Multiple Regression

José Pedro Conceição,Kiko Sánchez , Eloi Cirera

March 25, 2019

Table of Contents

Executive Summary

The first Objective was accurately Predicting Sales Volume, this has not been fully achieved because of the small data set provided to the team, we did train some models and made some predictions,however, they accurate and the errors are big, but it was the best information value we could extract from this sample. Predicting the sales of new Products using a reduced sample won't be accurate, besides being statistically unsound. We found out that what actually predicts success in volume are both, however the best predictor for successe comes from service reviews with an importance of 100% according to a random forest algorithm, followed by a a 50% importance of 4 star reviews.
Why 4 star reviews and not other reviews ? Well because they had to be taken out of our model training, they had levels of relationship with our predictor (Volume) so high that they were biasing the whole model, making the predictions even more unreliable.For example the 5 star review had a perfect correlation with the Volume, this means that the volume would grow at the same rate as a the 5 star reviews increased, which does not translate into reality. Nonetheless here are our final predictions.

Number
Game Console
Game Console
Tablet
Tablet
NoteBook

Technical Report

Pre-process

We always start by assesing the importance of each variable, we achieve this by doing a correlation matrix and training a simple model, followed by a varImp(), which will give us in percentage the importance of the variable for the model’s prediction.(We need to first use a correlation matrix to see the correlation values, and take out anything that migh bias our model, otherwise the “biased”, features will just appear at the top of the varImp() output).

We created a function to dummyfy the variables and to check if there was any NA values (in any attribute), and if they exist, remove them, I also included a function that removes outliers, and a function to subset the data into different product types.

Process functions

Pre-process

```
PPfunction <- function(data) {

  N <- dummyVars(" ~ .", data = data)

  N <- data.frame(predict(N, newdata = data))

  N <- N[,colSums(is.na(N)) == 0]

  N
}
```

Remove Outliers

```
RmOut <- function(D,V)

  {

  Out <- boxplot(D$V ,plot = FALSE)$out
  K <- D[-which(D$V %in% Out),]
```

```
  K

}
```

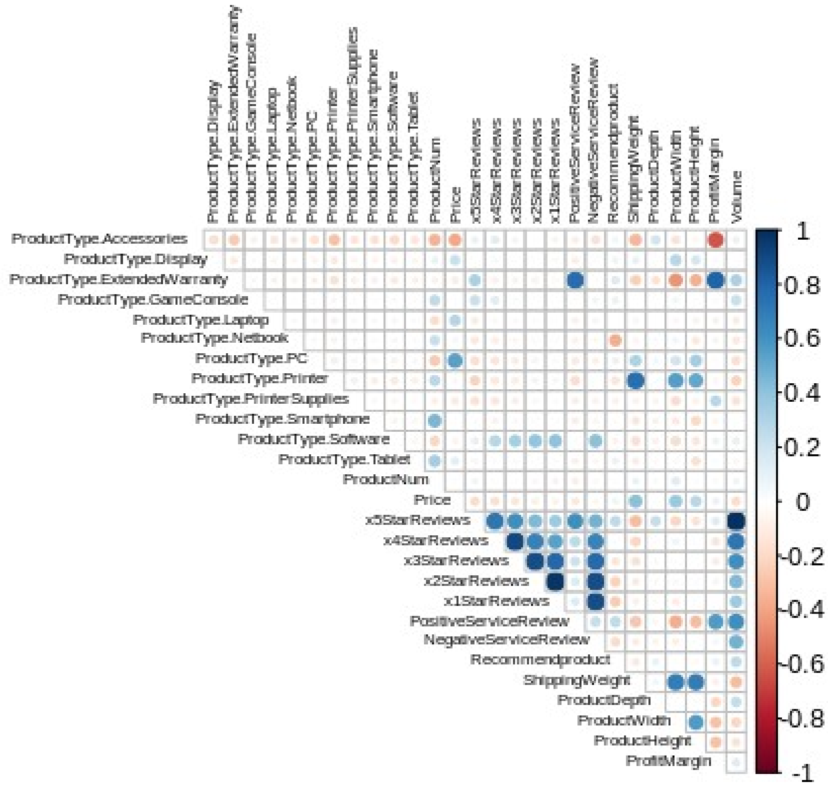
Sub-set by product types

```
SubSetDataProductTypes <- function(data,p,p1 = 0,p2 = 0 , p3 =
0 , p4 = 0)

  {
  if ( p1 == 0 && p2 == 0 && p3 == 0 && p4 == 0)
  {
    Nsub <- subset(data, data$ProductType == p)

    return(Nsub)
  }
  else if (p2 == 0 && p3 == 0 && p4 == 0){
    Nsub <- subset(data, data$ProductType == p)

    Nsub2 <-subset(data, data$ProductType == p1)

    Nsub2 <- rbind(Nsub,Nsub2)

    return(Nsub2)
  }
  else  if (p3 == 0 && p4 == 0)
  {

    Nsub <- subset(data, data$ProductType == p)

    Nsub2 <-subset(data, data$ProductType == p1)

    Nsub3 <- subset(data,data$ProductType == p2)

    Nsub3 <- rbind(Nsub,Nsub2,Nsub3)

    return(Nsub3)

  }

  else  if (p4 == 0){
    Nsub <- subset(data, data$ProductType == p)

    Nsub2 <-subset(data, data$ProductType == p1)

    Nsub3 <- subset(data,data$ProductType == p2)
```

```
    Nsub4 <- subeset(data,data$ProductType == p3)

    Nsub4 <- rbind(Nsub,Nsub2,Nsub3,Nsub4)

    return(Nsub4)
  }

  else{

    Nsub <- subset(data, data$ProductType == p)

    Nsub2 <-subset(data, data$ProductType == p1)

    Nsub3 <- subset(data,data$ProductType == p2)

    Nsub4 <- subeset(data,data$ProductType == p3)

    Nsub5 <- subset(data,data$ProductType == p4)

    Nsub5 <- rbind(Nsub,Nsub2,Nsub3,Nsub4,Nsub5)

    return(Nsub5)
  }
}

#### I know it's not the most pretty or effective way to do this,
but it works.
```

