  :character   Mode  :character   Mode  :character   Median :0.000
##                                                           Mean   :1.199
##                                                           3rd Qu.:2.000
##                                                           Max.   :8.000
##     weight          display_size          warranty     Touchscreen
##  Length:896         Length:896         Min.   :0.000   Length:896
##  Class :character   Class :character   1st Qu.:0.000   Class :character
##  Mode  :character   Mode  :character   Median :1.000   Mode  :character
##                                        Mean   :0.692
##                                        3rd Qu.:1.000
##                                        Max.   :3.000
##    msoffice          latest_price      old_price         discount
##  Length:896         Min.   : 13990   Min.   :     0   Min.   : 0.00
##  Class :character   1st Qu.: 45490   1st Qu.: 54940   1st Qu.:11.00
##  Mode  :character   Median : 63494   Median : 78052   Median :19.00
##                     Mean   : 76310   Mean   : 88134   Mean   :18.53
##                      3rd Qu.: 89090   3rd Qu.:111020   3rd Qu.:26.00
##                      Max.   :441990   Max.   :377798   Max.   :57.00
##   star_rating      ratings           reviews
##  Min.   :0.00   Min.   :    0.0   Min.   :   0.00
##  1st Qu.:0.00   1st Qu.:    0.0   1st Qu.:   0.00
##  Median :4.10   Median :   19.0   Median :   3.00
##  Mean   :2.98   Mean   :  367.4   Mean   :  46.15
##  3rd Qu.:4.40   3rd Qu.:  179.5   3rd Qu.:  23.25
##  Max.   :5.00   Max.   :15279.0   Max.   :1947.00
```

Cleaning Up the Data and Creating New Features

Making the Data More Intuitive

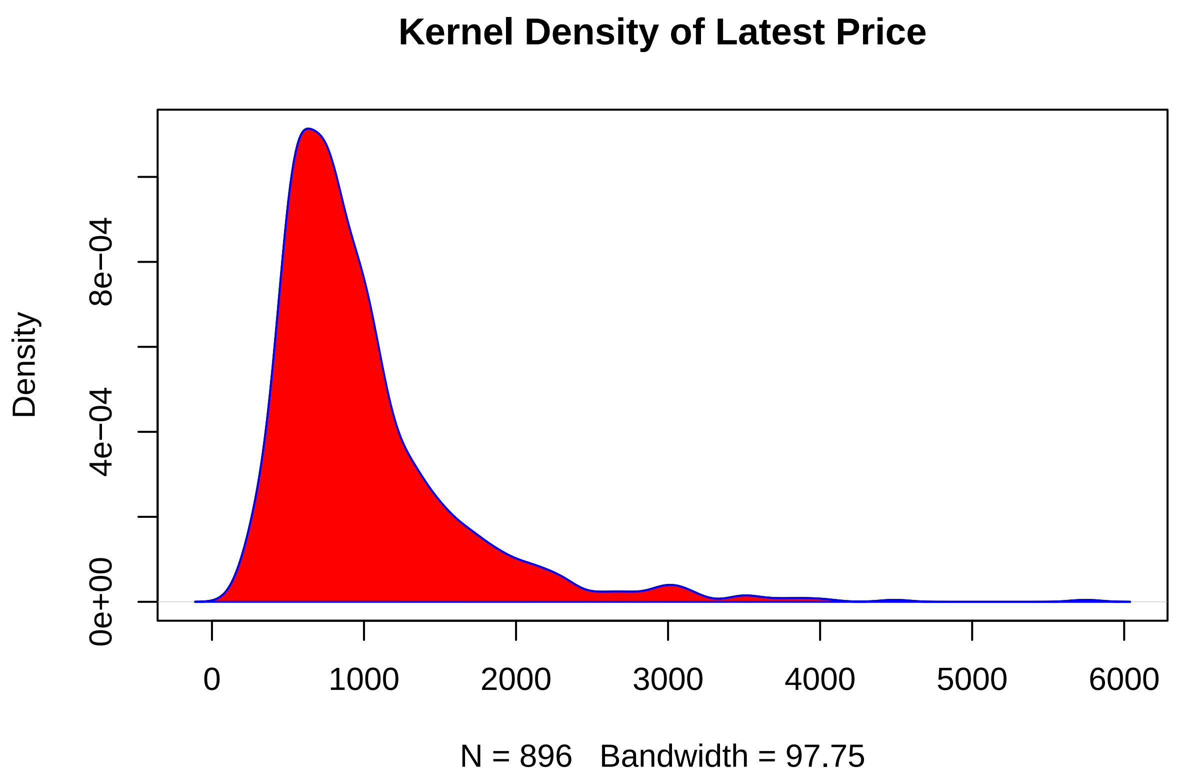
While our data set is relatively clean, we chose to further refine it to make analysis easier and more intuitive. For example, we removed redundant units / characters from columns (e.g., "GB" for ssd, hdd and "bit" for os_bit). We also identified missing data and reclassified those entries as "NA." The variables with the most NAs were model (95), processor_gnrtn (239), and display_size (332).

Converting Laptop Price to USD for Better Understanding

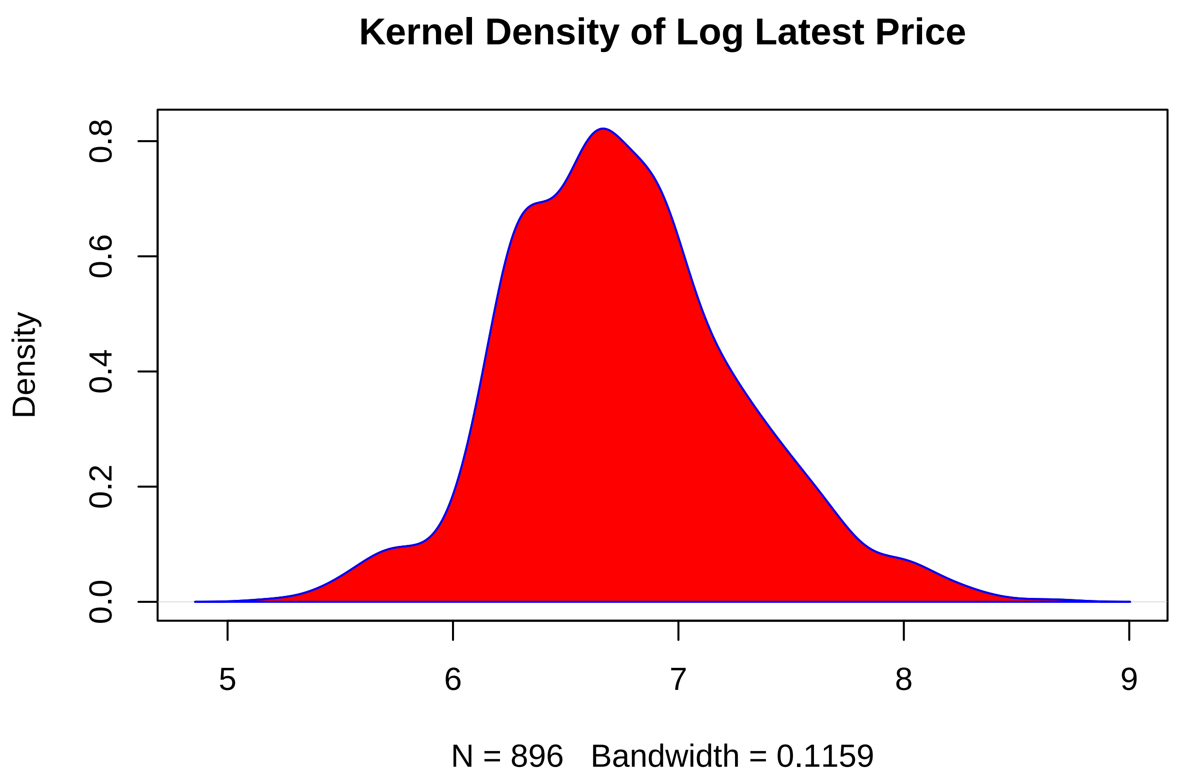
The information in this data set was pulled from Flipkart, so the prices are in Indian rupees. Although it was not necessary for modeling, we converted all prices from rupees to USD using a 1 to 0.013 ratio (current exchange rate as of 4/12/22) to make it easy for us to understand.

Applying Log Transform to the Target Variable

Next we studied our target variable a bit closer and observed the following distribution:



