IDS572-HW5

Kritika Raghuwanshi;Shixun Jiang;Yaze Gao

2023-05-05

Problem 4

Loading Data

Before we get started with any of the parts in problem 4, we need to load the different pages of the Champo Carpets Excel sheet. We do that as follows:

```
library("pacman")
library("tidyverse")
library("rpart")
library("rpart.plot")
library("readxl")

RawDataOrderSample <- read_xlsx("C:/Champo Carpets.xlsx", sheet = 2)
DataOrderOnly <- read_xlsx("C:/Champo Carpets.xlsx", sheet = 3)
DataOnSampleOnly <- read_xlsx("C:/Champo Carpets.xlsx", sheet = 4)
DataForRecommendation <- read_xlsx("C:/Champo Carpets.xlsx", sheet = 5)
DataAssociationRules <- read_xlsx("C:/Champo Carpets.xlsx", sheet = 7)

head(RawDataOrderSample)
```

```
## # A tibble: 6 x 16
##   OrderType OrderC~1 Custo~2 Count~3 Custo~4 Custorderdate       UnitN~5 QtyRe~6
##   <chr>     <chr>    <chr>   <chr>   <chr>   <dttm>              <chr>     <dbl>
## 1 Area Wise Order    H-1     USA     1873354 2017-01-16 00:00:00 Ft            2
## 2 Area Wise Order    H-1     USA     1873354 2017-01-16 00:00:00 Ft            2
## 3 Area Wise Order    H-1     USA     1873354 2017-01-16 00:00:00 Ft            2
## 4 Area Wise Order    H-1     USA     1918436 2017-02-01 00:00:00 Ft            5
## 5 Area Wise Order    H-1     USA     1873354 2017-01-16 00:00:00 Ft            5
## 6 Area Wise Order    H-1     USA     1918436 2017-02-01 00:00:00 Ft            4
## # ... with 8 more variables: TotalArea <dbl>, Amount <dbl>, ITEM_NAME <chr>,
## #   QualityName <chr>, DesignName <chr>, ColorName <chr>, ShapeName <chr>,
## #   AreaFt <dbl>, and abbreviated variable names 1: OrderCategory,
## #   2: CustomerCode, 3: CountryName, 4: CustomerOrderNo, 5: UnitName,
## #   6: QtyRequired
```

Cleaning the Data Set "RawDataOrderSample"

Now that we're done loading the data into R using the different sheets of the Excel file, we can proceed to the next step of cleaning the data. This is needed to ensure we're working on a data set devoid of any **"NA"** values, which has the potential of skewing our models we will be making further. First, we start with the **"RawDataOrderSample"** data set:

```r
ROS <- RawDataOrderSample #making a copy

#converting values from character categorical to factor

ROS$OrderType <- as.factor(ROS$OrderType)
ROS$OrderCategory <- as.factor(ROS$OrderCategory)
ROS$CustomerCode <- as.factor(ROS$CustomerCode)
ROS$CountryName <- as.factor(ROS$CountryName)
ROS$UnitName <- as.factor(ROS$UnitName)
ROS$ITEM_NAME <- as.factor(ROS$ITEM_NAME)
ROS$QualityName <- as.factor(ROS$QualityName)
ROS$DesignName <- as.factor(ROS$DesignName)
ROS$ColorName <- as.factor(ROS$ColorName)
ROS$ShapeName <- as.factor(ROS$ShapeName)

str(ROS)
```

```
## tibble [18,955 x 16] (S3: tbl_df/tbl/data.frame)
##  $ OrderType      : Factor w/ 2 levels "Area Wise","Pc Wise": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OrderCategory  : Factor w/ 2 levels "Order","Sample": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CustomerCode   : Factor w/ 46 levels "A-11","A-6","A-9",..: 19 19 19 19 19 19 19 19 19 19 ...
##  $ CountryName    : Factor w/ 15 levels "AUSTRALIA","BELGIUM",..: 15 15 15 15 15 15 15 15 15 15 ...
##  $ CustomerOrderNo: chr [1:18955] "1873354" "1873354" "1873354" "1918436" ...
##  $ Custorderdate  : POSIXct[1:18955], format: "2017-01-16" "2017-01-16" ...
##  $ UnitName       : Factor w/ 5 levels "Ft","INCH","Mtr",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ QtyRequired    : num [1:18955] 2 2 2 5 5 4 6 16 2 4 ...
##  $ TotalArea      : num [1:18955] 6 9 54 54 71.2 ...
##  $ Amount         : num [1:18955] 12 18 108 270 356 ...
##  $ ITEM_NAME      : Factor w/ 12 levels "-","DOUBLE BACK",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ QualityName    : Factor w/ 382 levels "D.B 30C H/S LEFA VISCOSE+45C WOOL",..: 300 300 300 300 300
##  $ DesignName     : Factor w/ 2254 levels "1 LOOP/1 CUT",..: 1793 1793 1793 1793 1793 1793 1793 1793
##  $ ColorName      : Factor w/ 815 levels "0620+18-1239",..: 125 125 125 125 125 125 385 385 385 385
##  $ ShapeName      : Factor w/ 5 levels "OCTAGON","OVAL",..: 3 3 3 3 3 3 3 3 3 4 4 ...
##  $ AreaFt         : num [1:18955] 6 9 54 54 71.2 ...
```

We have one variable within the RawDataOrderSample data set which has POSIX (UTC) format date, but is of no use to us, as we cannot form a decision tree with it.

```r
ROS <- select(ROS, -c('Custorderdate'))
```

Next up, we check if there are any NA values in the RawDataOrderSample data set. Upon running the is.na command, we notice that the variable **CustomerOrderNo** is the only variable with NA values - 9 of them. In order to replace those NA values with the most occurring value in that column, we write a Mode function, and then finally replace the NA value with the mode value as follows:

```
sum(is.na(ROS))#before removing NAs
```

```
## [1] 9
```

```
Modes <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}

mode_con <- Modes(ROS$CustomerOrderNo)
mode_con #this is the most occurring value in the column
```

```
## [1] "12985"
```

```
ROS$CustomerOrderNo <- replace(ROS$CustomerOrderNo, is.na(ROS$CustomerOrderNo),
                               mode_con)

sum(is.na(ROS))#after removing NAs
```

```
## [1] 0
```

Cleaning the Data Set "DataOnSampleOnly"

When we check the is.na command against this data set, we see there are 273 NAs, the spread of which is as follows:

```
DOS <- DataOnSampleOnly #making a copy
sum(is.na(DOS))
```

```
## [1] 273
```

```
colSums(is.na(DOS))
```

```
##     CustomerCode      CountryName              USA               UK
##                0                0               39               39
##            Italy          Belgium          Romania        Australia
##               39               39               39               39
##            India      QtyRequired        ITEM_NAME      Hand Tufted
##               39                0                0                0
##            Durry      Double Back       Hand Woven          Knotted
##                0                0                0                0
##         Jacquard         Handloom            Other        ShapeName
##                0                0                0                0
##              REC            Round           Square           AreaFt
##                0                0                0                0
## Order Conversion
##                0
```