Correlation Matrix :

```
EP <- PPfunction(EP)
EP <- RmOut(EP,Volume)

corr_all<-cor(EP)



corrplot::
corrplot(corr_all,type="upper",tl.pos="td",method="circle",tl.cex
= 0.5,tl.col='black',diag=FALSE)
```



This information does not differ from the module 1's counterpart, it's obvius because we are using the same data set.

We trained a random forest followed by the use of varImp() function that assess the importance of each variables (without the ones we took out by looking at the correlation matrix). But first we need to create Test and Training sets, we also came up with a simple function to automate the processs.

Train and Test Set function

```
TrainAndTestSets <- function(label,p,data,seed){
  set.seed(seed)

  inTrain <- createDataPartition(y= label, p = p , list = FALSE)
  training <- data[inTrain,]
  testing <- data[-inTrain,]


  list(trainingSet=training,testingSet = testing)

}
```

```r
EP <-
EP[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,20,21,22,23,24,25,26
,27,28)]

List <- TrainAndTestSets(EP$Volume,0.75,EP,123)
```

```r
 fitcontrol <-  trainControl(method = "repeatedcv", repeats = 4)

  Model <- train(Volume~., data = EP,method = "rf", trcontrol =
fitcontrol , tunelenght = 5
                 , preProcess = c("center",
"scale"),importance=T)

  varImp(Model)

## rf variable importance
##
##   only 20 most important variables shown (out of 24)
##
##                              Overall
## PositiveServiceReview        100.000
## x4StarReviews                 46.607
## ProductWidth                  11.786
## ShippingWeight                10.171
## Price                          9.865
## x2StarReviews                  9.387
## ProductType.ExtendedWarranty   8.821
## ProductType.Printer            7.122
## ProfitMargin                   6.530
## ProductDepth                   6.083
## ProductType.Tablet             6.030
## ProductType.Software           5.748
## ProductHeight                  4.535
## ProductType.GameConsole        3.915
## Recommendproduct               3.785
## ProductType.Accessories        3.465
## ProductType.Smartphone         3.014
## ProductType.PC                 2.530
## ProductType.Display            2.499
## NegativeServiceReview          2.344
```

So the only variables with a significant impact are only PostiveServiceReview and x4StarReviews.

Models and Predictions

Training Function

I created a function that trains every different model, by user specification

```
EP <- read.csv( file
="/home/zordo/Documents/Ubiqum/R-M2Task3/data/Epa.csv" , header =
TRUE , sep = ',')

EP <- EP[,c(1,5,9,18)]

EP <- PPfunction(EP)

EP <- RmOut(EP)

List <- TrainAndTestSets(EP$Volume,0.75,EP,123)


#### Random Forest ####
ModelRandomForest <-
TrainingFunction("rf",Volume~.,List$trainingSet,5)

PredictionRandomForest <-
predict(ModelRandomForest,List$testingSet)

TestResultsRF <-
postResample(PredictionRandomForest,List$testingSet$Volume)

#### SVM ####

svm.model <-
TrainingFunction("svm",Volume~.,List$trainingSet,5,10000000,0.000
0001)

svm.pred <- predict(svm.model,List$testingSet)

TestResultsSVM <- postResample(svm.pred,List$testingSet$Volume)


#### knn ####

 KNN <- TrainingFunction("knn",Volume~.,List$trainingSet,30)
```

```
KnnPrediction <- predict(KNN,List$testingSet)

TestResultsKNN <-
postResample(KnnPrediction,List$testingSet$Volume)

####


AllTestResults <-
cbind(TestResultsKNN,TestResultsRF,TestResultsSVM)

AllTestResults
```

```
##          TestResultsKNN TestResultsRF TestResultsSVM
## RMSE        429.1288137   271.9646084    788.7251446
## Rsquared      0.6828584     0.8447768      0.2698622
## MAE         250.9259259   156.6409058    553.4143148
```