As you can see, our target variable is right skewed which prompted us to we apply a log transformation and create a new column called log_latest_price.



This transformation makes the data close to normal (shown above), so we will be using log_latest_price as our target y variable for the rest of our analysis.

Missing Value Treatment:

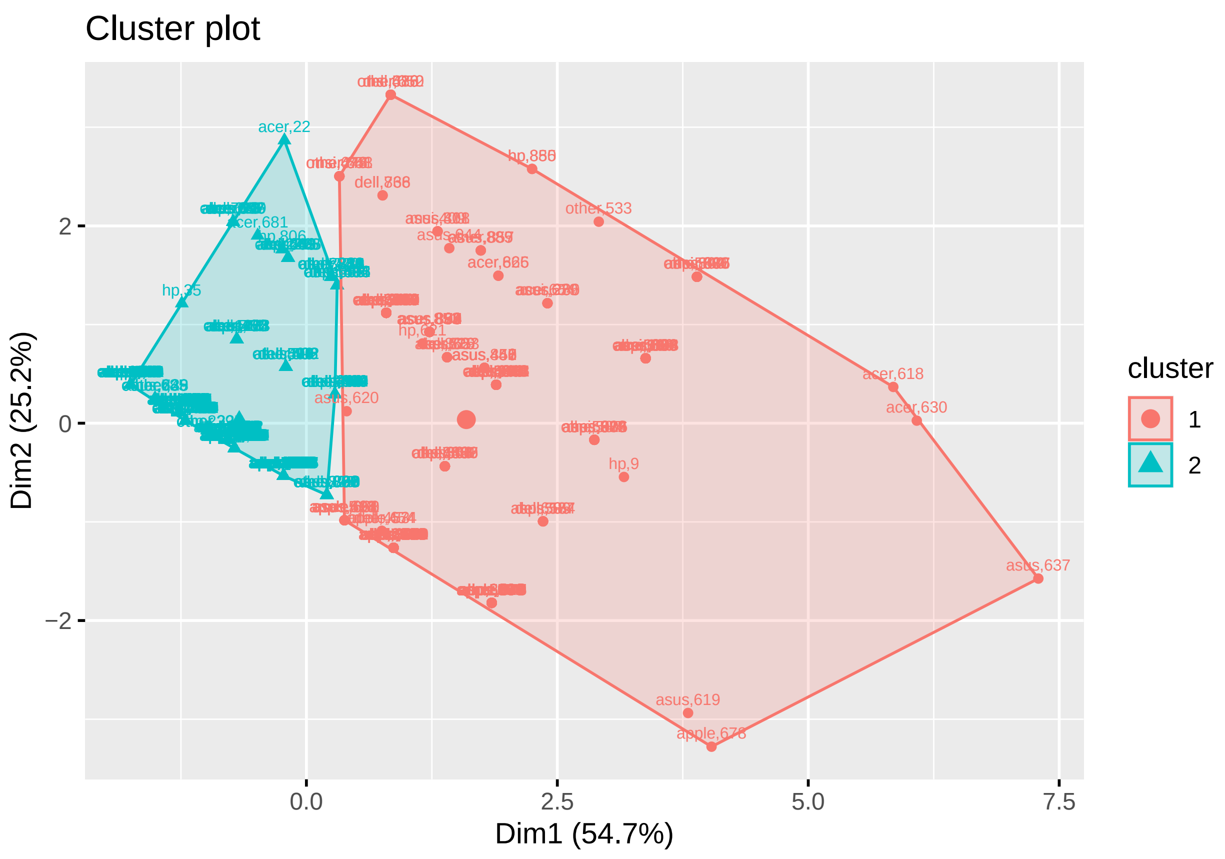
We saw that display sizes had many missing records and we know from our intuition that display sizes impact pricing of a laptop (i.e., Macbook 17inch more expensive than Macbook 13inch). So, we created a new column in the dataframe called "Display Size Given?" and populated it with "Yes" or "No" values. We then replaced any missing display size values with the average of display size from all available records. In addition to a Y/N column, we wanted to leverage the available records in the display size column, so we replaced the missing records with the column average. We also grouped a few levels in some categorical variables (such as processor name, brand, etc.) to make the levels more dense and make them easier to model.

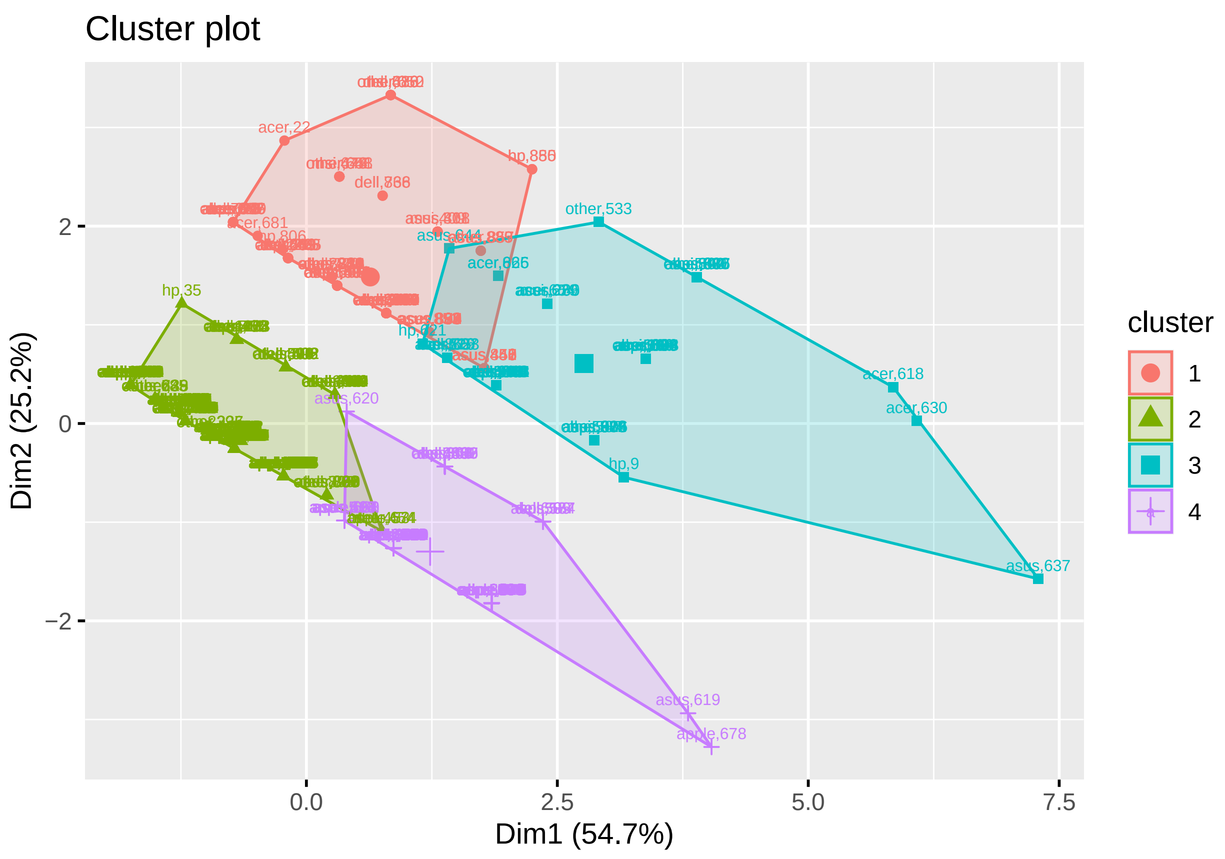
We chose to drop some columns for which there were many NAs (i.e., missing) or variables that did not seem to be useful for our use case: (1) model, (2) processor_gnrtn, (3) discount, and (4) old_price. We felt comfortable removing the first 2 columns from the list above because a laptop's model is dependent on the laptop brand which we were already considering, while processor's generation is reflected in the processor name (3 in i3 represents generation). We did not want to use other pricing related variables in our predictive models and so we removed discount and old_price variables. We wanted to make sure that we can predict the price of a laptop given it's attributes and not be dependent on discount or old listed price from a single e-commerce platform.

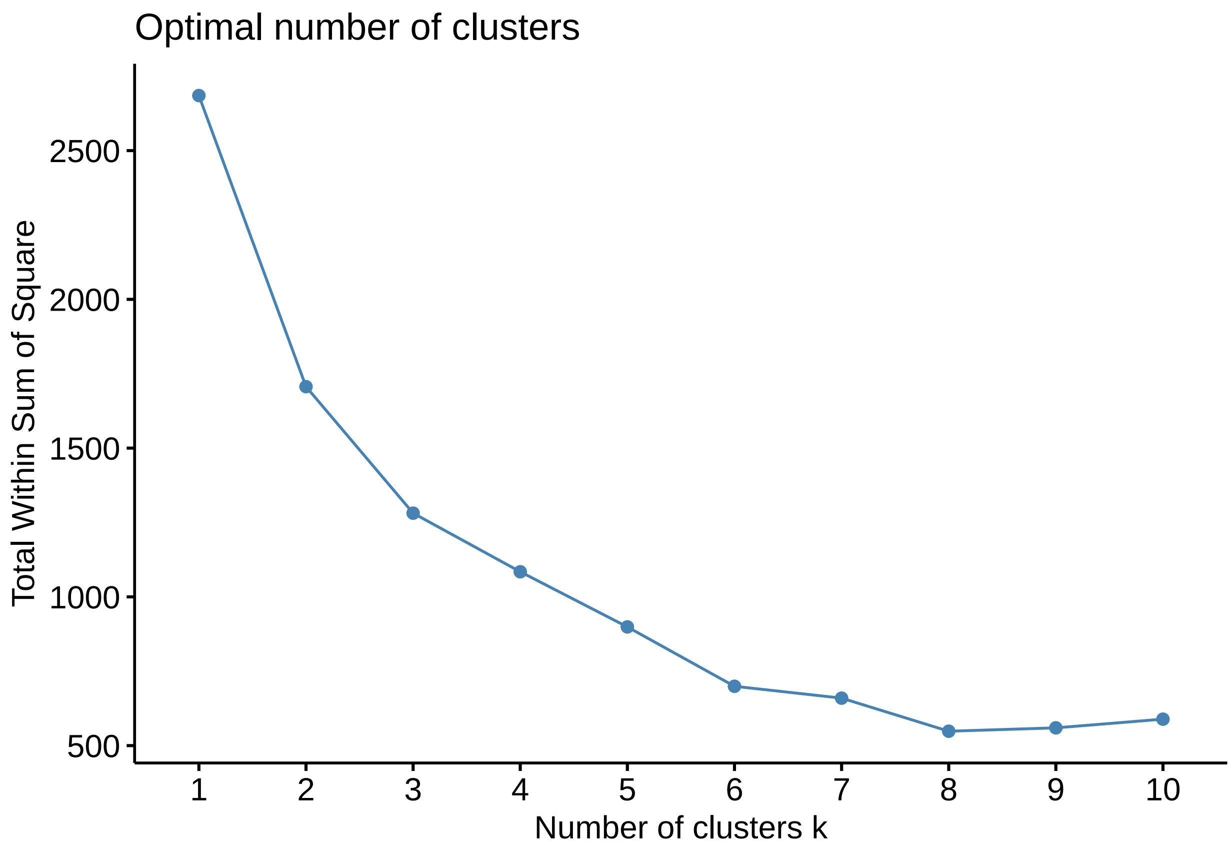
After applying all cleanup and variable selection, we created a new dataframe **laptop.data.clean** for the rest of this analysis.

Clustering the Laptops Based on Performance Attributes

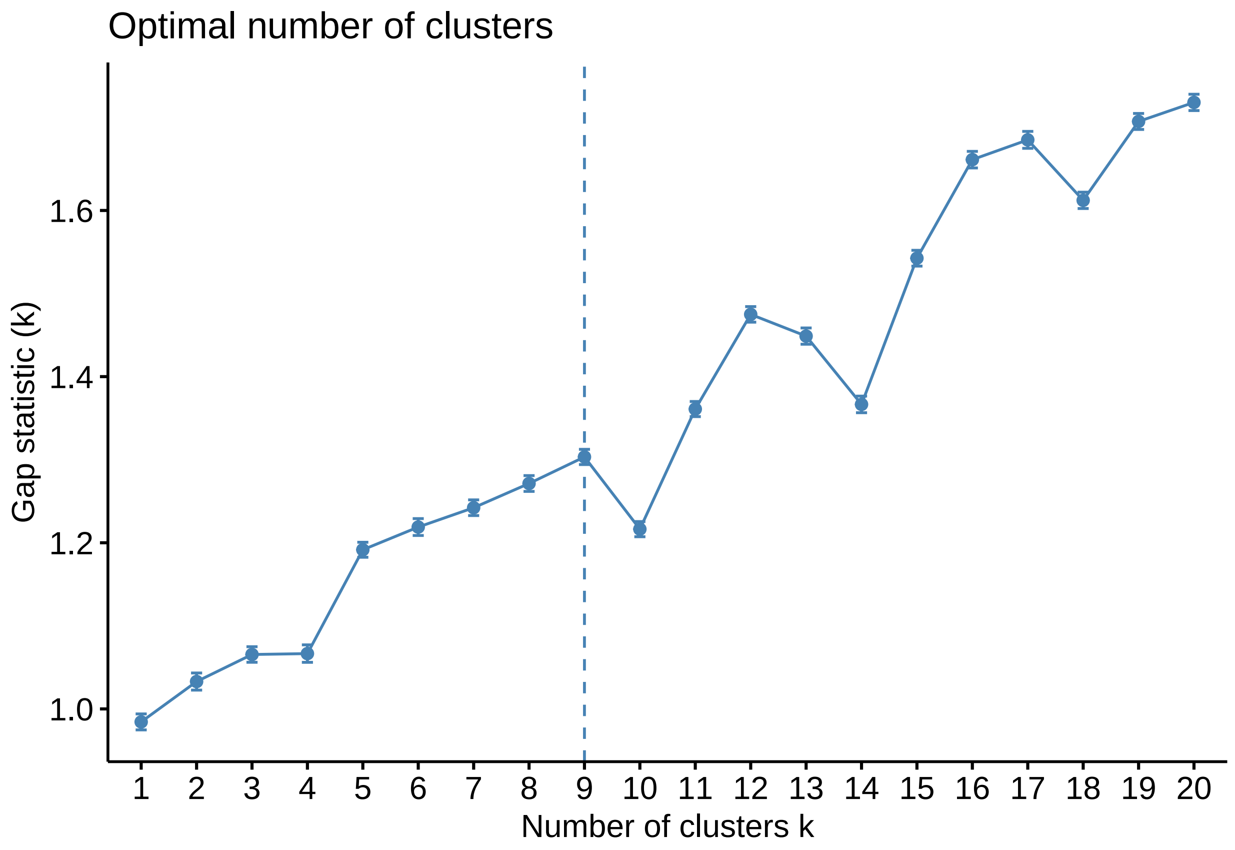
We believed it would be beneficial to perform cluster analysis on our data set. We know that RAM, ssd, and Graphic Card memory are common performance attributes that buyers consider when determining if a laptop would be suitable for their use case. We extracted these three columns to conduct cluster analysis as shown in the plots below.



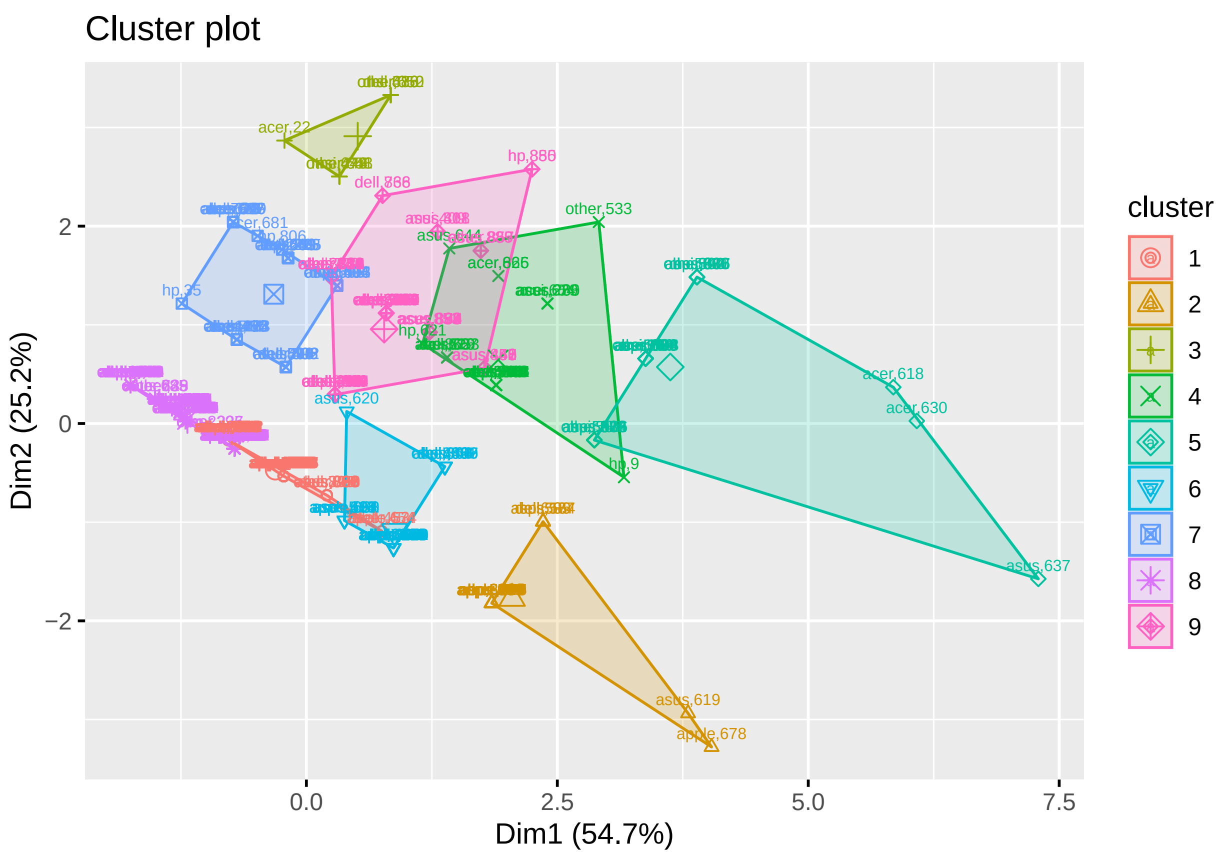




