M3T3.R

christiancobollogomez

2021-12-12

```
# Module 3 Task 3: Predicting Customers Preference
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(gbm)
```

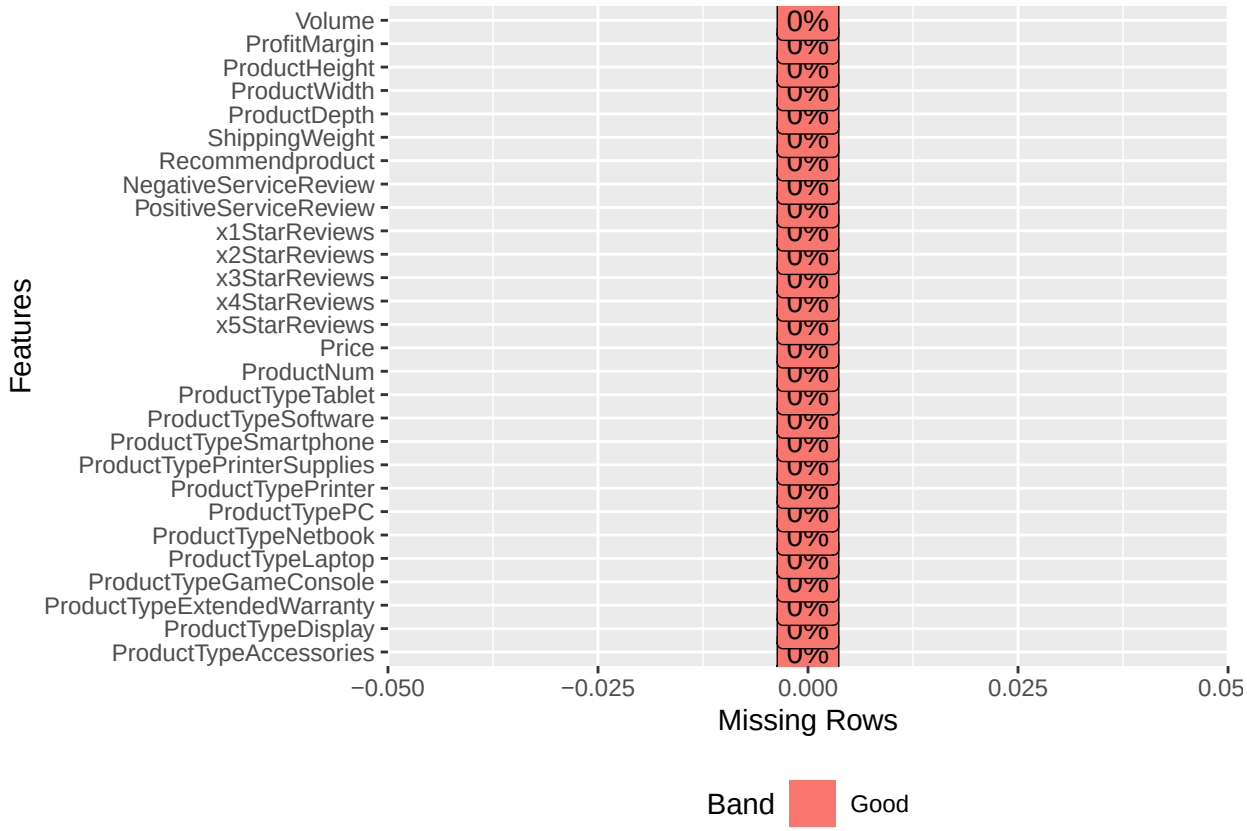
```
## Loaded gbm 2.1.8
```

```
library(reshape2) # Heatmap with melt()
library(ggplot2)

library(DataExplorer) # quick EDA package


set.seed(998)

existing_products<- read.csv("existingproductattributes2017.csv")

summary(existing_products) # Shows that in BestSellersRank we have NA's.
```