Based on the results, we take those variables that don't have any NA values and convert them into factors, to make it easy for us to build our models later:

```
DOS$CustomerCode <- as.factor(DOS$CustomerCode)
DOS$CountryName <- as.factor(DOS$CountryName)
#DOS$QtyRequired <- as.factor(DOS$QtyRequired)
DOS$ITEM_NAME <- as.factor(DOS$ITEM_NAME)
DOS$`Hand Tufted` <- as.factor(DOS$`Hand Tufted`)
DOS$Durry <- as.factor(DOS$Durry)
DOS$`Double Back` <- as.factor(DOS$`Double Back`)
DOS$`Hand Woven` <- as.factor(DOS$`Hand Woven`)
DOS$Knotted <- as.factor(DOS$Knotted)
DOS$Jacquard <- as.factor(DOS$Jacquard)
DOS$Handloom <- as.factor(DOS$Handloom)
DOS$Other <- as.factor(DOS$Other)
DOS$ShapeName <- as.factor(DOS$ShapeName)
DOS$REC <- as.factor(DOS$REC)
DOS$Round <- as.factor(DOS$Round)
DOS$Square <- as.factor(DOS$Square)
#DOS$AreaFt <- as.factor(DOS$AreaFt)
```

Variable **OrderConversion** can be considered as our **"target"** variable, so we will convert this into "YES" or "NO", depending on the values.

```
library(knitr)
DOS$target <- as.factor(ifelse(DOS$`Order Conversion` == 1,
                        "YES", "NO"))
kable(table(DOS$target))
```

Var1	Freq
NO	4651
YES	1169

```
DOS <- DOS[,-c(25)]#removal of Order Conversion due to new "target" variable
```

Now that we're done with converting those variable to factors that do not have any NA values, we shift our focus to those that have NA values and replace those NAs with the mode

1. Removing NAs from the "USA" variable:

```
sum(is.na(DOS$USA))#before removing NAs
```

```
## [1] 39
```

```
mode_USA <- Modes(DOS$USA)
DOS$USA <- replace(DOS$USA, is.na(DOS$USA), mode_USA)

sum(is.na(DOS$USA))#after removing NAs
```

```
## [1] 0
```

2. Removing NAs from the "UK" variable:

```
sum(is.na(DOS$UK))#before removing NAs
```

```
## [1] 39
```

```
mode_UK <- Modes(DOS$UK)
DOS$UK <- replace(DOS$UK, is.na(DOS$UK), mode_UK)

sum(is.na(DOS$UK))#after removing NAs
```

```
## [1] 0
```

3. Removing NAs from the "Italy" variable:

```
sum(is.na(DOS$Italy))#before removing NAs
```

```
## [1] 39
```

```
mode_Italy <- Modes(DOS$Italy)
DOS$Italy <- replace(DOS$Italy, is.na(DOS$Italy), mode_Italy)

sum(is.na(DOS$Italy))#after removing NAs
```

```
## [1] 0
```

4. Removing NAs from the "Belgium" variable:

```
sum(is.na(DOS$Belgium))#before removing NAs
```

```
## [1] 39
```

```
mode_Belgium <- Modes(DOS$Belgium)
DOS$Belgium <- replace(DOS$Belgium, is.na(DOS$Belgium), mode_Belgium)

sum(is.na(DOS$Belgium))#after removing NAs
```

```
## [1] 0
```

5. Removing NAs from the "Romania" variable:

```
sum(is.na(DOS$Romania))#before removing NAs
```

```
## [1] 39
```

```
mode_Romania <- Modes(DOS$Romania)
DOS$Romania <- replace(DOS$Romania, is.na(DOS$Romania), mode_Romania)

sum(is.na(DOS$Romania))#after removing NAs
```

```
## [1] 0
```

6. Removing NAs from the "Australia" variable:

```
sum(is.na(DOS$Australia))#before removing NAs
```

```
## [1] 39
```

```
mode_Australia <- Modes(DOS$Australia)
DOS$Australia <- replace(DOS$Australia, is.na(DOS$Australia), mode_Australia)

sum(is.na(DOS$Australia))#after removing NAs
```

```
## [1] 0
```

7. Removing NAs from the "India" variable:

```
sum(is.na(DOS$India))#before removing NAs
```

```
## [1] 39
```

```
mode_India <- Modes(DOS$India)
DOS$India <- replace(DOS$India, is.na(DOS$India), mode_India)

sum(is.na(DOS$India))#after removing NAs
```

```
## [1] 0
```

Finally, if we check our data set "DataOnSampleOnly", we should not see any NAs

```
sum(is.na(DOS))
```

```
## [1] 0
```

Part A

Exploratory Data Analysis

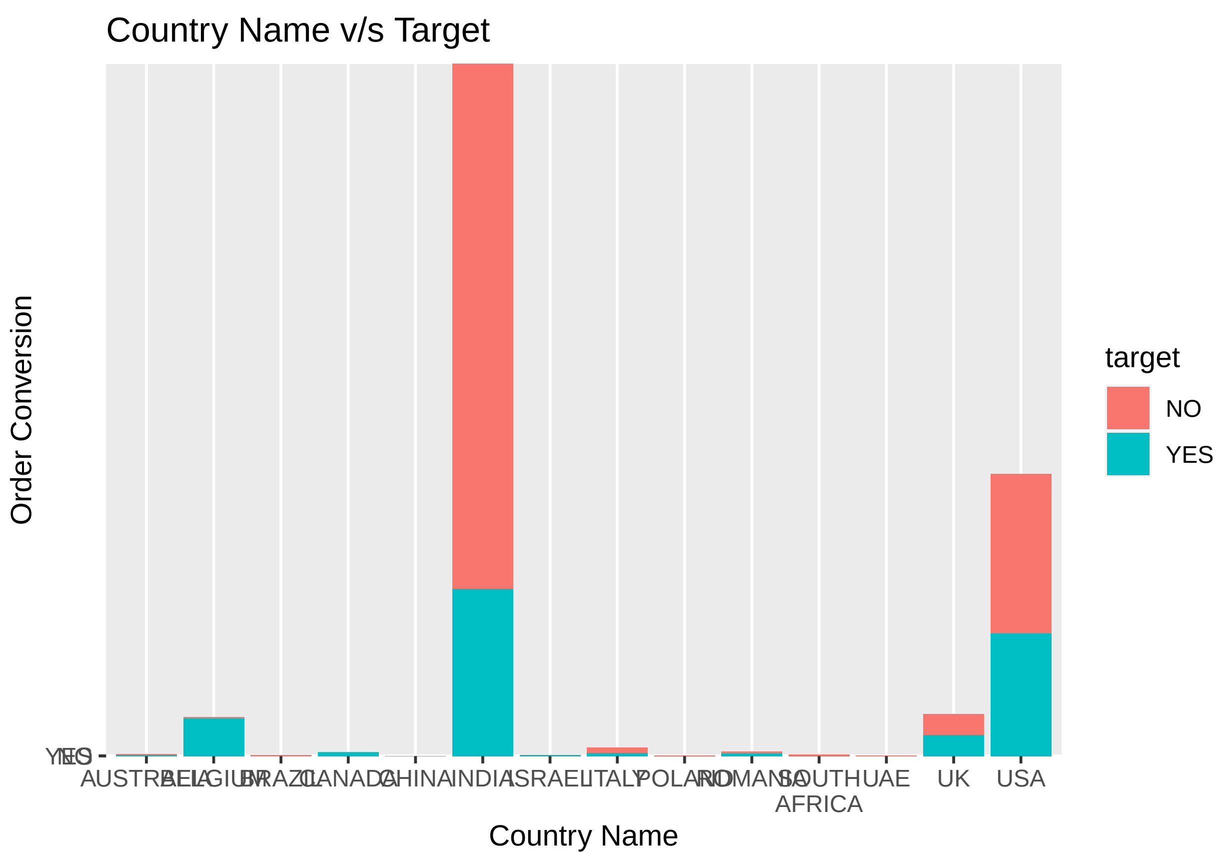
Q. With the help of data visualization, provide key insights using exploratory data analysis

