BLACKWELL ELECTRONICS

Sales prediction

Blackwell
Electronics

Maja Jurcan
maja.jurcan@gmail.com

Table of Contents

Goals

The goal of this task was to analyze historical sales data to:

- Predict sales of four different product types: PC, Laptops, Netbooks and Smartphones

- Assess the impact services reviews and customer reviews have on sales of different product types

The idea behind the task is to better understand how specific product types perform against each other which will help the sales team better understand how types of products might impact sales across the enterprise.

To do this we will run and optimize 4 different regression methods in R – linear regression, random forest, support vector machine and extreme gradient boosted machines - and compare which one works better for this data set.

Results and conclusions

The data exploration and our models showed that given dataset is a bad fit for sales volume prediction. Our models were especially sensitive to the size of the dataset which in the end caused low confidence in our models. This data is sensitive to both parametric and non-parametric machine learning models and can't be properly used for this kind of business question.

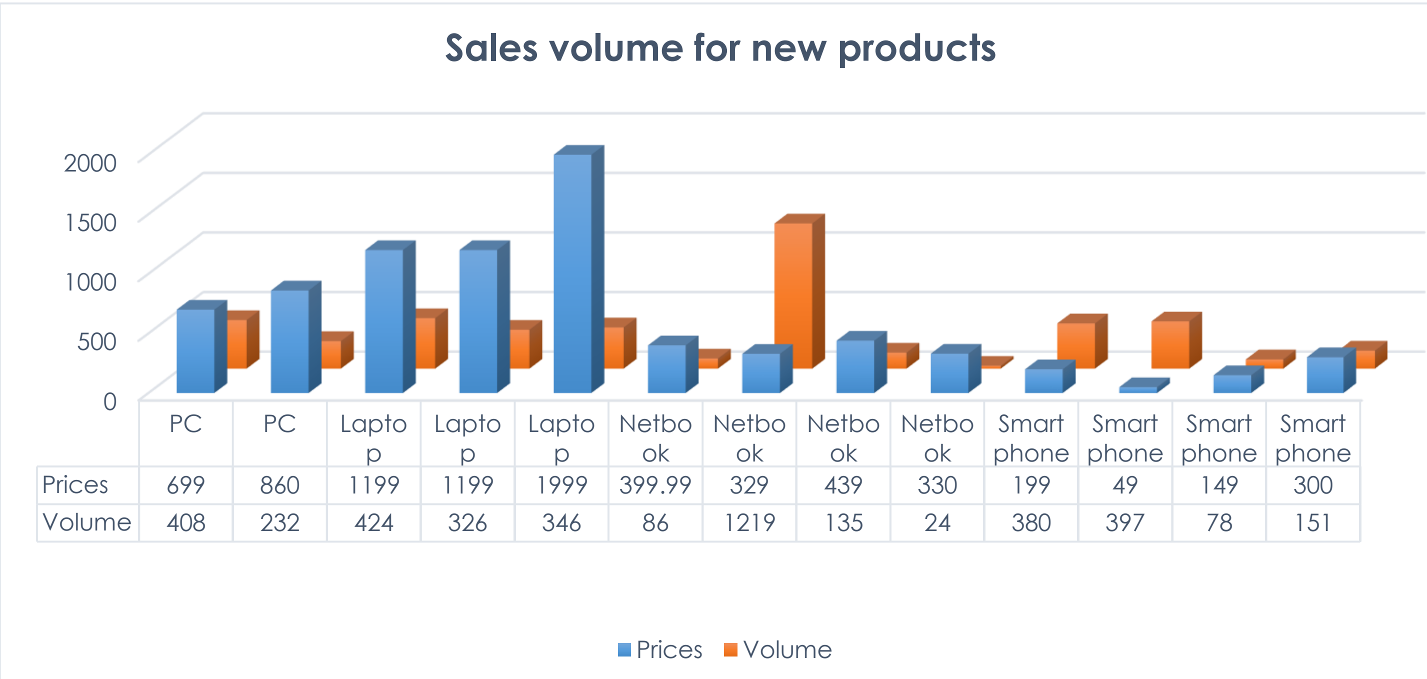
Additional attributes (Age, Gender, In store) were not found relevant for sales prediction and therefore were not used in building and training our predictive models. It is important to note that most of the collected data didn't seem relevant and in any way connected to the sales volume. One reason might be that data collected from different sources doesn't function as good as the data from one source.

The given dataset is a combination of objective and observed data (measure of products, age, in store, gender) and subjective and customer entered data (starred reviews and service reviews) which in the end doesn't make a good dataset for this kind of exploration.

Even though it was shown that sales volume is mostly and almost solely correlated to 5 star reviews, it should be noted that the given dataset with large amount of attributes doesn't get used and that should be changed. The dataset shouldn't be combined in previously described way and even though the dataset didn't function for sales volume prediction, additional exploration of the data might show other connections between attributes.