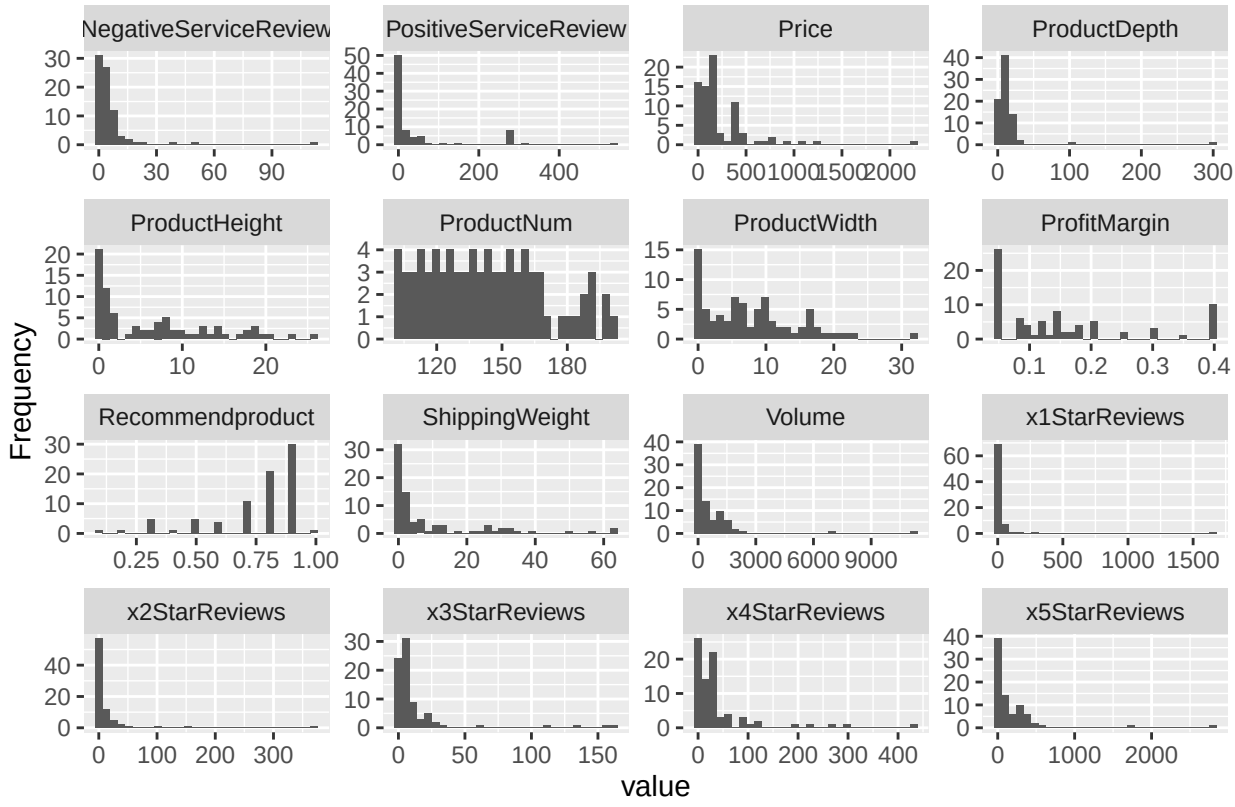
```
##  ProductType          ProductNum        Price          x5StarReviews
##  Length:80          Min.   :101.0   Min.   :   3.60   Min.   :   0.0
##  Class :character   1st Qu.:120.8   1st Qu.:  52.66   1st Qu.:  10.0
##  Mode  :character   Median :140.5   Median : 132.72   Median :  50.0
##                     Mean   :142.6   Mean   : 247.25   Mean   : 176.2
##                     3rd Qu.:160.2   3rd Qu.: 352.49   3rd Qu.: 306.5
##                     Max.   :200.0   Max.   :2249.99   Max.   :2801.0
##
##  x4StarReviews    x3StarReviews    x2StarReviews    x1StarReviews
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :   0.00
##  1st Qu.:  2.75   1st Qu.:  2.00   1st Qu.:  1.00   1st Qu.:   2.00
##  Median : 22.00   Median :  7.00   Median :  3.00   Median :   8.50
##  Mean   : 40.20   Mean   : 14.79   Mean   : 13.79   Mean   :  37.67
##  3rd Qu.: 33.00   3rd Qu.: 11.25   3rd Qu.:  7.00   3rd Qu.:  15.25
##  Max.   :431.00   Max.   :162.00   Max.   :370.00   Max.   :1654.00
##
##  PositiveServiceReview NegativeServiceReview Recommendproduct BestSellersRank
##  Min.   :  0.00        Min.   :  0.000       Min.   :0.100    Min.   :    1
```

```
##  1st Qu.:  2.00        1st Qu.:  1.000       1st Qu.:0.700    1st Qu.:    7
##  Median :  5.50        Median :  3.000       Median :0.800    Median :   27
##  Mean   : 51.75        Mean   :  6.225       Mean   :0.745    Mean   : 1126
##  3rd Qu.: 42.00        3rd Qu.:  6.250       3rd Qu.:0.900    3rd Qu.:  281
##  Max.   :536.00        Max.   :112.000       Max.   :1.000    Max.   :17502
##                                                               NA's   :15
##  ShippingWeight     ProductDepth      ProductWidth    ProductHeight
##  Min.   : 0.0100   Min.   :  0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.5125   1st Qu.:  4.775   1st Qu.: 1.750   1st Qu.: 0.400
##  Median : 2.1000   Median :  7.950   Median : 6.800   Median : 3.950
##  Mean   : 9.6681   Mean   : 14.425   Mean   : 7.819   Mean   : 6.259
##  3rd Qu.:11.2050   3rd Qu.: 15.025   3rd Qu.:11.275   3rd Qu.:10.300
##  Max.   :63.0000   Max.   :300.000   Max.   :31.750   Max.   :25.800
##
##   ProfitMargin        Volume
##  Min.   :0.0500   Min.   :    0
##  1st Qu.:0.0500   1st Qu.:   40
##  Median :0.1200   Median :  200
##  Mean   :0.1545   Mean   :  705
##  3rd Qu.:0.2000   3rd Qu.: 1226
##  Max.   :0.4000   Max.   :11204
##
```

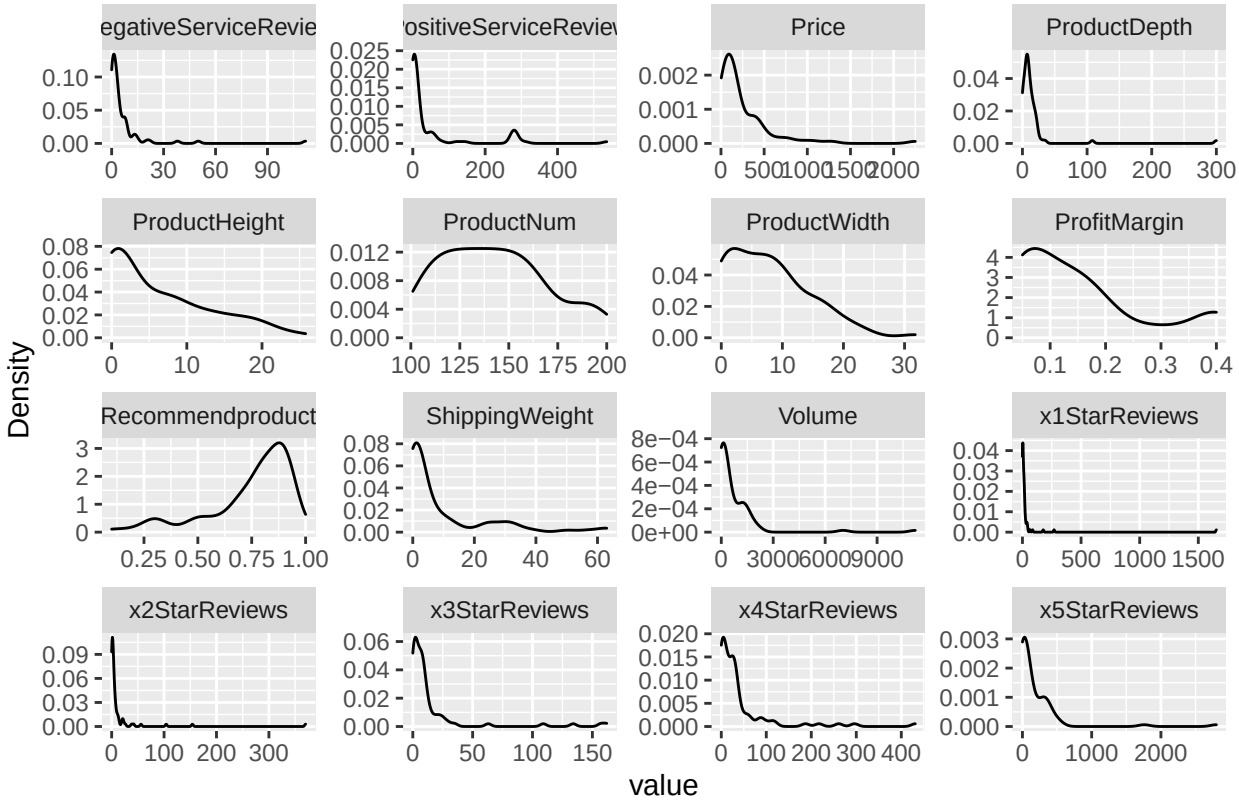
```r
# Delete the attribute with missing data:
existing_products$BestSellersRank<-NULL

# dummify the data

EPdata <- dummyVars(" ~ .", data = existing_products)

readyEPdata <- data.frame(predict(EPdata, newdata = existing_products))

# quick EDA:

plot_str(readyEPdata)

plot_missing(readyEPdata)
```



