Multiple Regression in R

Elias Smouni, Joël Ribera Zaragoza

2019-07-17

Contents

Load packages and datasets

```r
pacman::p_load(readr,readxl,rstudioapi,ggplot2,cowplot,GGally,caret,dplyr,party)
products <- read_csv("datasets/existingproductattributes2017.csv")
```

```
## Parsed with column specification:
## cols(
##   ProductType = col_character(),
##   ProductNum = col_double(),
##   Price = col_double(),
##   x5StarReviews = col_double(),
##   x4StarReviews = col_double(),
##   x3StarReviews = col_double(),
##   x2StarReviews = col_double(),
##   x1StarReviews = col_double(),
##   PositiveServiceReview = col_double(),
##   NegativeServiceReview = col_double(),
##   Recommendproduct = col_double(),
##   BestSellersRank = col_double(),
##   ShippingWeight = col_double(),
##   ProductDepth = col_double(),
##   ProductWidth = col_double(),
##   ProductHeight = col_double(),
##   ProfitMargin = col_double(),
##   Volume = col_double()
## )
```

```r
newproducts <- read_csv("datasets/newproductattributes2017.csv")
```

```
## Parsed with column specification:
## cols(
##   ProductType = col_character(),
##   ProductNum = col_double(),
##   Price = col_double(),
##   x5StarReviews = col_double(),
##   x4StarReviews = col_double(),
##   x3StarReviews = col_double(),
##   x2StarReviews = col_double(),
##   x1StarReviews = col_double(),
##   PositiveServiceReview = col_double(),
##   NegativeServiceReview = col_double(),
##   Recommendproduct = col_double(),
##   BestSellersRank = col_double(),
##   ShippingWeight = col_double(),
##   ProductDepth = col_double(),
##   ProductWidth = col_double(),
##   ProductHeight = col_double(),
##   ProfitMargin = col_double(),
##   Volume = col_double()
## )
```

Executive Summary

Introduction

We were requested to conduct further analysis regarding an earlier data mining operation, in which we used RapidMiner to predict projected sales volumes and subsequent profitability for a shortlist of products considered as possible additions to Blackwell's selection.

Specifically, we were requested to pick up where we left off, by providing a more elaborate analysis based on the same data. For this analysis, we were asked to explore what, if any, influence the respective product types have on the projecting of sales volumes. Furthermore, we were requested to:

A. Predict sales volumes of the following product types:

i) PC;
ii) Laptops;
iii) Netbooks; and
iv) Smartphones

B. Assess the impact that services' and customer satisfaction

The purpose of this summary is to provide the reader with an overview of our key findings and consists appropriately of weighted conclusions. We refer you to the Technical Report that is appended hereto for technical documentation and argumentation.

About the data

We implemented our analysis on the same historical data that was used as basis for our projection regarding the envisaged new products. Consequently, the dataset imposes the same limitations as they did in regard of predicting the sales volumes of new products.

The data consisted of 80 observations of 18 attributes in regard of products sold by Blackwell. Our tests show that the most influential attributes in predicting sales were the attributes related to customer sentiment as is evidenced in the technical report.

We note that the data is unevenly distributed and contained a few abnormalities. There were also a few obvious errors from the data collection phase as well as missing data which resulted in the deletion of a number of observations and one attribute, which we would have been glad to avoid considering the sparsity of the dataset.

Results

We began by analyzing the influence of the attribute “ProductType” on predicting sales volume. We did this through a variance-analysis method, namely ANOVA. The analysis showed that said attribute had no significance toward an inferential analysis of sales.

This task was a so-called multiple-regression task. We trained the number of supervised-learning algorithms and received the best results with a decision-tree algorithm called Random Forest. Our training yielded a model that inferred the volume from the data with a 92% accuracy.

Prediction

The projected sales volumes for the product types are presented in the following table.

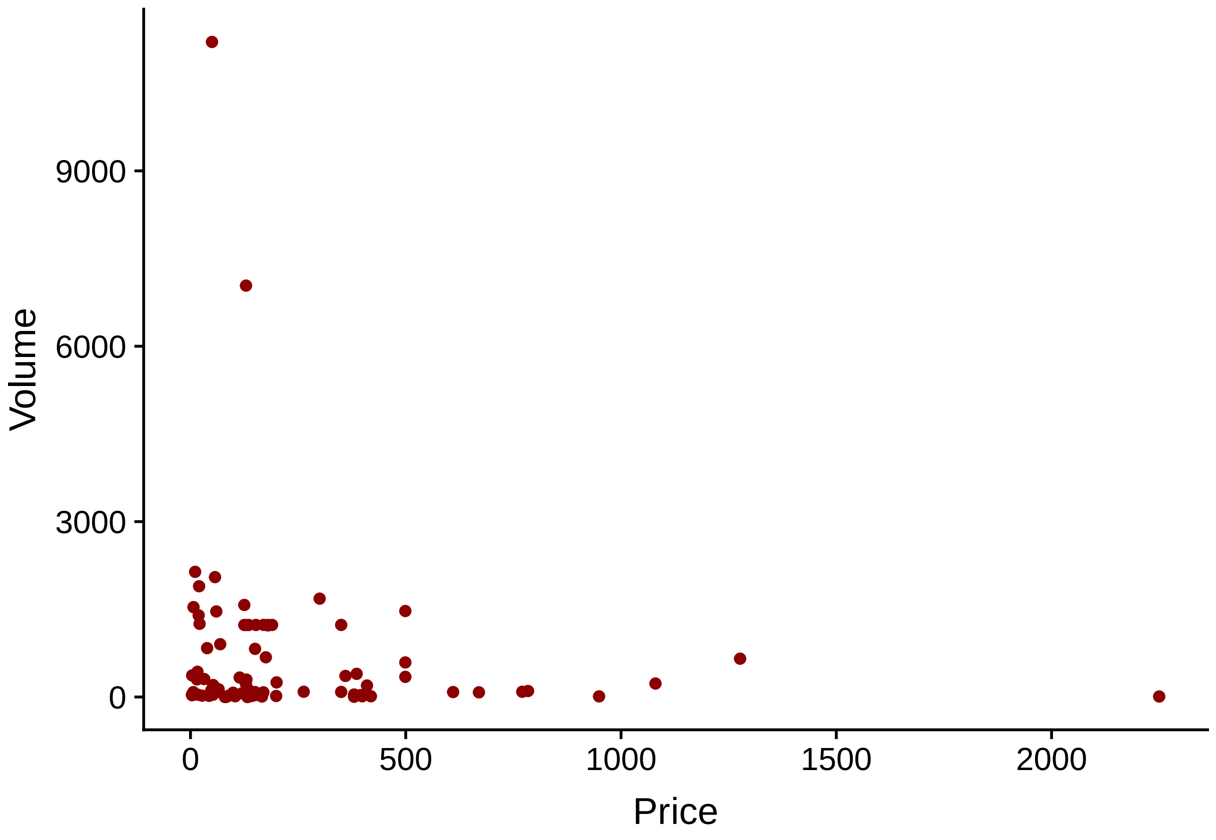
ProductType	ProductNum	Brand	Volume
Netbook	180	Acer	1191
Smartphone	194	Samsung	899
Smartphone	193	Motorola	261
PC	171	Dell	217
Laptop	173	Apple	177
Netbook	181	Asus	159
PC	172	Dell	104
Smartphone	196	Motorola	95
Smartphone	195	HTC	84
Netbook	178	HP	62
Laptop	175	Toshiba	54
Netbook	183	Samsung	43
Laptop	176	Razer	19