```
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```



We can see from above plots that the chosen attributes are able to cluster our laptops into cleanly separable clusters. We started by building clusters with 2 and 4 centers and then applied the elbow method and gapstatistics to learn that the optimal number of clusters possible from our attributes would be 9. Therefore, we finally created clusters with 9 centers using the K-means method. Below is a visual:



Upon visually inspecting the clusters, we realized that these groupings across RAM, ssd, and graphics card is possibly an indication of the use cases (such as gaming vs. office use vs. personal use) of the laptops in each groups. We expect a natural shakeout of high-end, high-price laptops all the way down to low-end, low-price laptops (along with multiple other “buckets” in between).

Building a Feature Based on the Clusters Created

We had previously converted ram, ssd and graphic card into denser multilevel categorical variables. We felt that the the clusters created based on their raw values would provide additional predictive power over the previously created variables. Hence, we turned these clusters into a feature with 9 levels for our predictive models. See below the number of records in each of the nine levels:

```
##
##   1   2   3   4   5   6   7   8   9
## 256  33  11  37  37  76  51 283 112
```

Predictive Analysis

Overview of the Approaches

To predict the price of a laptop, we explored the following techniques:

1. Stepwise AIC/Stepwise BIC
2. Regularized Regression
3. Decision Trees
4. Ensemble : Random Forest
5. Ensemble : Tuned XGBoost

Splitting Data Into Train and Validation We split our clean dataframe (**laptop.data.clean**) into training and validation (holdout) subsets. We used 2/3rd of the records for training and 1/3rd for validation. The same validation dat aset was used for all models for calculating R^2 and RMSE's.

Stepwise Regression

Comparing AIC and BIC

From our validation metrics, we can see that StepAIC is a slightly better stepwise approach if we were to use a linear model to predict laptop prices. Below is a brief summary of the R^2 and RMSE. We wanted to look at the validation metrics at both log scale and original scale. Although our underlying model is optimizing at the log scale, we would care more about about actual prices and how far we would be from them with our predictions. Our R^2 for both stepAIC and stepBIC are fairly high (log scale) indicating that our predictions and actuals are somewhat correlated. Additionally our RMSE in the log scale is fairly small indicating that we have a fairly good model. The larger discrepancy in RMSE in regular scale is possibly due to outliers (i.e., very highly priced laptops).

StepAIC Model Fit Summary:

[1] "The R^2 value of stepAIC is 0.804889136995938" [1] "The RMSE value of stepAIC is 253.418380673218" [1] "The R^2 value of stepAIC in log scale is 0.857404190399198" [1] "The RMSE value of stepAIC in log scale is 0.190957188483678"

StepBIC Model Fit Summary:

[1] "The R^2 value of stepBIC is 0.799925677459252" [1] "The RMSE value of stepBIC is 256.653973304625" [1] "The R^2 value of stepBIC in log scale is 0.85636750739433" [1] "The RMSE value of stepBIC in log scale is 0.191323071789122"

Coefficients and their interpretations:

The annova analysis we conducted (shown below) identifies the list of significant variables and the coefficients show that certain levels in categorical variables are significant whereas others are not. It is good to see that the presence of clusters is significant for the model. This is a further validation that our feature engineering was an useful trick. Also, both display size given (Y/N) and display size turned out be significant. StepwiseBIC had 29 significant variables (counting only stat sig ones), whereas stepwiseAIC picked 30 which is expected as BIC punishes for complexity more than AIC. (Note: Multi-levels are counted as separate variables)

```
## Analysis of Variance Table
##
## Response: log_latest_price
##                      Df Sum Sq Mean Sq  F value    Pr(>F)
## processor_name       10 90.974  9.0974 218.8416 < 2.2e-16 ***
## cluster               8 24.850  3.1063  74.7220 < 2.2e-16 ***
## processor_brand       2  8.573  4.2867 103.1179 < 2.2e-16 ***
## ram_type              5  3.630  0.7260  17.4631 4.364e-16 ***
## Touchscreen           1  2.379  2.3792  57.2318 1.599e-13 ***
## os                    2  2.004  1.0021  24.1067 9.057e-11 ***
## star_rating           1  1.843  1.8425  44.3229 6.653e-11 ***
## display_size          1  1.035  1.0353  24.9039 8.056e-07 ***
## graphic_card_gb       1  0.942  0.9421  22.6635 2.460e-06 ***
## ssd_cat               1  0.512  0.5121  12.3184 0.0004846 ***
## display_size_given    1  0.415  0.4146   9.9723 0.0016747 **
## weight                2  0.332  0.1658   3.9877 0.0190719 *
## ratings               1  0.159  0.1589   3.8236 0.0510331 .
## Residuals           560 23.280  0.0416
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "The R^2 value of stepAIC is 0.804889136995938"
```

```
## [1] "The RMSE value of stepAIC is 253.418380673218"
```

```
## [1] "The R^2 value of stepAIC in log scale is 0.857404190399198"
```

```
## [1] "The RMSE value of stepAIC in log scale is 0.190957188483678"
```

```
##
## Call:
## lm(formula = log_latest_price ~ processor_name + cluster + processor_brand +
##     Touchscreen + ram_type + os + star_rating + display_size +
##     graphic_card_gb + ssd_cat + display_size_given, data = pred.laptop.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90398 -0.11945 -0.00547  0.09710  1.10748
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                4.462205   0.254550  17.530  < 2e-16 ***
## processor_namecore i3      0.517700   0.062088   8.338 5.82e-16 ***
## processor_namecore i5      0.869765   0.063667  13.661  < 2e-16 ***
## processor_namecore i7      1.004909   0.072684  13.826  < 2e-16 ***
## processor_namem1           2.008513   0.197600  10.165  < 2e-16 ***
## processor_nameother        1.037052   0.087998  11.785  < 2e-16 ***
## processor_namepentium quad 0.147120   0.101696   1.447 0.148548
## processor_nameryzen 3      1.385709   0.112377  12.331  < 2e-16 ***
## processor_nameryzen 5      1.694818   0.108917  15.561  < 2e-16 ***
## processor_nameryzen 7      1.782434   0.112653  15.822  < 2e-16 ***
## processor_nameryzen 9      2.070597   0.133385  15.523  < 2e-16 ***
## cluster2                   0.152507   0.073593   2.072 0.038692 *
## cluster3                   0.473053   0.134193   3.525 0.000458 ***
## cluster4                   0.029352   0.076765   0.382 0.702341
## cluster5                   0.149718   0.105117   1.424 0.154914
## cluster6                   0.169686   0.037700   4.501 8.23e-06 ***
## cluster7                  -0.136276   0.061123  -2.230 0.026173 *
## cluster8                  -0.088613   0.025765  -3.439 0.000626 ***
## cluster9                  -0.110375   0.058277  -1.894 0.058740 .
## processor_brandintel       0.933830   0.081254  11.493  < 2e-16 ***
## processor_brandother      -0.288711   0.130632  -2.210 0.027499 *
## TouchscreenYes             0.268039   0.030902   8.674  < 2e-16 ***
## ram_typeDDR4               0.108920   0.075165   1.449 0.147875
## ram_typeDDR5               0.483772   0.132030   3.664 0.000272 ***
## ram_typeLPDDR3             0.537994   0.099006   5.434 8.22e-08 ***
## ram_typeLPDDR4             0.068075   0.102540   0.664 0.507038
## ram_typeLPDDR4X            0.170916   0.080627   2.120 0.034457 *
## osMac                      0.703859   0.148427   4.742 2.68e-06 ***
## osWindows                 -0.222420   0.070968  -3.134 0.001814 **
## star_rating               -0.030141   0.004632  -6.507 1.69e-10 ***
## display_size               0.058268   0.012344   4.720 2.97e-06 ***
## graphic_card_gb            0.058192   0.013598   4.280 2.20e-05 ***
## ssd_catless_than_1gb      -0.190275   0.055027  -3.458 0.000586 ***
## display_size_givenYes     -0.059956   0.019134  -3.134 0.001817 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2055 on 563 degrees of freedom
## Multiple R-squared:  0.8523, Adjusted R-squared:  0.8436
## F-statistic: 98.44 on 33 and 563 DF,  p-value: < 2.2e-16
```

```
## Analysis of Variance Table
##
## Response: log_latest_price
##                     Df Sum Sq Mean Sq  F value    Pr(>F)
## processor_name      10 90.974  9.0974 215.4741 < 2.2e-16 ***
## cluster              8 24.850  3.1063  73.5722 < 2.2e-16 ***
## processor_brand      2  8.573  4.2867 101.5311 < 2.2e-16 ***
## Touchscreen          1  2.713  2.7129  64.2555 6.343e-15 ***
## ram_type             5  3.296  0.6592  15.6135 2.104e-14 ***
## os                   2  2.004  1.0021  23.7357 1.269e-10 ***
## star_rating          1  1.843  1.8425  43.6409 9.151e-11 ***
## display_size         1  1.035  1.0353  24.5207 9.732e-07 ***
## graphic_card_gb      1  0.942  0.9421  22.3147 2.926e-06 ***
## ssd_cat              1  0.512  0.5121  12.1289  0.000535 ***
## display_size_given   1  0.415  0.4146   9.8188  0.001817 **
## Residuals          563 23.770  0.0422
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## [1] "The R^2 value of stepBIC is 0.799925677459252"
```

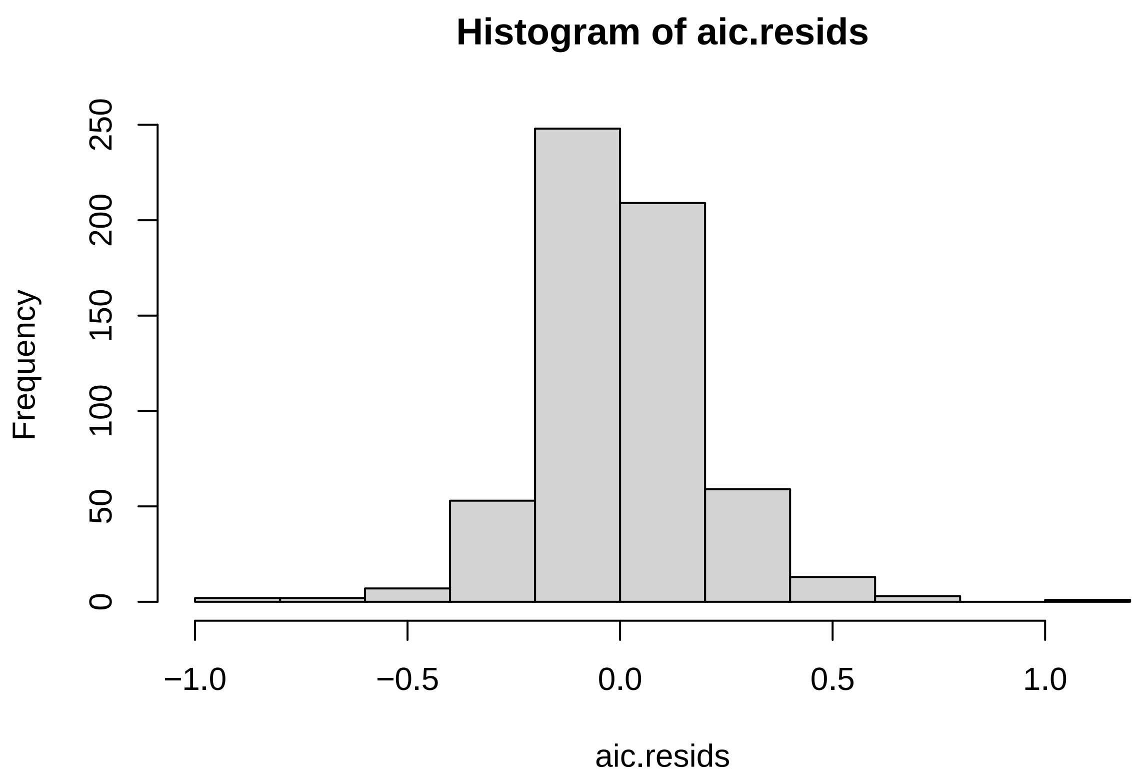
```
## [1] "The RMSE value of stepBIC is 256.653973304625"
```