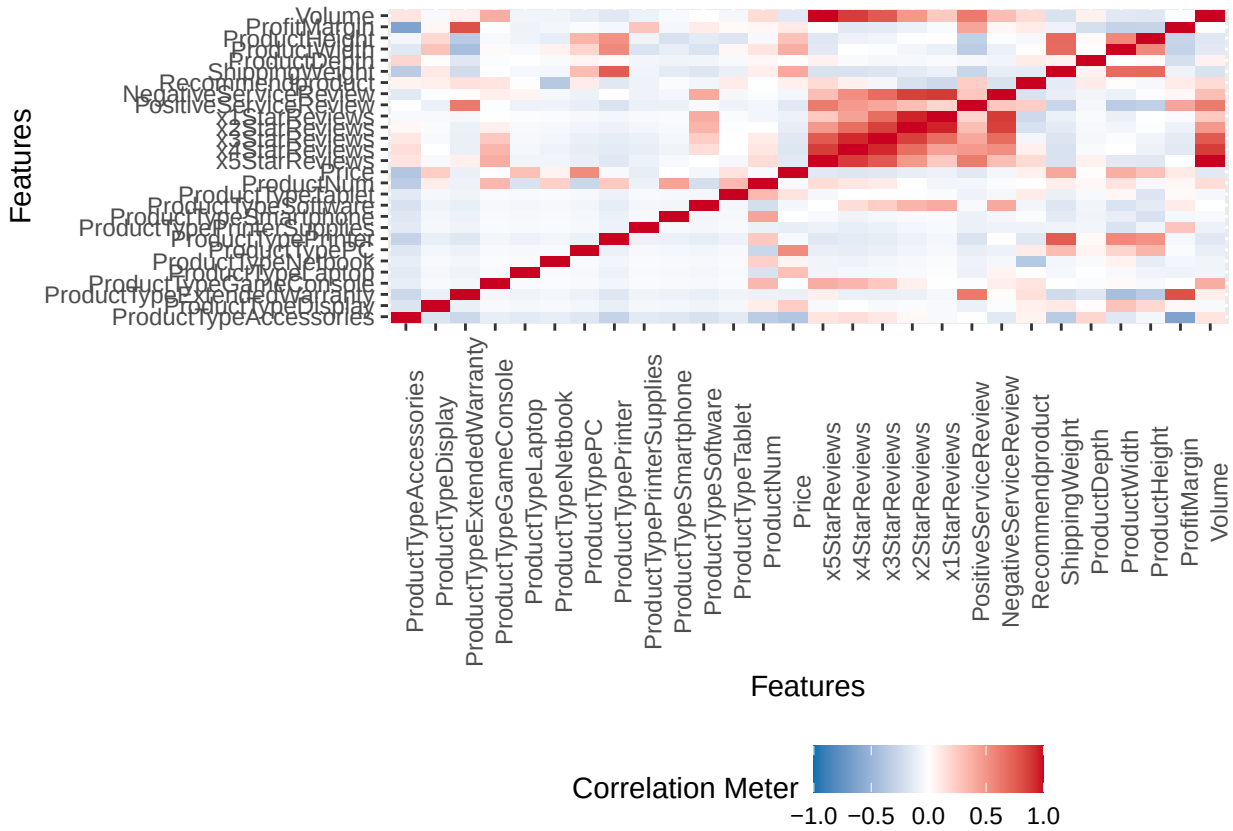

```
plot_histogram(readyEPdata)
```



```
plot_density(readyEPdata)
```