And then did another one t train the three models at the same time with a for loop

```
TrainAll3Models <- function (formula,data)
  {

  Model <- vector(mode="list", length=length(methods))

      methods <- c("rf","svm","knn")

      for(i in 1:length(methods))
          {

            Model[[i]] <-
TrainingFunction(methods[i],formula,data,5)



      }
    Model
}
```

I didn’t use this function that much since the mentors showed us another way of training without any function, and it’s much easier and cleaner.

```
a <- c("Volume ~ x4StarReviews","Volume ~.","Volume ~
PositiveServiceReview")
```

```
b <- c("lm","rf", "knn","svmLinear")
compare_var_mod <- c()

for ( i   in a) {
  for (j in b) {

    model <- train(formula(i), data = List$trainingSet, method =
b,trainControl=trainControl(method = "repeatedcv", repeats = 4))

    pred <- predict(model, newdata = List$testingSet)

    pred_metric <- postResample(List$testingSet$Volume, pred)

    compare_var_mod <- cbind(compare_var_mod , pred_metric)

  }

}
      compare_var_mod

##          pred_metric pred_metric pred_metric pred_metric
pred_metric
## RMSE     459.4407974 459.4407974 459.4407974 459.4407974
482.8296811
## Rsquared   0.5286655   0.5286655   0.5286655   0.5286655
0.5398693
## MAE      338.0400480 338.0400480 338.0400480 338.0400480
254.6791507
##          pred_metric pred_metric pred_metric pred_metric
pred_metric
## RMSE     482.8296811 482.8296811 482.8296811 568.6757233
568.6757233
## Rsquared   0.5398693   0.5398693   0.5398693   0.2851607
0.2851607
## MAE      254.6791507 254.6791507 254.6791507 436.9034926
436.9034926
##          pred_metric pred_metric
## RMSE     568.6757233 568.6757233
## Rsquared   0.2851607   0.2851607
## MAE      436.9034926 436.9034926

names_var <- c()
for (i in a) {
  for(j in b) {
    names_var <- append(names_var,paste(i,j))
  }
}


names_var
```

```
##  [1] "Volume ~ x4StarReviews lm"
##  [2] "Volume ~ x4StarReviews rf"
##  [3] "Volume ~ x4StarReviews knn"
##  [4] "Volume ~ x4StarReviews svmLinear"
##  [5] "Volume ~. lm"
##  [6] "Volume ~. rf"
##  [7] "Volume ~. knn"
##  [8] "Volume ~. svmLinear"
##  [9] "Volume ~ PositiveServiceReview lm"
## [10] "Volume ~ PositiveServiceReview rf"
## [11] "Volume ~ PositiveServiceReview knn"
## [12] "Volume ~ PositiveServiceReview svmLinear"

colnames(compare_var_mod) <- names_var

compare_var_mod

##          Volume ~ x4StarReviews lm Volume ~ x4StarReviews rf
## RMSE                  459.4407974               459.4407974
## Rsquared                0.5286655                 0.5286655
## MAE                   338.0400480               338.0400480
##          Volume ~ x4StarReviews knn Volume ~ x4StarReviews
svmLinear
## RMSE                   459.4407974
459.4407974
## Rsquared                 0.5286655
0.5286655
## MAE                    338.0400480
338.0400480
##          Volume ~. lm Volume ~. rf Volume ~. knn Volume ~.
svmLinear
## RMSE      482.8296811  482.8296811   482.8296811
482.8296811
## Rsquared    0.5398693    0.5398693     0.5398693
0.5398693
## MAE       254.6791507  254.6791507   254.6791507
254.6791507
##          Volume ~ PositiveServiceReview lm
## RMSE                          568.6757233
## Rsquared                        0.2851607
## MAE                           436.9034926
##          Volume ~ PositiveServiceReview rf
## RMSE                          568.6757233
## Rsquared                        0.2851607
## MAE                           436.9034926
##          Volume ~ PositiveServiceReview knn
## RMSE                           568.6757233
## Rsquared                         0.2851607
## MAE                            436.9034926
##          Volume ~ PositiveServiceReview svmLinear
```