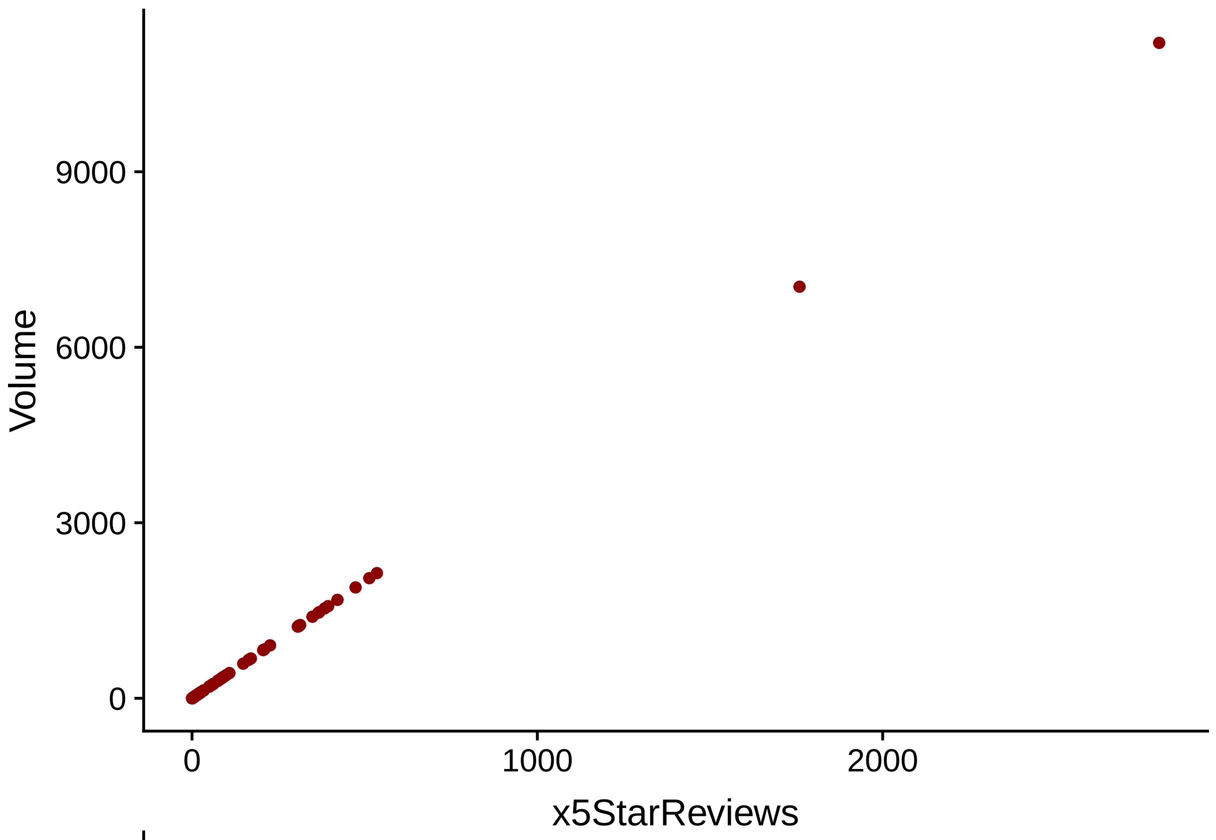
Discussion

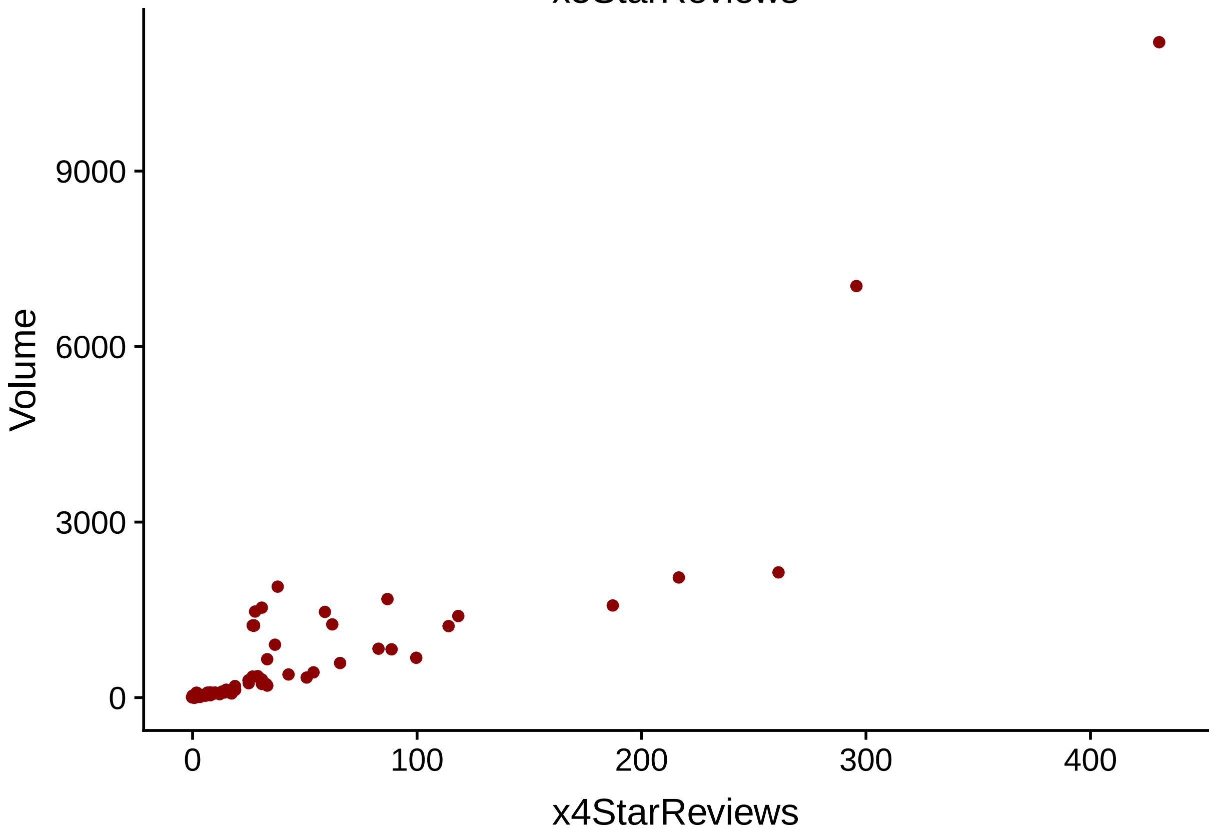
As the data and available algorithms were the same as last time and so were the algorithms available to us, there was no material difference nor material knowledge gains achieved by this exercise. The central take-away is that, in relation to this dataset, product type carries little weight in predicting sales.

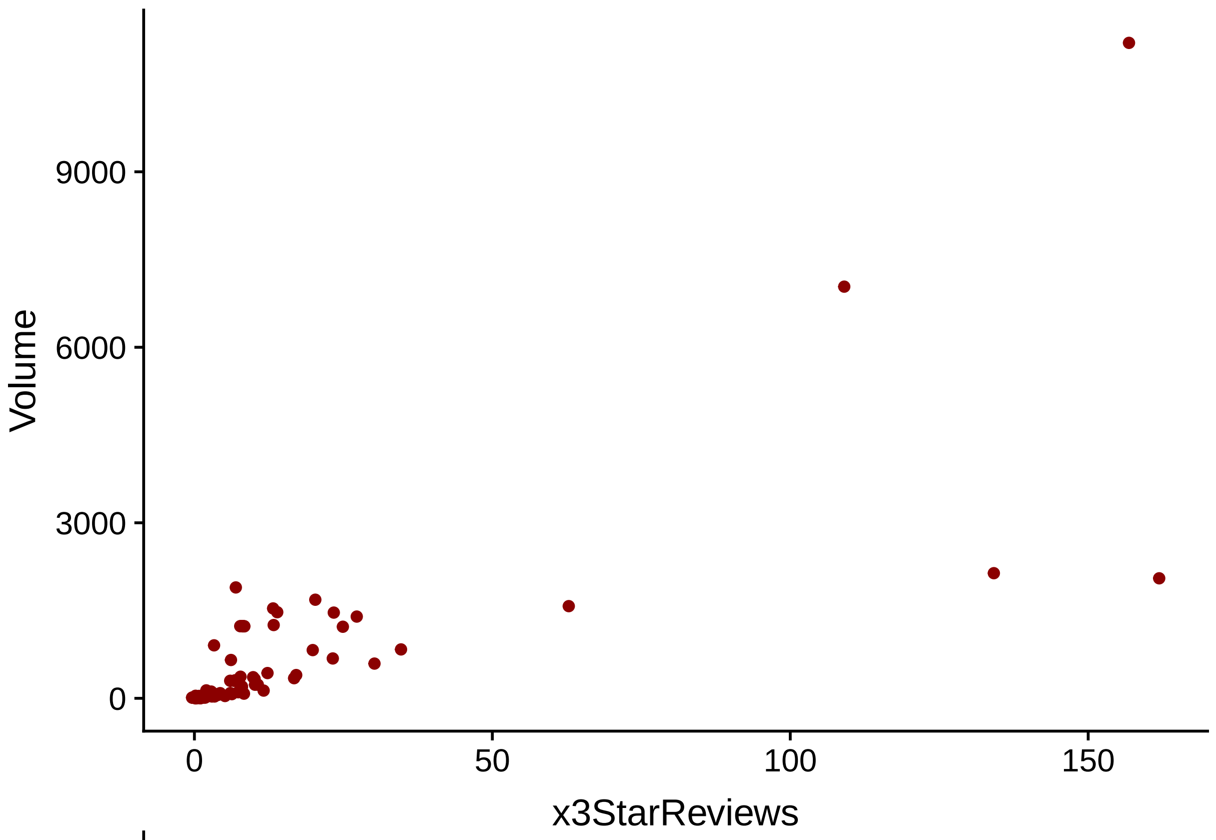
Visualization and data exploration:

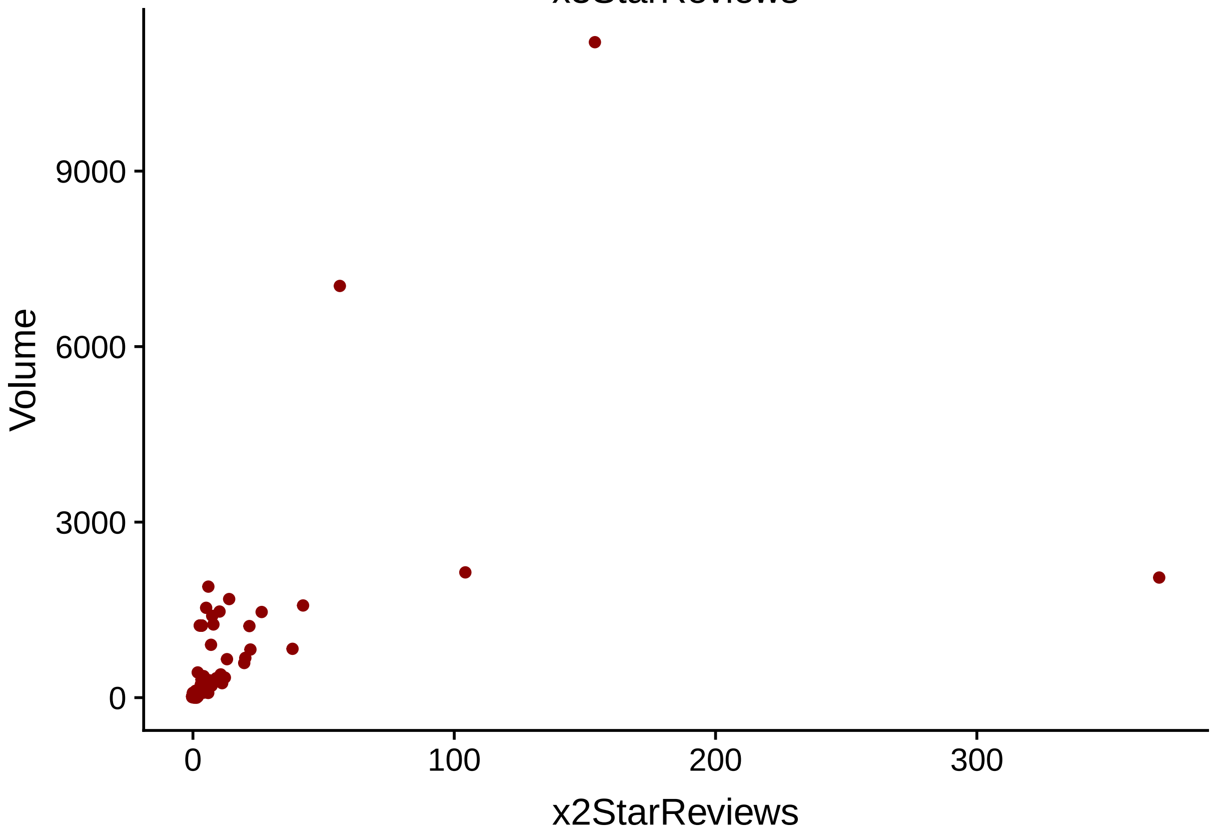
```r
plotprod <- products[, c(3:11, 13:18)]

for (i in names(plotprod[, -which(names(plotprod) == "Volume")])) {
 print(ggplot(data = plotprod,
              aes_string(x=i, y = plotprod$Volume))
       + geom_jitter(color = "darkred")
       + ylab("Volume")
 )
}
```