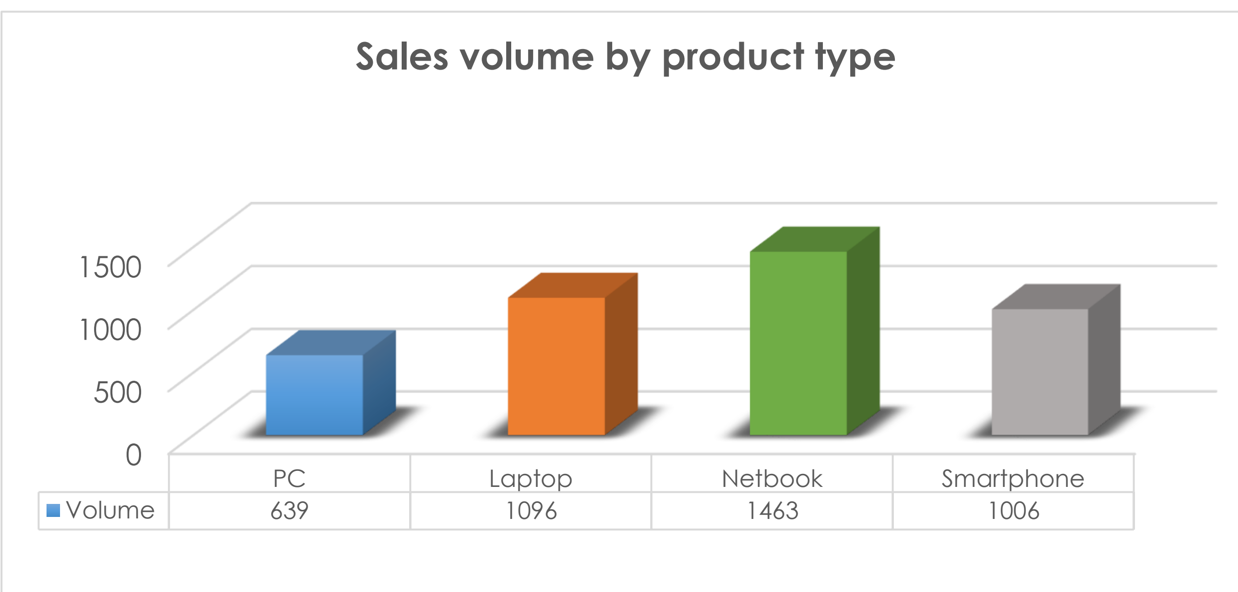
Regarding the impact of starred reviews and service reviews to sales volume, the data shows low correlation to positive or negative service reviews and visible correlation between high starred reviews and high sales volume (Appendix). Only exception is 1 star reviews which don't follow the descending order of starred reviews. In other words, as star reviews go down the sales volume goes down. One star reviews need further exploration to explain this phenomenon (Appendix - Plots).

Final results of our predictions were visualized with a chart of predicted sales volume with prices for the new product dataset with focus on products from 4 categories (PC, Laptop, Netbook, Smartphone).



	PC	PC	Laptop	Laptop	Laptop	Netbook	Netbook	Netbook	Netbook	Smartphone	Smartphone	Smartphone	Smartphone
Prices	699	860	1199	1199	1999	399.99	329	439	330	199	49	149	300
Volume	408	232	424	326	346	86	1219	135	24	380	397	78	151

To visualize the predicted sales volume grouped by product type we used easily readable table with the sums of sales volume by each of four targeted product types.



	PC	Laptop	Netbook	Smartphone
Volume	639	1096	1463	1006

Appendix

1. Overview of the data

The dataset consists of 250 observations through 21 attributes.

- **Product Type**: Accessories, Display, Extended Warranty, Game Console, Laptop, Netbook, PC, Printer, Printer Supplies, Smartphone, Software, Tablet
- **Product Number**: Internal product indexing
- **Price**
- **x5StarReviews**
- **x4StarReviews**
- **x3StarReviews**
- **x2StarReviews**
- **x1StarReviews**
- **Positive Service Review**
- **Negative Service Review**
- **Recommend product:** would consumer recommend product
- **Best Sellers Rank**
- **Weight**
- **Depth**
- **Width**
- **Height**
- **Profit Margin**
- **Volume:** Sales volume
- **Gender:** 0 – Female; 1 - Male
- **Age:** 1 - 18-34; 2 – 35-51; 3 – 52-66; 4 – 67-80 ;
- **In store:** 1 – In store; 0 - Online

2. Preprocessing

Preprocessing of the data included looking at data distributions. It was immediately obvious that our dataset has right skewed distribution. That information will affect our model building and should be taken into consideration. During this step the outliers were detected and removed only for our dependent variable Volume, but in further data exploration outliers in other attributes should be removed as well.

In the preprocessing step it was decided to:

- remove **outliers** from Volume attribute (values over 2000 were removed; altogether 4 observations)
- remove **Product ID** attribute because it represents internal product indexing, thus it doesn't add any information to our models
- remove **Best Seller Rank** because of the large number of NA values; another approach would be instead of deleting the column or the observations, to exchange NAs with the medians
- remove **3 Star** and **1 star** reviews because of collinearity above 0.85 (see correlation matrix) to avoid overfitting
- remove 7 observations of the same product in Extended Warranty product type
- remove NA value from attribute **Width**
- change data type from factor to numeric (with prior transformation to character) for attribute **Depth**
- dummify the **Product type** attribute for our regression models, because they don't accept non numeric data as input

Every step outlined above was done for the existing products dataset, and later for new products dataset.