A. In order to perform data visualization, we chose two different data sets - the **Data On Sample Only** data set and the **Raw Data-Order and Sample** data set.

EDA on Data on Sample Only Data Set

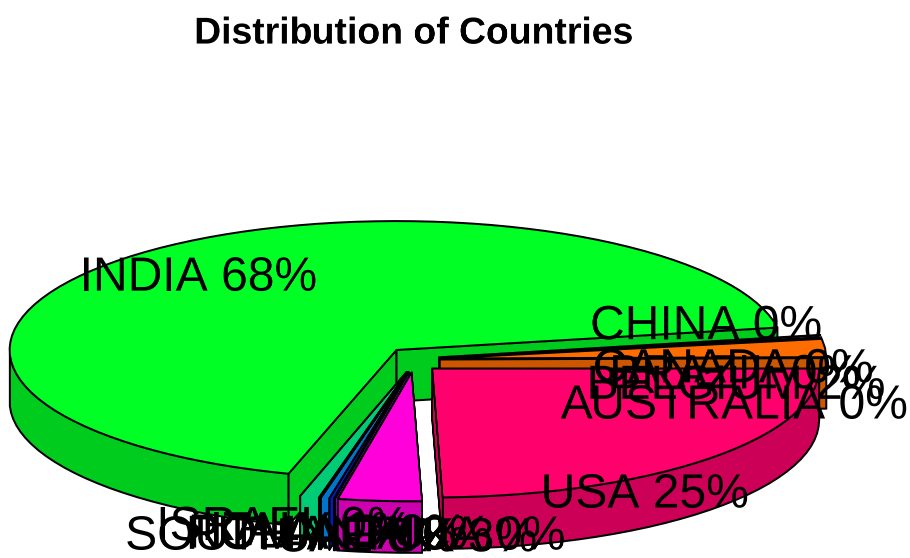
1. CountryName v/s Target

```
library(ggplot2)
library(scales)
ggplot(DOS, aes(CountryName, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Country Name v/s Target", x = "Country Name",
         y= "Order Conversion")
```



Pie chart for country distribution

```
library(ggplot2)
library(plotrix)
pietable <- table(DOS$CountryName)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Countries", radius = 1.5)
```



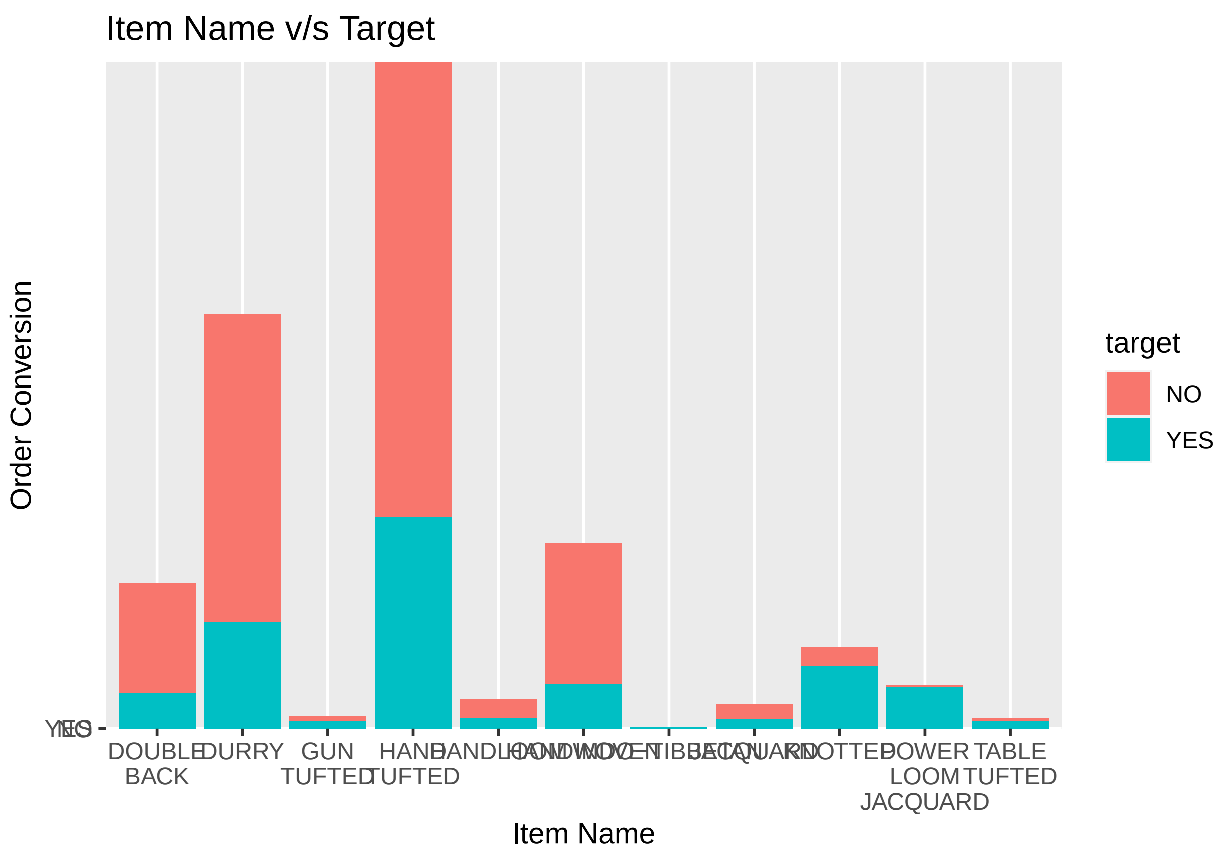
Analysis

From the first chart, we notice that **India** has the highest ratio of order conversion among all the countries, but they also have the highest numbers of orders that didn't convert or materialize. The next best country is **USA**, who also have a similar story, with more orders not getting converted compared to orders getting converted. The country that has more orders getting converted v/s not is **Belgium**.

