Sales Offering Optimization

Business Analytics with R Project
BUAN 6356

Group 14
Chirag Hamirani
Hyunsoo Park
Risto Pitkanen
Yanyi Tang

Contents

1. **Background**

Brazil is the largest E-Commerce market in the Latin America, accounting for over 40% of the region’s online sales, followed by Mexico (18%). The online sales of Brazil are also estimated to grow at a compound annual rate of almost 11 percent between 2018 and 2022. Olist, founded in 2014, is the largest online sales platform in Brazil. It connects small business all over brazil directly to end customers. Therefore, the analysis of Olist dataset will help identify the business pattern from the data and provide a deep insight of Brazilian E-Commerce market as well.

1.1 Objective of the Project

Our project is to help solve following business problems of Olist to boost its online sales to the future.

A. Which product categories should merchants expand to?
Find association among different products from transaction records of Olist store, filter rules based on judgment, and recommend the merchants the product categories that should be linked.

B. Which customer groups should be targeted with the extension of product offering?
Identify different customer groups based on clustering tool, get the picture of different groups, and recommend marketing policy for targeted group.

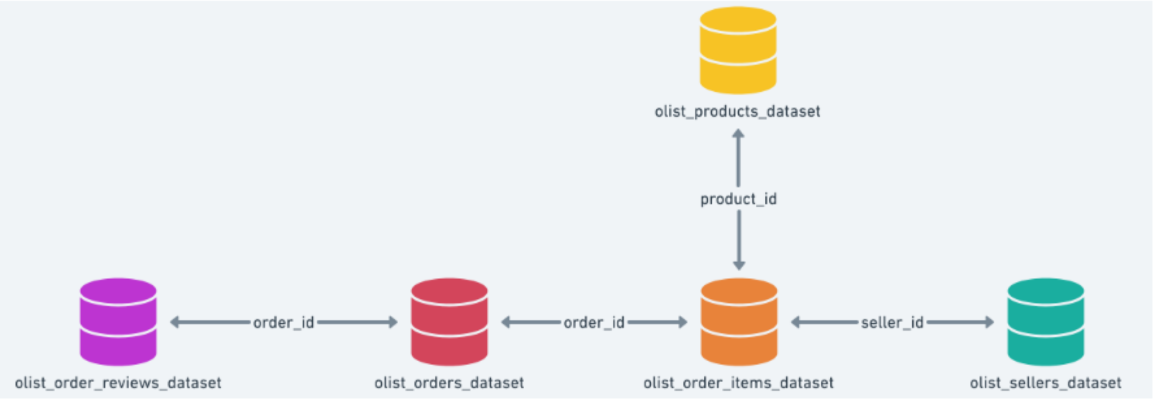
C. How would customer groups respond to the product?
Predict factors affecting review score given by customers, identify the gap between customer expectation and platform’s intuition, and further improve customer satisfaction from prediction.

1.2 Data description

The source of the dataset is Olist public dataset (second-hand) which was extracted in .csv-format. The dataset has information of approximately 100k transactions made at Olist store from 2016 to 2018. The whole dataset consists of 7 separate datasets, 99,441 rows in each table.

1.3 Data selection:

We selected five separate datasets for data processing



Olist products dataset:

The products dataset includes 9 attributes and we selected three of them in our analysis. The attributes of product description length, photo numbers and product volume are not directly related to our analysis.

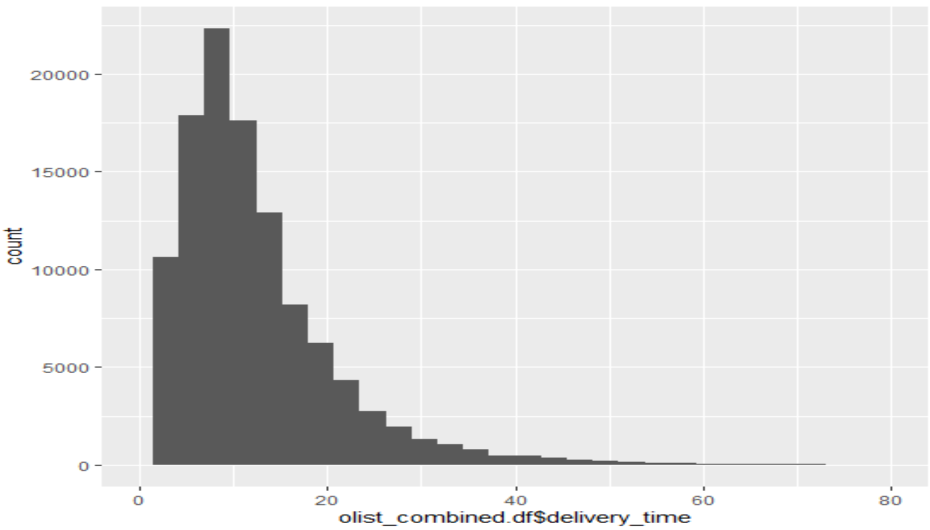
Attributes	Description	Selected or not?
product_id	unique id for product	Y
product_category_name	product category name	Y
product_name_lenght	length of product name	Y
product_description_lenght	length fo product description	N
product_photos_qty	number of product photos	N
product_weight_g	product weight	N
product_length_cm	part of product cube	N
product_height_cm	part of product cube	N
product_width_cm	part of product cube	N

Olist orders dataset:

The orders dataset includes 8 attributes. We selected order_id for mapping and order_purchase_timestamp together with order_delivered_customer_date to obtain delivery time, which is crucial for the prediction of factors influencing customer satisfaction.