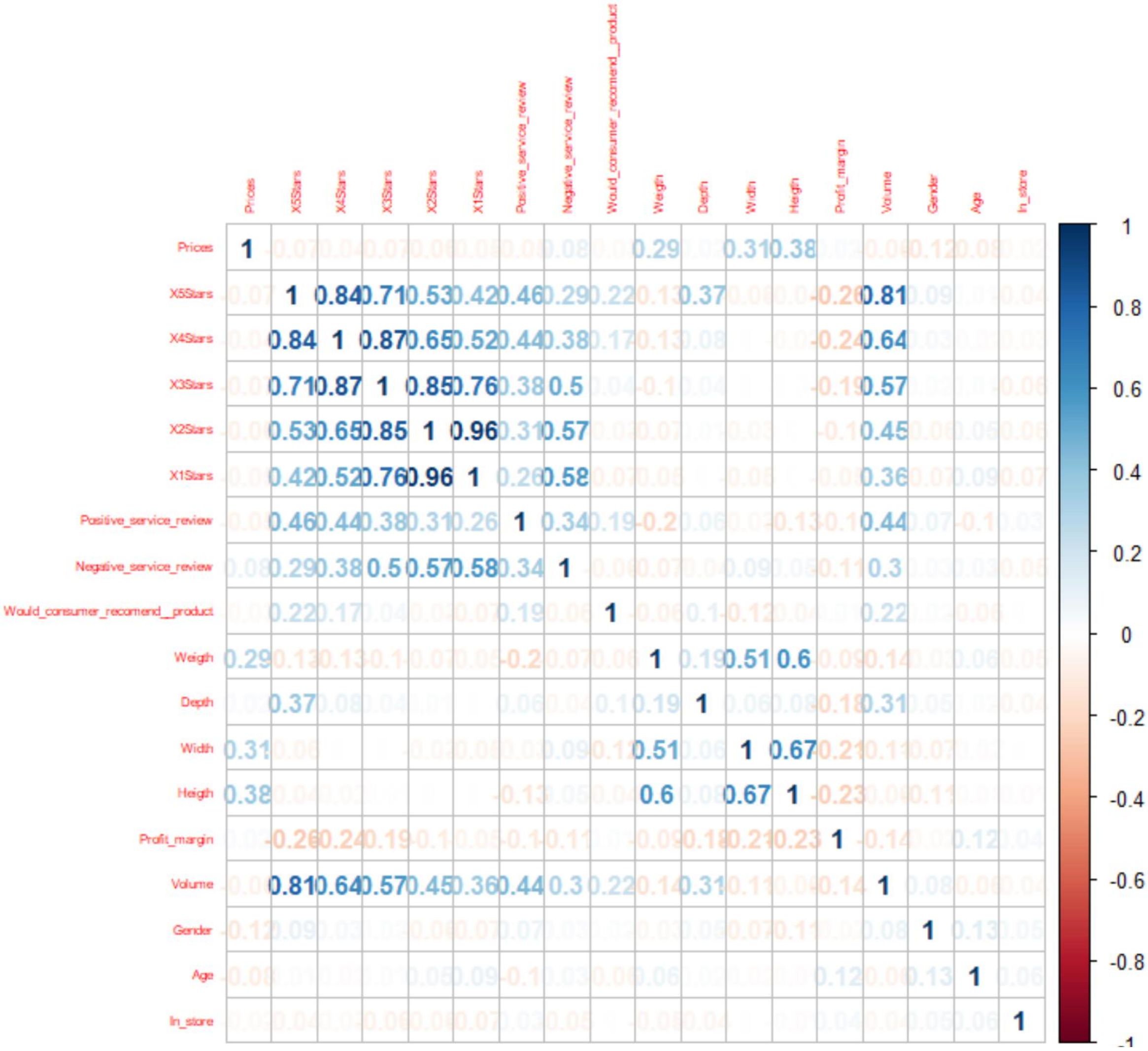
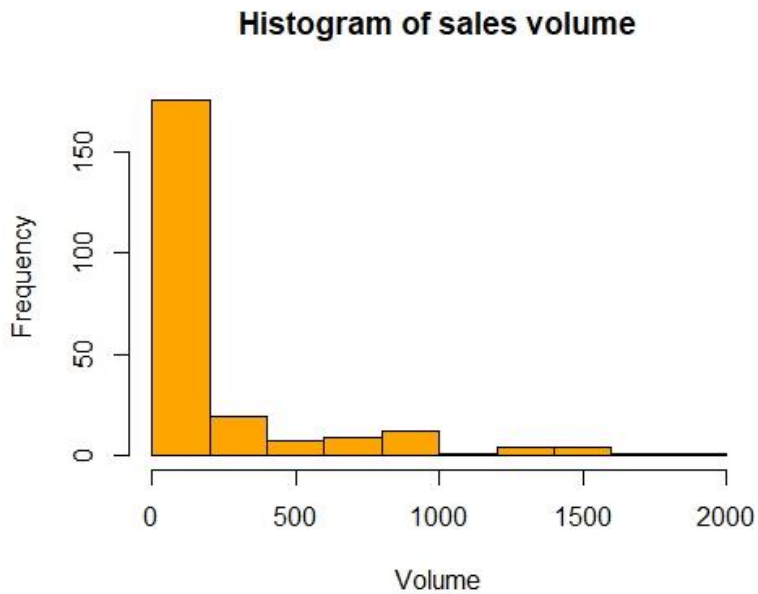


Figure: Correlation matrix

3. Methodology

As mentioned previously, our data showed right skewed distribution which played important role in our model building techniques. Parametric methods require normal data distribution, while non parametric methods work with other distributions of data.



Linear regression (parametric) model showed poor metrics with the R^2 value of 0.52, RMSE 226 and MAE 152. As we can see the metrics of our base model were not good and considering the non normal data distribution it was expected.

It was expected that non parametric models would perform better. Even though that was true, we still couldn't get metrics that would make one model a good model for our data.

We chose to build our models using three different algorithms and then choose the best performing one. The algorithms used for our model building were: random forest, support vector machine and extreme gradient boosted trees.

The model that performed the best was random forest model. The metrics can be seen in next sections. Our chosen model was better than other models, but as it was said before, our level of confidence in our model was not even remotely high. Even though the model showed poor metrics on our training set, testing set was predicted with relevant accuracy. But it must be said once again, our model is unstable and can't be used properly for sales volume prediction in this dataset.