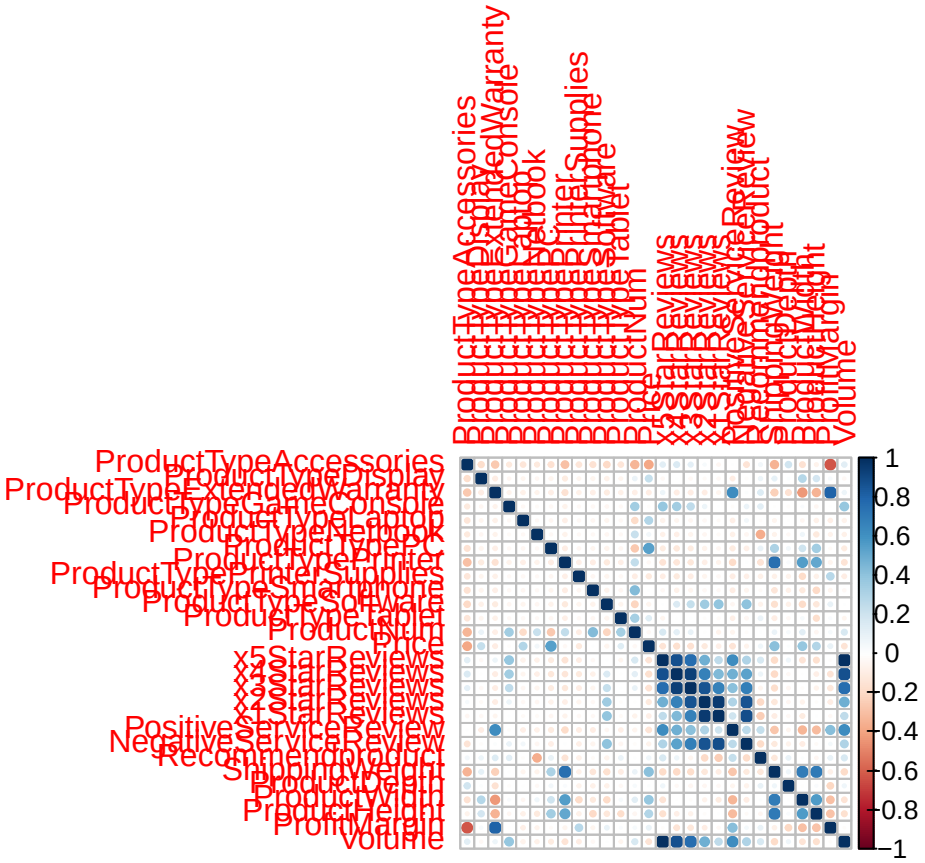

```
#plot correlation matrix in two different ways:

plot_correlation(readyEPdata, type = 'continuous')
```




```
# We compute correlation
corrData <- cor(readyEPdata)


melted_cormat <- melt(corrData)
head(melted_cormat)
```

```
##                            Var1                    Var2      value
## 1      ProductTypeAccessories ProductTypeAccessories  1.0000000
## 2          ProductTypeDisplay ProductTypeAccessories -0.1791613
## 3 ProductTypeExtendedWarranty ProductTypeAccessories -0.2622653
## 4      ProductTypeGameConsole ProductTypeAccessories -0.1111111
## 5           ProductTypeLaptop ProductTypeAccessories -0.1369636
## 6          ProductTypeNetbook ProductTypeAccessories -0.1111111
```

```
#ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
 # geom_tile()

corrplot(corrData) # heat map correlation matrix.
```




```
# From the correlation Matrix we can observe that one of the variables
# that has a higher value from the positive correlation is the x5StarReviews.
# We may want to build the linear model using this as a feature:




##############################
# Linear Model 1: Predicting Volume using 5 Star Reviews

Data<-readyEPdata
# We select the features according to variable importance and the correlation matrix.
Data<-Data[c("x5StarReviews","x4StarReviews","x3StarReviews","x2StarReviews","x1StarReviews","PositiveS


trainSize<-round(nrow(Data)*0.7)

testSize<-nrow(Data)-trainSize

trainSize
```

```
## [1] 56
```

```
testSize
```

```
## [1] 24
```

```
training_indices<-sample(seq_len(nrow(Data)),size =trainSize)

trainSet<-Data[training_indices,]

testSet<-Data[-training_indices,]

LRmodel<-lm(Volume~ x5StarReviews, trainSet)

summary(LRmodel)
```

```
## Warning in summary.lm(LRmodel): essentially perfect fit: summary may be
## unreliable
```