Attributes	Description	Selected or not?
order_id	unique id for orders	Y
customer_id	id for customers	N
order_status	8 status	N
order_purchase_timestamp	order purchase time	Y
order_approved_at	order approval time	N
order_delivered_carrier_date	arrier delivery date	N
order_delivered_customer_date	customer receipt date	Y
order_estimated_delivery_date	estimated delivery date	N

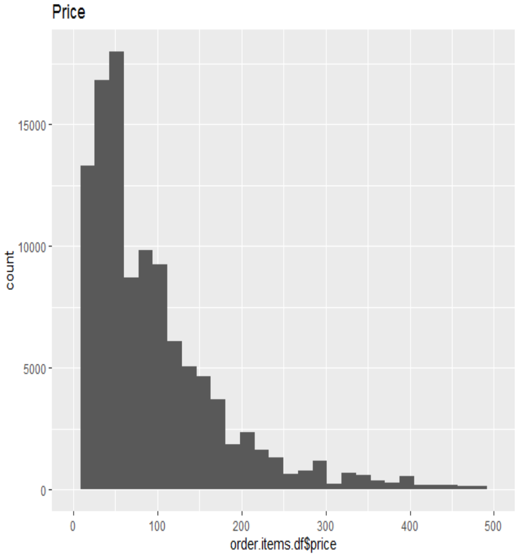
We calculated the delivery time with variable order order_purchase_timestamp and order_delivered_customer_date, and plotted the its distribution. From the histogram, the distribution of delivery time is positively skewed and majority of delivery time fall within 20 days.

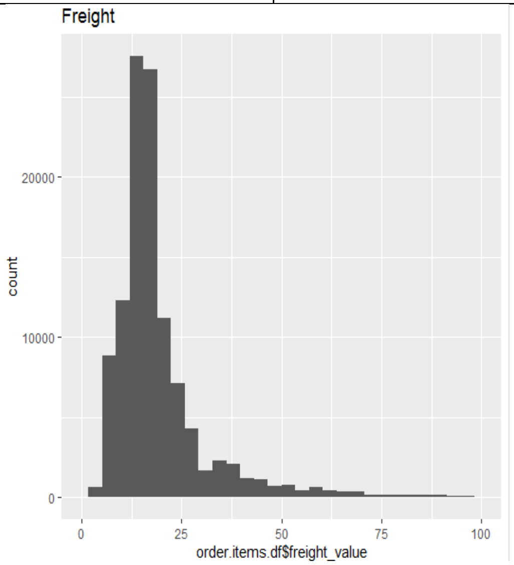


Olist order items dataset:

The order items dataset includes 7 attributes. The attributes order_id, product_id, and seller_id are used for mapping. The attributes price and freight_value are selected for clustering and prediction analysis.

Attributes	Description	Selected or not?
order_id	unique id for orders	Y
order_item_id	unique id for order item	N
product_id	unique id for product	Y
seller_id	unique id for sellers	Y
shipping_limit_date	last shipping date	N
price	selling price	Y
freight_value	freight	Y





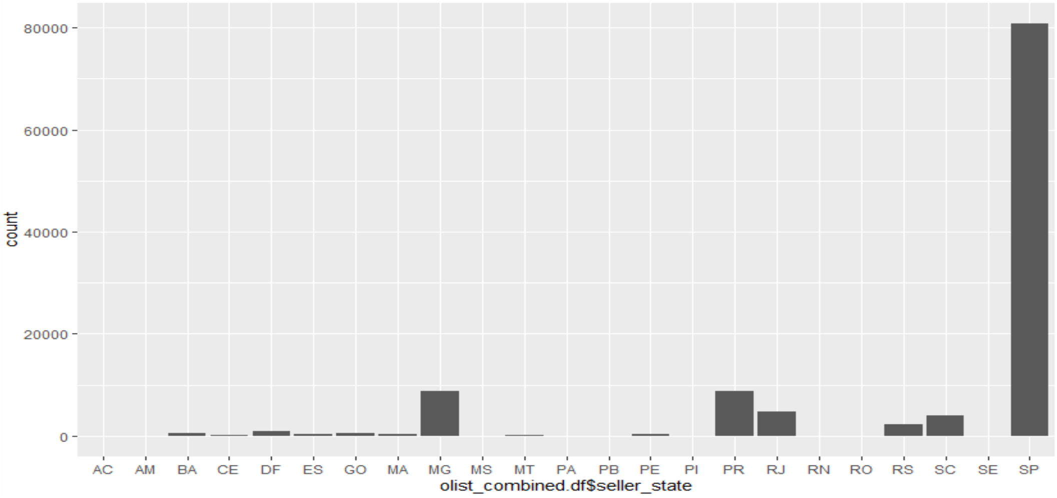
The distribution of price and freight are both positively skewed. Most product prices are lower than $200, and freight values are mainly below $25.

Olist seller dataset:

The seller dataset includes 4 attributes and we selected two of them for the analysis.

Attributes	Description	Selected or not?
seller_id	unique id for sellers	Y
seller_zip_code_prefix	seller zip code	N
seller_city	seller city	N
seller_state	seller state	Y

From the graph, we can see that most of sellers are from SP state.

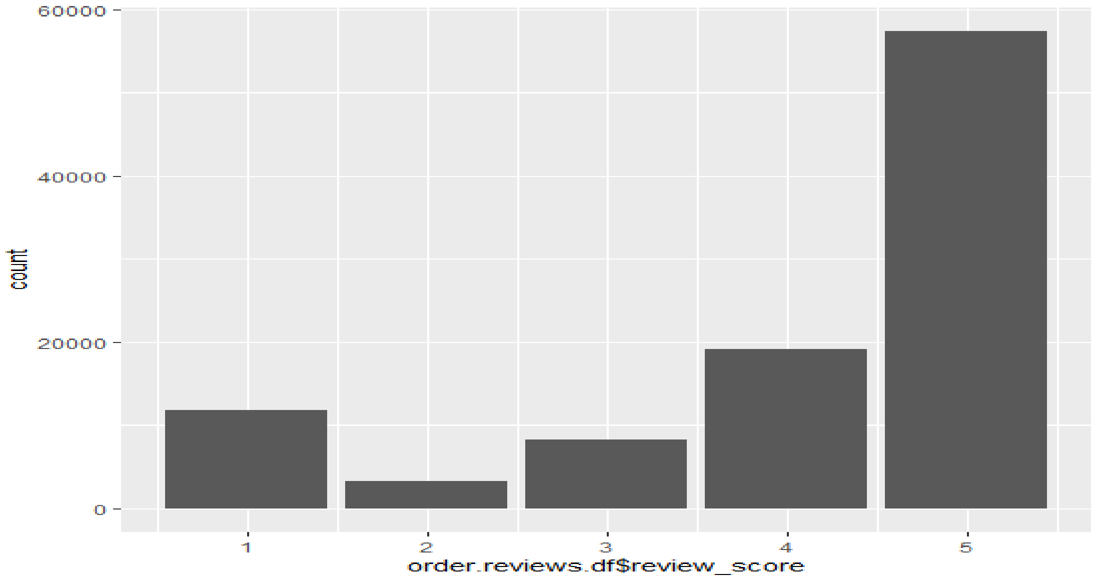


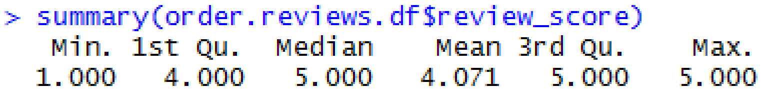
Olist order review dataset:

The order review dataset includes 7 attributes and we selected order_id to merge with other datasets and review_score to predict factors influencing customer satisfaction.