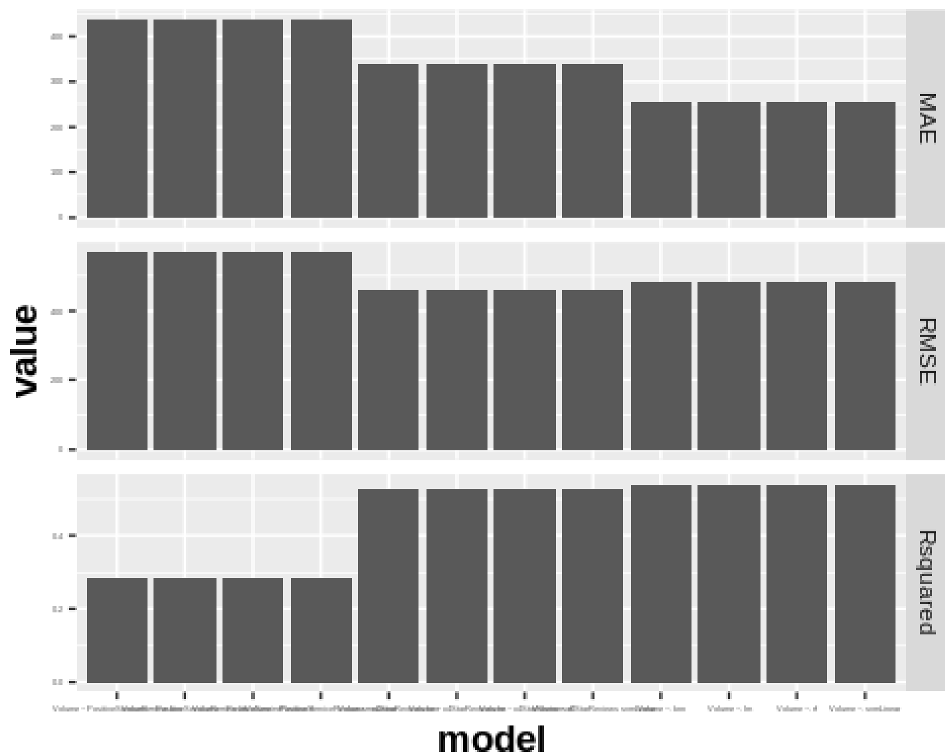
```
## RMSE                                  568.6757233
## Rsquared                                0.2851607
## MAE                                   436.9034926
```

```
compare_var_mod_melt <- melt(compare_var_mod, varnames =
c("metric", "model"))
compare_var_mod_melt <- as.data.frame(compare_var_mod_melt)
compare_var_mod_melt
```

```
##      metric                                    model
value
## 1      RMSE               Volume ~ x4StarReviews lm
459.4407974
## 2  Rsquared               Volume ~ x4StarReviews lm
0.5286655
## 3       MAE               Volume ~ x4StarReviews lm
338.0400480
## 4      RMSE               Volume ~ x4StarReviews rf
459.4407974
## 5  Rsquared               Volume ~ x4StarReviews rf
0.5286655
## 6       MAE               Volume ~ x4StarReviews rf
338.0400480
## 7      RMSE              Volume ~ x4StarReviews knn
459.4407974
## 8  Rsquared              Volume ~ x4StarReviews knn
0.5286655
## 9       MAE              Volume ~ x4StarReviews knn
338.0400480
## 10     RMSE        Volume ~ x4StarReviews svmLinear
459.4407974
## 11 Rsquared        Volume ~ x4StarReviews svmLinear
0.5286655
## 12      MAE        Volume ~ x4StarReviews svmLinear
338.0400480
## 13     RMSE                             Volume ~. lm
482.8296811
## 14 Rsquared                             Volume ~. lm
0.5398693
## 15      MAE                             Volume ~. lm
254.6791507
## 16     RMSE                             Volume ~. rf
482.8296811
## 17 Rsquared                             Volume ~. rf
0.5398693
## 18      MAE                             Volume ~. rf
254.6791507
## 19     RMSE                            Volume ~. knn
482.8296811
## 20 Rsquared                            Volume ~. knn
0.5398693
```

```
## 21      MAE                                 Volume ~. knn
254.6791507
## 22     RMSE                           Volume ~. svmLinear
482.8296811
## 23 Rsquared                           Volume ~. svmLinear
0.5398693
## 24      MAE                           Volume ~. svmLinear
254.6791507
## 25     RMSE        Volume ~ PositiveServiceReview lm
568.6757233
## 26 Rsquared        Volume ~ PositiveServiceReview lm
0.2851607
## 27      MAE        Volume ~ PositiveServiceReview lm
436.9034926
## 28     RMSE        Volume ~ PositiveServiceReview rf
568.6757233
## 29 Rsquared        Volume ~ PositiveServiceReview rf
0.2851607
## 30      MAE        Volume ~ PositiveServiceReview rf
436.9034926
## 31     RMSE       Volume ~ PositiveServiceReview knn
568.6757233
## 32 Rsquared       Volume ~ PositiveServiceReview knn
0.2851607
## 33      MAE       Volume ~ PositiveServiceReview knn
436.9034926
## 34     RMSE Volume ~ PositiveServiceReview svmLinear
568.6757233
## 35 Rsquared Volume ~ PositiveServiceReview svmLinear
0.2851607
## 36      MAE Volume ~ PositiveServiceReview svmLinear
436.9034926
```

```
ggplot(compare_var_mod_melt, aes(x=model,y=value)) + geom_col() +
facet_grid(metric~., scales="free")
+theme(axis.text=element_text(size=3),
        axis.title=element_text(size=14,face="bold"))
```



I only used RF, and KNN because from past results the SVM did not look like a good fit.