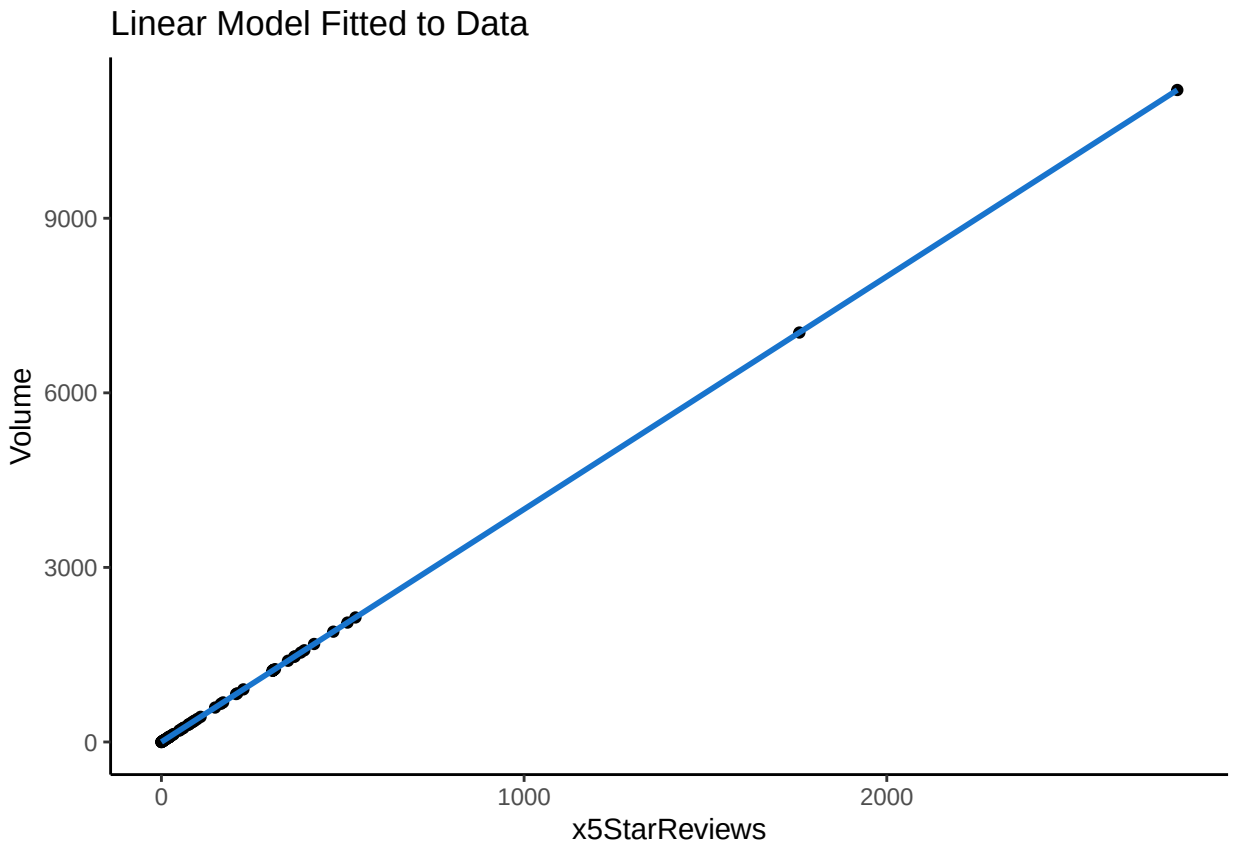
```
##
## Call:
## lm(formula = Volume ~ x5StarReviews, data = trainSet)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.217e-12 -1.935e-13 -1.665e-13 -1.429e-13  9.638e-12
##
## Coefficients:
##                Estimate Std. Error   t value Pr(>|t|)
## (Intercept)   4.861e-13  1.917e-13 2.536e+00   0.0141 *
## x5StarReviews 4.000e+00  4.705e-16 8.502e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.334e-12 on 54 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.228e+31 on 1 and 54 DF,  p-value: < 2.2e-16
```

```
PredictLR <- predict(LRmodel,testSet)


PredictLR
```

```
##   12   13   14   20   22   23   24   25   26   31   33   39   44   48   52   53
##  300   40 1252   80 1576 2052  116  308 1224   20   20 1232  368 2140  204 1896
##   56   57   59   63   65   73   75   80
##  360  656   84   32   80 7036   12 1684
```

```
## We represent the model vs the scatter plot of the data.
ggplot(data = Data, aes(x = x5StarReviews, y = Volume)) +
  geom_point() +
  stat_smooth(method = "lm", col = "dodgerblue3") +
  theme(panel.background = element_rect(fill = "white"),
        axis.line.x=element_line(),
        axis.line.y=element_line()) +
  ggtitle("Linear Model Fitted to Data")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Apparently there is nothing wrong, but it is very suspicious having practically perfect
# scores even using only one variable (even though, the correlation matrix
# show a very high value of positive correlation). It may be a case of overfitting.
```

```
# Linear Model 2: Predicting Volume using 4 Star Reviews
```

```
LRmodel<-lm(Volume~ x4StarReviews, trainSet)

summary(LRmodel)
```

```
##
## Call:
## lm(formula = Volume ~ x4StarReviews, data = trainSet)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1563.02  -186.55    86.75   143.85  1166.39
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

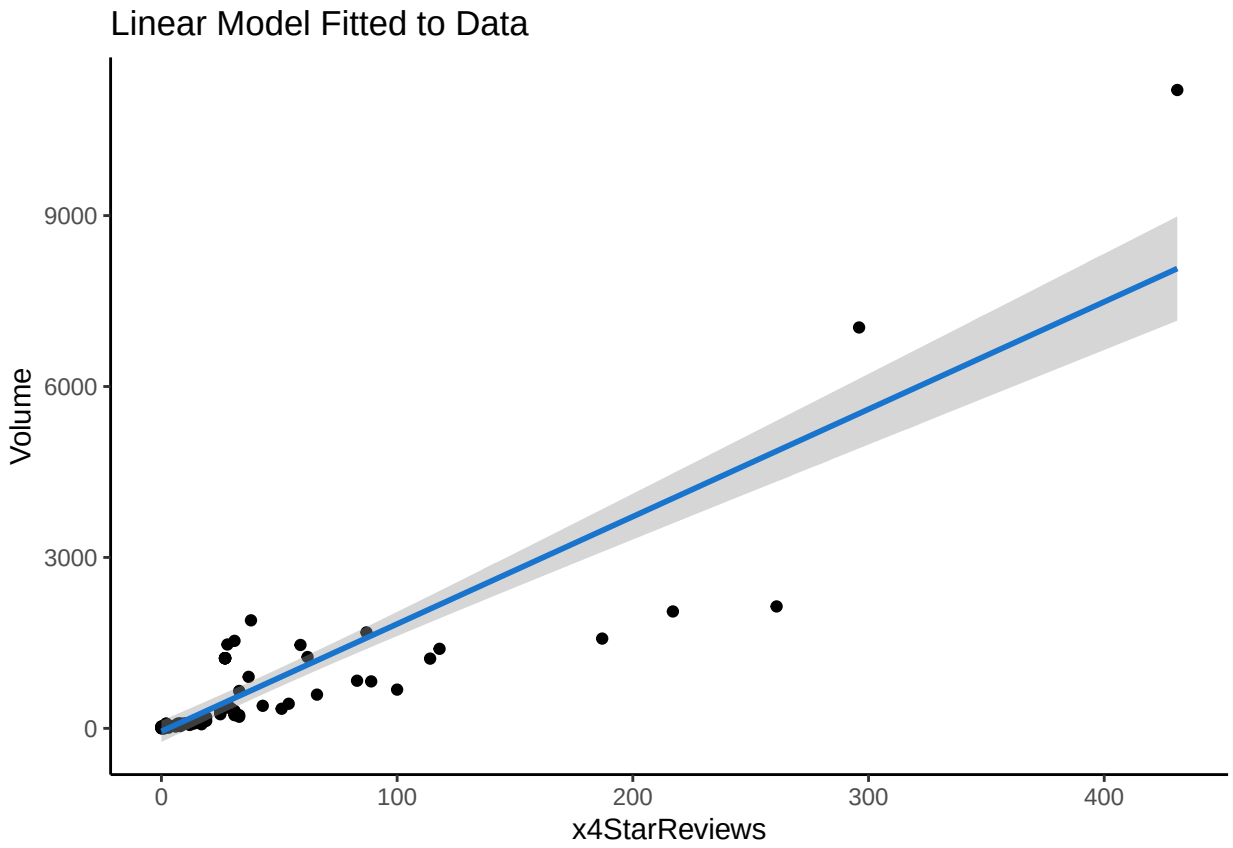
```
## (Intercept)   -111.846     81.376  -1.374    0.175
## x4StarReviews   23.549      1.208  19.499   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 544.1 on 54 degrees of freedom
## Multiple R-squared:  0.8756, Adjusted R-squared:  0.8733
## F-statistic: 380.2 on 1 and 54 DF,  p-value: < 2.2e-16
```