Models were build using combination of most relevant attributes for sales volume according to decision tree and correlation matrix. The attribute combinations were:

- Volume~. – rest of the dataset; only random forest significantly benefited from additional attributes
- Volume ~ 5 stars, 4 stars, Positive and negative service review, Product type software and printer supplies
- Volume ~ 5 stars, 4 stars, Positive service review
- Volume ~ 5 stars, 4 stars, Negative service review
- Volume ~ 5 stars, Positive and negative service review
- Volume ~ 5 stars
- Volume ~ 5 stars, Positive service review, Product type software

4. Models

4.1. Random forest:

```
> RFFit
Random Forest

236 samples
 27 predictor

Pre-processing: re-scaling to [0, 1] (27)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 212, 212, 212, 213, 213, 213, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
   2    264.3656  0.6015832  176.5690
  14    213.5207  0.6355034  132.6928
  27    209.6046  0.6313508  128.6719

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 27.
> predictionRFFit <- predict(RFFit, testSet)
> #performace measurment
> postResample(predictionRFFit, testSet$Volume)
     RMSE  Rsquared       MAE
110.46256   0.93682  62.51669
```

Chosen model:

```
> RFFit2 <- train(Volume ~ X5Stars + X4Stars + Positive_service_review +
+                 Product_type.Software + Product_type.Printer.Supplies +
+                 Negative_service_review,
+                 data = ExistingDummy,
+                 method = 'rf',
+                 trControl=fitControl,
+                 tuneLength = 3,
+                 preProc = "range")
> RFFit2
Random Forest

236 samples
  6 predictor

Pre-processing: re-scaling to [0, 1] (6)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 212, 212, 212, 215, 212, 212, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     229.3447  0.6151691  141.5977
  4     221.6693  0.6224082  134.1201
  6     219.6340  0.6249389  131.5769

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 6.
> predictionRFFit2 <- predict(RFFit2, testSet)
> #performace measurment
> postResample(predictionRFFit2, testSet$Volume)
       RMSE    Rsquared         MAE
135.2945495   0.8988452  78.1262968
```

```
> RFFit3 <- train(Volume ~ X5Stars + X4Stars + Positive_service_review,
+                 data = ExistingDummy,
+                 method = 'rf',
+                 trControl=fitControl,
+                 preProc = "range")
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2

> RFFit3
Random Forest

236 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 213, 212, 212, 213, 212, 212, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     236.0977  0.5545534  146.5820
  3     235.4254  0.5508618  145.5583

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
> predictionRFFit3 <- predict(RFFit3, testSet)
> #performace measurment
> postResample(predictionRFFit3, testSet$Volume)
       RMSE    Rsquared         MAE
144.5092892   0.8879281  88.3492698
```

```
> RFFit4 <- train(Volume ~ X5Stars + X4Stars + Negative_service_review,
+                  data = ExistingDummy,
+                  method = 'rf',
+                  trControl=fitControl,
+                  preProc = "range")
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .

> RFFit4
Random Forest

236 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 212, 212, 212, 212, 213, 212, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     238.1740  0.5607365  156.2477
  3     238.6834  0.5567805  156.1787

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
> predictionRFFit4 <- predict(RFFit3, testSet)
> predictionRFFit4 <- predict(RFFit4, testSet)
> #performace measurment
> postResample(predictionRFFit4, testSet$Volume)
       RMSE    Rsquared         MAE
180.6891323   0.8075887 110.6562957


> RFFit5
Random Forest

236 samples
  1 predictor

Pre-processing: re-scaling to [0, 1] (1)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 212, 213, 213, 212, 212, 213, ...
Resampling results:

  RMSE      Rsquared   MAE
  238.3357  0.5415186  156.3731

Tuning parameter 'mtry' was held constant at a value of 2
> predictionRFFit5 <- predict(RFFit5, testSet)
> #performace measurment
> postResample(predictionRFFit5, testSet$Volume)
       RMSE    Rsquared         MAE
254.1832678   0.6121778 151.2557781
```

```
> RFFit7 <- train(Volume ~ X5Stars + Positive_service_review + Product_type.Software,
+                 data = ExistingDummy,
+                 method = 'rf',
+                 trControl=fitControl,
+                 preProc = "range")
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .

> RFFit7
Random Forest

236 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 213, 212, 213, 213, 212, 212, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     217.7143  0.6228147  129.6049
  3     218.4239  0.6163094  128.3805

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
> predictionRFFit7 <- predict(RFFit7, testSet)
> #performace measurment
> postResample(predictionRFFit7, testSet$Volume)
       RMSE    Rsquared         MAE
155.9357625   0.8690092  92.7011001
```

2. SVM

```
> SVMFit <- train(Volume ~ .,
+                 data = ExistingDummy,
+                 method = 'svmLinear',
+                 trControl=fitControl,
+                 tuneLength = 3,
+                 preProc = "range")
> SVMFit
Support Vector Machines with Linear Kernel

233 samples
 27 predictor

Pre-processing: re-scaling to [0, 1] (27)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 209, 209, 211, 210, 209, 212, ...
Resampling results:

  RMSE      Rsquared   MAE
  226.6515  0.5266505  118.8137

Tuning parameter 'C' was held constant at a value of 1
> predictionSVMFit <- predict(SVMFit, testSet)
> #performace measurment
> postResample(predictionSVMFit, testSet$Volume)
      RMSE    Rsquared         MAE
276.282154   0.584146 128.587377
```

```
> SVMFit2 <- train(Volume ~ X5Stars + X4Stars + Positive_service_review +
+                  Product_type.Software + Product_type.Printer.Supplies +
+                  Negative_service_review, data = ExistingDummy,
+                  method = 'svmLinear',
+                  trControl=fitControl,
+                  tuneLength = 3,
+                  preProc = "range")
> SVMFit2
Support Vector Machines with Linear Kernel

233 samples
  6 predictor

Pre-processing: re-scaling to [0, 1] (6)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 209, 210, 210, 209, 209, 209, ...
Resampling results:

  RMSE      Rsquared   MAE
  227.8197  0.5146647  117.6643

Tuning parameter 'C' was held constant at a value of 1
> predictionSVMFit2 <- predict(SVMFit2, testSet)
> #performace measurment
> postResample(predictionSVMFit2, testSet$Volume)
       RMSE    Rsquared         MAE
287.1045835   0.5450966 134.4222804
```

```
> SVMFit3 <- train(Volume ~ X5Stars + X4Stars + Positive_service_review,
+                  data = ExistingDummy,
+                  method = 'svmLinear',
+                  trControl=fitControl,
+                  tuneLength = 3,
+                  preProc = "range")
> SVMFit3
Support Vector Machines with Linear Kernel

233 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 209, 210, 211, 209, 209, 210, ...
Resampling results:

  RMSE     Rsquared   MAE
  232.714  0.4763872  116.4975

Tuning parameter 'C' was held constant at a value of 1
> predictionSVMFit3 <- predict(SVMFit3, testSet)
> #performace measurment
> postResample(predictionSVMFit3, testSet$Volume)
       RMSE    Rsquared         MAE
294.8135446   0.5246245 138.9903510
```