For this reason, we pull a pie chart of the distribution of countries. The pie chart shows us that the above result is due to **India** occupying **68%** of the distribution, while the rest of the countries don't have much spread across the data set. **USA** is at **25%**, which explains the theory of them being next best. Finally, out of the **2%** distribution Belgium has, Champo Carpets are successful in selling to them the most.

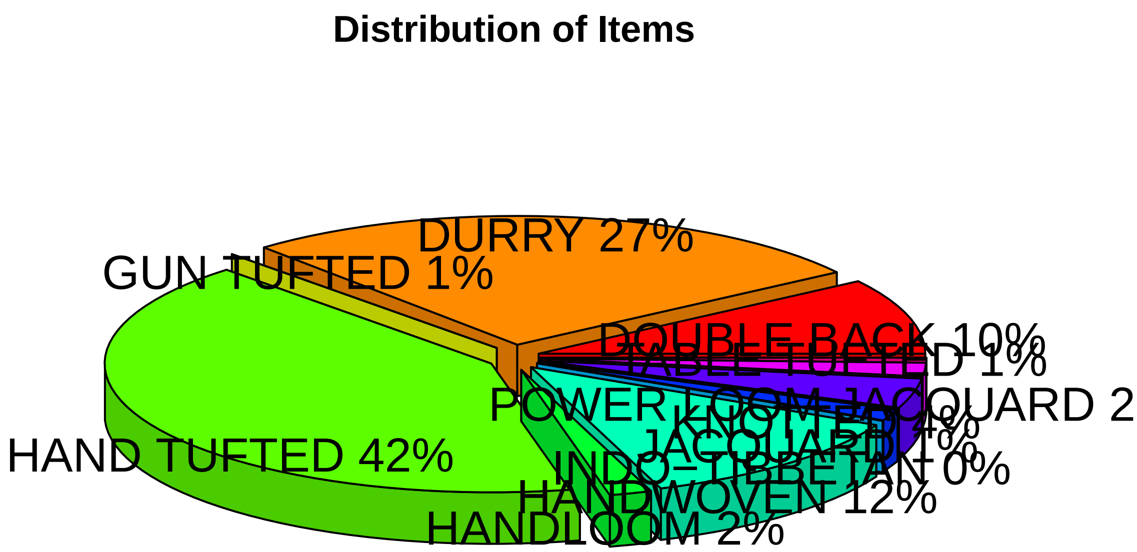
2. ITEM_NAME v/s Target

```
ggplot(DOS, aes(ITEM_NAME, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Target", x = "Item Name",
         y= "Order Conversion")
```



Pie chart for item distribution

```
pietable <- table(DOS$ITEM_NAME)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Items", radius = 1.5)
```



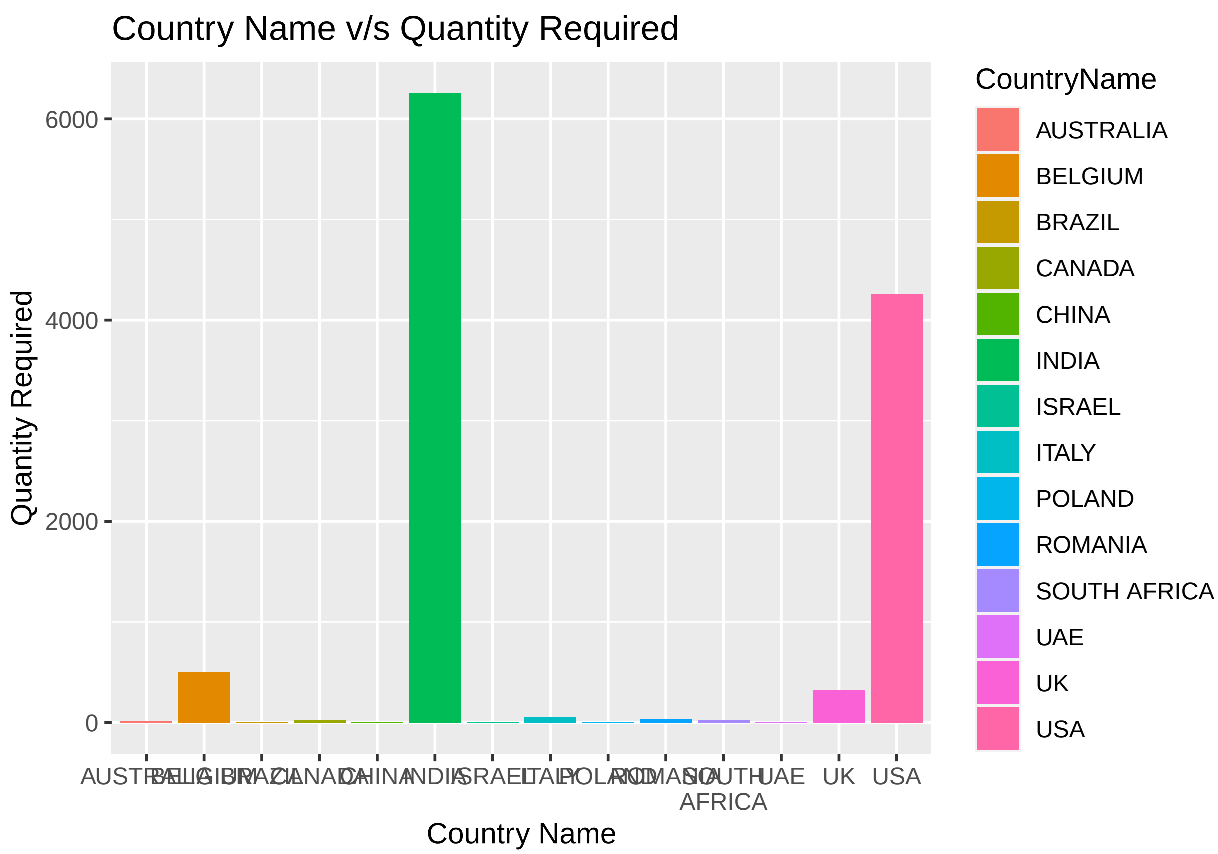
Analysis

From the first chart, we notice that **Hand Tufted** item has the highest ratio of order conversion among all the items, but it also has the highest numbers of orders that didn't convert or materialize. The next best item is **Durry**, that has a similar story, with more orders not getting converted compared to orders getting converted. The item that has more orders getting converted v/s not is **Power Loom Jacquard**.

For this reason, we pull a pie chart of the distribution of items. The pie chart shows us that the above result is due to **Hand Tufted** occupying **42%** of the distribution, while the rest of the items don't have much spread across the data set. **Durry** is at **27%**, which explains the theory of them being next best. Finally, out of the **2%** distribution that Power Loom Jacquard has, Champo Carpets are successful in selling it the most.