```
PredictLR <- predict(LRmodel,testSet)


PredictLR
```

```
##         12         13         14         20         22         23         24
##  476.86969   76.54298 1348.16901  -64.74880 4291.74779 4998.20669  312.02928
##         25         26         31         33         39         44         48
##  618.16147 2572.69778 -111.84606 -111.84606  523.96695  571.06421 6034.34642
##         52         53         56         57         59         63         65
##  665.25873  783.00188  523.96695  665.25873  123.64024  -41.20017  194.28613
##         73         75         80
## 6858.54848  -88.29743 1936.88476
```

```
## We represent the model vs the scatter plot of the data.
ggplot(data = Data, aes(x = x4StarReviews, y = Volume)) +
  geom_point() +
  stat_smooth(method = "lm", col = "dodgerblue3") +
  theme(panel.background = element_rect(fill = "white"),
        axis.line.x=element_line(),
        axis.line.y=element_line()) +
  ggtitle("Linear Model Fitted to Data")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# $ Star also has a high correlation value and the results are not even close
# to the 5StarReviews case, but this may be a more realistic result.


## Non-parametric Machine Learning Models


# define an 70%/30% train/test split of the dataset
inTraining <- createDataPartition(Data$Volume, p = .70, list = FALSE)
training <- Data[inTraining,]
testing <- Data[-inTraining,]

#10 fold cross validation
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)

##############################################################################
## Support Vector Machines (SVM)


#train model with a tuneLenght = 5 (trains with 5 mtry value for SVM)
SVMfit1 <- train(Volume~., data = training, method = "svmLinear", trControl=fitControl, tuneLength = 5)

SVMfit1
```

```
## Support Vector Machines with Linear Kernel
##
## 57 samples
##  7 predictor
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 52, 52, 51, 50, 51, 52, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   128.197  0.9915989  89.78558
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
summary(SVMfit1) # Shows the relative influence of each variable in the model.
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
VarImpSVM<-varImp(SVMfit1,numTrees = NULL) # variable importance

VarImpSVM
```

```
## loess r-squared variable importance
##
##                       Overall
## x5StarReviews          100.00
## x4StarReviews           93.97
## PositiveServiceReview   86.96
## x2StarReviews           73.08
## x3StarReviews           56.03
## NegativeServiceReview   29.33
## x1StarReviews            0.00
```

```
# First, we will evaulate on the testing set to evaluate

SVMpred1 <- predict(SVMfit1,testing)
SVMpred1 # It predicts negative values of Volumes, which is impossible. We should
```

```
##          2          3          8         11         28         32         37
##  -60.55569  -58.67490   68.70480   10.58997   21.37841  -75.82039 1194.28238
##         40         42         43         44         45         46         56
## 1194.28238   10.69556   32.61904  309.01604 1394.56702 1379.94315  293.16816
##         60         62         63         67         69         70         73
##  238.35294  -57.69371  -37.51256  735.00203  329.05710  -69.16386 6783.11289
##         76         80
##  195.21420 1589.00531
```

```
# discard this model

summary(SVMpred1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -75.82  -13.46  195.21  670.42  964.64 6783.11
```

```
ResultsSVM1<-postResample(SVMpred1,testing$Volume)#

ResultsSVM1 # #Rsquared is very high, but RMSE and MAE may be suspicious.
```

```
##       RMSE   Rsquared        MAE
## 89.1214820  0.9997906 78.4534667
```

```
##############################################################################

## Random Forest

#train model with a tuneLenght = 5 (trains with 5 mtry value for Random Forest)
RFfit1 <- train(Volume~., data = training, method = "rf", trControl=fitControl, tuneLength = 5);

RFfit1
```

```
## Random Forest
##
## 57 samples
##  7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 52, 51, 51, 52, 50, 50, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     696.2914  0.9674234  355.7056
##   3     635.2545  0.9801397  315.0501
##   4     627.9415  0.9826814  310.3012
##   5     604.7272  0.9844233  298.2291
##   7     590.4946  0.9889196  287.8835
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 7.
```

```
summary(RFfit1) # Shows the relative influence of each variable in the model.
```

```
##                 Length Class      Mode
## call               4   -none-     call
## type               1   -none-     character
## predicted         57   -none-     numeric
## mse              500   -none-     numeric
## rsq              500   -none-     numeric
## oob.times         57   -none-     numeric
## importance         7   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            11   -none-     list
## coefs              0   -none-     NULL
## y                 57   -none-     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames             7   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          1   -none-     logical
## param              0   -none-     list
```

```r
VarImpRF<-varImp(RFfit1,numTrees = NULL) # variable importance

VarImpRF
```

```
## rf variable importance
##
##                         Overall
## x5StarReviews          100.0000
## PositiveServiceReview   36.5442
## x4StarReviews           33.9747
## x1StarReviews            8.8308
## x2StarReviews            6.1680
## x3StarReviews            0.8417
## NegativeServiceReview    0.0000
```

```r
# First, we will evaulate on the testing set to evaluate

RFpred1 <- predict(RFfit1,testing)

RFpred1 # Every volume predicted is a positive number, as it should be.
```

```
##          2          3          8         11         28         32         37
##    9.52040   16.89933  156.65667   75.15853   74.05773   10.93213 1232.48813
##         40         42         43         44         45         46         56
## 1232.48813   80.98547   93.07493  343.69960 1449.84400 1422.69640  323.36467
##         60         62         63         67         69         70         73
##  293.69240   15.34627   33.63467 1277.22000  397.11040    4.35600 4040.75707
##         76         80
##  240.90053 1528.49200
```