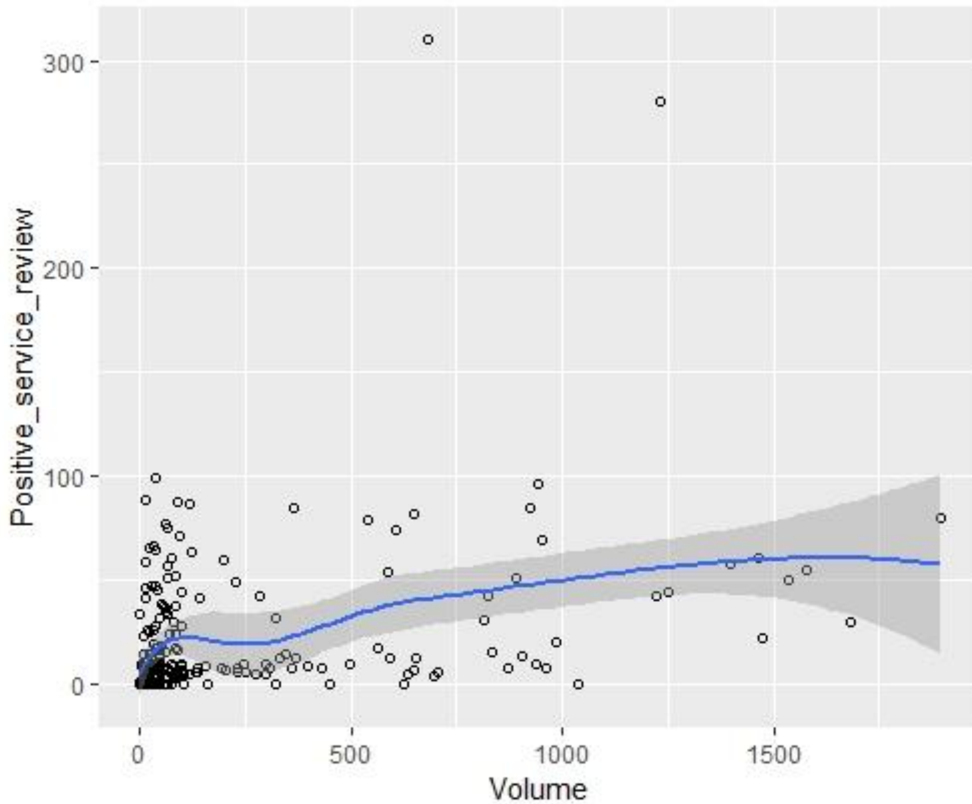
```
> SVMFit4 <- train(Volume ~ X5Stars + X4Stars + Negative_service_review,
+                  data = ExistingDummy,
+                  method = 'svmLinear',
+                  trControl=fitControl,
+                  tuneLength = 3,
+                  preProc = "range")
> SVMFit4
Support Vector Machines with Linear Kernel

233 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 210, 210, 209, 210, 209, 209, ...
Resampling results:

  RMSE      Rsquared   MAE
  231.3436  0.4981807  116.6493

Tuning parameter 'C' was held constant at a value of 1
> predictionSVMFit4 <- predict(SVMFit4, testSet)
> #performace measurment
> postResample(predictionSVMFit4, testSet$Volume)
       RMSE    Rsquared         MAE
293.4044607   0.5281107 137.4354514
```

```
> SVMFit5 <- train(Volume ~ X5Stars + Positive_service_review + Product_type.Software,
+          data = ExistingDummy,
+          method = 'svmLinear',
+          trControl=fitControl,
+          tuneLength = 3,
+          preProc = "range")
> SVMFit5
Support Vector Machines with Linear Kernel

233 samples
  3 predictor

Pre-processing: re-scaling to [0, 1] (3)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 210, 210, 209, 210, 211, 209, ...
Resampling results:

  RMSE      Rsquared   MAE
  232.0838  0.4886749  117.7016

Tuning parameter 'C' was held constant at a value of 1
> predictionSVMFit5 <- predict(SVMFit5, testSet)
> #performace measurment
> postResample(predictionSVMFit5, testSet$Volume)
       RMSE    Rsquared         MAE
291.8280546   0.5307913 137.0536797
```