Attributes	Description	Selected or not?
review_id	unique id for customer review	N
order_id	unique id for order	Y
review_score	1 to 5	Y
review_comment_title	comment tile	N
review_comment_message	comments	N
review_creation_date	review date	N
review_answer_timestamp	answer to review time	N

We plot the review score to understand the data distribution as follows:





```
> summary(order.reviews.df$review_score)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   4.000   5.000   4.071   5.000   5.000 
```

The median of review score is 5, which means at least half customers were satisfied with Olist store. The mean of review score is 4.071, suggesting that some extreme review score lowered the mean. Therefore, we will use review score of 5 to explore factors influencing customer review score.

1.4 Data cleaning

- There were some missing values for both numeric and categorical values. We replace the null value of numeric attributes with mean and categorical attributes will not be used in the analysis.
- Merge selected attribute column in separate datasets into one table.

2. Associations

2.1 Objective:

Olist acts as platform to sell products for about 3050 sellers. To encourage new sellers to join Olist, we offer our analytical decisions to them. These business decisions would give them incentive to join and continue their business with Olist. This would ensure client acquisition, retention and indirectly higher profits for Olist.

2.2 Business Explanation:

One such decision offering is new product categories, which our clients (merchants) should expand to. Many merchants typically sell in single or a few product categories. There are total of 22 product categories on Olist platform. We can encourage merchants to expand their product catalog to different categories. With the order details of customers who buy product across different categories and merchants, we can identify such associated product categories. This analysis will enable Olist to find and suggest relevant associated categories to merchants. When merchant expands to these new associated categories, customers can then buy from same merchants. Thus, ensuring sales for merchants and in turn, improves profits for Olist.

2.3 Technical Implementation:

Database has many tables, of which 5 are mainly used in this project. Orders, Order-items, Products, Customers, Sellers are combined to form parent data frame, of which few columns are used in each technical implementation. Apart from this, a name translation table is used for translating categories to English from Portuguese.

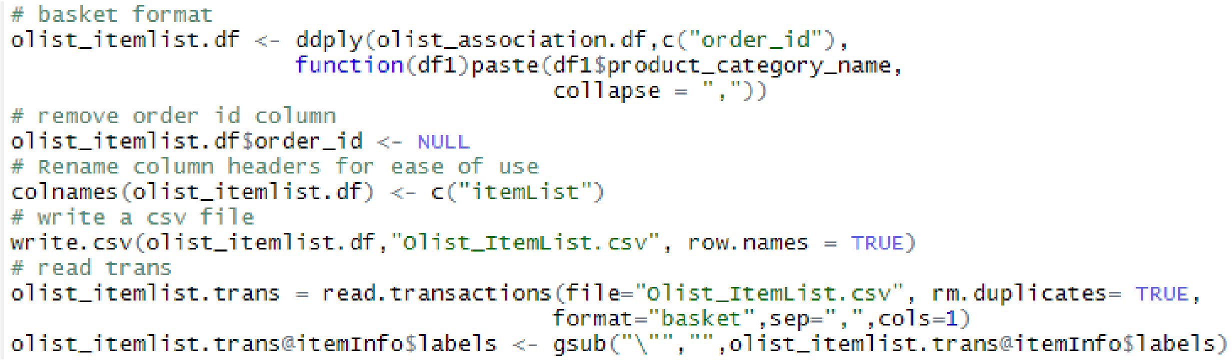
A subset of data frame, with useful columns like Order ID, Customer ID, Product category name, Delivery date is selected. After transforming this set to usable transactions, Apriori function is applied to obtain rules.

Discarded Scripts:

Initial approach was to select column subset data frame of orders and product categories. Transform available data frame to matrix, then transactions for application of Apriori. However, few limitations prevented this implementation.

Limitations	Workaround/ Change in Approach
Order IDs are 32-digit Hex numbers: can't be converted to numerical values	The field was used as string values, sorting was done based on string sort
Product categories field converting dummy matrix: there were 22 categories hence data frame would expand to 22 x 100k which requires 70Gb RAM size	Changed approach from df > matrix > transactions > rules to df > csv file > transactions > rules
Null product category names: Some orders with products not belonging to any product categories	Such orders records were deleted, not considered for associations
Error reading transactions: Transactions were written in basket format to csv file which were not being read in correct format despite correct syntax	Changed approach: csv file written with transactions in single format

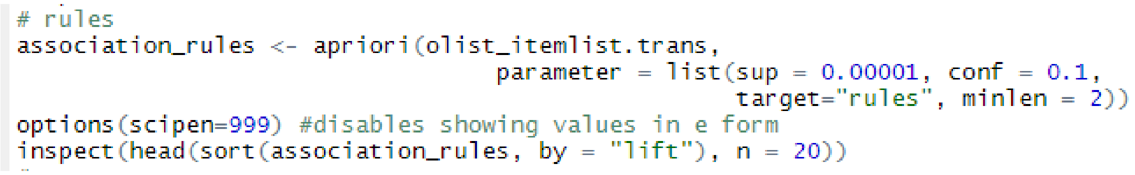
Basket Format script:



```
# basket format
olist_itemlist.df <- ddply(olist_association.df,c("order_id"),
                      function(df1)paste(df1$product_category_name,
                                    collapse = ","))
# remove order id column
olist_itemlist.df$order_id <- NULL
# Rename column headers for ease of use
colnames(olist_itemlist.df) <- c("itemList")
# write a csv file
write.csv(olist_itemlist.df,"Olist_ItemList.csv", row.names = TRUE)
# read trans
olist_itemlist.trans = read.transactions(file="Olist_ItemList.csv", rm.duplicates= TRUE,
                                    format="basket",sep=",",cols=1)
olist_itemlist.trans@itemInfo$labels <- gsub("\"","",olist_itemlist.trans@itemInfo$labels)
```