```r
summary(RFpred1)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    4.356   53.846  240.901  624.060 1232.488 4040.757
```

```r
ResultsRF1<-postResample(RFpred1,testing$Volume)#

ResultsRF1 #
```

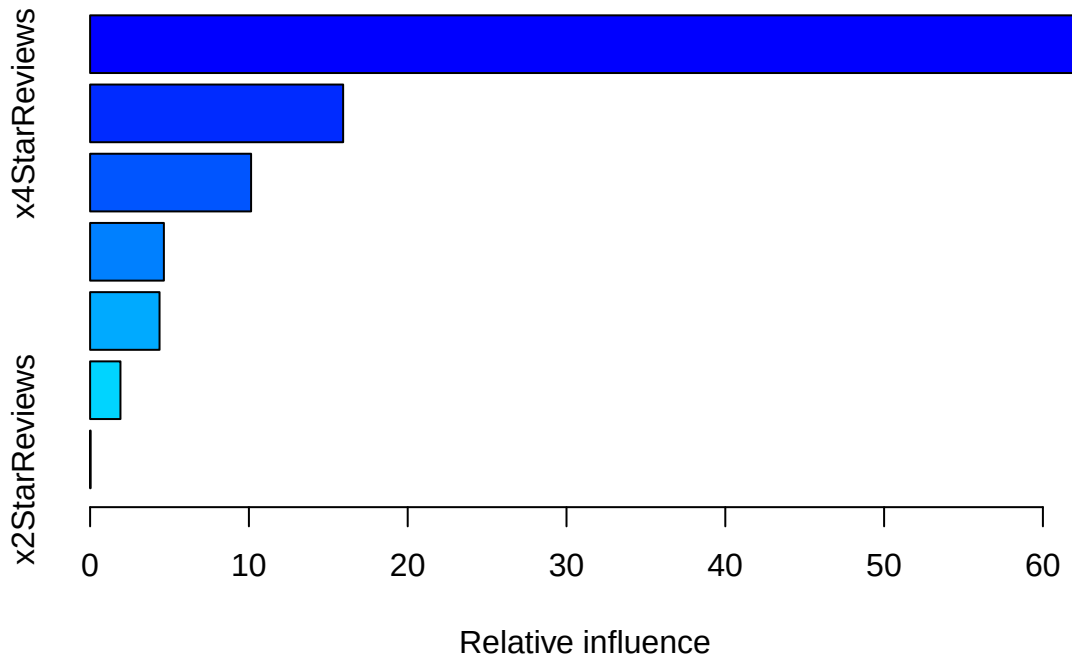
```
##        RMSE    Rsquared         MAE
## 632.9109656   0.9289243 168.6689623
```

```r
#####################################################
## Gradient Boosting


#train model with a tuneLenght = 5 (trains with 5 mtry value for Random Forest)
GBfit1<-train(Volume~., data = training, method = "gbm", trControl=fitControl, tuneLength = 5,verbose=F

GBfit1
```

```
## Stochastic Gradient Boosting
##
## 57 samples
##  7 predictor
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 51, 53, 52, 52, 50, 52, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      839.3976  0.8517723  510.3453
##   1                  100      872.9437  0.8100929  520.0436
##   1                  150      855.9724  0.8102823  521.9233
##   1                  200      927.4407  0.7960992  575.9741
##   1                  250      942.5258  0.7567870  581.6163
##   2                   50      844.8608  0.8590836  524.9232
##   2                  100      855.7702  0.8244638  531.9524
##   2                  150      869.5934  0.8065430  540.3207
##   2                  200      924.9816  0.7710412  575.1151
##   2                  250      923.2832  0.7690634  568.5782
##   3                   50      825.2804  0.8563175  507.6042
##   3                  100      860.7717  0.8249499  524.8827
##   3                  150      901.9480  0.8112257  564.6943
##   3                  200      912.0463  0.7903811  559.6767
##   3                  250      946.5058  0.7926152  591.6768
##   4                   50      794.4756  0.8532225  489.1598
##   4                  100      846.8452  0.8314044  530.2877
##   4                  150      851.1168  0.8082610  529.9923
##   4                  200      881.4377  0.8039912  551.6418
##   4                  250      922.3068  0.7790274  580.0237
##   5                   50      830.7915  0.8561415  510.0523
##   5                  100      831.1536  0.8498755  510.1181
##   5                  150      861.2729  0.8318469  527.3245
##   5                  200      900.9535  0.8215679  569.7378
##   5                  250      929.3953  0.8123446  574.2867
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  4, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
summary(GBfit1) # Shows the relative influence of each variable in the model.
```




```
##                                       var     rel.inf
## x5StarReviews                 x5StarReviews 62.96649265
## x4StarReviews                 x4StarReviews 15.93491569
## PositiveServiceReview PositiveServiceReview 10.13763464
## NegativeServiceReview NegativeServiceReview  4.64650470
## x1StarReviews                 x1StarReviews  4.37350842
## x3StarReviews                 x3StarReviews  1.91165065
## x2StarReviews                 x2StarReviews  0.02929325
```

```
VarImpGB<-varImp(GBfit1,numTrees = NULL) # variable importance

VarImpGB
```

```
## gbm variable importance
##
##                         Overall
## x5StarReviews           100.000
## x4StarReviews            25.272
## PositiveServiceReview    16.061
## NegativeServiceReview     7.336
## x1StarReviews             6.902
## x3StarReviews             2.991
## x2StarReviews             0.000
```

```
# First, we will evaulate on the testing set to evaluate

GBpred1 <- predict(GBfit1,testing)# The values are discrete because
# the imputs were introduced as factors. In other case, the model returns
# continuous values, close to 0 and 1.