3. CountryName v/s QtyRequired

```
ggplot(DOS, aes(CountryName, QtyRequired, fill = CountryName)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Country Name v/s Quantity Required", x = "Country Name",
         y= "Quantity Required")
```

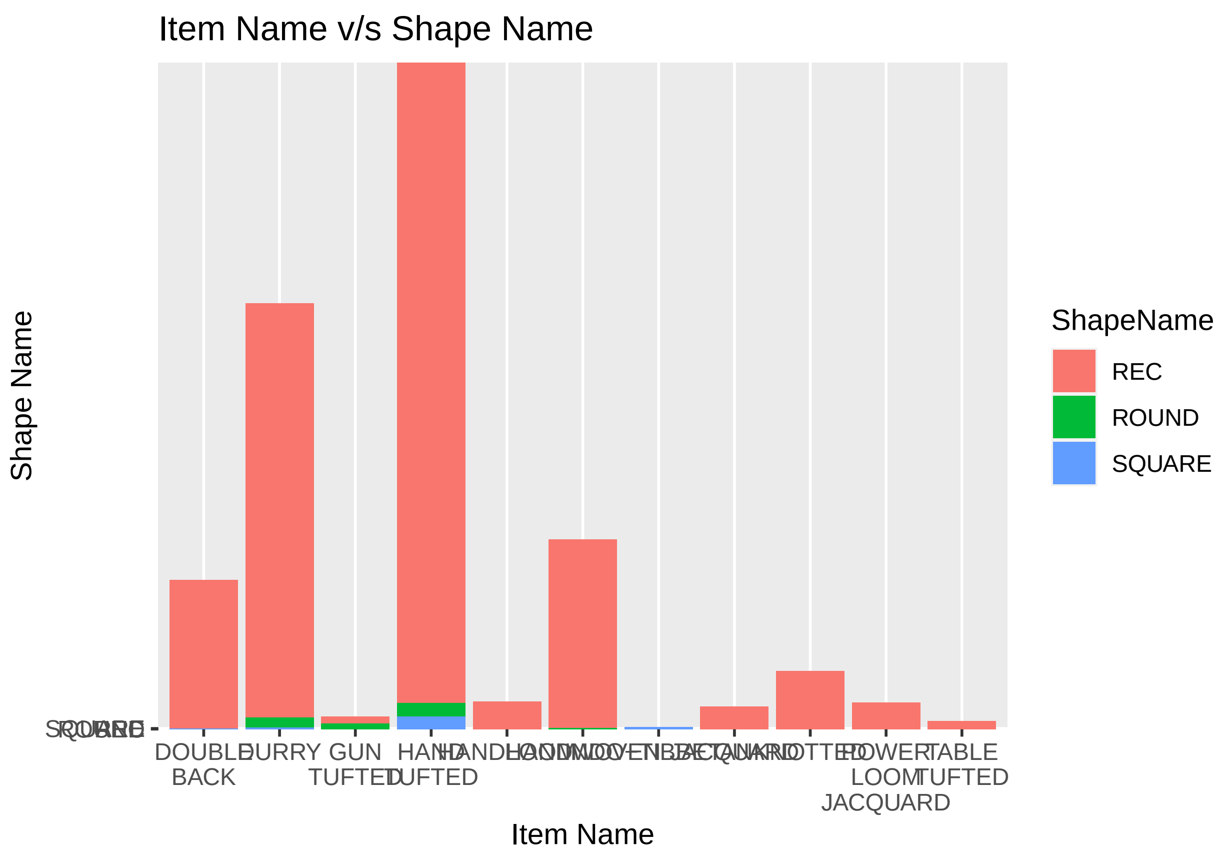


Analysis

From the above chart, we again notice that **India** requires the highest number of quantity among all the countries, but as we noticed from the above charts, not all those get converted into an order, so even though India needs more, they don't end up ordering more. Similarly, **USA** is the next best, followed by **Belgium** and **UK**.

4. ITEM_NAME v/s ShapeName

```r
ggplot(DOS, aes(ITEM_NAME, ShapeName, fill = ShapeName)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Shape Name", x = "Item Name",
         y= "Shape Name")
```