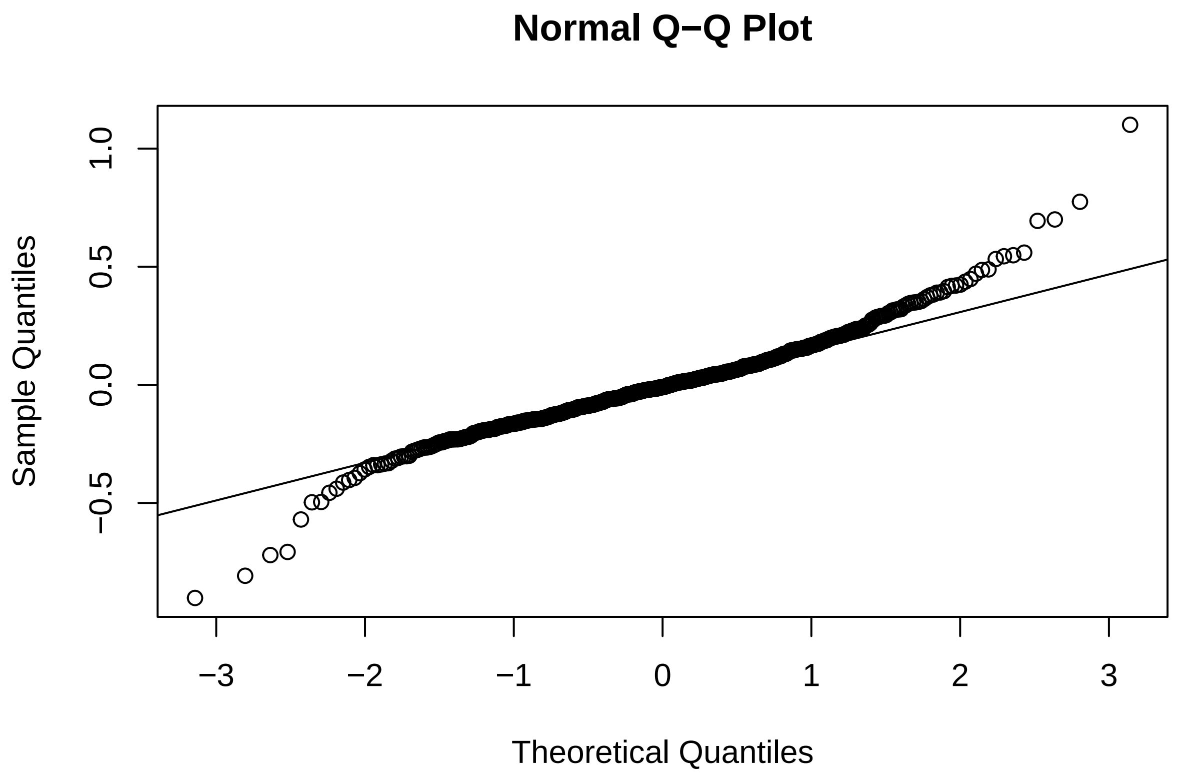
```
## [1] "The R^2 value of stepBIC in log scale is 0.85636750739433"
```

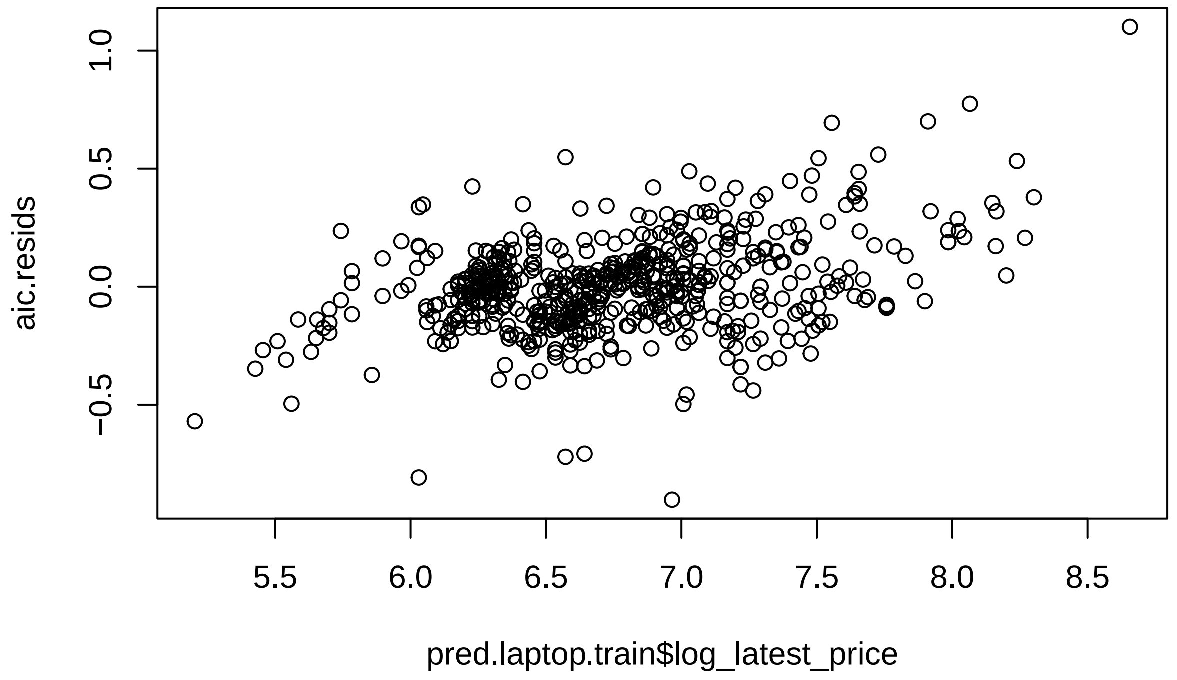
```
## [1] "The RMSE value of stepBIC in log scale is 0.191323071789122"
```

Residuals

The residuals seems to be aligned with assumptions of linear regression (normally distributed, centered around 0, and equal variance). The qq plot showcases presence of heavy tails but it does not seem to be a big concern from the plot.


Histogram of aic.resids
Frequency
0
50
100
150
200
250
−1.0
−0.5
0.0
0.5
1.0
aic.resids


Normal Q−Q Plot
Sample Quantiles
−0.5
0.0
0.5
1.0
−3
−2
−1
0
1
2
3
Theoretical Quantiles



Regularized Regression

Analysis - Validation Metrics

We applied a lasso regression to see if we could build a model with fewer features. We chose the penalty multiple (lambda) and number of features based on 1 standard error criteria. We found that while lasso could provide a fairly good model (high r square and low RMSE), it wasn't better than our Stepwise regression approach.

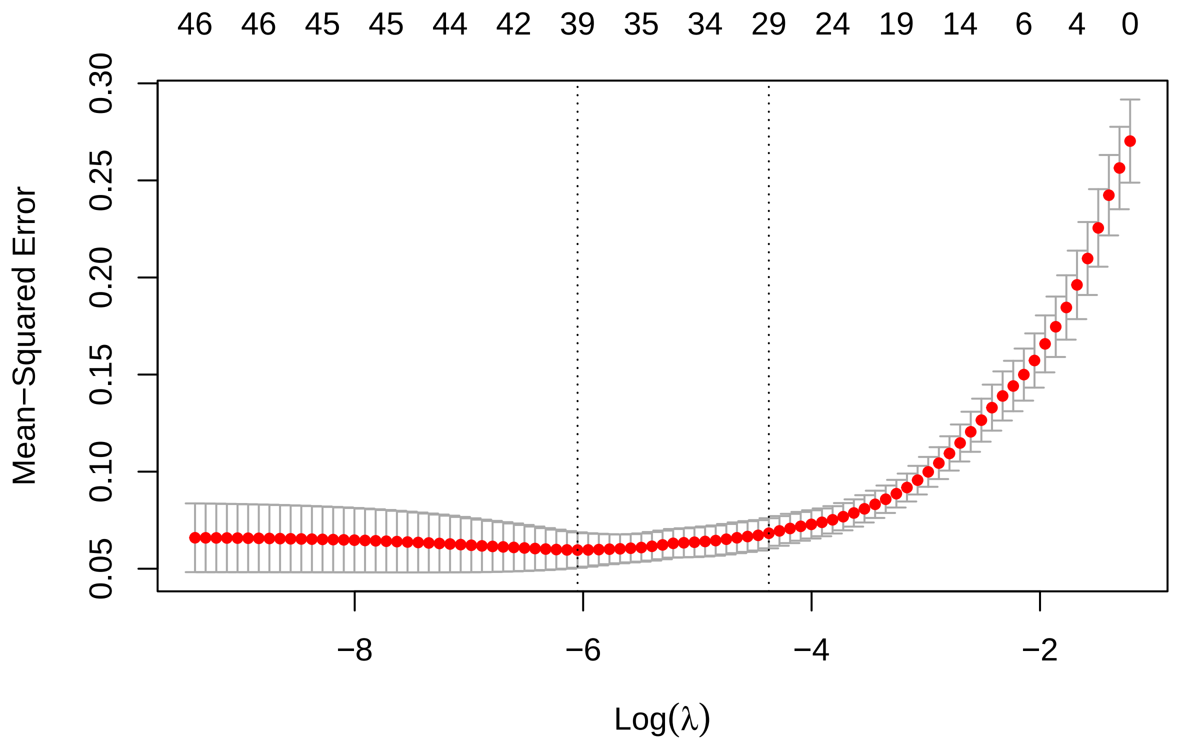
Regularized Regression Model Fit Summary:

[1] "The R^2 value of regularized regression is 0.739109086771972" [1] "The RMSE value of regularized regression is 296.652788037578" [1] "The R^2 value of regularized regression in log scale is 0.787245668133689" [1] "The RMSE value of regularized regression in log scale is 0.232685415075645"

Coefficients and Their Interpretations:

It is interesting to observe that lasso gave us 29 significant variables which is similar to our best stepwise regression (stepAIC). Cluster seems to be as important for the regularized regression as it was for stepwise regression. However, regularized regressions shrunk display size given (Y/N) unlike stepwise regression.

```
## [1] 597  49
```



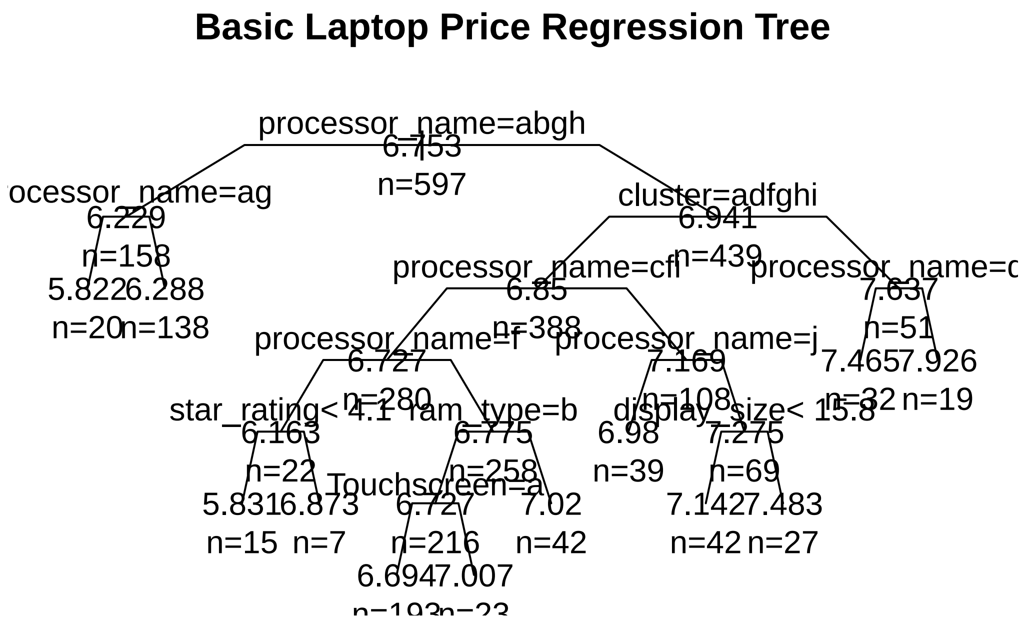
```
## [1] 50  1

## [1] "The lasso regression chose 29 variables and 1 intercept"

## 50 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                 6.7416819323
## (Intercept)                 .
## brandapple                  0.5558859411
## brandasus                  -0.0020370338
## branddell                   0.0048670322
## brandhp                     .
## brandmsi                    .
## brandother                  .
## processor_brandintel        .
## processor_brandother        .
## processor_namecore i3      -0.0097100993
## processor_namecore i5       0.1985940376
## processor_namecore i7       0.2869211681
## processor_namem1            0.1365601575
## processor_nameother         .
## processor_namepentium quad -0.2503282289
## processor_nameryzen 3      -0.0362703686
## processor_nameryzen 5       0.0907451217
## processor_nameryzen 7       0.1362783961
## processor_nameryzen 9       0.3215072423
## ram_typeDDR4                .
## ram_typeDDR5                0.3005891972
## ram_typeLPDDR3             0.3802361275
## ram_typeLPDDR4             .
## ram_typeLPDDR4X            .
## osMac                      0.0012034446
## osWindows                 -0.2671547136
## os_bit64                   .
## graphic_card_gb            0.0448758096
## weightGaming               .
## weightThinNlight           0.0083628155
## display_size               0.0332215507
## warranty                   0.0263151957
## TouchscreenYes             0.2750043592
## msofficeYes                .
## star_rating               -0.0281194638
## ratings                   -0.0000391211
## reviews                    .
## display_size_givenYes      .
## ram_gb_catless_than_8     -0.1557618406
## ssd_catless_than_1gb      -0.2561305868
## hdd_catlow_hdd             .
## cluster2                   .
## cluster3                   0.5303553466
## cluster4                   .
## cluster5                   0.1315500117
## cluster6                   0.0068286880
## cluster7                   .
## cluster8                  -0.2059538924
## cluster9                   .

## [1] "The R^2 value of regularized regression is 0.739109086771972"

## [1] "The RMSE value of regularized regression is 296.652788037578"

## [1] "The R^2 value of regularized regression in log scale is 0.787245668133689"

## [1] "The RMSE value of regularized regression in log scale is 0.232685415075645"
```