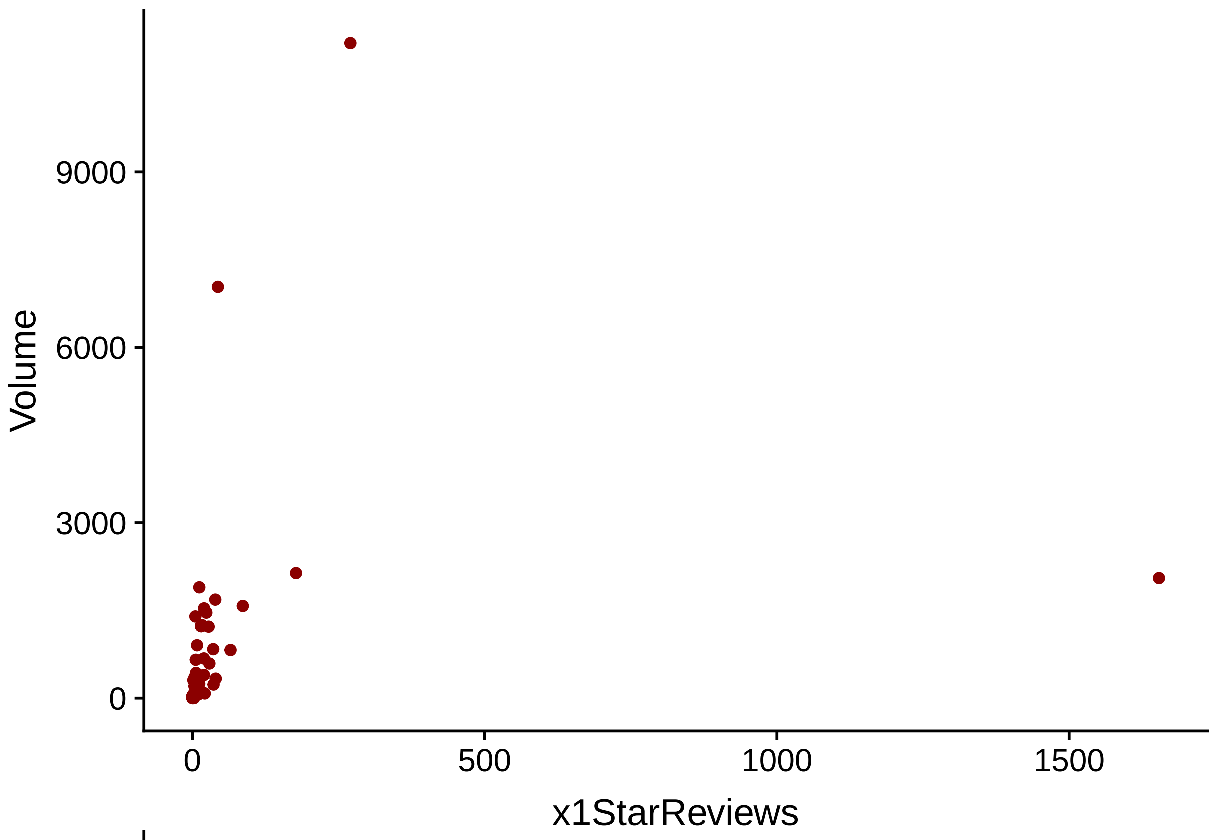


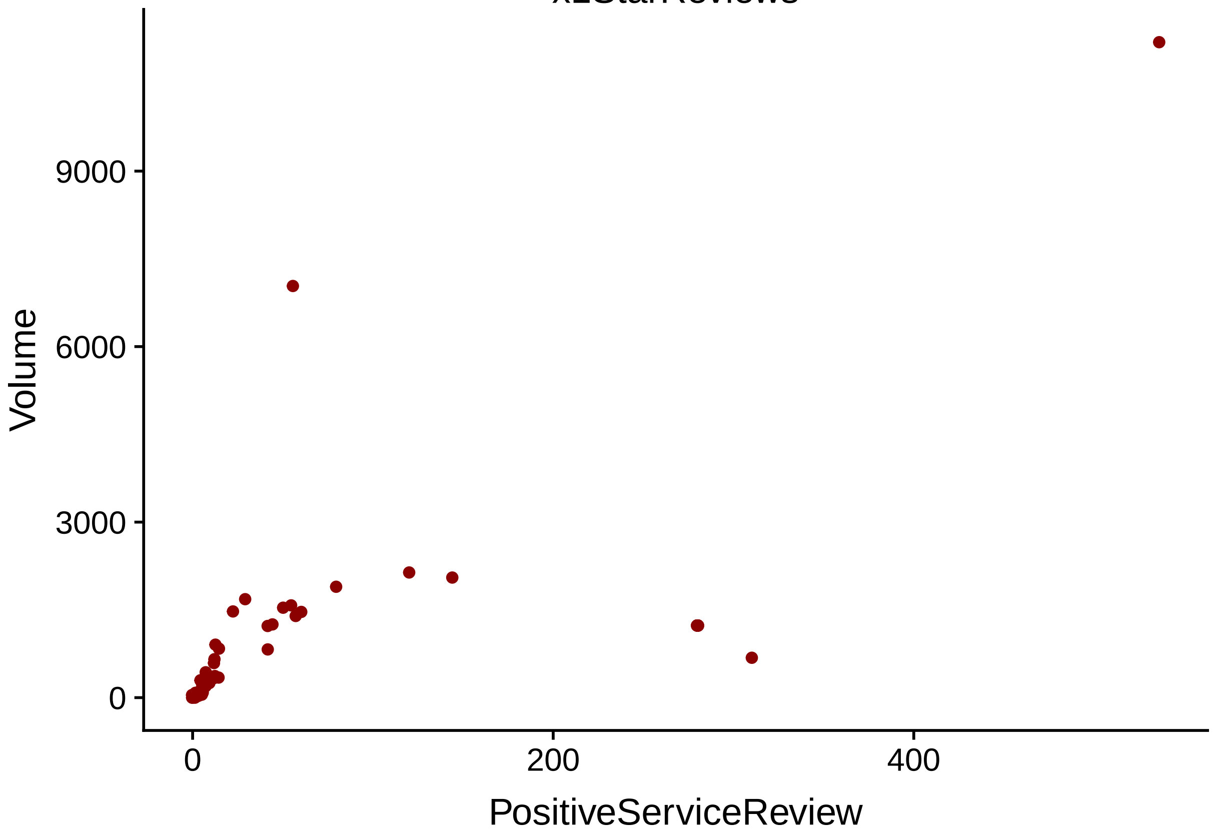

Volume
9000
6000
3000
0
0
1000
2000
x5StarReviews

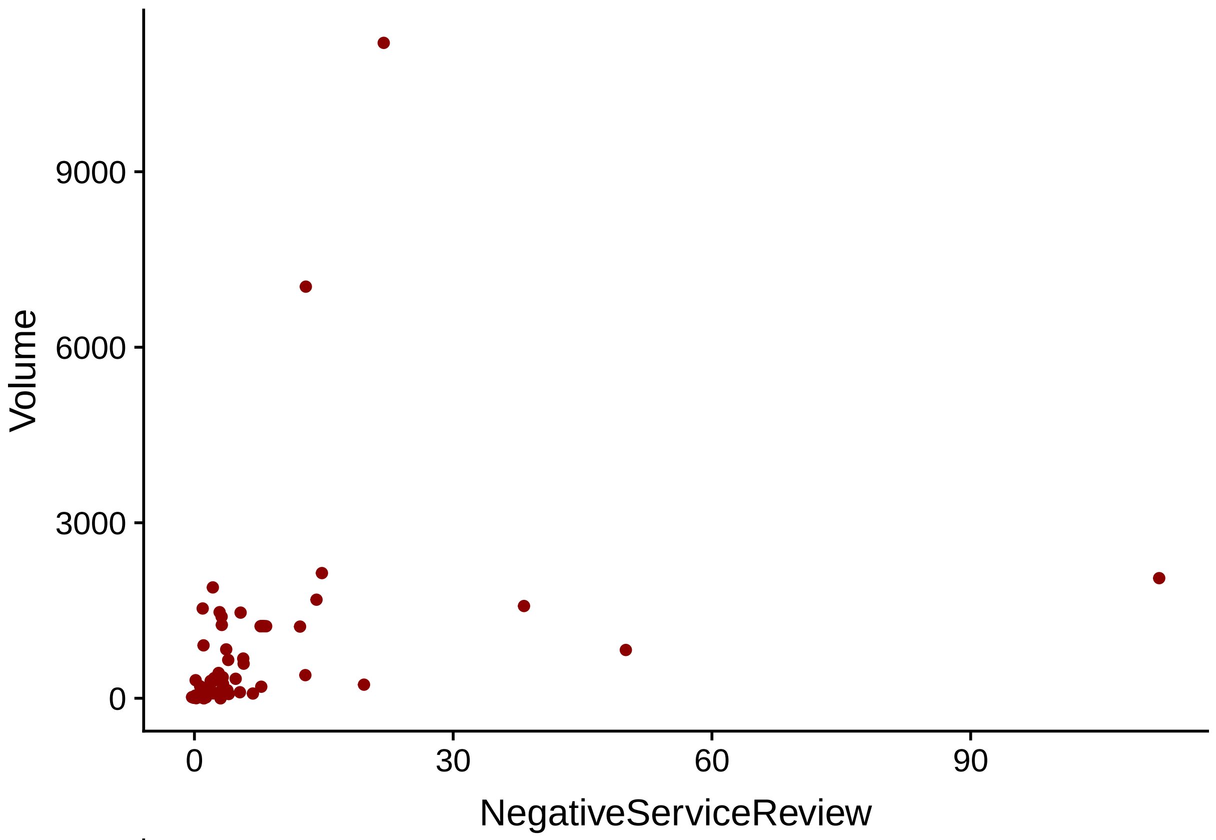

Volume
9000
6000
3000
0
0
100
200
300
400
x4StarReviews

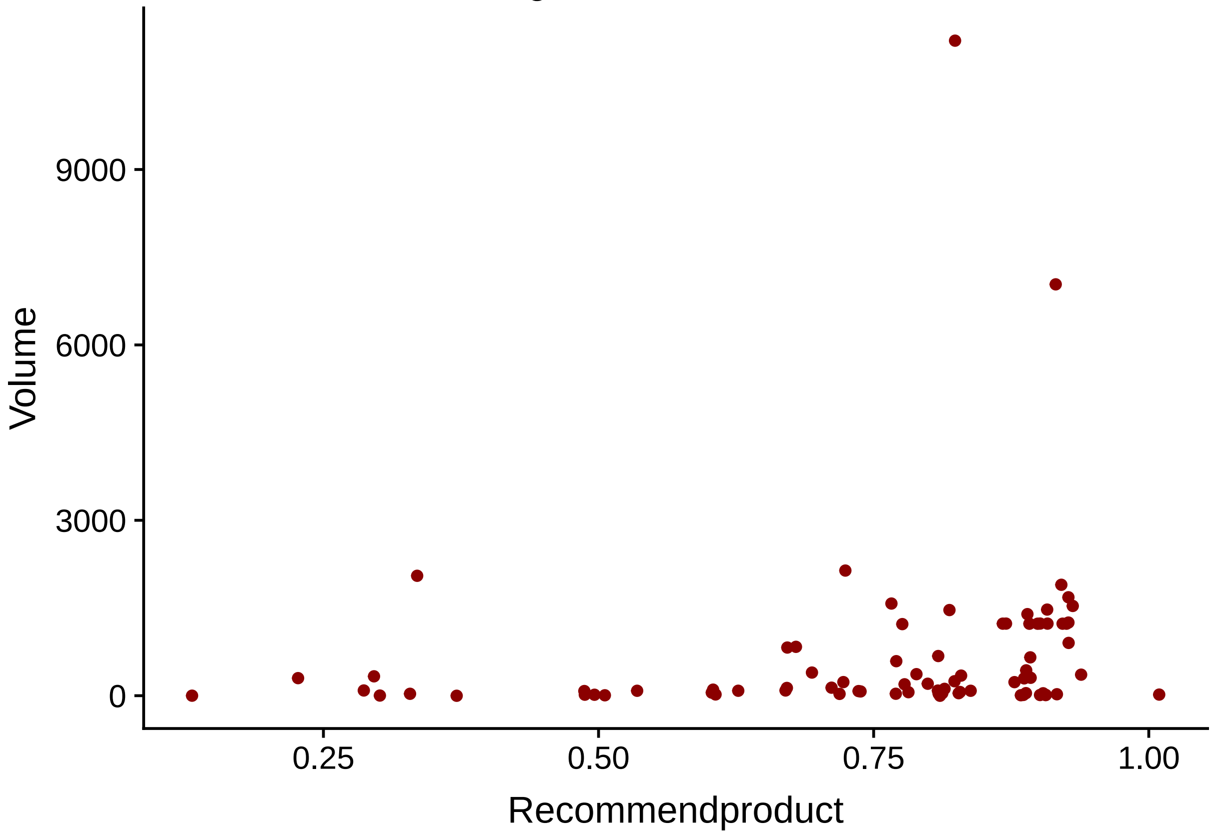

Volume
9000
6000
3000
0
0
50
100
150
x3StarReviews

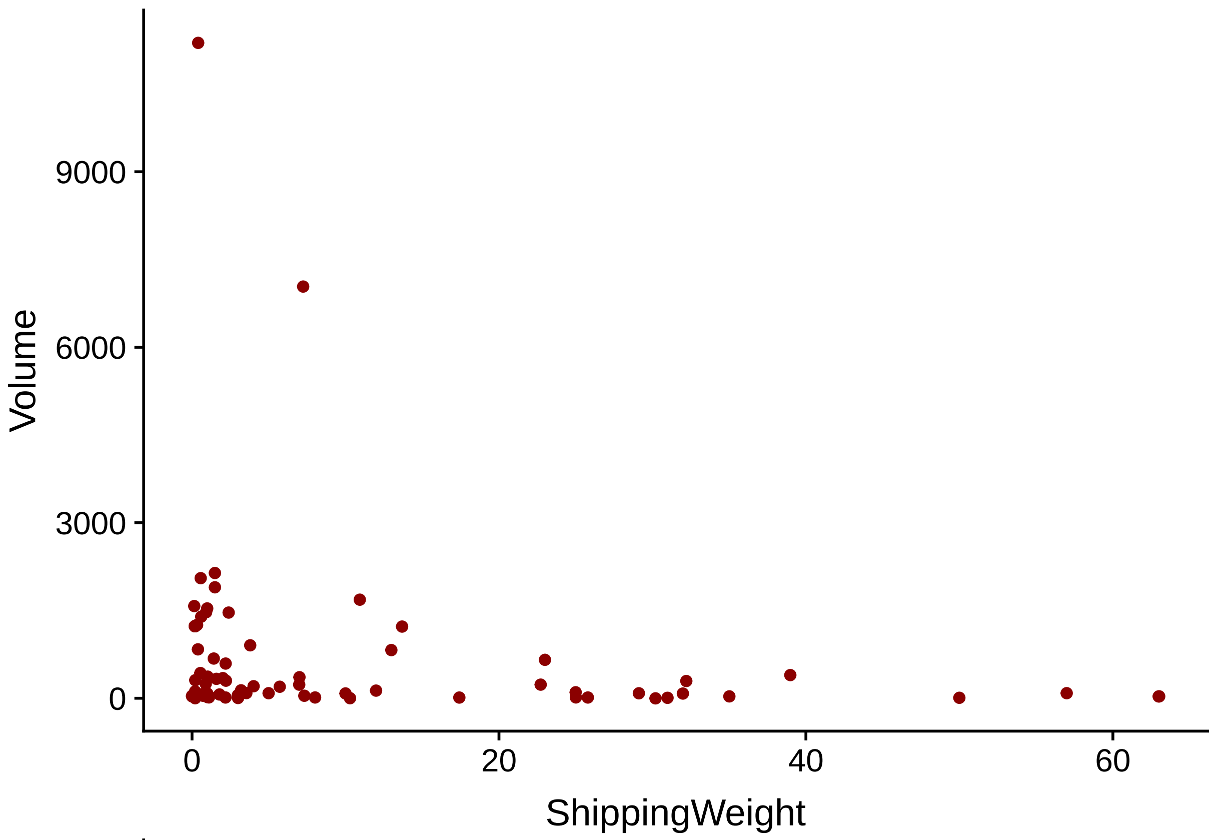

Volume
9000
6000
3000
0
0
100
200
300
x2StarReviews

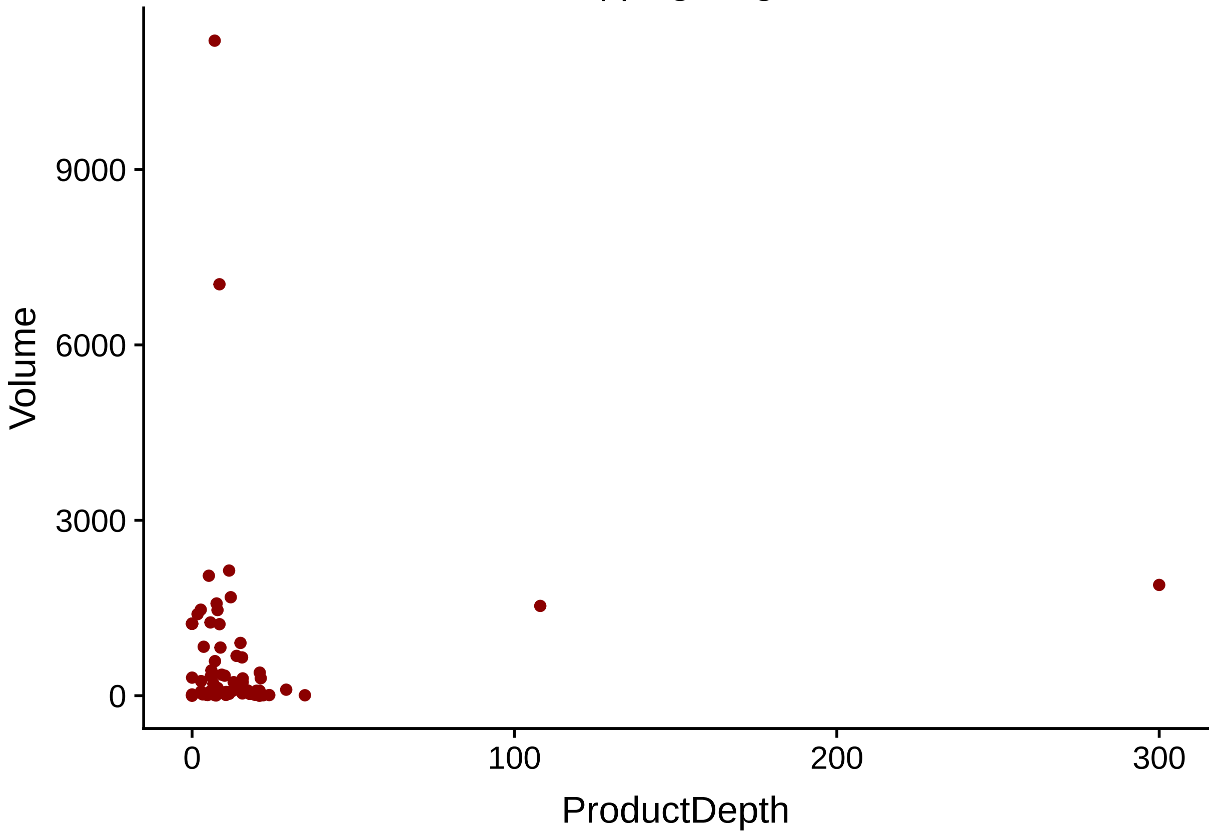

Volume
9000
6000
3000
0
0
500
1000
1500
x1StarReviews

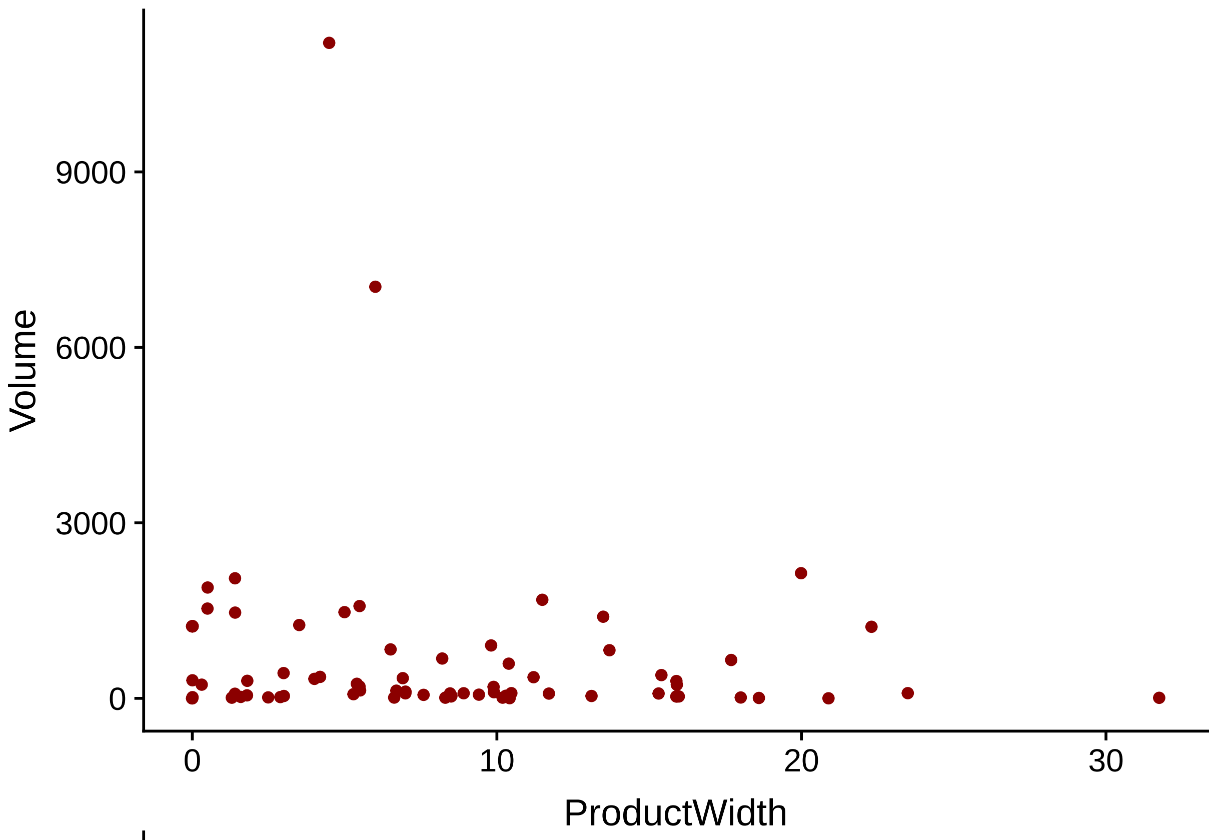