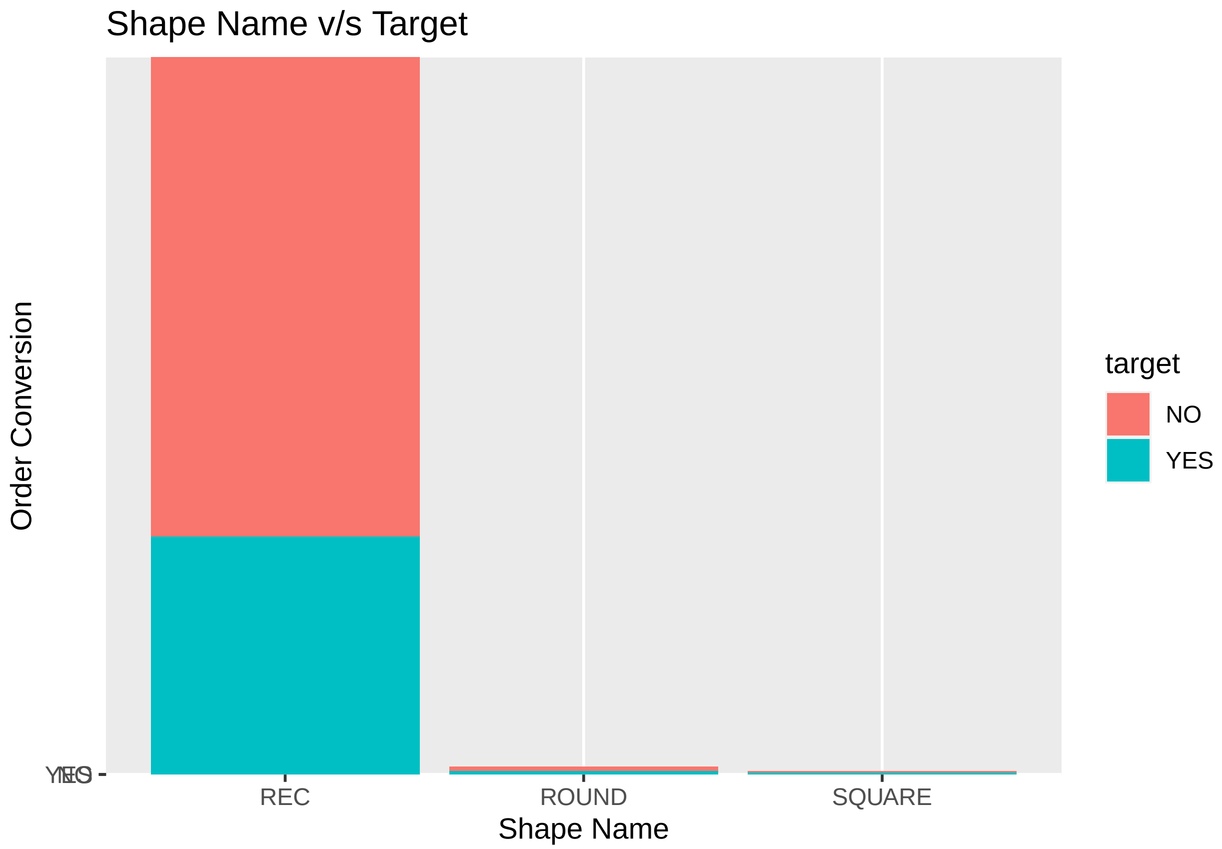


Analysis

From the above chart, we notice that *Hand tufted **is the only item that comes in all shapes -** Rectangle**,** Round**, and** Square**. However, it's largely available in "Rectangular" shape v/s the other shapes.** Durry **and** Gun Tufted **come in two shapes - "Rectangular" and "Round".** Indo-Tibetan** is the only item that comes in only "Square" shape.

5. ShapeName v/s Target

```
ggplot(DOS, aes(ShapeName, target, fill = target)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Shape Name v/s Target", x = "Shape Name",
         y= "Order Conversion")
```

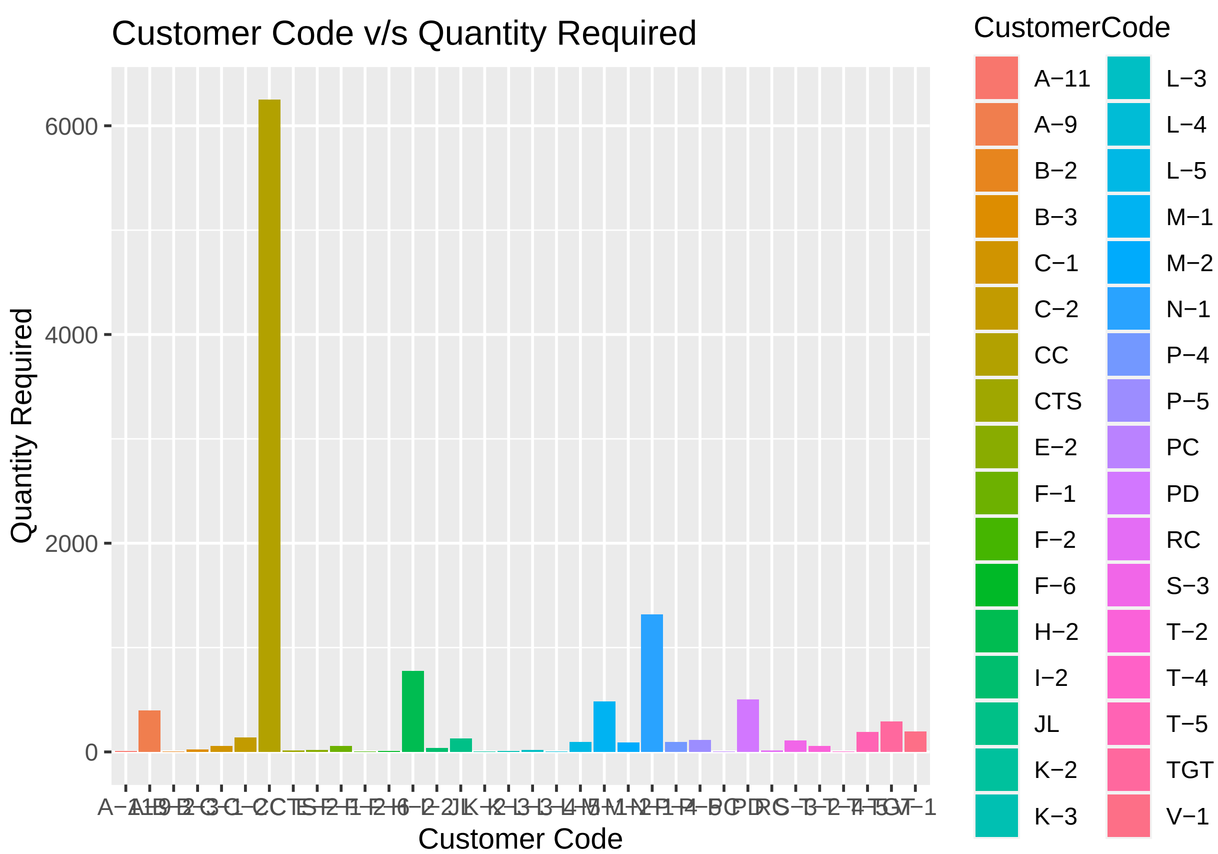


Analysis

From the above chart, we can see that out of all the shapes, **Rectangular** shape is the only one with the most order conversions, while also being the one that has the most orders that don't materialize. This theory can be explained by the above graphs, where **Round** and **Square** shaped carpets were not as prevalent as the the rectangular. Hence, **more sales for rectangular.**

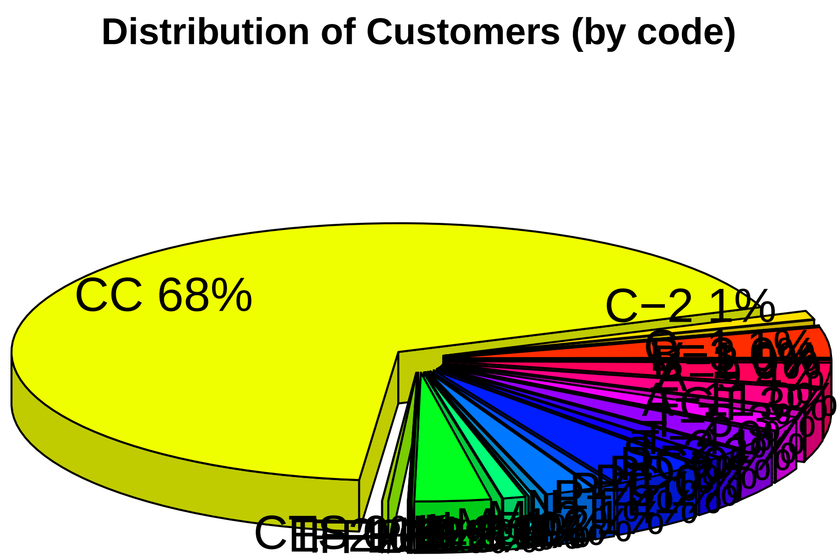
6. CustomerCode v/s QtyRequired

```
ggplot(DOS, aes(CustomerCode, QtyRequired, fill = CustomerCode)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Customer Code v/s Quantity Required", x = "Customer Code",
         y= "Quantity Required")
```



Pie chart for Customer distribution by code

```
pietable <- table(DOS$CustomerCode)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Customers (by code)", radius = 1.5)
```