Decision Trees

Analysis - Validation Metrics Across Simple and Pruned Trees

We started by building a simple tree with default parameters and then pruned it based on complexity parameter (cp). Pruning did not change anything as we got the exact same tree after pruning. Additionally the R^2 and RMSE for a single tree seemed to be worse compared to linear models.

```
## Call:
## rpart(formula = log_latest_price ~ ., data = pred.laptop.train,
##     parms = list(split = "information"), control = rpart.control(minsplit = 20))
##   n= 597
##
##            CP nsplit rel error    xerror       xstd
## 1  0.36654808      0 1.0000000 1.0060243 0.06412652
## 2  0.17344585      1 0.6334519 0.7029555 0.06058484
## 3  0.09453099      2 0.4600061 0.5687281 0.05477197
## 4  0.04720661      3 0.3654751 0.4107763 0.03427384
## 5  0.03220955      4 0.3182685 0.3551096 0.03287640
## 6  0.02356044      5 0.2860589 0.3482968 0.03289189
## 7  0.01869464      6 0.2624985 0.3234842 0.03071704
## 8  0.01578159      7 0.2438038 0.3211953 0.03089951
## 9  0.01353413      8 0.2280223 0.3109761 0.03093254
## 10 0.01249174      9 0.2144881 0.2934783 0.03088260
## 11 0.01186462     10 0.2019964 0.2877812 0.03056464
## 12 0.01000000     11 0.1901318 0.2773660 0.03007320
##
## Variable importance
##  processor_name         cluster         ssd_cat graphic_card_gb    display_size
##              43              26               5               5               3
## processor_brand           brand     star_rating        ram_type      ram_gb_cat
##               3               3               2               2               2
##              os         reviews         ratings     Touchscreen
##               2               2               1               1
##
## Node number 1: 597 observations,    complexity param=0.3665481
##   mean=6.752586, MSE=0.2695605
##   left son=2 (158 obs) right son=3 (439 obs)
##   Primary splits:
##       processor_name splits as  LLRRRRLLRRR, improve=0.3665481, (0 missing)
##       cluster        splits as  LRRRRRLLR,   improve=0.3478018, (0 missing)
##       ram_gb_cat     splits as  RL,          improve=0.2637135, (0 missing)
##       ssd_cat        splits as  RL,          improve=0.2517967, (0 missing)
##       display_size   < 15.8   to the left,   improve=0.1923546, (0 missing)
##   Surrogate splits:
##       cluster splits as  RRRRRRRLR,  agree=0.821, adj=0.323, (0 split)
##       reviews < 794    to the right, agree=0.742, adj=0.025, (0 split)
##       ratings < 4914   to the right, agree=0.740, adj=0.019, (0 split)
##
## Node number 2: 158 observations,    complexity param=0.02356044
##   mean=6.228627, MSE=0.04839155
##   left son=4 (20 obs) right son=5 (138 obs)
##   Primary splits:
##       processor_name splits as  LR----LR---, improve=0.49589220, (0 missing)
##       ram_type       splits as  LR--LL,      improve=0.32979860, (0 missing)
##       brand          splits as  L-LRRRL,     improve=0.12698630, (0 missing)
##       warranty       < 0.5    to the left,   improve=0.09250302, (0 missing)
##       cluster        splits as  R------L-,   improve=0.08363798, (0 missing)
##   Surrogate splits:
##       ram_type splits as  LR--LR, agree=0.911, adj=0.3, (0 split)
##
## Node number 3: 439 observations,    complexity param=0.1734458
##   mean=6.941164, MSE=0.2147928
##   left son=6 (388 obs) right son=7 (51 obs)
##   Primary splits:
##       cluster         splits as  LRRLRLLLL,   improve=0.2960126, (0 missing)
##       processor_name  splits as  --LRRL--LLR, improve=0.2477640, (0 missing)
##       ssd_cat         splits as  RL,          improve=0.2446011, (0 missing)
##       ram_gb_cat      splits as  RL,          improve=0.2060381, (0 missing)
##       graphic_card_gb < 5      to the left,   improve=0.1837011, (0 missing)
##   Surrogate splits:
##       ssd_cat         splits as  RL,         agree=0.925, adj=0.353, (0 split)
##       graphic_card_gb < 5      to the left,  agree=0.913, adj=0.255, (0 split)
##       display_size    < 13.15  to the right, agree=0.886, adj=0.020, (0 split)
##
## Node number 4: 20 observations
##   mean=5.821713, MSE=0.0929088
##
## Node number 5: 138 observations
##   mean=6.2876, MSE=0.01446495
##
## Node number 6: 388 observations,    complexity param=0.09453099
##   mean=6.849745, MSE=0.1527882
##   left son=12 (280 obs) right son=13 (108 obs)
##   Primary splits:
##       processor_name splits as  --LRRL--LRR, improve=0.2566158, (0 missing)
##       display_size   < 15.8   to the left,   improve=0.1844874, (0 missing)
##       cluster        splits as  R--R-RRLR,   improve=0.1435471, (0 missing)
##       ratings        < 4.5    to the right,  improve=0.1039789, (0 missing)
##       reviews        < 95.5   to the right,  improve=0.1030018, (0 missing)
##   Surrogate splits:
##       ram_gb_cat   splits as  RL,         agree=0.784, adj=0.222, (0 split)
##       cluster      splits as  L--R-RLLL,  agree=0.784, adj=0.222, (0 split)
##       display_size < 15.8   to the left,  agree=0.763, adj=0.148, (0 split)
##       brand        splits as  LRLLLLL,    agree=0.755, adj=0.120, (0 split)
##       os           splits as  LRL,        agree=0.755, adj=0.120, (0 split)
##
## Node number 7: 51 observations,    complexity param=0.01578159
##   mean=7.636661, MSE=0.1392158
##   left son=14 (32 obs) right son=15 (19 obs)
##   Primary splits:
##       processor_name     splits as  --RLRR---LL, improve=0.3577031, (0 missing)
##       display_size_given splits as  RL,          improve=0.2341135, (0 missing)
##       msoffice           splits as  RL,          improve=0.2025396, (0 missing)
##       star_rating        < 4.75   to the left,   improve=0.1384419, (0 missing)
##       brand              splits as  LRRLLRR,     improve=0.1362138, (0 missing)
##   Surrogate splits:
##       os         splits as  RRL,     agree=0.863, adj=0.632, (0 split)
##       brand      splits as  LRLLLRL, agree=0.765, adj=0.368, (0 split)
##       ram_type   splits as  -LLLRL,  agree=0.765, adj=0.368, (0 split)
##       ram_gb_cat splits as  LR,      agree=0.765, adj=0.368, (0 split)
##       ssd_cat    splits as  LR,      agree=0.765, adj=0.368, (0 split)
##
## Node number 12: 280 observations,    complexity param=0.04720661
##   mean=6.726769, MSE=0.1167087
##   left son=24 (22 obs) right son=25 (258 obs)
##   Primary splits:
##       processor_name  splits as  --R--L--R--, improve=0.2324729, (0 missing)
##       processor_brand splits as  LRL,         improve=0.1943622, (0 missing)
##       ratings         < 218.5  to the right,  improve=0.1519869, (0 missing)
##       cluster         splits as  R--R-RRLR,   improve=0.1429508, (0 missing)
##       reviews         < 102    to the right,  improve=0.1403495, (0 missing)
##   Surrogate splits:
##       reviews         < 249    to the right, agree=0.939, adj=0.227, (0 split)
##       processor_brand splits as  RRL,        agree=0.936, adj=0.182, (0 split)
##       ratings         < 1887.5 to the right, agree=0.936, adj=0.182, (0 split)
##
## Node number 13: 108 observations,    complexity param=0.01353413
##   mean=7.168571, MSE=0.1054698
##   left son=26 (39 obs) right son=27 (69 obs)
##   Primary splits:
##       processor_name splits as  ---RR----LR, improve=0.19120930, (0 missing)
##       display_size   < 15.8   to the left,   improve=0.15777240, (0 missing)
##       cluster        splits as  L--L-RLLL,   improve=0.12301120, (0 missing)
##       brand          splits as  LRLLRLL,     improve=0.09585890, (0 missing)
##       os             splits as  RRL,         improve=0.08480929, (0 missing)
##   Surrogate splits:
##       processor_brand splits as  LRR,       agree=0.898, adj=0.718, (0 split)
##       cluster         splits as  L--R-RRRL, agree=0.704, adj=0.179, (0 split)
##       brand           splits as  LRLRRRR,   agree=0.657, adj=0.051, (0 split)
##       ram_type        splits as  -RLR-R,    agree=0.657, adj=0.051, (0 split)
##       os_bit          splits as  LR,        agree=0.657, adj=0.051, (0 split)
##
## Node number 14: 32 observations
##   mean=7.464709, MSE=0.07727019
##
## Node number 15: 19 observations
##   mean=7.926265, MSE=0.1098771
##
## Node number 24: 22 observations,    complexity param=0.03220955
##   mean=6.162695, MSE=0.3764164
##   left son=48 (15 obs) right son=49 (7 obs)
##   Primary splits:
##       star_rating     < 4.1    to the left,  improve=0.6259276, (0 missing)
##       processor_brand splits as  LRL,        improve=0.5654994, (0 missing)
##       brand           splits as  R-RRLRL,    improve=0.2535976, (0 missing)
##       ratings         < 52     to the right, improve=0.1595410, (0 missing)
##       reviews         < 9.5    to the right, improve=0.1595410, (0 missing)
##   Surrogate splits:
##       cluster         splits as  ---L--RL-,  agree=0.909, adj=0.714, (0 split)
##       graphic_card_gb < 2      to the left,  agree=0.864, adj=0.571, (0 split)
##       brand           splits as  R-LRLRL,    agree=0.818, adj=0.429, (0 split)
##       processor_brand splits as  LRL,        agree=0.818, adj=0.429, (0 split)
##       display_size    < 15.36  to the left,  agree=0.773, adj=0.286, (0 split)
##
## Node number 25: 258 observations,    complexity param=0.01869464
##   mean=6.774869, MSE=0.0651179
##   left son=50 (216 obs) right son=51 (42 obs)
##   Primary splits:
##       ram_type     splits as  RL-RRR,     improve=0.17907190, (0 missing)
##       Touchscreen  splits as  LR,         improve=0.15686400, (0 missing)
##       ratings      < 2.5    to the right, improve=0.10946810, (0 missing)
##       star_rating  < 0.8    to the right, improve=0.09652365, (0 missing)
##       display_size < 15.8   to the left,  improve=0.09615770, (0 missing)
##   Surrogate splits:
##       os          splits as  RRL,        agree=0.872, adj=0.214, (0 split)
##       brand       splits as  LRLLLLL,    agree=0.841, adj=0.024, (0 split)
##       star_rating < 4.85   to the left,  agree=0.841, adj=0.024, (0 split)
##
## Node number 26: 39 observations
##   mean=6.97968, MSE=0.04893071
##
## Node number 27: 69 observations,    complexity param=0.01186462
##   mean=7.275335, MSE=0.1058613
##   left son=54 (42 obs) right son=55 (27 obs)
##   Primary splits:
##       display_size    < 15.8   to the left,   improve=0.2613956, (0 missing)
##       cluster         splits as  L--L-RLLR,   improve=0.1052999, (0 missing)
##       processor_brand splits as  RLR,         improve=0.1037542, (0 missing)
##       processor_name  splits as  ---LR-----R, improve=0.1037542, (0 missing)
##       ratings         < 3.5    to the right,  improve=0.0797980, (0 missing)
##   Surrogate splits:
##       processor_brand splits as  RLL,         agree=0.725, adj=0.296, (0 split)
##       processor_name  splits as  ---LL-----R, agree=0.725, adj=0.296, (0 split)
##       ssd_cat         splits as  RL,          agree=0.710, adj=0.259, (0 split)
##       brand           splits as  LLRLRLL,     agree=0.696, adj=0.222, (0 split)
##       Touchscreen     splits as  LR,          agree=0.696, adj=0.222, (0 split)
##
## Node number 48: 15 observations
##   mean=5.831106, MSE=0.05287271
##
## Node number 49: 7 observations
##   mean=6.873242, MSE=0.3292376
##
## Node number 50: 216 observations,    complexity param=0.01249174
##   mean=6.727252, MSE=0.04898566
##   left son=100 (193 obs) right son=101 (23 obs)
##   Primary splits:
##       Touchscreen     splits as  LR,         improve=0.18999010, (0 missing)
##       cluster         splits as  L--R-RLLR,  improve=0.11033980, (0 missing)
##       ratings         < 25     to the right, improve=0.10179550, (0 missing)
##       display_size    < 15.8   to the left,  improve=0.09614241, (0 missing)
##       graphic_card_gb < 1      to the left,  improve=0.09077926, (0 missing)
##   Surrogate splits:
##       warranty < 2.5    to the left,  agree=0.903, adj=0.087, (0 split)
##
## Node number 51: 42 observations
##   mean=7.019756, MSE=0.07645314
##
## Node number 54: 42 observations
##   mean=7.14196, MSE=0.07555488
##
## Node number 55: 27 observations
##   mean=7.482808, MSE=0.08228808
##
## Node number 100: 193 observations
##   mean=6.693949, MSE=0.03826758
##
## Node number 101: 23 observations
##   mean=7.006708, MSE=0.05152147
```