Volume
9000
6000
3000
0
0
200
400
PositiveServiceReview

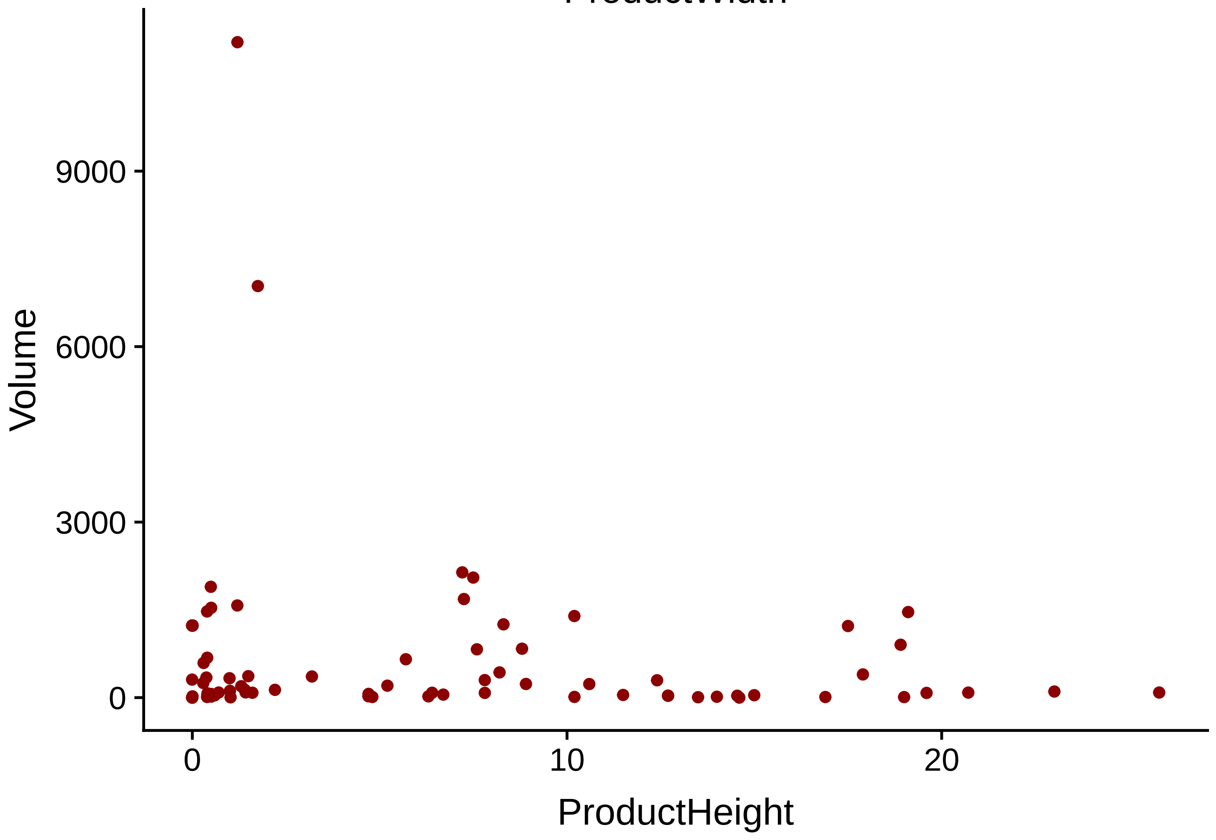

Volume
9000
6000
3000
0
0
30
60
90
NegativeServiceReview

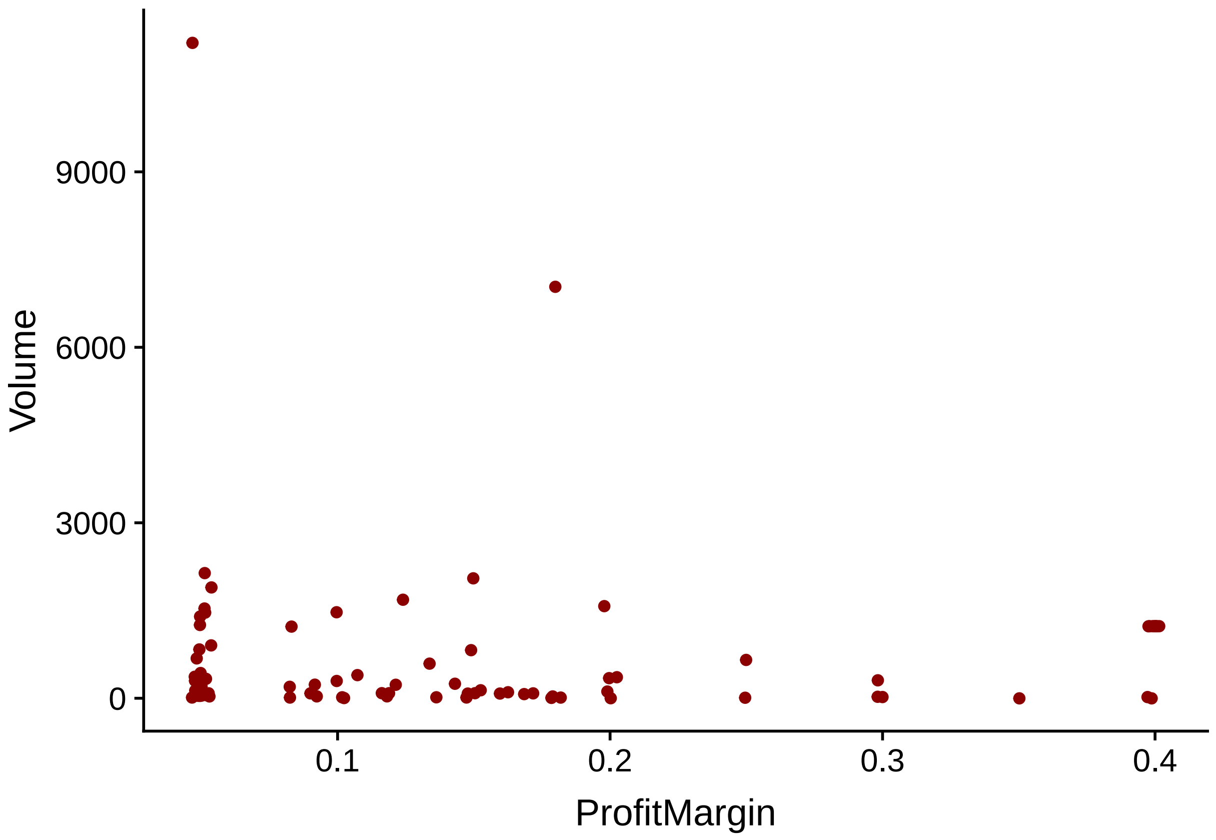

Volume
9000
6000
3000
0
0.25
0.50
0.75
1.00
Recommendproduct


Volume
9000
6000
3000
0
0
20
40
60
ShippingWeight


Volume
9000
6000
3000
0
0
100
200
300
ProductDepth


Volume
9000
6000
3000
0
0
10
20
30
ProductWidth


Volume
9000
6000
3000
0
0
10
20
ProductHeight



We can infer the following already from the visual exploration of the dataset:

i) x5StarReviews has a suspiciously strong positive correlation with Volume;
ii) x4StarReviews has a very strong positive correlation with Volume;
iii) all StarReviews are positively correlated with Volume, even 1 and 2 starreviews, which are generally considered as expressions of negative customer sentiment. This observation suggests that it is the number of reviews rather than the quality of the same that is related to Volume.
iv) Attributes relating to physical or financial aspects of the products are largely irrelevant for the purposes of inferential statistics.