Successful Scripts:

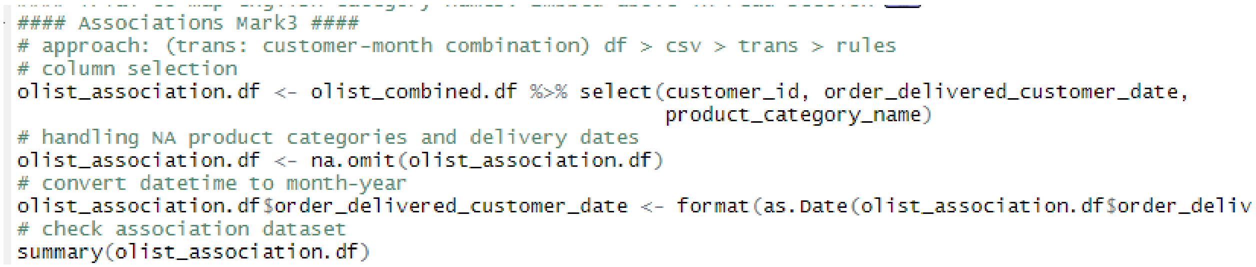
The approach accepted to carry out associations was to take subset from parent data frame, write the transactions in a csv file with single transaction format, read transactions from this csv file. Then these transactions are applied Apriori rules. There are possible variations in selecting subset to get association rules.



```
# rules
association_rules <- apriori(olist_itemlist.trans,
                      parameter = list(sup = 0.00001, conf = 0.1,
                                    target="rules", minlen = 2))
options(scipen=999) #disables showing values in e form
inspect(head(sort(association_rules, by = "lift"), n = 20))
```

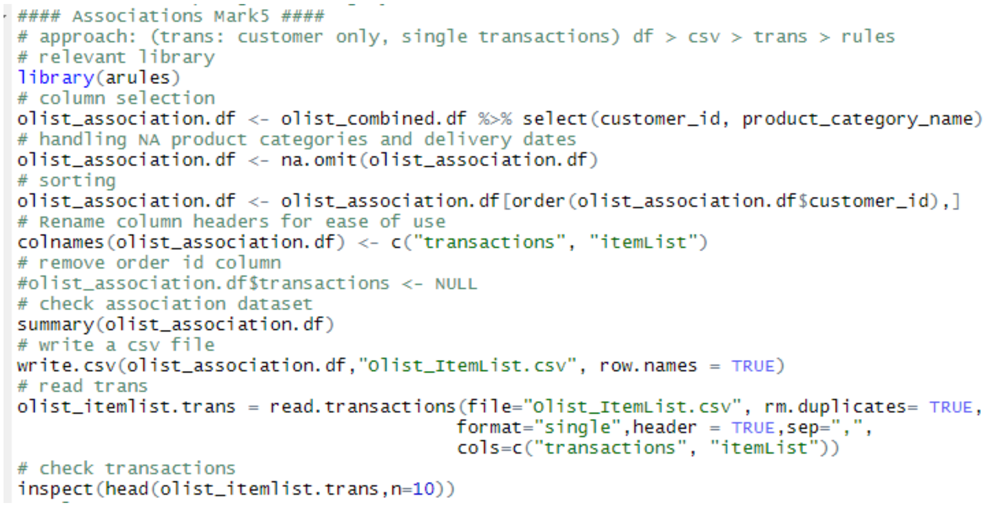
Variations:

- If we consider only single order as transaction, there are really few orders with more than one order items. Thus, association gets low support values and are less reliable.
- We can consider customer – delivery month combination as single transactions, it will show association at customer level who buy from similar categories in one-month duration.



```
#### Associations Mark3 ####
# approach: (trans: customer-month combination) df > csv > trans > rules
# column selection
olist_association.df <- olist_combined.df %>% select(customer_id, order_delivered_customer_date,
                                                product_category_name)
# handling NA product categories and delivery dates
olist_association.df <- na.omit(olist_association.df)
# convert datetime to month-year
olist_association.df$order_delivered_customer_date <- format(as.Date(olist_association.df$order_deliv
# check association dataset
summary(olist_association.df)
```

- Considering each customer as a transaction, helps understand what product categories are bought by customer over time, depicting long-term behavior. This helps to get insight in buying behavior of customers and need of certain product categories.
- Note that total number of records is over 100k, hence having threshold support in 0.0001 range will also yield significant rules, as it would be still applied to over 1000 transactions. At customer level, there are only 2% repeat customers, thus even lower support threshold is required. Correspondingly Lift values will also be high given low support values.