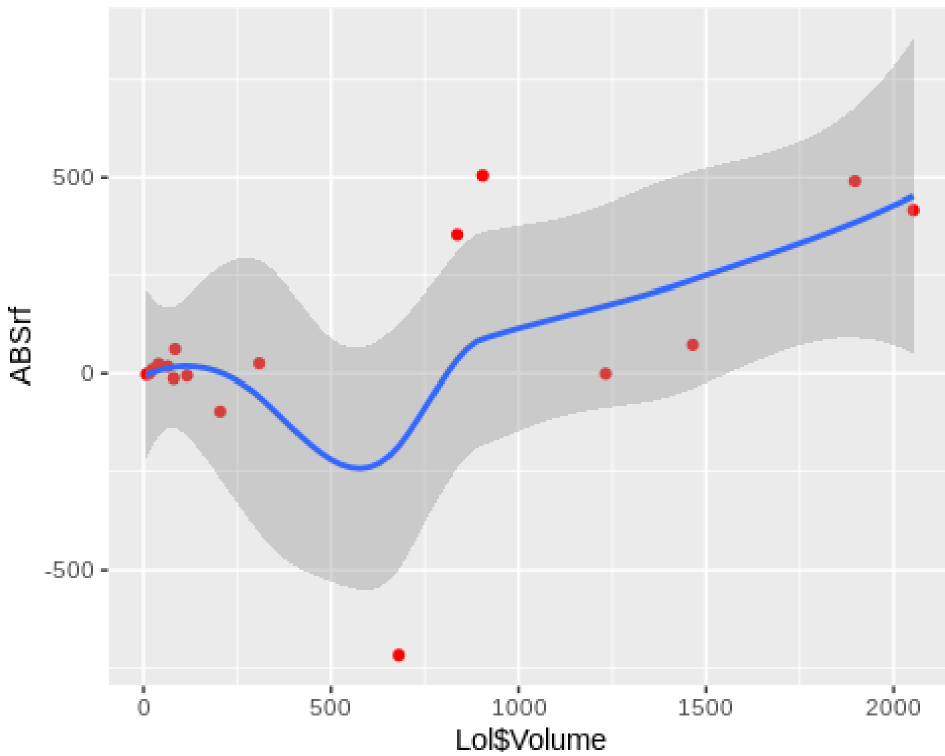
Error Analysis

```
ABSrf <- (List$testingSet$Volume - PredictionRandomForest)

RLTrf <-  (ABSrf / List$testingSet$Volume)

ABSsvm <- (List$testingSet$Volume - svm.pred)

RLTsvm <- (ABSsvm / List$testingSet$Volume)



Absknn <- (List$testingSet$Volume - KnnPrediction)

RLTknn <-  (Absknn / List$testingSet$Volume)
 #abline(0, 0)                  # the horizon


Lol <- cbind(List$testingSet,ABSrf)
```