Data preprocessing

```
#Check duplicated rows
sum(duplicated(products[,-which(names(products) %in% c("ProductNum","Price"))]))
```

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

```
#6 rows from the extended warranty are duplicated, so we'll remove them (but if we search manually we c
products <- products[-c(35:41),]
```

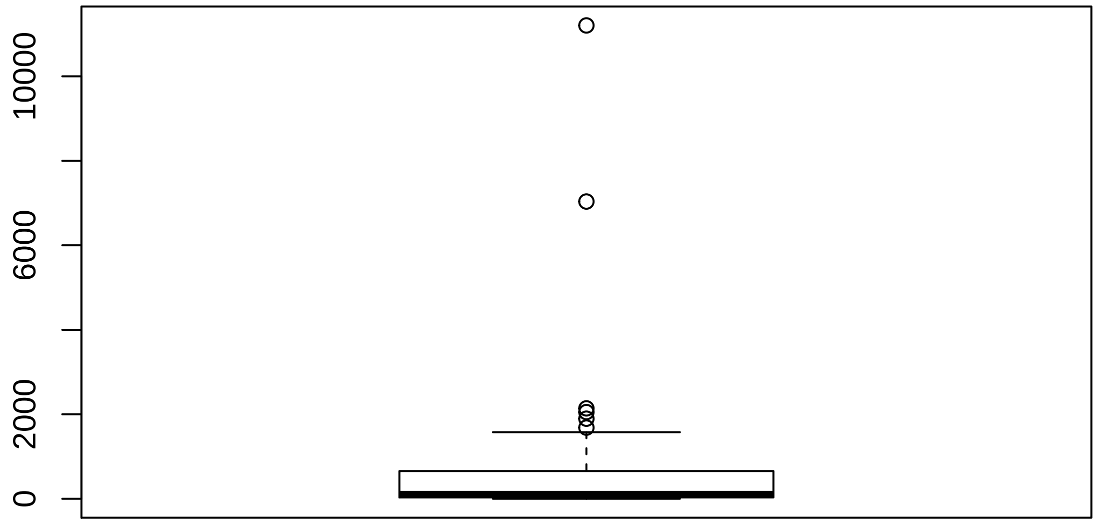
```
#Check NA
any(is.na(products))
```

```
## [1] TRUE
```

```r
for (i in c(1:ncol(products))){
  print(paste(i,any(is.na(products[,i]))))
}
```

```
## [1] "1 FALSE"
## [1] "2 FALSE"
## [1] "3 FALSE"
## [1] "4 FALSE"
## [1] "5 FALSE"
## [1] "6 FALSE"
## [1] "7 FALSE"
## [1] "8 FALSE"
## [1] "9 FALSE"
## [1] "10 FALSE"
## [1] "11 FALSE"
## [1] "12 TRUE"
## [1] "13 FALSE"
## [1] "14 FALSE"
## [1] "15 FALSE"
## [1] "16 FALSE"
## [1] "17 FALSE"
## [1] "18 FALSE"
```

```r
#There are missing values in the 12 column, which is "BestSellersRank".
#There are 15 missing values in Best Sellers Rank attribute, so we'll remove it.
products <- products[,-which(names(products) %in% "BestSellersRank")]
#Check outliers
boxplot(products$Volume)
```



```r
#Cleaning outliers
products <- filter(products,
                   products$Volume < 7000)
```

We an observe that there are missing values in the "BestSellerRank" column. As we cannot predict or obtain the value of this column based on the other features. We are also removing the two biggest outliers that make the biggest impact, corresponding to the volumes of 11204 and 7036.

```
anova_test <- aov(Volume ~ ProductType, data = products)
summary(anova_test)
```

```
##             Df   Sum Sq Mean Sq F value Pr(>F)
## ProductType 11  5075621  461420   1.473  0.166
## Residuals   59 18487579  313349
```

The PValue is too big, so our categorical variables have no relation between the dependent variable (Volume). This means we are not considering the ProductType for our model but we will store it as a vector for later uses (plots).

Feature Selection

```
ProductType <- as.vector(products$ProductType)
products <- products[,-which(colnames(products) %in% "ProductType")]
#Correlation Matrix
corr_products <- cor(products)
#Colinearity:
colinear <- findCorrelation(x = corr_products, cutoff = 0.80, names = T)
colinear
```

```
## [1] "x5StarReviews"         "Volume"                "x3StarReviews"
## [4] "x2StarReviews"         "NegativeServiceReview"
```

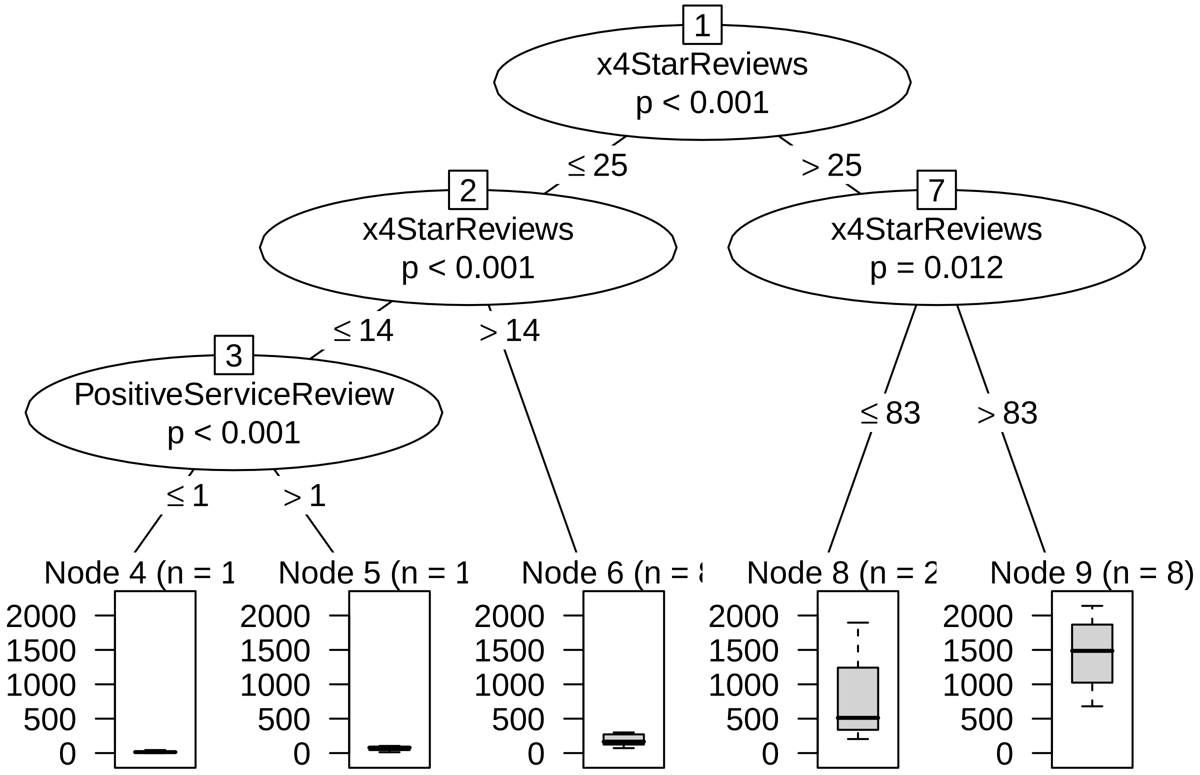
```
pairwiseCor <- function(dataframe){
  pairs <- combn(names(dataframe), 2, simplify=FALSE)
  df <- data.frame(Variable1=rep(0,length(pairs)), Variable2=rep(0,length(pairs)),
                   AbsCor=rep(0,length(pairs)), Cor=rep(0,length(pairs)))
  for(i in 1:length(pairs)){
    df[i,1] <- pairs[[i]][1]
    df[i,2] <- pairs[[i]][2]
    df[i,3] <- round(abs(cor(dataframe[,pairs[[i]][1]], dataframe[,pairs[[i]][2]])),4)
    df[i,4] <- round(cor(dataframe[,pairs[[i]][1]], dataframe[,pairs[[i]][2]]),4)
  }
  pairwiseCorDF <- df
  pairwiseCorDF <- pairwiseCorDF[order(pairwiseCorDF$AbsCor, decreasing=TRUE),]
  row.names(pairwiseCorDF) <- 1:length(pairs)
  pairwiseCorDF <<- pairwiseCorDF
  pairwiseCorDF
}
#x5StarReviews has perfect correlation, and we'll remove it. There's also colinearity between x4StarRev
pairw <- (pairwiseCor(products))
pairw[which(pairw$Variable2 == "Volume"),]
```

	Variable1	Variable2	AbsCor	Cor
1	x5StarReviews	Volume	1.0000	1.0000
8	x4StarReviews	Volume	0.8041	0.8041
12	x3StarReviews	Volume	0.6864	0.6864
18	PositiveServiceReview	Volume	0.5658	0.5658

	Variable1	Variable2	AbsCor	Cor
22	x2StarReviews	Volume	0.5159	0.5159
25	NegativeServiceReview	Volume	0.4997	0.4997
28	x1StarReviews	Volume	0.4124	0.4124
34	ProductDepth	Volume	0.3203	0.3203
38	ShippingWeight	Volume	0.2690	-0.2690
45	Recommendproduct	Volume	0.1834	0.1834
49	Price	Volume	0.1742	-0.1742
51	ProfitMargin	Volume	0.1681	-0.1681
69	ProductNum	Volume	0.0989	0.0989
78	ProductWidth	Volume	0.0900	-0.0900
97	ProductHeight	Volume	0.0421	-0.0421

Here we can observe that the reviews are highly correlated with each other. To sum up, we are removing the x3StarReviews and the x1StarReviews, as they are highly correlated to x4StarReviews and x2StarReviews respectively. Furthermore, we will plot a decision tree to see the variables that have the biggest impact.

```
decisiontree <- ctree(Volume~.,data =
                        products[,-which(colnames(products) %in% c("x5StarReviews","x3StarReviews", "x1S
                      controls = ctree_control(maxdepth = 3))
plot(decisiontree)
```



We can observe that the variables that have the biggest impact are x4StarReviews and PositiveServiceReview, which are also the ones that have the highest correlation.