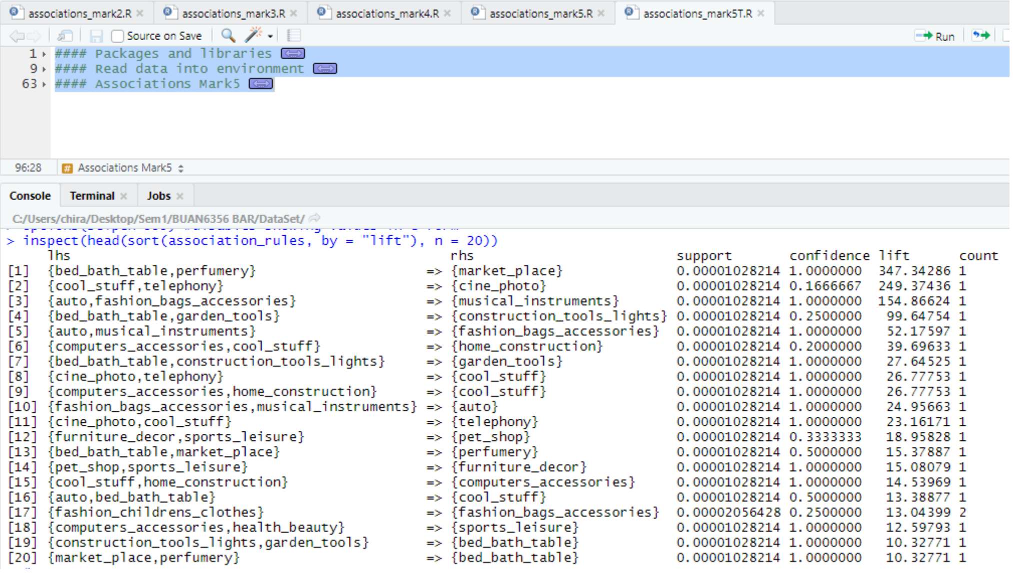
```
#### Associations Mark5 ####
# approach: (trans: customer only, single transactions) df > csv > trans > rules
# relevant library
library(arules)
# column selection
olist_association.df <- olist_combined.df %>% select(customer_id, product_category_name)
# handling NA product categories and delivery dates
olist_association.df <- na.omit(olist_association.df)
# sorting
olist_association.df <- olist_association.df[order(olist_association.df$customer_id),]
# Rename column headers for ease of use
colnames(olist_association.df) <- c("transactions", "itemList")
# remove order id column
#olist_association.df$transactions <- NULL
# check association dataset
summary(olist_association.df)
# write a csv file
write.csv(olist_association.df,"Olist_ItemList.csv", row.names = TRUE)
# read trans
olist_itemlist.trans = read.transactions(file="Olist_ItemList.csv", rm.duplicates= TRUE,
                                         format="single",header = TRUE,sep=",",
                                         cols=c("transactions", "itemList"))

# check transactions
inspect(head(olist_itemlist.trans,n=10))
```

Optimized Solution:

Final rules can be selected from above discussed variations, even combination of them, depending upon business requirements. These requirements can be how recent data closely affects the buying behavior or is a group of customers is to be targeted for marketing campaign, etc. In this case, top three rules on customer level are listed below:

A. {bed_bath_table, perfumery} => {market_place}
B. {cool_stuff, telephony} => {cine_photo}
C. {auto, fashion_bags_accessories} => {musical_instruments}



```
#### Packages and libraries
#### Read data into environment
#### Associations Mark5
```

```
> inspect(head(sort(association_rules, by = "lift"), n = 20))
     lhs                                          rhs                           support      confidence lift      count
[1]  {bed_bath_table,perfumery}                => {market_place}                0.00001028214 1.0000000  347.34286 1
[2]  {cool_stuff,telephony}                    => {cine_photo}                  0.00001028214 0.1666667  249.37436 1
[3]  {auto,fashion_bags_accessories}           => {musical_instruments}         0.00001028214 1.0000000  154.86624 1
[4]  {bed_bath_table,garden_tools}             => {construction_tools_lights}   0.00001028214 0.2500000   99.64754 1
[5]  {auto,musical_instruments}                => {fashion_bags_accessories}    0.00001028214 1.0000000   52.17597 1
[6]  {computers_accessories,cool_stuff}        => {home_construction}           0.00001028214 0.2000000   39.69633 1
[7]  {bed_bath_table,construction_tools_lights} => {garden_tools}               0.00001028214 1.0000000   27.64525 1
[8]  {cine_photo,telephony}                    => {cool_stuff}                  0.00001028214 1.0000000   26.77753 1
[9]  {computers_accessories,home_construction} => {cool_stuff}                  0.00001028214 1.0000000   26.77753 1
[10] {fashion_bags_accessories,musical_instruments} => {auto}                   0.00001028214 1.0000000   24.95663 1
[11] {cine_photo,cool_stuff}                   => {telephony}                   0.00001028214 1.0000000   23.16171 1
[12] {furniture_decor,sports_leisure}          => {pet_shop}                    0.00001028214 0.3333333   18.95828 1
[13] {bed_bath_table,market_place}             => {perfumery}                   0.00001028214 0.5000000   15.37887 1
[14] {pet_shop,sports_leisure}                 => {furniture_decor}             0.00001028214 1.0000000   15.08079 1
[15] {cool_stuff,home_construction}            => {computers_accessories}       0.00001028214 1.0000000   14.53969 1
[16] {auto,bed_bath_table}                     => {cool_stuff}                  0.00001028214 0.5000000   13.38877 1
[17] {fashion_childrens_clothes}               => {fashion_bags_accessories}    0.00002056428 0.2500000   13.04399 2
[18] {computers_accessories,health_beauty}     => {sports_leisure}              0.00001028214 1.0000000   12.59793 1
[19] {construction_tools_lights,garden_tools}  => {bed_bath_table}              0.00001028214 1.0000000   10.32771 1
[20] {market_place,perfumery}                  => {bed_bath_table}              0.00001028214 1.0000000   10.32771 1
```

2.4 Business Implementation:

The rules can be used for making informed business decisions for helping merchants to expand their product catalog into new product category. For example, a merchant who already sells bed, tables and perfumes they can expand to marketplace category. Or merchants selling funk material and telephone accessories should expand to photography department. Expansion into new category helps in business expansion of merchant and increases profits for Olist.

3. Clustering Analysis

Clustering analysis is done to gain further insight to transactions by categories. Clustering analysis is a common method of exploratory data mining. In the analysis the whole dataset of transactions is grouped several groups that are distinctively different from each other. This kind of analysis reveals different kind of groups of transactions and the split of transactions between the different groups. For example, the transaction is divided into premium and basic service and the split between the groups is 75%/25%. This analysis can be done on any of the categories in our data set.

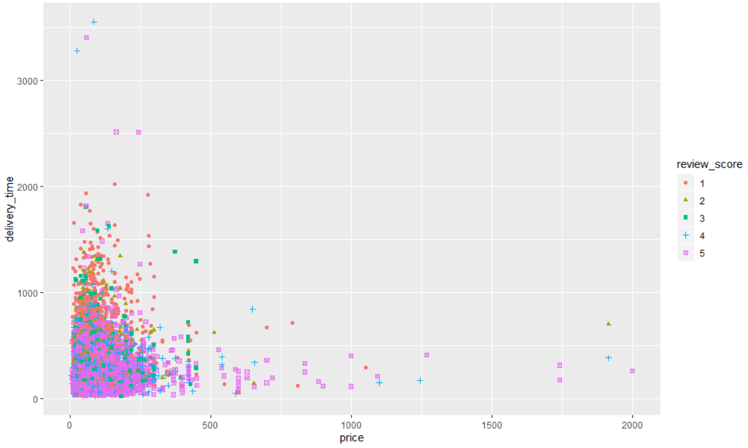
More specifically our clustering analysis looks at a category that is recommended to our client and breaks it down to clusters. For example, if a client is selling home goods and our recommendation is to expand to selling beds too, we will look at the market of bed sales for our client. Then when the client evaluates if it is reasonable to enter this new category, they can already see how the market breaks down and what are the volumes in the different groups. Also, when review score is used as a feature in the analysis the clusters with low review scores can be identified and avoided.