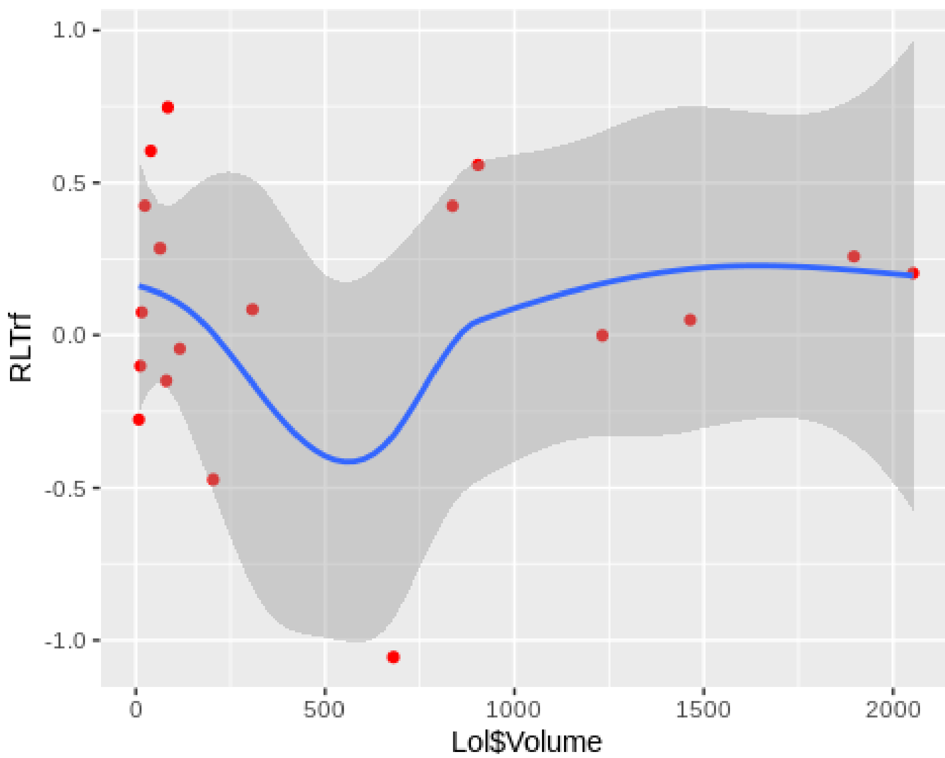
Random Forest Residuals

```
ggplot(Lol,
      aes(Lol$Volume,ABSrf))+
```

```
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



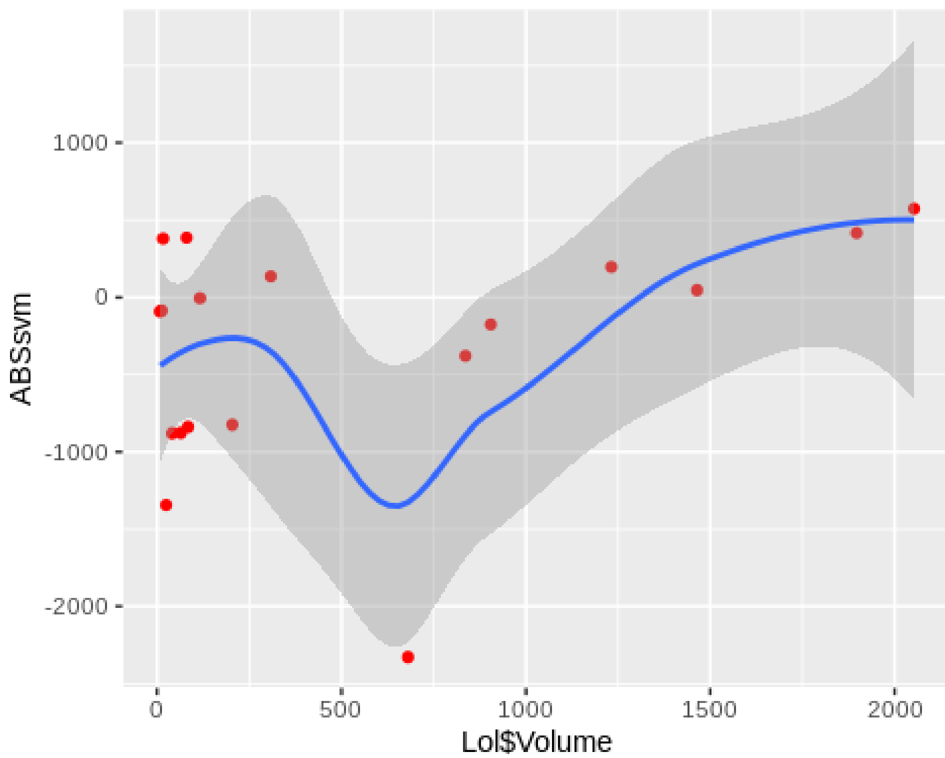

```
ggplot(Lol,
      aes(Lol$Volume,RLTrf))+
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