Here we create a new variable with the StarReviews. We will do this by using a linear regression and we'll create an "average weighted star review" based on the coefficients of the regression model and its respectively variable.

```r
lm_model <- train(Volume~x4StarReviews + x3StarReviews + x2StarReviews + x1StarReviews,
                  products, method = "lm")
summary(lm_model)$coefficients
```

```
##                  Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)    92.8996017  50.904372  1.82498276 7.252930e-02
## x4StarReviews  14.2183632   2.572673  5.52668819 5.977552e-07
## x3StarReviews -14.7529735   9.979575 -1.47831683 1.440798e-01
## x2StarReviews   3.2275066  12.659522  0.25494696 7.995568e-01
## x1StarReviews   0.0484845   2.178431  0.02225662 9.823104e-01
```

```r
products$"Avg_WghtStar" <-  summary(lm_model)$coefficients[2]*products$x4StarReviews +
  summary(lm_model)$coefficients[3]*products$x3StarReviews +
  summary(lm_model)$coefficients[4]*products$x2StarReviews +
  summary(lm_model)$coefficients[5]*products$x1StarReviews
```

```r
pairw_avg <- (pairwiseCor(products))
pairw_avg[which(pairw_avg$Variable1 == "Volume"),]
```

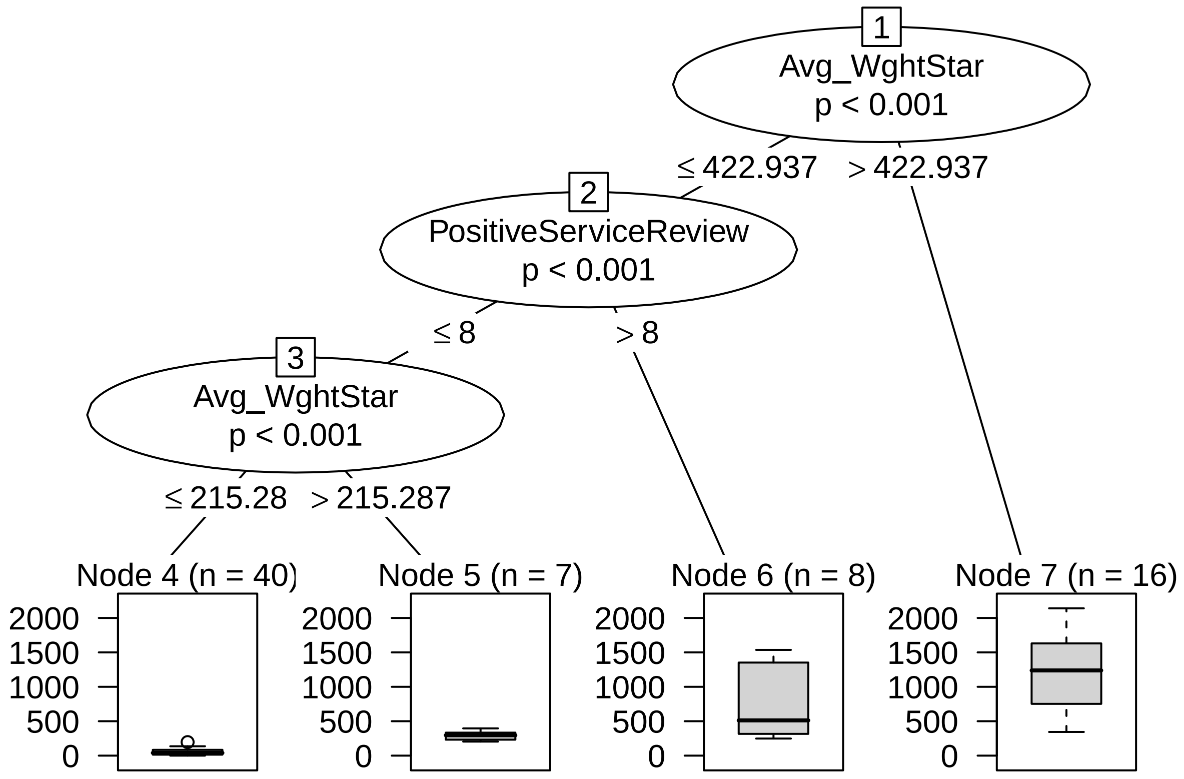
	Variable1	Variable2	AbsCor	Cor
10	Volume	Avg_WghtStar	0.8163	0.8163

```r
pairw_avg[which(pairw_avg$Variable1 == "x4StarReviews" &
                        pairw_avg$Variable2 == "Volume"),]
```

	Variable1	Variable2	AbsCor	Cor
12	x4StarReviews	Volume	0.8041	0.8041

Here we can observe that the new variable has better correlation than x4StarReviews, which was the one with the highest relationship with volume.

```r
#Decision Tree
avg_decisiontree <- ctree(Volume~.,data =
                        products[,-which(colnames(products) %in% c("x5StarReviews",
                                                                    "x4StarReviews",
                                                                    "x3StarReviews",
                                                                    "x2StarReviews",
                                                                    "x1StarReviews"))],
                      controls = ctree_control(maxdepth = 3))
plot(avg_decisiontree)
```



Modeling

We create a loop to train several features with several models.

```r
 #Cross validation:
set.seed(69)
indexing <- createDataPartition(products$Volume, p = 0.75, list = F)
trainSet <- products[indexing,]
testSet <- products[-indexing,]

form <- c("Volume ~ x4StarReviews + PositiveServiceReview",
             "Volume ~ Avg_WghtStar + PositiveServiceReview")
models <- c("rf","knn", "svmLinear", "svmRadial","glm")
combined <- c()
cnames <- vector()
for (i in form){
  for (j in models) {
    model <- train(formula(i), data = trainSet, method = j, tuneLength = 3, metric = "MAE")
    predictions <- predict(model, testSet)
    results <- postResample(predictions, testSet$Volume)
    combined <- cbind(results, combined)
    cnames <- c(paste(i,j),cnames)
  }
}
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
##
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```r
colnames(combined) <-cnames
min(combined[3,] )
```

```
## [1] 123.048
```

```r
combined
```

```
##           Volume ~ Avg_WghtStar + PositiveServiceReview glm
## RMSE                                            359.7378551
## Rsquared                                          0.7569293
## MAE                                             197.1320486
##           Volume ~ Avg_WghtStar + PositiveServiceReview svmRadial
## RMSE                                                  417.5630488
## Rsquared                                                0.7671428
## MAE                                                   248.5718783
##           Volume ~ Avg_WghtStar + PositiveServiceReview svmLinear
## RMSE                                                  349.2470074
## Rsquared                                                0.7995331
## MAE                                                   170.4095309
##           Volume ~ Avg_WghtStar + PositiveServiceReview knn
## RMSE                                            398.5383294
## Rsquared                                          0.7991097
## MAE                                             227.7000000
##           Volume ~ Avg_WghtStar + PositiveServiceReview rf
## RMSE                                           218.6926097
## Rsquared                                         0.9215047
## MAE                                            123.0479857
##           Volume ~ x4StarReviews + PositiveServiceReview glm
## RMSE                                             395.2775566
## Rsquared                                           0.6851846
## MAE                                              233.7248491
##           Volume ~ x4StarReviews + PositiveServiceReview svmRadial
## RMSE                                                   432.1760144
## Rsquared                                                 0.7425961
## MAE                                                    259.8334941
##           Volume ~ x4StarReviews + PositiveServiceReview svmLinear
## RMSE                                                   377.3323953
## Rsquared                                                 0.7309845
## MAE                                                    187.7202784
##           Volume ~ x4StarReviews + PositiveServiceReview knn
## RMSE                                              249.506351
## Rsquared                                            0.939719
## MAE                                               134.294048
##           Volume ~ x4StarReviews + PositiveServiceReview rf
## RMSE                                            219.9479104
## Rsquared                                          0.9186288
## MAE                                             125.8509355
```

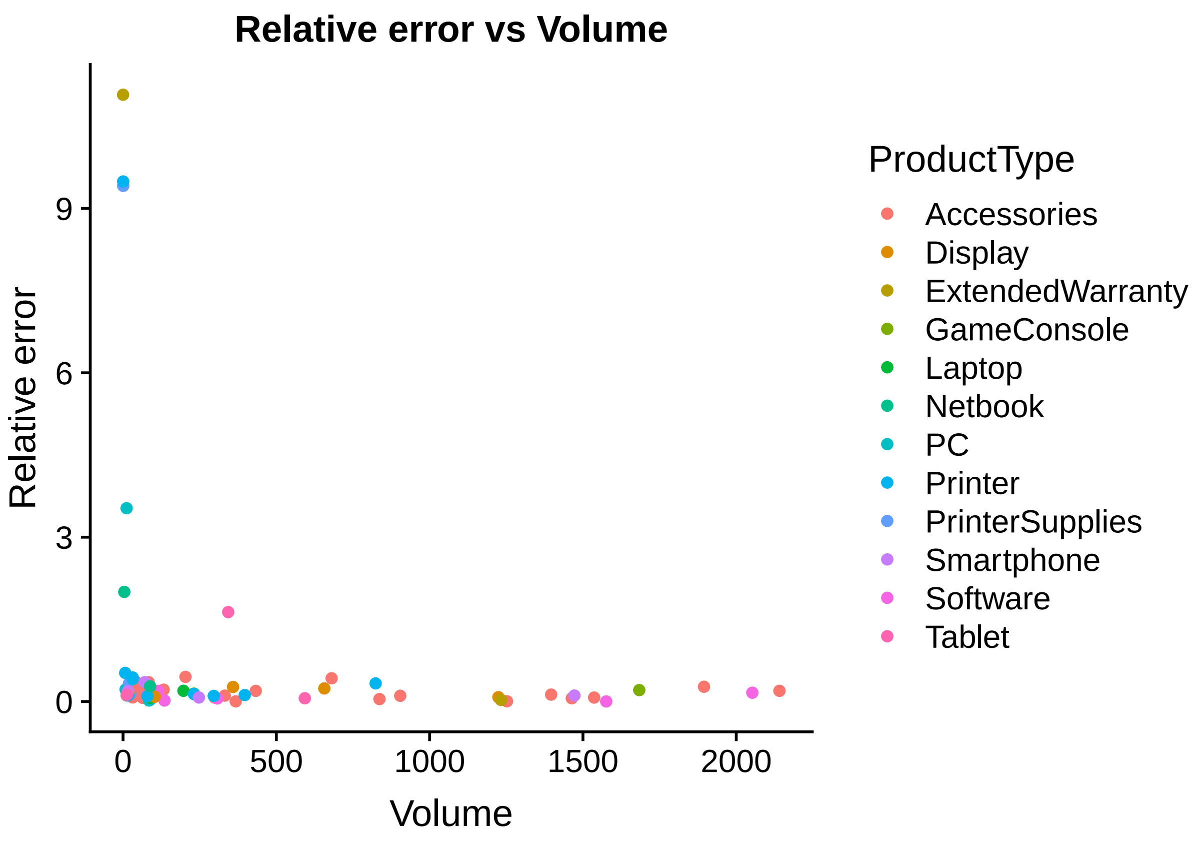
```r
#Best model for MAE is rf with variables= Avg_WghtStar and PositiveServiceReview
```

```
#We create the new avg_wght star variable for our predictions
newproducts$"Avg_WghtStar" <-  summary(lm_model)$coefficients[2]*newproducts$x4StarReviews +
  summary(lm_model)$coefficients[3]*newproducts$x3StarReviews +
  summary(lm_model)$coefficients[4]*newproducts$x2StarReviews +
  summary(lm_model)$coefficients[5]*newproducts$x1StarReviews

rf_model <- train(Volume ~ Avg_WghtStar + PositiveServiceReview,
                  trainSet, tuneLength = 3, metric = "MAE")
```