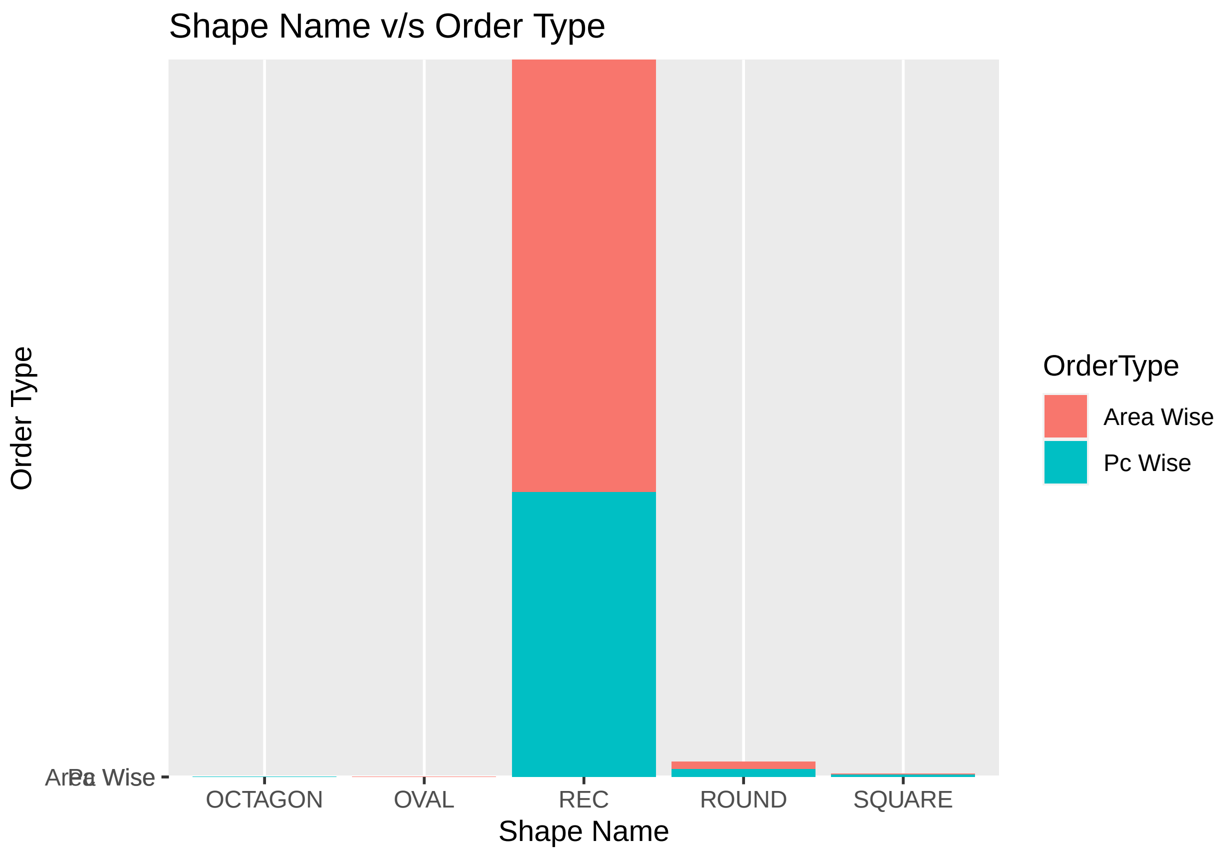
Analysis

From the first chart, we notice that the customer with code **CC** orders the most quantity of carpets from Champo. The next best sales is for the customer with code **N-1**. To understand why CC has the most sales, we pull a pie chart of the distribution of customers by their code. The pie chart shows us that the previous result is due to customer with code **CC** occupying **68%** of the distribution, while the rest of the items don't have much spread across the data set. Even though the customer with code **N-1** does not occupy much space on the distribution chart, it still converts a lot of the orders, thereby giving a Champo a lot of sales for the small customer they are.

EDA on Raw Data-Order and Sample Data Set

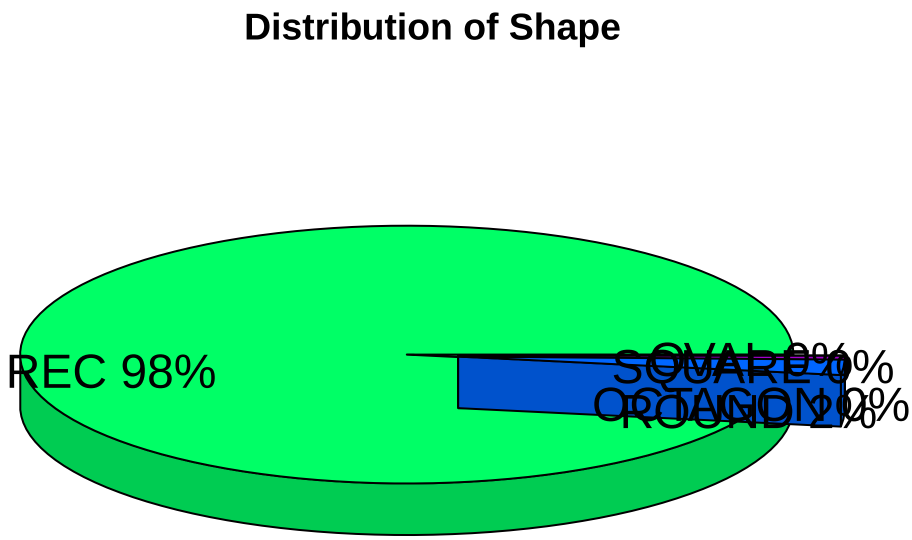
1. ShapeName v/s OrderType

```
ggplot(ROS, aes(ShapeName, OrderType, fill = OrderType)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Shape Name v/s Order Type", x = "Shape Name",
         y= "Order Type")
```



Pie chart for distribution of Shape

```
pietable <- table(ROS$ShapeName)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Shape", radius = 1.5)
```