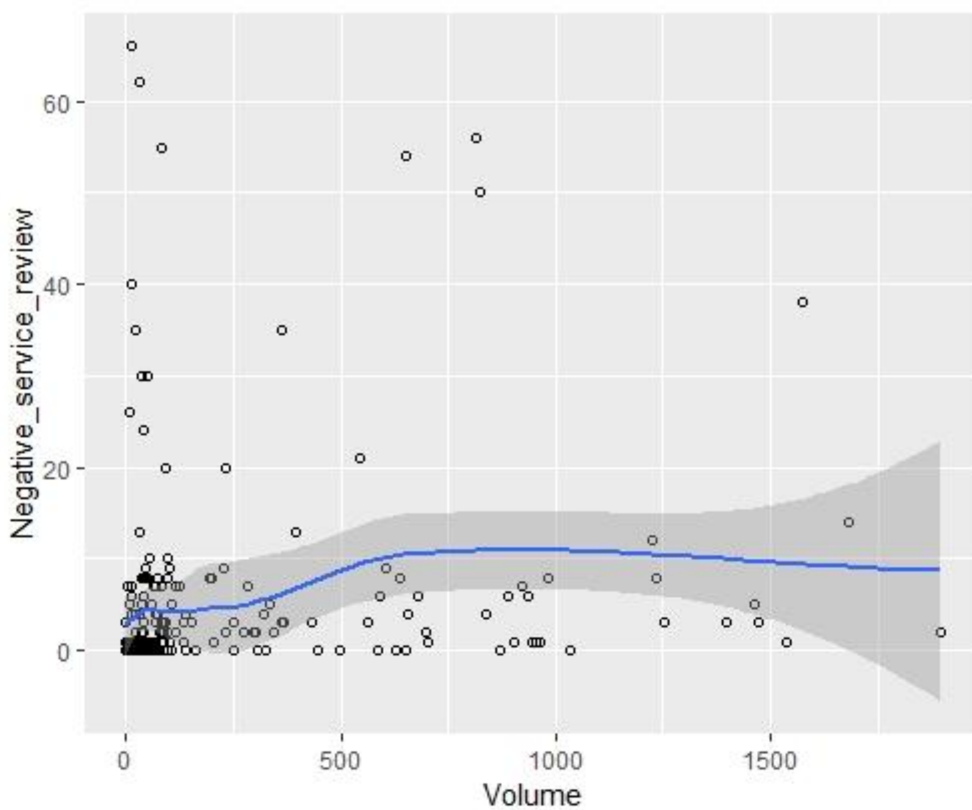
3. XGBT

```
> XGBTFit2 <- train(Volume ~ X5Stars + X4Stars + Positive_service_review +
+         Product_type.Software + Product_type.Printer.Supplies +
+         Negative_service_review, data = ExistingDummy,
+         method = 'xgbTree',
+         trControl=fitControl,
+         tuneLength = 3,
+         preProc = "range")
> XGBTFit2
eXtreme Gradient Boosting

233 samples
  6 predictor

Pre-processing: re-scaling to [0, 1] (6)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 209, 209, 209, 209, 209, 211, ...
Resampling results across tuning parameters:

  eta  max_depth  colsample_bytree  subsample  nrounds  RMSE      Rsquared   MAE
  0.3  1          0.6               0.50        50      221.2656  0.5376029  145.5583
  0.3  1          0.6               0.50       100      222.9549  0.5329118  147.2889
  0.3  1          0.6               0.50       150      223.8986  0.5317823  148.2083
  0.3  1          0.6               0.75        50      223.2812  0.5355271  147.1002
  0.3  1          0.6               0.75       100      224.0456  0.5347972  146.6649
  0.3  1          0.6               0.75       150      223.3740  0.5367157  146.0174
  0.3  1          0.6               1.00        50      221.7602  0.5341221  145.3046
  0.3  1          0.6               1.00       100      221.7658  0.5345500  144.6694
  0.3  1          0.6               1.00       150      222.3234  0.5343620  144.1232
  0.3  1          0.8               0.50        50      219.3720  0.5453888  143.9975
  0.3  1          0.8               0.50       100      222.0538  0.5402683  146.7094
  0.3  1          0.8               0.50       150      225.8415  0.5298305  149.0363
  0.3  1          0.8               0.75        50      220.1316  0.5430903  144.4037
  0.3  1          0.8               0.75       100      220.9111  0.5415779  144.2917


  Tuning parameter 'gamma' was held constant at a value of 0
  Tuning parameter
   'min_child_weight' was held constant at a value of 1
  RMSE was used to select the optimal model using the smallest value.
  The final values used for the model were nrounds = 50, max_depth = 1, eta = 0.3, gamma =
   0, colsample_bytree = 0.8, min_child_weight = 1 and subsample = 0.5.
  > predictionXGBTFit2 <- predict(XGBTFit2, testSet)
  > #performace measurment
  > postResample(predictionXGBTFit2, testSet$Volume)
        RMSE    Rsquared         MAE
  235.068296    0.664413 147.601076
```