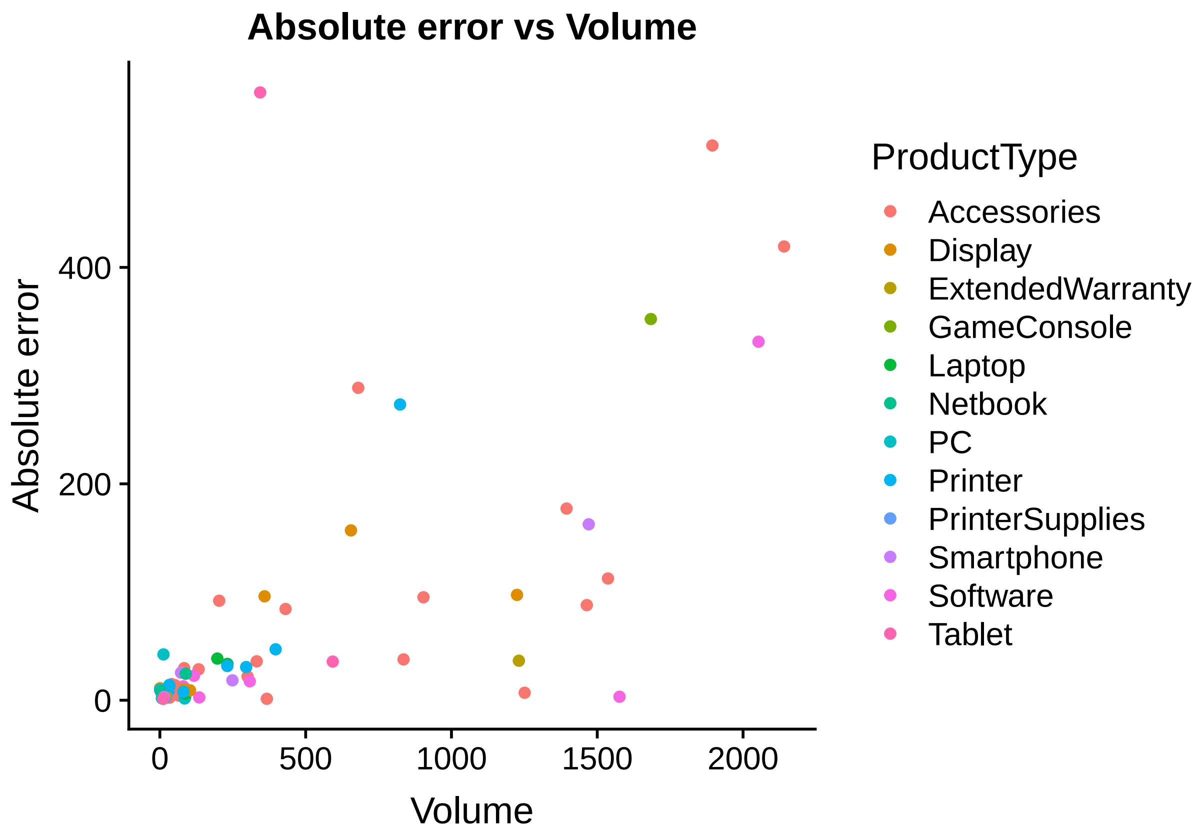
```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
newproducts$Volume <- predict(rf_model,newproducts)
```

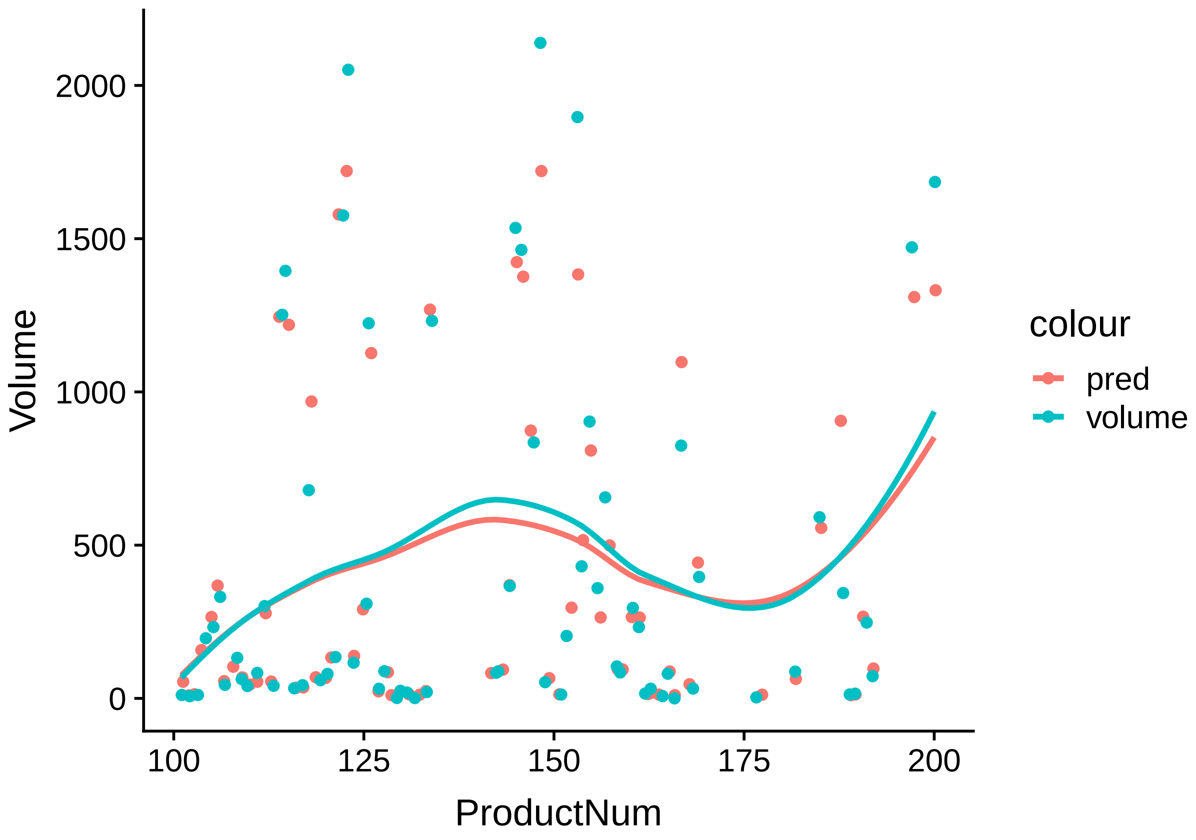
```
products$Volume[products$Volume == 0] <- 1
Volume <- as.numeric(products$Volume)
ex_preds <- as.numeric(predict(rf_model,products))
ae_errors <- as.numeric(abs(ex_preds - products$Volume))
re_errors <- as.numeric(ae_errors/products$Volume)
errors_df <- as.data.frame(cbind(Volume,ex_preds,ae_errors,re_errors))
errors_df$ProductType <- ProductType
errors_df$ProductNum <- products$ProductNum

ggplot(errors_df, aes(x = Volume, y = re_errors, color = ProductType)) + geom_jitter() + ylab("Relative
```



```
ggplot(errors_df, aes(x = Volume, y = ae_errors, color = ProductType)) + geom_jitter() + ylab("Absolute
```



```
ccols <- c("volume"="#f04546","pred"="#3591d1")
ggplot(errors_df, aes()) +geom_jitter(aes(x=ProductNum, y = ex_preds, color = "pred")) + geom_jitter(ae
  geom_smooth(aes(x=ProductNum, y = ex_preds, color = "pred"),method = "loess",
              se = F) +
  geom_smooth(aes(x=ProductNum, y = Volume, color = "volume"),method = "loess",
              se = F) + ylab("Volume")
```



As we can see from the plots, the relative error is at its largest at low volumes, which is nearly always the case as relative error is the absolute error as a fraction of the observation. The absolute error is greater at greater volumes.

```
filtered <- newproducts[which(newproducts$ProductType == "PC"|newproducts$ProductType == "Laptop"|
                    newproducts$ProductType == "Netbook"|
                    newproducts$ProductType == "Smartphone"), ]
finalresults <- filtered[,which(colnames(filtered) %in% c("ProductType","ProductNum","Volume"))]
finalresults$Volume <- round(finalresults$Volume,0)
finalresults$Volume <- as.integer(finalresults$Volume)
finalresults
```

ProductType	ProductNum	Volume
PC	171	217
PC	172	104
Laptop	173	177
Laptop	175	54
Laptop	176	19
Netbook	178	62
Netbook	180	1191
Netbook	181	159
Netbook	183	43
Smartphone	193	261
Smartphone	194	899
Smartphone	195	84
Smartphone	196	95