3.1 Preprocessing of data

In order to carry out the clustering analysis correct data needs to be preprocessed and selected from the larger dataset. As the clustering is done within certain category transactions are first filtered to match the specific category. Second delivery time is calculated as a difference between purchase timestamp and the delivery timestamp. Transactions with incomplete information are omitted. Next the analysis can be started as the for every order the selected columns are available. We chose to use price, review score and delivery time as features to gain a rough understanding of the target market.

3.2 Visualization of a category market

In the next figure visualization of example market can be seen. Price is on the x-axis and y-axis is the delivery time. Data points are marked with different colors and shapes to show the review score for each order. For this example, market it seems that the longer the order takes to deliver the more likely it is that the order gets a bad review. Also seem that the price does not seem to play a large role in the rating. Most of the orders with a rating of 4 or 5 can be seen closer to origo which group is the largest by volume in the dataset. Then as the delivery time increases it seem that the orders with a review score of one can be seen there. This visualization of the bed category market is an example of the visualization and similar visualization can be done on any of the product categories.



Caption: Beds Category market

4.3 Technical Implementation

After the exploratory analysis the actual clustering analysis is done for each category. Here we'll look into the beds market once again. First from preprocessed dataset for clustering the transactions related to the category are chosen and the features that will not be used in the analysis are filtered away. As a result of this we end up with a dataset of certain category with the columns of price, review score and delivery time. After selection and filtering the data is scaled using base-R scale function. The analysis should be run multiple times as the number of clusters need to be chosen for each category separately. We found from the results that using a K of 5 made sense for the beds market.

Code snippet: Selecting data and carrying out k-means clustering

```
cat1_data <- clustering_data %>% filter(product_category_name == 'cama_mesa_banho')  %>% select(price, review_score,
delivery_time)
cat1_data <- scale(cat1_data)
km1 <- kmeans(cat1_data, 5)
```

The kmeans function from stats-package runs kmeans clustering on a given data matrix. We give the function the data matrix containing the selected data and the number of K as parameters.

Code snippet: Results of the clustering - Beds category

```
> km1$centers #
       price review_score delivery_time
1 -0.2298155    0.5930267    -0.5087105
2 -0.1740422   -1.4147710    -0.1869851
3  2.0963348    0.3734312    -0.2596247
4 -0.1350948    0.4291760     0.7656352
5  0.0123088   -1.6718130     2.8519468
> km1$withinss
[1] 2211.393 2377.389 4724.550 1538.585 1908.949
> km1$size
[1] 5184 2339  894 2095  597
```

As results of the clustering we look at the centers, withinss and sizes. First from the centers we can see that the number of five different clusters creates distinctive clusters. First cluster is with close to average price, close to average delivery and below average delivery time. Second cluster is close to the average price and delivery time but with significantly below average review score. Third center is with very high price and close to average review and delivery times. Forth center is with close to average price, close to average score but high delivery time. Fifth center is with close to average price but with low review score and very high delivery time. Each of the centers represent a cluster in the target category market. We can see that the third cluster seems to be the premium beds transactions and the first cluster is the basic bed transactions with decent review scores and quick delivery times.

When our client is thinking considering entering new category looking at the clusters and identifying different types of transactions helps them to think about how they should possibly position themselves on the existing market. Entering a saturated market like beds is never easy and the company should be very aware of what kind of customers inside the existing market they wish to serve and with what kind of players they are competing in the market.

Withinss is the within cluster sum of squares. It results in a vector with a number of each cluster. This ratio is good when it is as low as possible for each cluster and to evaluate what is a good number for k sum of all the withinss is useful metric. The withinss is lower when the homogeneity is higher inside a certain cluster. When evaluating these values, it must be remembered that it's the sum for each cluster and the number of observations inside the cluster also effects the total number of this value. So, if the cluster has a lot of values in it - it's expected that the withinss is also larger. It can be for example be seen that the first cluster which has the most observations in it has relatively low withinss meaning that this cluster is able to capture quite well homogeneous transactions. And for example, for the third cluster where there is only 894 observations the withinss is relatively large when compared to other clusters being the largest value of all the clusters being approximately twice that of the first cluster.

The size shows the sizes of each cluster. As mentioned before the first cluster captures the largest amount of transactions and for example the fifth cluster (where something went wrong with delivery) captures only 597 of the observations.

From these numbers we can see how the market splits up between different types of transactions and calculate the relative share of the market.

Name (suggestions)	Cluster	# observations	Share of the market
The mode transaction	1	5184	47%
Bad reviews with okay delivery and average price	2	2339	21%
High price order	3	894	8%
Average with longer delivery	4	2095	19%
Delivery fail	5	507	5%

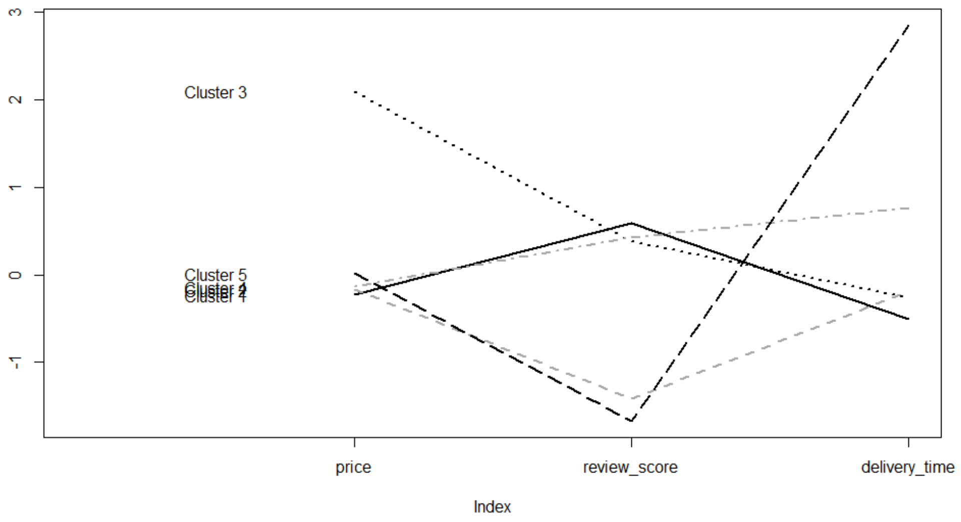
3.4 Visualization of the centroids

As per usual the different cluster are wanted to be visualized in an understandable way. We have done centroid plotting which quickly show how the different clusters are relative to each other. Below here is the code used to do the centroid plotting and an image of the resulting plot.