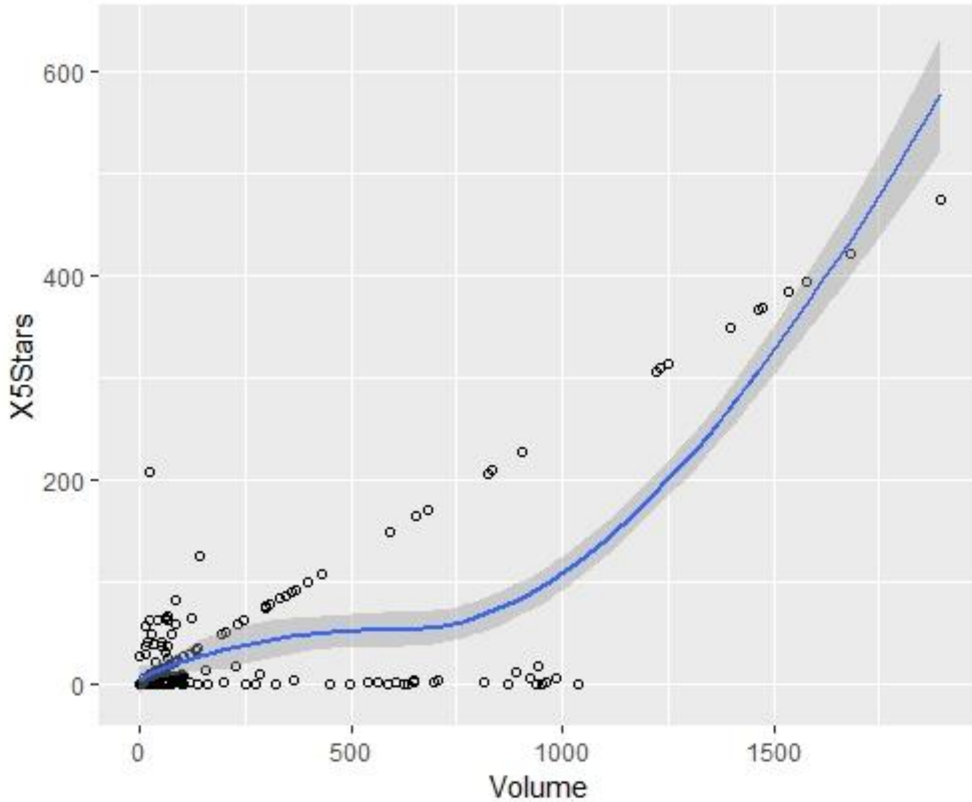
5. Plots

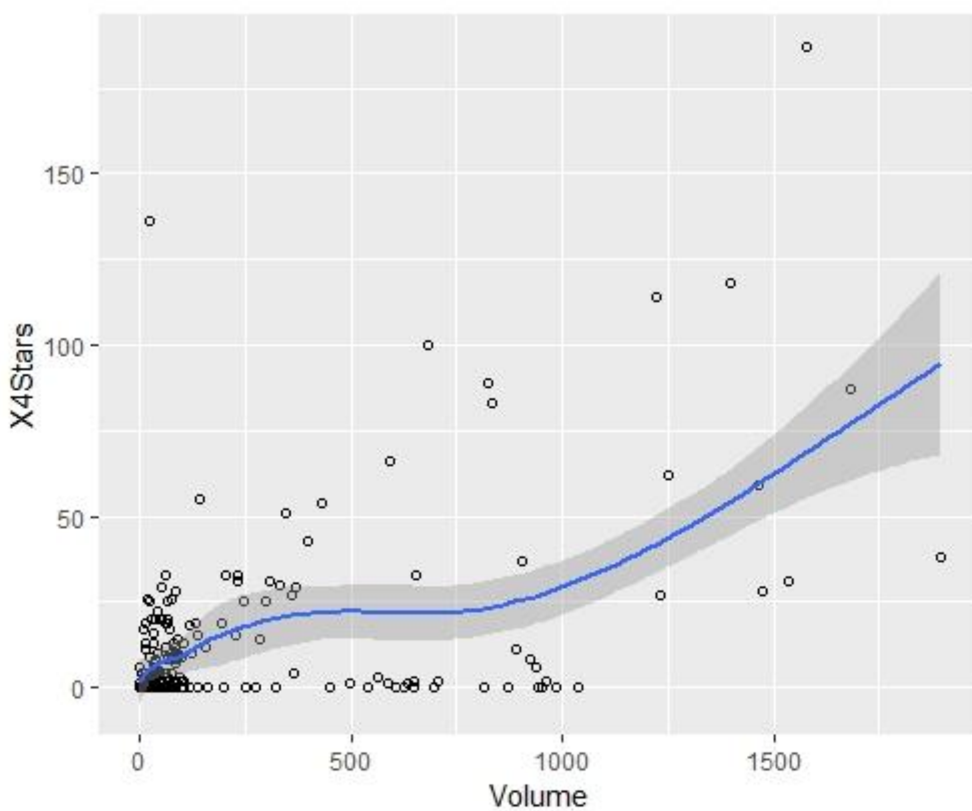


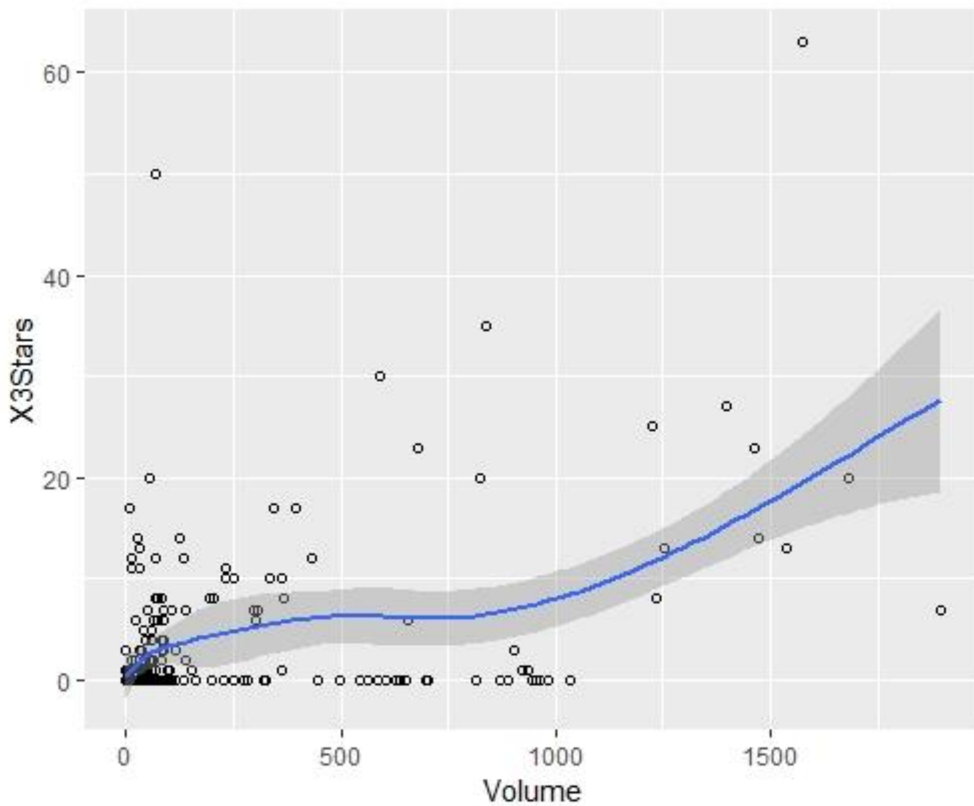
300
200
100
0
Positive_service_review
0
500
1000
1500
Volume