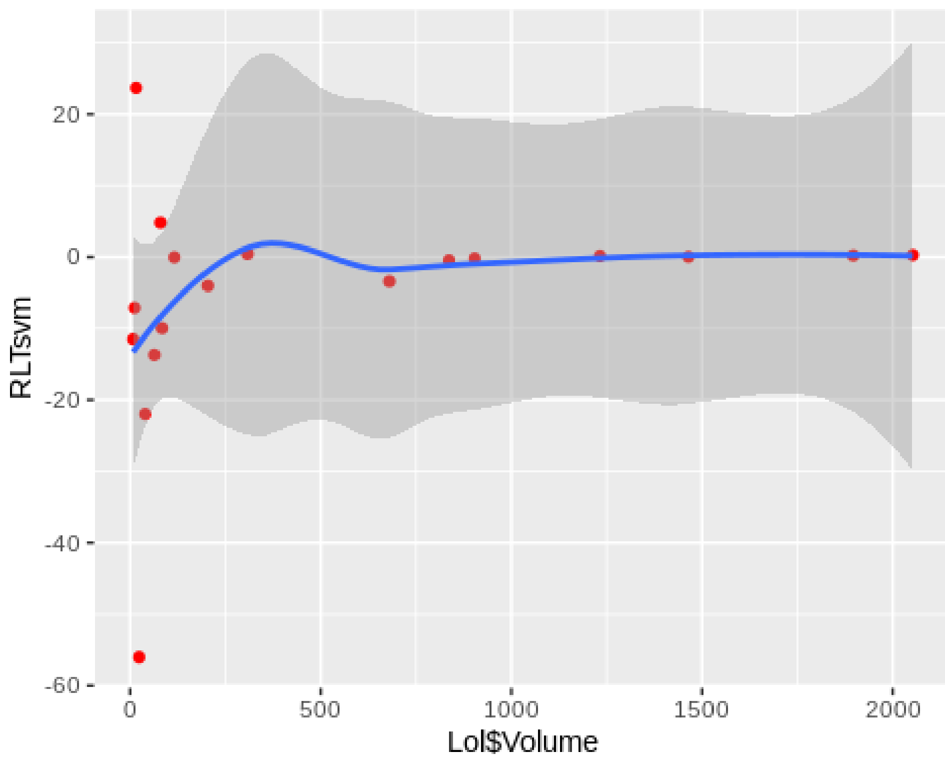
Svm

Residuals

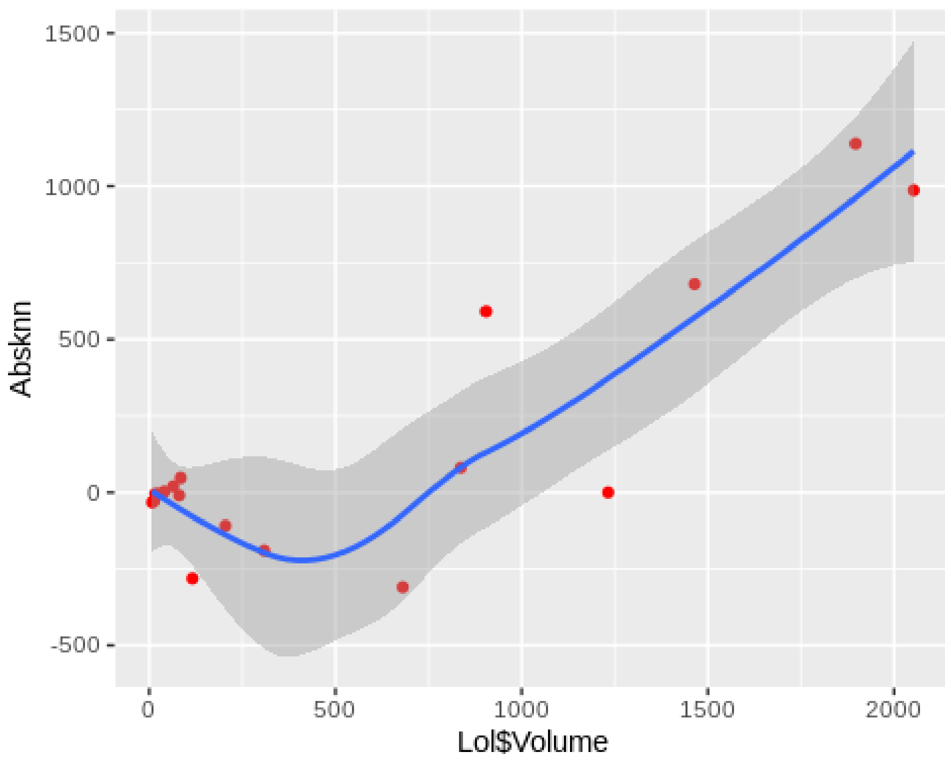
```
ggplot(Lol,
      aes(Lol$Volume,ABSsvm))+
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



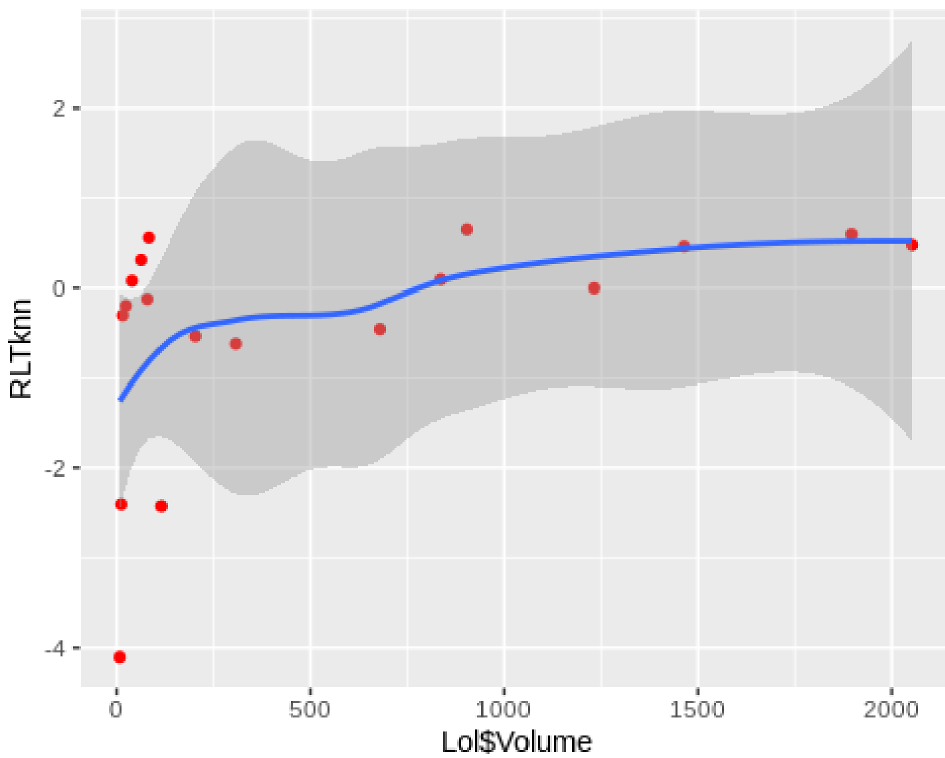

```
ggplot(Lol,
      aes(Lol$Volume,RLTsvm))+
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(Lol,
      aes(Lol$Volume,Absknn))+
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
ggplot(Lol,
      aes(Lol$Volume,RLTknn))+
 geom_point(color="red")+
 geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Let’s now apply the current models into the new product list and make a top 5 for most probably sold products in volume.