Code snippet: Centroid plotting

```
# category one
plot(c(0), xaxt = 'n', ylab = "", type = "l",
     ylim = c(min(km1$centers), max(km1$centers)), xlim = c(0, 3))
axis(1, at = c(1:3), labels = colnames(clustering_data))
for (i in c(1:5))
  lines(km1$centers[i,], lty = i, lwd = 2, col = ifelse(i %in% c(1, 3, 5),
                                                     "black", "dark grey"))
text(x = 0.5, y = km1$centers[, 1], labels = paste("Cluster", c(1:5)))
```

Image Caption: Centroid plot of the category one market



3.5 Summary

Clustering is done a category that is recommended to our client in order to gain insights to that category markets. Transactions in the market are broken down to clusters and the clusters and their shares of the total markets are analyzed. First data is selected for the analysis, second kmeans algorythm is run on the data and lastly the resuts are analyzed and knowledge of the target market is shared with our client. In this way the target market is already roughly analyzed when the client starts to think about entering it.

4. Decision Tree

4.1 Objective and business explanation of the Decision Tree

Another business question we wanted to answer was “What would lead to a high customer satisfaction for products?” It is important to analyze this problem as it directly affects the business outcomes we are interested in, which is higher product sales, a metric that we desire in order to satisfy our business stakeholders. Because customer satisfaction can be quantitatively measured by the reviews of the products, we must understand the relevant predictors that would optimize this measure.

There are several assumptions that must hold for us to care about product reviews. First, we must assume that the consumer is a rational entity who optimizes utility maximization. Following from this, we must also assume that consumer behavior is rational as well. Based on these two assumptions, we can presume that the customer will choose to read reviews generated by other previous users of the product before they make their purchase decisions on their own. As a result, the review ratings will influence whether the consumer chooses to make a purchase.

4.2 Preprocessing and Technical Requirements

To proceed with the decision tree, we had to gather the most relevant variables and aggregate them into one unified dataset. We extracted the freight_value variable in the order items dataframe, the order_purchase_timestamp and order_delivered_customer_date from the orders dataframe, and the review_score from the order.reviews dataframe. Next, we connected these columns with the order_id key so that the values of the products would align.

Lastly, we assigned the variables as the target and the predictors to be used for classification. We set the target variable to be the review score and the relevant predictors to be the freight value and the delivery time. Since the target variable was not in categorical form, we had to convert it initially to make sure that it was suitable for classification. The review scores were ranked from a scale of one to five, and scores that were rated a five were labeled as a yes response, and any scores lower than that were labeled with a no response. To get the delivery time variable, we had to find the difference between two datetime columns, the datetime of the product order and the datetime the delivery was registered. We expressed delivery units in terms of days. To deal with NA, we imputed the mean values.

Code snippet:

```
#Create new column delivery_time
delivery_time <- difftime(orders.df$order_delivered_customer_date,
orders.df$order_purchase_timestamp,units='days')
delivery_time <- as.data.frame(delivery_time)
orders.df['delivery_time'] <- delivery_time
#replace NA values with the mean

olist_combined.df$delivery_time <- replace_na(olist_combined.df$delivery_time, mean(olist_combined.df$delivery_time,
na.rm = TRUE))
```

4.3 Technical Implementation

Dataset split

Next, we split the data into training and validation set with a split ratio of 70/30. Although it is a good rule of thumb to split 80/20, we saw that the performance of the model was not affected in any way as these were not extreme split percentages that could potentially introduce bias.

Code snippet:

```
n <- nrow(olist_combined.df)
n_olist <- round(0.7 * n)
set.seed(123)
olist_indices <- sample(1:n, n_olist)
train <- olist_combined.df[olist_indices, ]
test <- olist_combined.df[-olist_indices, ]
```

Model fitting and parameters used

To fit the data into the model, we used four relevant parameters of the rpart function: minbucket, minsplit, cp, and maxdepth. Each parameter has a strict definition and changing the values of these parameters can alter the structure of the tree. The value of the minbucket parameter states the minimum number of observations that are allowed in the terminal node. The value of the minsplit parameter requires the minimum number of observations in the parent node for further splitting. The value of the cp states the minimum improvement in the model that are needed at each node. The value of the maxdepth parameter prevents the growth of branches of the tree.

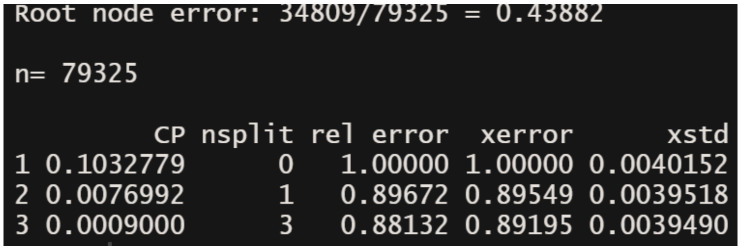
Code Snippet:

```
model <- rpart(review_score ~.,
            data = train, minbucket=8000,minsplit=20000,cp=0.0009, maxdepth=3,
            method = "class")#acc: 0.6133, 4 leaves
model <- rpart(review_score ~.,
            data = train, minbucket=500,minsplit=1000,cp=0.001,
            method = "class",)#acc: 0.6273, 5 leaves
model <- rpart(review_score ~.,
            data = train, minbucket=0,minsplit=0,cp=0.00167,
            method = "class",)#acc: 0.6273
model <- rpart(review_score ~.,
            data = train, minbucket=0,minsplit=0,cp=0.0009,
            method = "class",maxdepth=7) #acc: 0.6273
model <- rpart(review_score ~.,
            data = train, minbucket=1000,minsplit=10000,cp=0.001,
            method = "class",) #acc: 0.6273
```

Pruning

Pruning was a technique that we also used to reduce the size of the decision tree in order to avoid overfitting issues and at the same time improve predictive accuracy. However, in our case, pruning did not do much help to improve the accuracy as the cp value at 0.0009 was already optimized at the smallest tree having the smallest cross validation error.