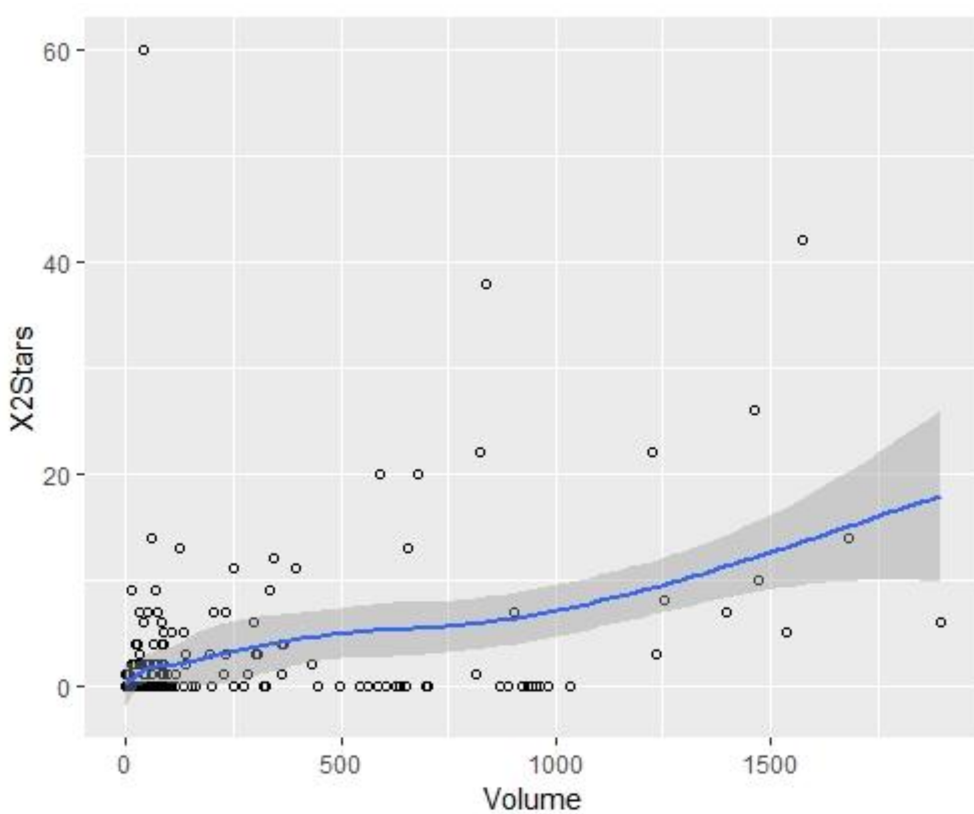
60
40
20
0
Negative_service_review
0
500
1000
1500
Volume


600
400
200
0
X5Stars
0
500
1000
1500
Volume

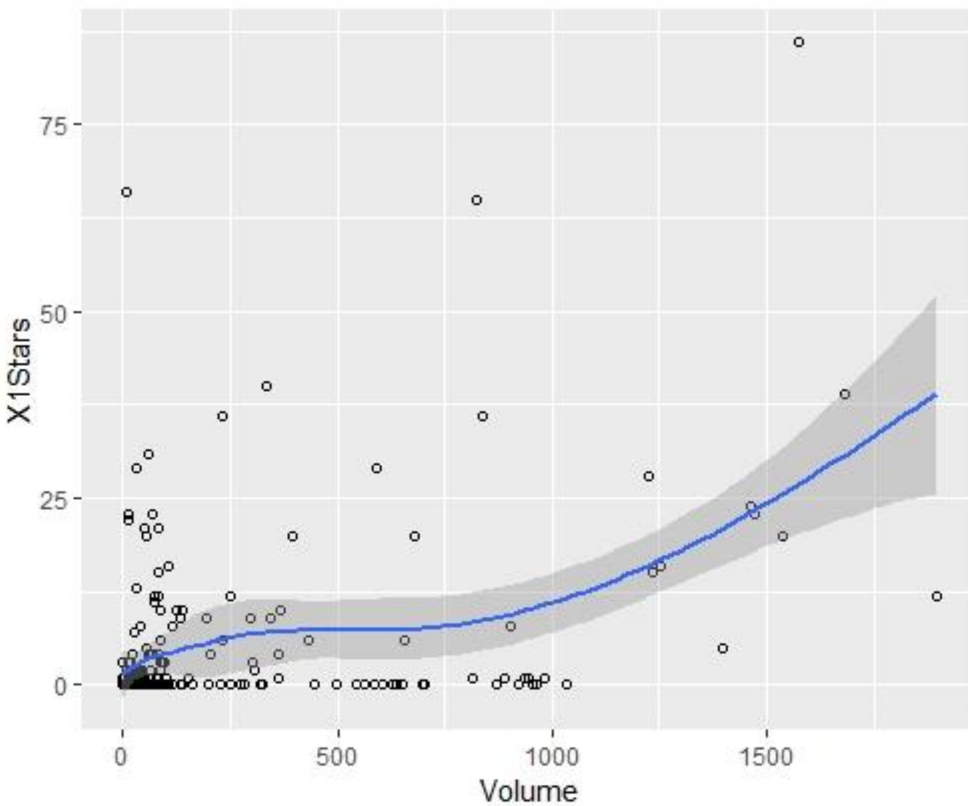

150
100
50
0
X4Stars
0
500
1000
1500
Volume



60
40
20
0
X3Stars
0
500
1000
1500
Volume



60
40
20
0
X2Stars
0
500
1000
1500
Volume


X1Stars
Volume
75
50
25
0
0
500
1000
1500