Basic Tree and Pruned Tree Model Fit Summary (values identical):

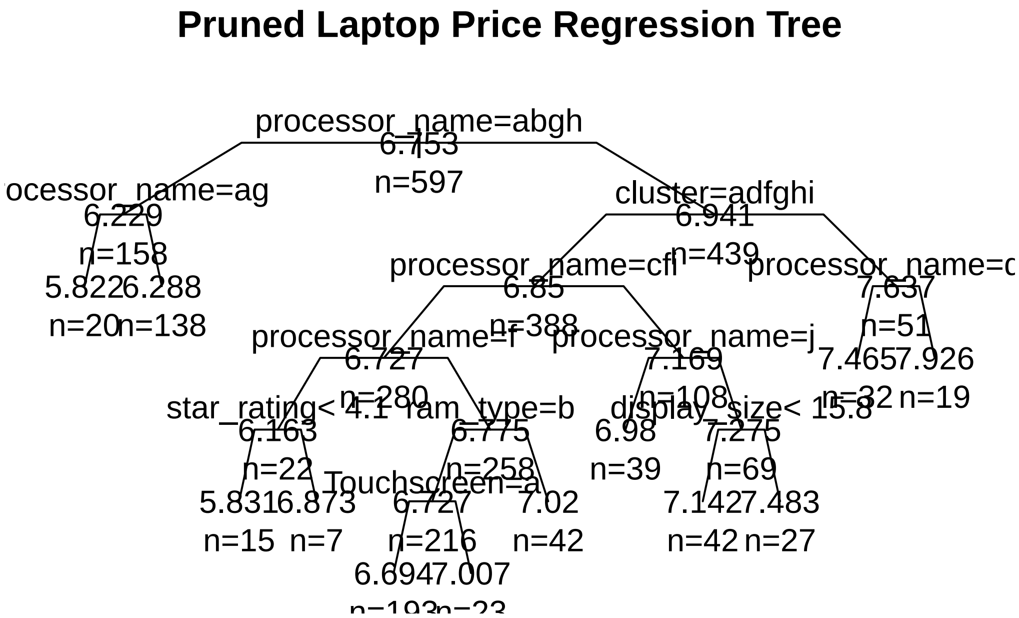
[1] "The R^2 value of pruned tree is 0.648006191517393" [1] "The RMSE value of pruned tree is 342.789650527593" [1] "The R^2 value of pruned tree in log scale is 0.737397757156823" [1] "The RMSE value of pruned tree in log scale is 0.261736301177269"

```
## [1] "Basic Decision Tree Model Summary:"

## [1] "The R^2 value of basic tree is 0.648006191517393"

## [1] "The RMSE value of basic tree is 342.789650527593"

## [1] "The R^2 value of basic tree in log scale is 0.737397757156823"

## [1] "The RMSE value of basic tree in log scale is 0.261736301177269"

##
## Regression tree:
## rpart(formula = log_latest_price ~ ., data = pred.laptop.train,
##     parms = list(split = "information"), control = rpart.control(minsplit = 20))
##
## Variables actually used in tree construction:
## [1] cluster        display_size   processor_name ram_type       star_rating
## [6] Touchscreen
##
## Root node error: 160.93/597 = 0.26956
##
## n= 597
```

```
##
##          CP nsplit rel error  xerror     xstd
## 1  0.366548      0   1.00000 1.00602 0.064127
## 2  0.173446      1   0.63345 0.70296 0.060585
## 3  0.094531      2   0.46001 0.56873 0.054772
## 4  0.047207      3   0.36548 0.41078 0.034274
## 5  0.032210      4   0.31827 0.35511 0.032876
## 6  0.023560      5   0.28606 0.34830 0.032892
## 7  0.018695      6   0.26250 0.32348 0.030717
## 8  0.015782      7   0.24380 0.32120 0.030900
## 9  0.013534      8   0.22802 0.31098 0.030933
## 10 0.012492      9   0.21449 0.29348 0.030883
## 11 0.011865     10   0.20200 0.28778 0.030565
## 12 0.010000     11   0.19013 0.27737 0.030073
```

```
## [1] 0.01
```

```
##  processor_name         cluster         ssd_cat graphic_card_gb    display_size
##      90.8721761      54.4265141      11.2820669      10.0768284       6.1913430
## processor_brand           brand     star_rating        ram_type      ram_gb_cat
##       5.7321403       5.5959114       5.2550370       5.1933107       4.3162652
##              os         reviews         ratings     Touchscreen        warranty
##       4.0798443       3.2199162       2.5012649       2.4345657       0.1748058
##          os_bit
##       0.1116931
```



```
## [1] "The R^2 value of pruned tree is 0.648006191517393"
## [1] "The RMSE value of pruned tree is 342.789650527593"
```

```
## [1] "The R^2 value of pruned tree in log scale is 0.737397757156823"
```

```
## [1] "The RMSE value of pruned tree in log scale is 0.261736301177269"
```

Below is a table of variable importance from the pruned decision tree model:

```
##  processor_name         cluster         ssd_cat graphic_card_gb    display_size
##      90.8721761      54.4265141      11.2820669      10.0768284       6.1913430
## processor_brand           brand     star_rating        ram_type      ram_gb_cat
##       5.7321403       5.5959114       5.2550370       5.1933107       4.3162652
##              os         reviews         ratings     Touchscreen        warranty
##       4.0798443       3.2199162       2.5012649       2.4345657       0.1748058
##          os_bit
##       0.1116931
```

Ensemble Random Forest with Default Parameters

Analysis of Validation Metrics Compared to Linear Models

Our first attempt at an ensemble method gave us better results than a single tree. It was also superior to our initial StepAIC model.

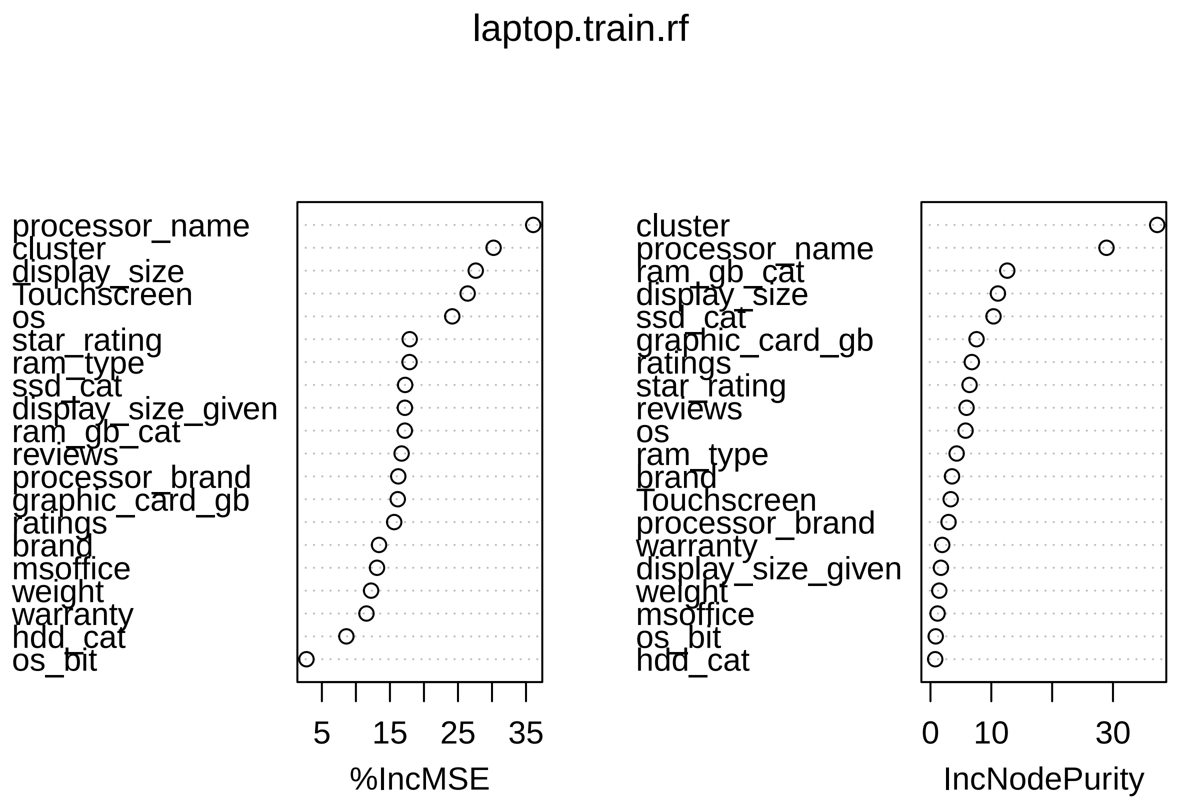
Random Forest Model Fit Summary:

[1] "The R^2 value of Random Forest is 0.783106449815662" [1] "The RMSE value of Random Forest is 275.955222895206" [1] "The R^2 value of Random Forest in log scale is 0.856022265086323" [1] "The RMSE value of Random Forest in log scale is 0.191797362743207"

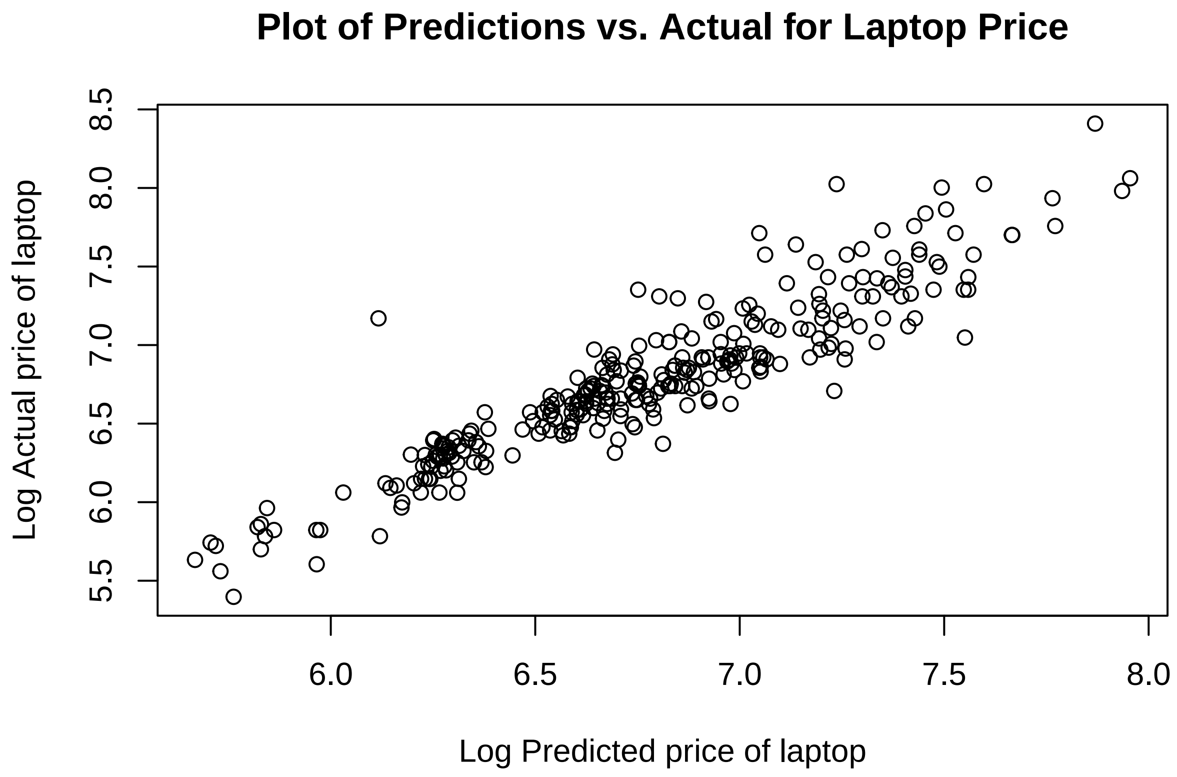
Important variables and their interpretations:

Processor name followed by cluster and ssd_cat had the most predictive power (most impact on %IncMSE). Our feature creation is further validated here.

```
##                         %IncMSE IncNodePurity
## brand              0.0059684506     3.5434391
## processor_brand    0.0153051988     2.9744342
## processor_name     0.0923899754    28.9346092
## ram_type           0.0087423660     4.3039938
## os                 0.0216309357     5.7705393
## os_bit             0.0006633697     0.8664901
## graphic_card_gb    0.0204087236     7.5760371
## weight             0.0042694026     1.4592317
## display_size       0.0275332441    11.0879834
## warranty           0.0036213301     1.9348522
## Touchscreen        0.0095114446     3.3301613
## msoffice           0.0028293931     1.1686366
## star_rating        0.0149630137     6.4254561
## ratings            0.0206146725     6.7990646
## reviews            0.0157708219     5.9301584
## display_size_given 0.0064951843     1.7197173
## ram_gb_cat         0.0225982440    12.6229860
## ssd_cat            0.0214271808    10.3615372
## hdd_cat            0.0017975182     0.7782949
## cluster            0.0872089169    37.2697937
```



```
## [1] "The most predictive variable with regard to price is:"

##
## processor_name
```



```
## [1] "The R^2 value of Random Forest is 0.783106449815662"

## [1] "The RMSE value of Random Forest is 275.955222895206"

## [1] "The R^2 value of Random Forest in log scale is 0.856022265086323"

## [1] "The RMSE value of Random Forest in log scale is 0.191797362743207"
```

Ensemble Method - Use Caret to Set Up a Computer Experiment and Tune XGBoost

Analysis of Validation Metrics and Tuning Strategy Used

Next, we decided to expand our analysis to using XGBoost (known to have exceptional predictive power). We used the caret package to set up a grid and ran an experiment across a few parameters to find one of the best tuned XGBoost. XGBoost ended up delivering on its promise and gave us the best model (lowest RMSE and highest R^2) of all the ones we had tried out:

XGboost Model Fit Summary:

[1] "The R^2 value for XGboost is 0.848757272940201" [1] "The RMSE value of XGboost is 227.363013941365" [1] "The R^2 value for XGboost in log scale is 0.875737860057181" [1] "The RMSE value of XGboost in log scale is 0.177554536560491"

Below are the tuning steps we followed:

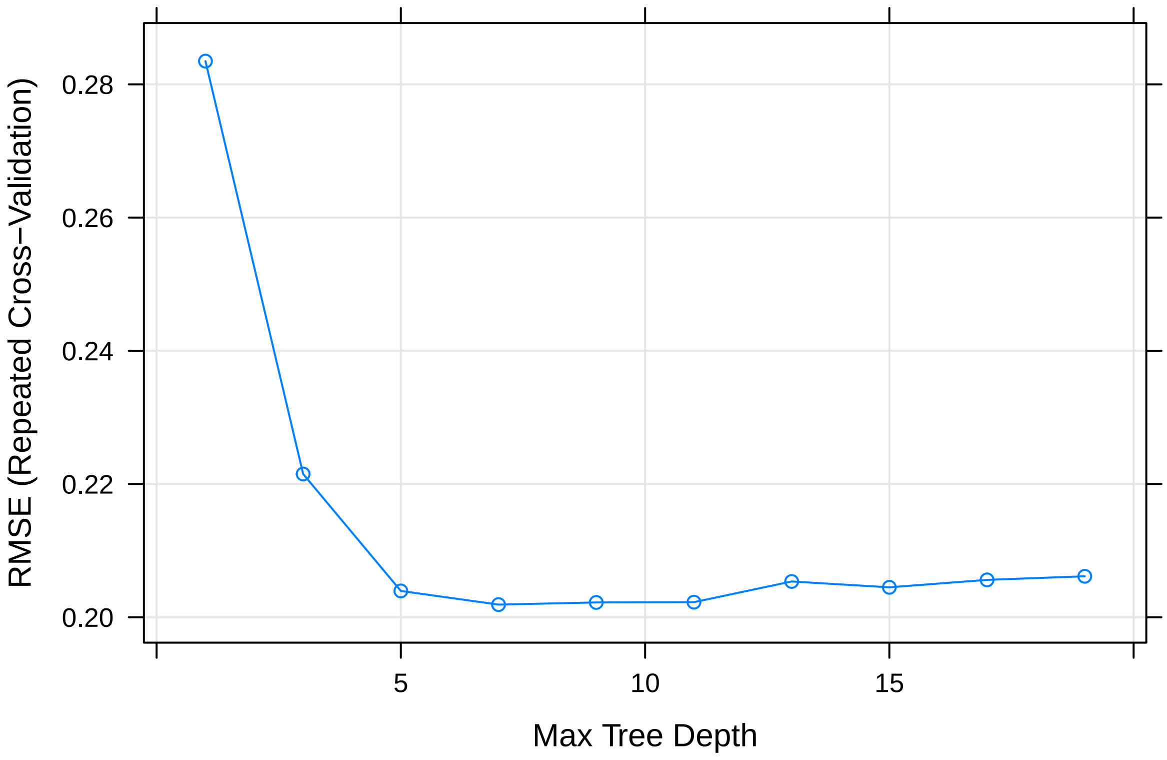
1. We defined our tuning parameters to be chosen via 5 fold cross-validation repeated 5 times.
2. We ran 100 iterations of boosting while running our grid search across various hyper parameters.
3. We chose a learning rate of 0.1 and searched for the max depth of trees by reviewing our reduction in RMSE. This gave us a max-depth of 9.

Important variables and their interpretations:

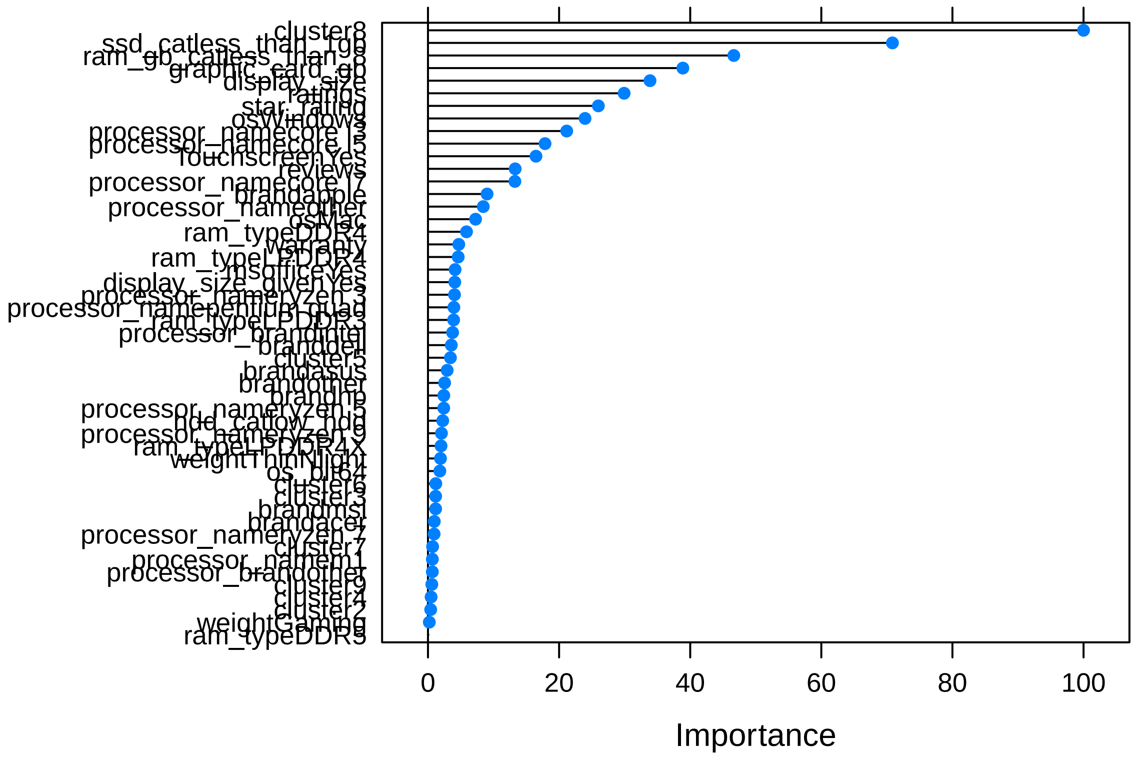
Analysis of important variables showed that laptop cluster8 was providing the best predictive power followed by ram being below 1gb. Other variables such as ram, display size, and ram < 1 gb were also important. All of the important variables were suggesting that our clustering exercise and various treatments on multilevel variables and missing data treatment were useful.

We also applied a partial dependence calculation to understand the direction/sign of associations across the top variables. Our partial dependence plot showcased the following :

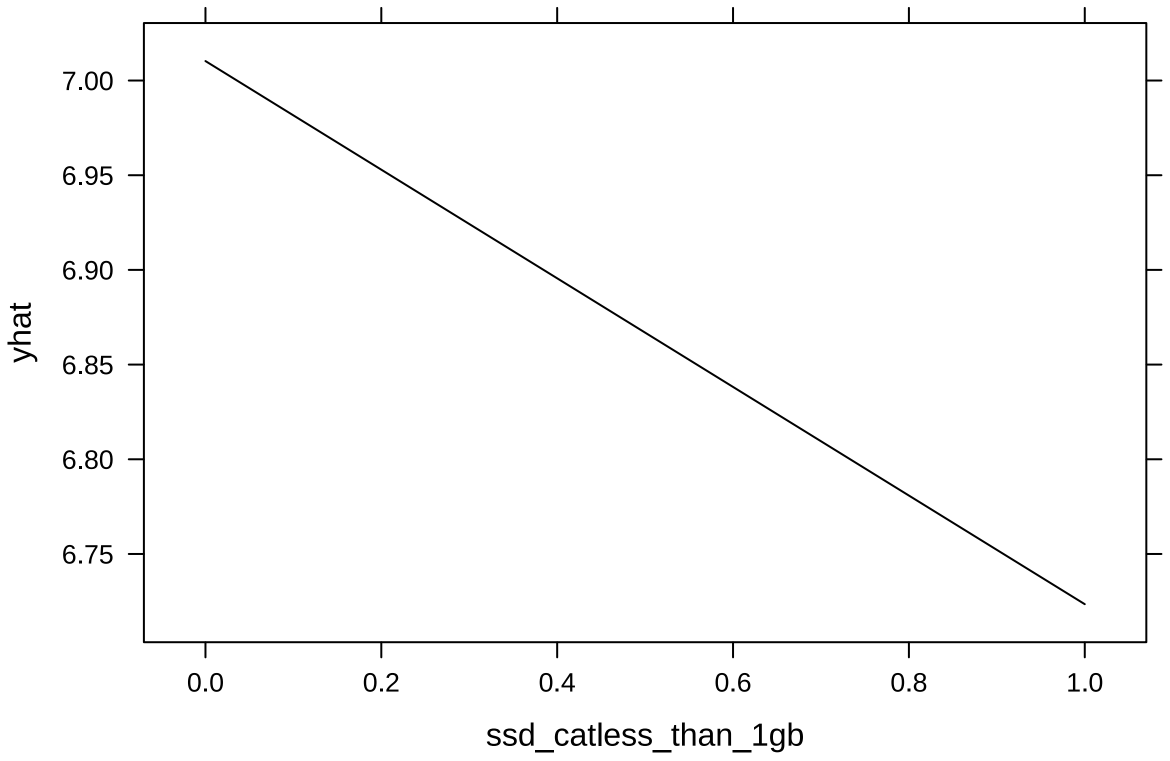
1. We saw that higher performance factors such as RAM being higher than 8 gb and ssd greater than 1 gb were associated with highly priced laptops. Additionally, laptops with higher graphic card specs had higher average prices.
2. It is interesting to observe that as display sizes increased beyond 15.5 inches average prediction of prices went up a lot and then it came down when the display size was around 17.5 inches. This is perhaps an indication that customers are willing to pay higher for an optimal display size of laptops.
3. Lastly laptops not belonging to cluster8 had higher average price predictions. We investigated the reasons behind this a bit further and observed that the centers across the 3 variables used for clustering (ram, sdd and graphic card) were usually lower for cluster8. Hence, we can conclude that low performance laptops are priced lower.