Image caption: pruning



```
Root node error: 34809/79325 = 0.43882

n= 79325

         CP nsplit rel error  xerror      xstd
1 0.1032779      0   1.00000 1.00000 0.0040152
2 0.0076992      1   0.89672 0.89549 0.0039518
3 0.0009000      3   0.88132 0.89195 0.0039490
```

Code snippet:

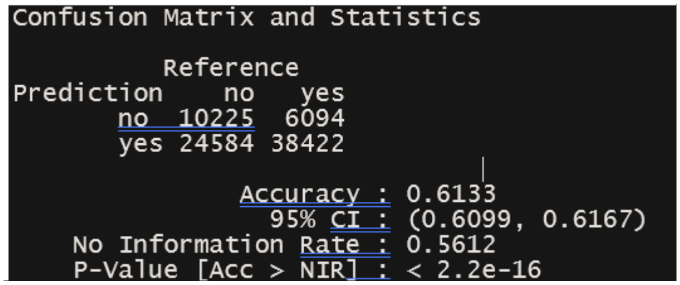
```
printcp(model)
bestcp <- model$cptable[which.min(model$cptable[,"xerror"]),"CP"]
pruned <- prune(model, cp=bestcp)
rpart.plot(pruned,type=1,extra=2,under=TRUE,tweak=1.2)
```

Results of the decision tree

The parameter values for this model were the following: minbucket = 8000, minsplit = 20000, cp = 0.0009, and maxdepth = 3. After much parameter tuning, the values for minbucket and minsplit were arbitrary as changing these values had no effect on the accuracy rate but only the tree structure. However, changing the cp did influence both the accuracy and the structure of the tree.

The confusion matrix gave us an accuracy rate of 0.6133. As mentioned previously, further tuning the values of the parameters did not drastically alter the rates so it was not helpful to change the usage of the models.

Image caption: confusion matrix



```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  10225  6094
       yes 24584 38422

               Accuracy : 0.6133
                 95% CI : (0.6099, 0.6167)
    No Information Rate : 0.5612
    P-Value [Acc > NIR] : < 2.2e-16
```

Code snippet:

```
model.pred.train <- predict(model,train,type = "class")

confusionMatrix(model.pred.train, as.factor(train$review_score))
```

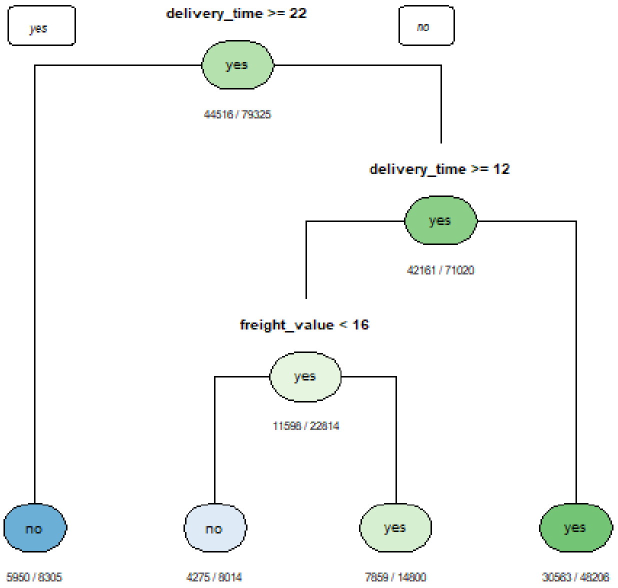
4.4 Visualization and Interpretation

The classification tree shows us the delivery time above the root and internal node, the freight value above the internal node, and four leaves at the bottom. Based on the responses of the leaves shown below, we can interpret what they mean in order to extract relevant business insights.

The first and the fourth leaves are very intuitive – since they are products based on their delivery times, we can reasonably say that those that have been delivered within the expectations of a time frame for the consumer would have higher ratings. From the tree, we confirm this hypothesis and see that products that took longer than 22 days received a low rating, while those take took less than 12 days received a perfect score. Although this is not a very interesting insight, we are given a specific range value that can potentially resolve the logistical issues that merchants might face.

For the second and third leaves, we see that product is now associated with delivery time and freight value. For this interpretation, we assume that freight value is a function of the weight and size of the product, and the distance from the consumer's house. If this holds true and given the no response in the second leaf, we can say that customer expects faster deliveries for products that are smaller or lighter or those that are closer to the customer's home. Oppositely, we can interpret the third leaf as customers who are more patient with their deliveries when they are larger, heavier, or farther away. This is a more interesting insight since we have associated the product with both predictor variables. The stakeholder will possibly act on the insight by making the necessary adjustments.

Image caption: classification decision tree



Code snippet:

```
rpart.plot(model,type=1,extra=2,under=TRUE,tweak=1.2)
```

4.5 Business Implementation and Potential Recommendations

Based on the results of the decision tree, we can directly try to help merchants and indirectly improve Olist sales. To do this, we can optimize the logistics of delivering products that are in the categories that we are looking to expand and advise merchants to sell more high rated products based on these range values. To generate further revenue growth, we can also tell them to curtail selling products that are not within this range or adjust their logistics so that more products on the no response side could be converted into a yes response.

One recommendation would be to use a different shipping provider that guarantees faster deliveries with little or no change in costs. Another recommendation would be to discriminate product deliveries based on shipping services so that products that are relatively smaller or lighter would be assigned with a faster shipping service.

By doing this, we can have a greater basket of products offered by the merchants that have review scores in the high rating. This would satisfy the merchants selling online, Olist Executives, investors or other relevant stakeholders who are looking for the end goal of sustainable business outcomes such as revenue growth.