Prediction

The random forest was the one who gave us the best results, with both variables (ProductServiceReview,x4StarReviews).

```
NP <- read.csv(file =
"/home/zordo/Documents/Ubiqum/R-M2Task3/data/Npa.csv", header =
TRUE , sep =',')

NP <- PPfunction(NP)

NewProductsVolume <- predict(ModelRandomForest,NP)

NP$Volume<-NewProductsVolume
```

We got the volume, now we need to calculate the profit, to see which products types we should invest on.

Profit = profit margin * Volume

```
  Profit <- NP$ProfitMargin * NP$Volume

  NP <- cbind(NP,Profit)


Top5 <-  top_n(NP, 5, Profit)

Top5 <- cbind (Top5,sort(Top5$Profit))

Top5

##   ProductType.Accessories ProductType.Display
ProductType.ExtendedWarranty
## 1                       0                   0
0
## 2                       0                   0
0
## 3                       0                   0
0
## 4                       0                   0
0
## 5                       0                   0
0
##   ProductType.GameConsole ProductType.Laptop
ProductType.Netbook
## 1                       0                  0
1
## 2                       0                  0
0
## 3                       0                  0
0
## 4                       1                  0
0
## 5                       1                  0
0
##   ProductType.PC ProductType.Printer
ProductType.PrinterSupplies
## 1              0                   0
0
## 2              0                   0
0
## 3              0                   0
0
## 4              0                   0
0
## 5              0                   0
0
##   ProductType.Smartphone ProductType.Software
ProductType.Tablet
## 1                      0                    0
```

```
0
## 2                     0                    0
1
## 3                     0                    0
1
## 4                     0                    0
0
## 5                     0                    0
0
##   ProductNum  Price x5StarReviews x4StarReviews x3StarReviews
## 1        180 329.00           312           112            28
## 2        186 629.00           296            66            30
## 3        187 199.00           943           437           224
## 4        199 249.99           462            97            25
## 5        307 425.00          1525           252            99
##   x2StarReviews x1StarReviews PositiveServiceReview
NegativeServiceReview
## 1            31            47                    28
16
## 2            21            36                    28
9
## 3           160           247                    90
23
## 4            17            58                    32
12
## 5            56            45                    59
13
##   Recommendproduct BestSellersRank ShippingWeight ProductDepth
## 1              0.7            2699            4.6        10.17
## 2              0.8              34            3.0         7.31
## 3              0.8               1            0.9         5.40
## 4              0.8             115            8.4         6.20
## 5              0.9             215           20.0         8.50
##   ProductWidth ProductHeight ProfitMargin   Volume   Profit
## 1         7.28          0.95         0.09 1386.986 124.8287
## 2         9.50          0.37         0.10 1371.081 137.1081
## 3         7.60          0.40         0.20 1835.990 367.1979
## 4        13.20         13.20         0.09 1513.556 136.2200
## 5         6.00          1.75         0.18 1843.867 331.8960
##   sort(Top5$Profit)
## 1          124.8287
## 2          136.2200
## 3          137.1081
## 4          331.8960
## 5          367.1979
```

Our top 5 most profitable product types are

Number
Game Console
Game Console
Tablet
Tablet
NoteBook

Conclusion

All three models used are non-parametric, but before explaining what a non-parametric model is, I would like to explain what parametric models are. Parametric models are algorithms that simplify the function to a known form, and no matter how much data that’s fed to the algorithm, the model won’t change the quantity of parameters needed.All you need to know for predicting a future data balue from the current state of the model are his parameters,EG : Linear regression with on variable, you have two parameters (coefficient and intercept).Knowing this parameters will enable you to predict new values. In a mathematical way :
$Y_i = B_0 + B_1X_1 + B_2X_2 + \dots + e_i$ Non-parametric models do not make any fixed assumptions about the form of the mapping, they are free to learn from the training data. The parameters are usually said to be infinite in dimension and so can express the characteristics in data much better than parametric models. E.g : KNN, makes predictions based on the k most similar training patterns for a new data instance. The method does not assume anything about the form of the mapping function other than patterns that are close are likely have a similar output variable. In mathematical form   **$Y_i=F(x_i) + e_i$** Where F can be any function,the data will decide what the function looks like.It will not provide the analytical expression but it will give you its graph given your data set.   I find this picture very helpfull in understanding how both types of models work, summing up in easy non- technical language, parametric models follow an equation while non-parametric models follow the data.   One of the requirements to use a non-parametric model is to have a big set of data, we have 80 rows, 78 after cutting outliers and only 60 of them are for training.This is not a big set of data at all, I think it actually couldn’t be any smaller than this. Also they are good methods when we don’t

have any prior knowledge and not worry about choosing the right features. So using this models is not efficient at all, if we are using algorithms that “follow the data” and we have almost no data, it’s obvious why this approach is not efficient and why the errors are so big.