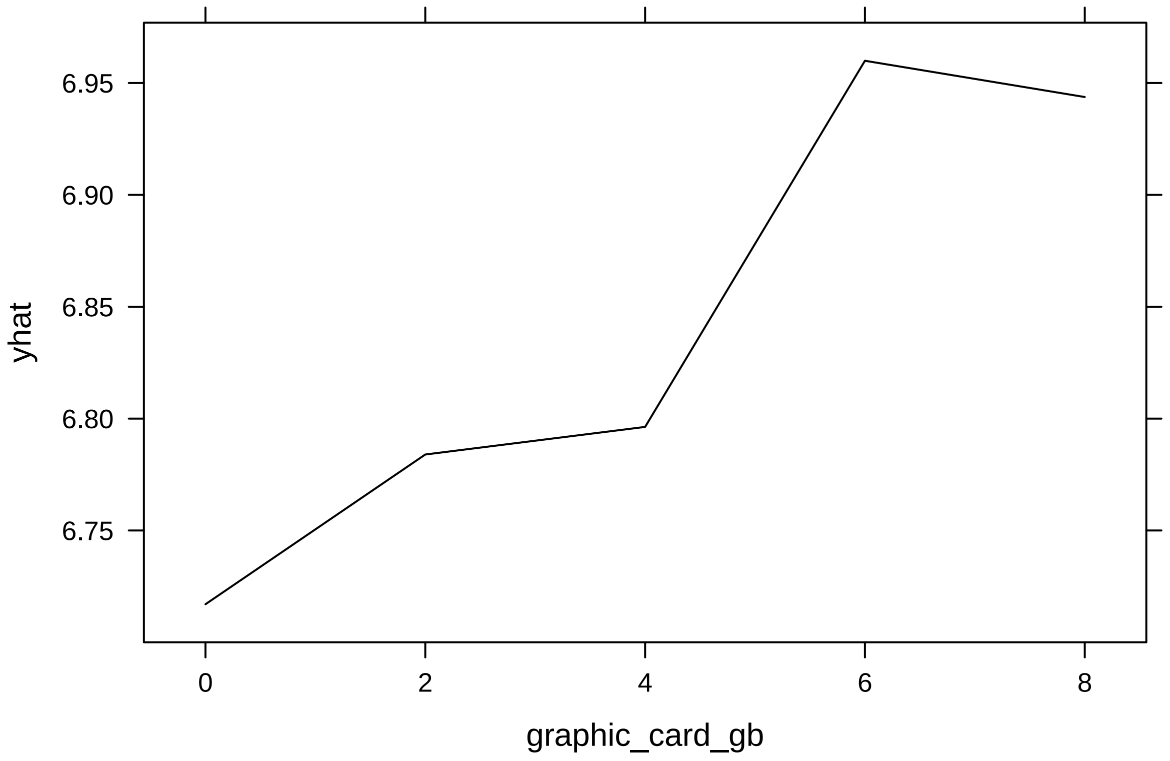


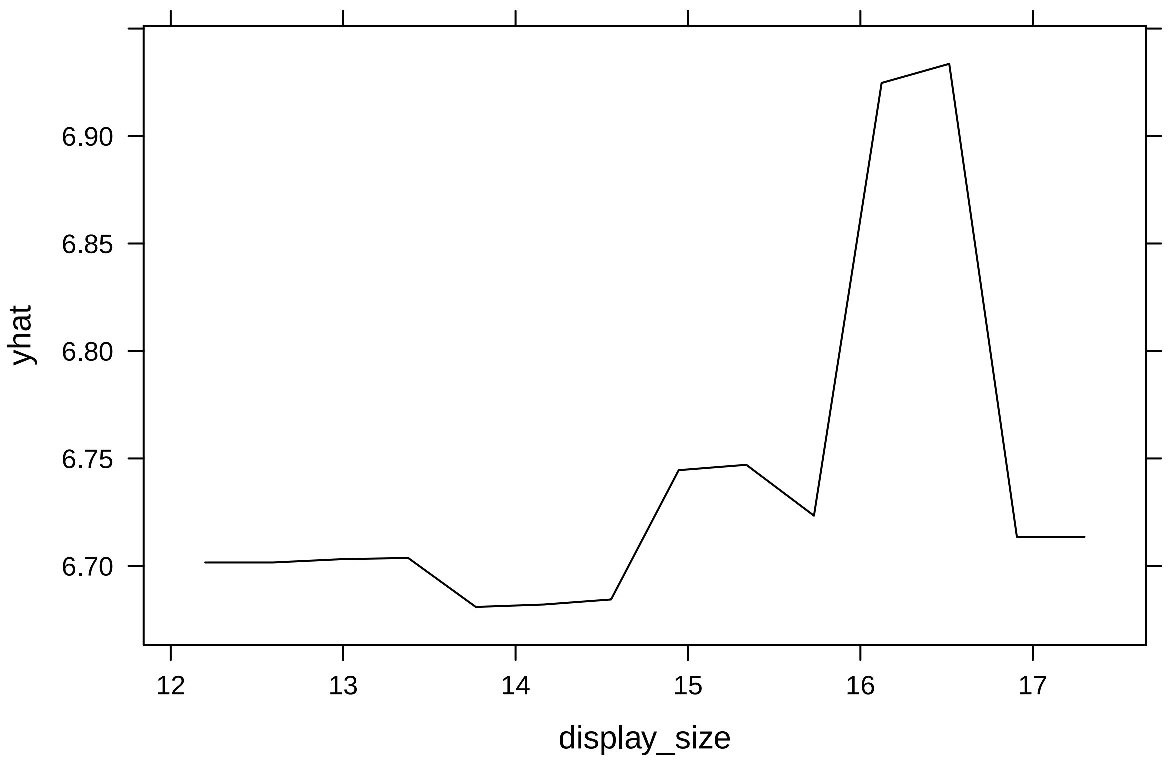
```
## xgbTree variable importance
##
##   only 20 most important variables shown (out of 49)
##
##                       Overall
## cluster8              100.000
## ssd_catless_than_1gb   70.847
## ram_gb_catless_than_8  46.651
## graphic_card_gb        38.878
## display_size           33.874
## ratings                29.922
## star_rating            25.994
## osWindows              23.958
## processor_namecore i3  21.165
## processor_namecore i5  17.858
## TouchscreenYes         16.475
## reviews                13.303
## processor_namecore i7  13.261
## brandapple              9.018
## processor_nameother     8.435
## osMac                   7.264
## ram_typeDDR4            5.880
## warranty                4.700
## ram_typeLPDDR4          4.615
## msofficeYes             4.138
```



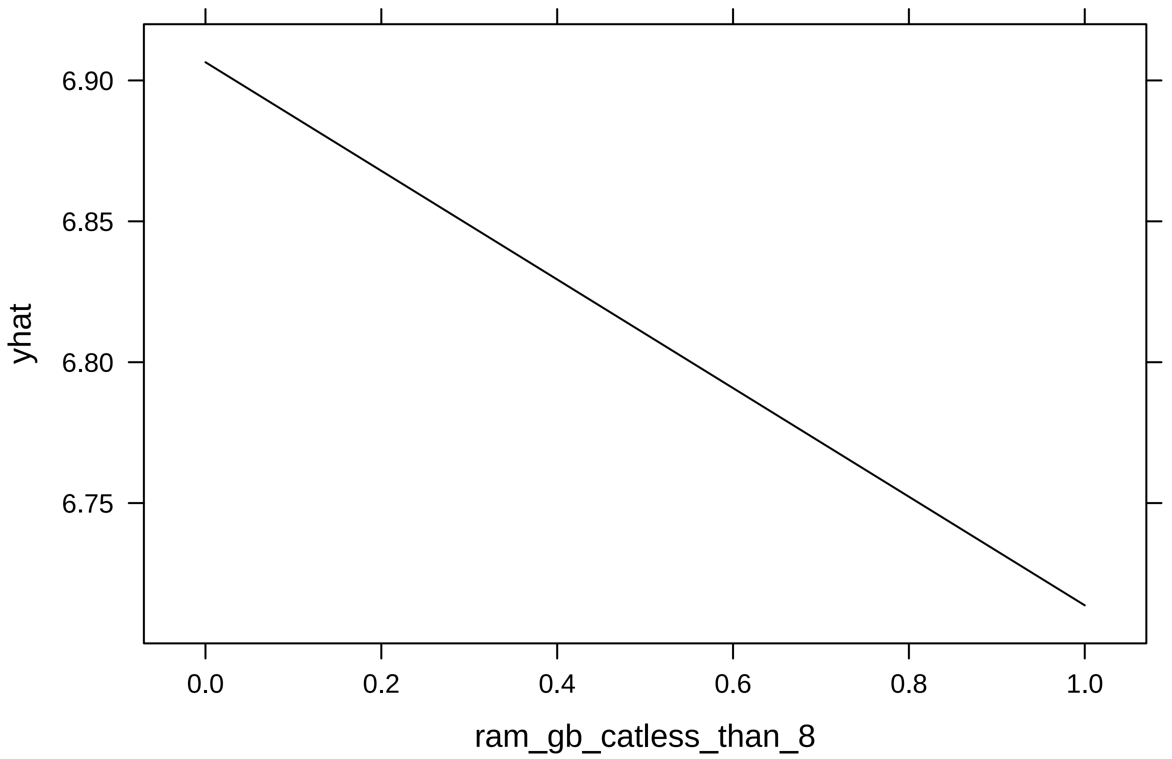
```
## [1] "The R^2 value for XGboost is 0.848757272940201"

## [1] "The RMSE value of XGboost is 227.363013941365"

## [1] "The R^2 value for XGboost in log scale is 0.875737860057181"

## [1] "The RMSE value of XGboost in log scale is 0.177554536560491"
```


yhat
7.00
6.95
6.90
6.85
6.80
6.75
0.0
0.2
0.4
0.6
0.8
1.0
ssd_catless_than_1gb

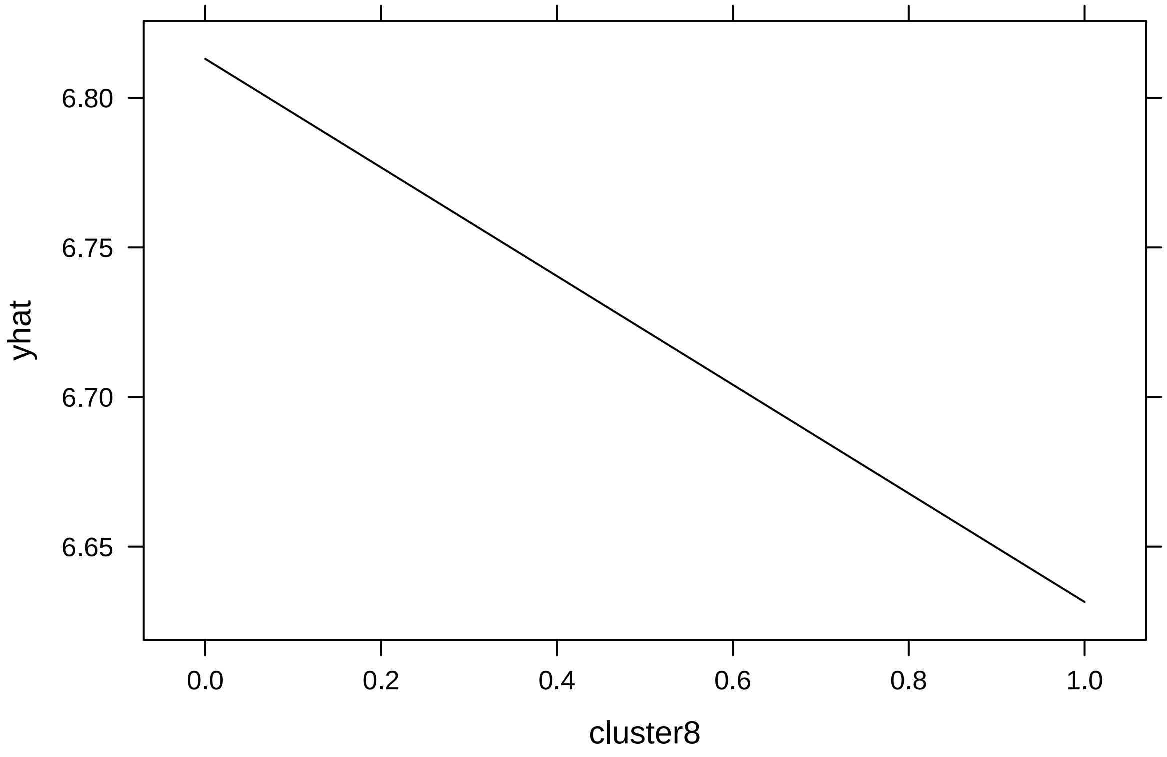

yhat
6.95
6.90
6.85
6.80
6.75
0
2
4
6
8
graphic_card_gb



yhat
6.70
6.75
6.80
6.85
6.90
12
13
14
15
16
17
display_size



yhat
6.75
6.80
6.85
6.90
0.0
0.2
0.4
0.6
0.8
1.0
ram_gb_catless_than_8



```
##       ram_gb         ssd graphic_card_gb
## 1 -0.3749096  0.36593430      -0.5825943
## 2  1.8175269  1.96985951      -0.4353101
## 3 -0.2044961 -1.36759329       2.7754842
## 4  1.8055490 -0.01880008       1.5980072
## 5  1.8055490  2.13434446       2.3861764
## 6  1.7067312  0.13484168      -0.4291087
## 7 -0.3364735 -1.04207616       0.9612901
## 8 -0.5800469 -0.88112711      -0.5825943
## 9 -0.3907221  0.62804222       1.1966508
```

In conclusion, it is wise to review multiple predictive model methodologies when trying to model a particular y variable as results can vary. While our best performing model (XGboost) is still not as accurate as we would like, it is clear that it is a powerful tool for predictive analysis. We are confident the models we presented could be improved with the addition of other explanatory variables and more data points.