GBpred1 #Negative Volumes. We don't keep this model.
```

```
##  [1]  -18.90685  -18.90685   43.31941  131.00902  -18.90685  -12.22936
##  [7] 1700.84274 1700.84274  173.13157  -25.59932  520.77797 2100.30498
```

```
## [13] 2282.47726   33.86956   88.98957  -18.90685  -18.90685 1833.09890
## [19]  874.45856  -18.90685 2282.47726  256.43911 2282.47726
```

```
summary(GBpred1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -25.60  -18.91  131.01  702.32 1700.84 2282.48
```

```
ResultsGB1<-postResample(GBpred1,testing$Volume)#

ResultsGB1 #
```

```
##         RMSE     Rsquared          MAE
## 1060.1573183    0.4688859  455.5858295
```

```
# Random Forest is the only one that does not predict any negative Volumes,
# and has one of the better performance. Thus, we will use Random Forest to
# do the final prediction.


new_products<- read.csv("newproductattributes2017.csv")


summary(new_products) # Shows that in BestSellersRank we have NA's.
```

```
##  ProductType          ProductNum        Price        x5StarReviews
##  Length:24          Min.   :171.0   Min.   :   8.5   Min.   :   0.00
##  Class :character   1st Qu.:179.5   1st Qu.: 130.0   1st Qu.:  16.00
##  Mode  :character   Median :193.5   Median : 275.0   Median :  46.00
##                     Mean   :219.5   Mean   : 425.6   Mean   : 178.50
##                     3rd Qu.:301.2   3rd Qu.: 486.5   3rd Qu.:  99.25
##                     Max.   :307.0   Max.   :1999.0   Max.   :1525.00
##  x4StarReviews    x3StarReviews    x2StarReviews    x1StarReviews
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  2.00   1st Qu.:  1.75   1st Qu.:  1.00   1st Qu.:  1.75
##  Median : 10.50   Median :  4.50   Median :  4.00   Median : 13.00
##  Mean   : 48.04   Mean   : 21.92   Mean   : 17.50   Mean   : 27.58
##  3rd Qu.: 26.00   3rd Qu.: 16.75   3rd Qu.: 20.25   3rd Qu.: 35.25
##  Max.   :437.00   Max.   :224.00   Max.   :160.00   Max.   :247.00
##  PositiveServiceReview NegativeServiceReview Recommendproduct
##  Min.   : 0.00         Min.   : 0.000        Min.   :0.3000
##  1st Qu.: 2.00         1st Qu.: 1.000        1st Qu.:0.6000
##  Median : 5.00         Median : 3.500        Median :0.7000
##  Mean   :13.46         Mean   : 5.667        Mean   :0.6708
##  3rd Qu.:12.50         3rd Qu.: 7.500        3rd Qu.:0.8000
##  Max.   :90.00         Max.   :23.000        Max.   :1.0000
##  BestSellersRank    ShippingWeight    ProductDepth     ProductWidth
##  Min.   :    1.00   Min.   : 0.200   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:   93.25   1st Qu.: 0.900   1st Qu.: 5.225   1st Qu.: 5.832
##  Median :  750.50   Median : 4.450   Median : 8.000   Median : 9.950
##  Mean   : 3957.62   Mean   : 7.802   Mean   : 9.094   Mean   :10.408
##  3rd Qu.: 3150.00   3rd Qu.: 9.575   3rd Qu.:11.425   3rd Qu.:12.875
##  Max.   :44465.00   Max.   :42.000   Max.   :21.890   Max.   :27.010
##  ProductHeight     ProfitMargin        Volume
##  Min.   : 0.000   Min.   :0.0500   Min.   :0
##  1st Qu.: 0.400   1st Qu.:0.0975   1st Qu.:0
```

```
##  Median : 0.985   Median :0.1150   Median :0
##  Mean   : 3.541   Mean   :0.1817   Mean   :0
##  3rd Qu.: 2.888   3rd Qu.:0.2000   3rd Qu.:0
##  Max.   :25.800   Max.   :0.9000   Max.   :0
```

```
# Delete the attribute with missing data:
new_products$BestSellersRank<-NULL

new_products$Volume<-NULL #Delete the empty Volume Variable

# dummify the data

NPdata <- dummyVars(" ~ .", data = new_products)

readyNPdata <- data.frame(predict(NPdata, newdata = new_products))

Ndata<-readyNPdata[c("x5StarReviews","x4StarReviews","x3StarReviews","x2StarReviews","x1StarReviews","P


FinalPred <- predict(RFfit1,Ndata) # We predict with RF model.

FinalPred<-round(FinalPred)

FinalPred
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
##  420  190  309   28    7   74 1241  156   12 1208 4369  439  504  138  223 1737
##   17   18   19   20   21   22   23   24
##   16   92   90  117  260   19   11 1867
```

```
## Prepare the output. We will leave the original file intact, adding the predictions.

output<-read.csv("newproductattributes2017.csv")

output$predictions<-FinalPred

write.csv(output, file="C2.T3output.csv", row.names = TRUE)


###############################

## Conclusions:

# In the correlation matrix we could appreciate that the reviews and opinions of
# the customers have a great impact on the Volume of sales. Specifically,
# the 5 4 and 3 Stars reviews and the Positive Service Reviews. Accordingly, those
# are the variables with high relative importance when building the models.

# We can get nice results by reducing the number of features. As it is shown by
# the correlation matrix, the majority of the variables are just very minorly
# related with sales Volumes.

# A sample of 80 to test is not so big. We may want a bigger dataframe.
# Linear Regression, GBM and SVM were used and even provided very nice Rsquare values.
# but were rejected because overfitting or predicting negative volumes.
```

```
# RF provided 0.92 Rsquared and apparently good results. Also, every prediction
# was a non-negative value.

# We focus on PC, laptops, Netbooks and SmartPhones. In the final .csv  output
# with Excel we can seethe predictions on the volums of this and other products:

# Total Volumes Smartphones: 1304. The Product Number 194 has 504 of volume sales.

# Total Volumes Netbooks: 1483. ProdNum 180 has a predicted volume of 1241, so apparently
# it will be nice product to sell. Probably because it is the cheapest Netbook.
# 9% Margin of profit.

# Total Volumes Laptop: 344.The product number 173 has 309 predicted sells. 10%,
# the smalles profit margin on laptops.

# Total Volumes PC: 610. The product number 171 has 420 sales by its own, and
# also has a 25% of profit margin. Very nice product from the economic perspective.
```