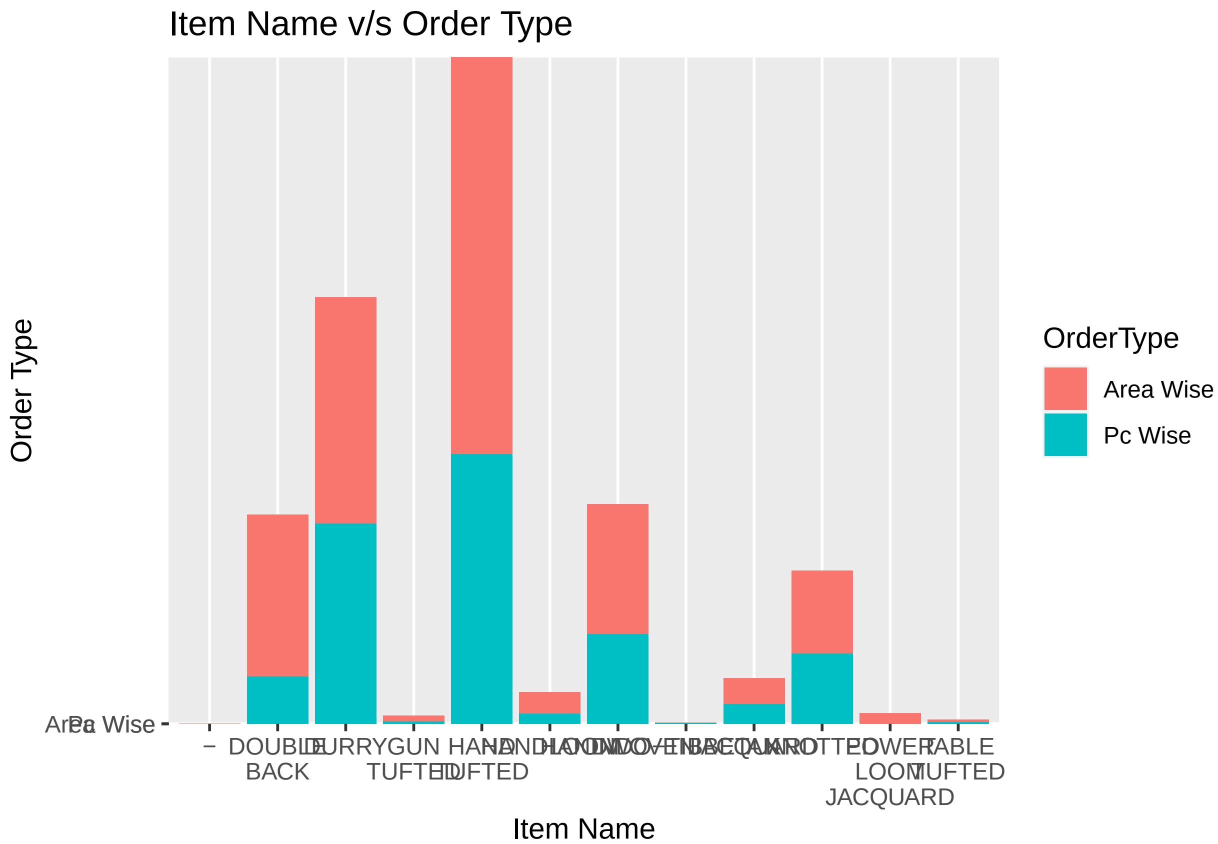
Analysis

From the first chart, we notice that **Rectangular** shaped carpets are sold more in terms of **Area** v/s **per piece**, whereas **Round** shaped carpets are sold equally in terms of area and per piece. **Oval** and **Octagon** shaped carpets are not sold as much as the other shapes.

We also pull a pie chart of the distribution of shapes. The pie chart shows us that the above result is due to **Rectangular** shaped carpets occupying **98%** of the distribution, while **Round** shaped carpets are at **2%**, which explains the theory of them being next best. Since **Octagon** and **Oval** shaped carpets occupy **0%** on the distribution chart, they are likely not sold at all due to their odd shape.

2. ITEM_NAME v/s OrderType

```
ggplot(ROS, aes(ITEM_NAME, OrderType, fill = OrderType)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Order Type", x = "Item Name",
         y= "Order Type")
```

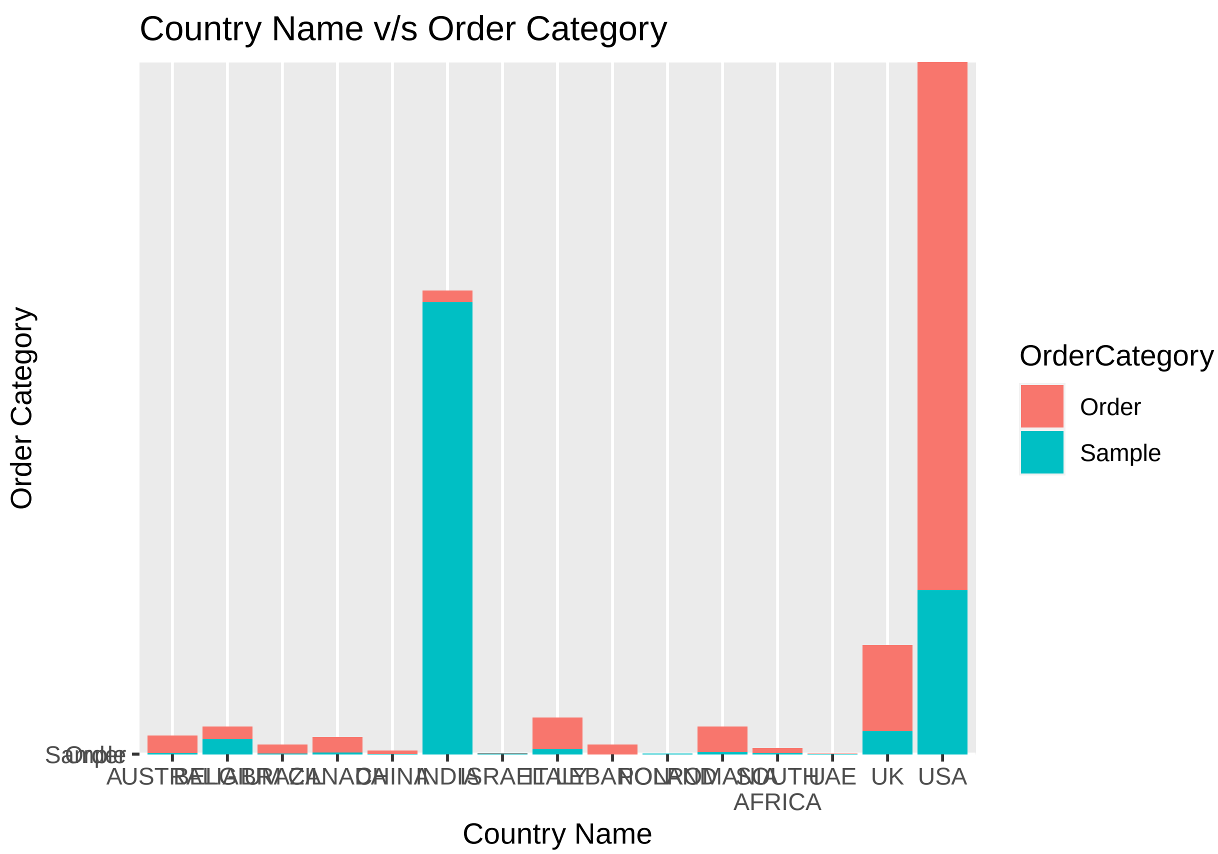


Analysis

From the above answer, **Hand Tufted** is the item type that's sold the most in both **Area** and **Per Piece** order types. Second best is **Durry**, followed by **Handwoven**, **Knotted**, and **Double Back**. **Power Loom Jacquard** is the only item type that's sold only area wise and not per piece.

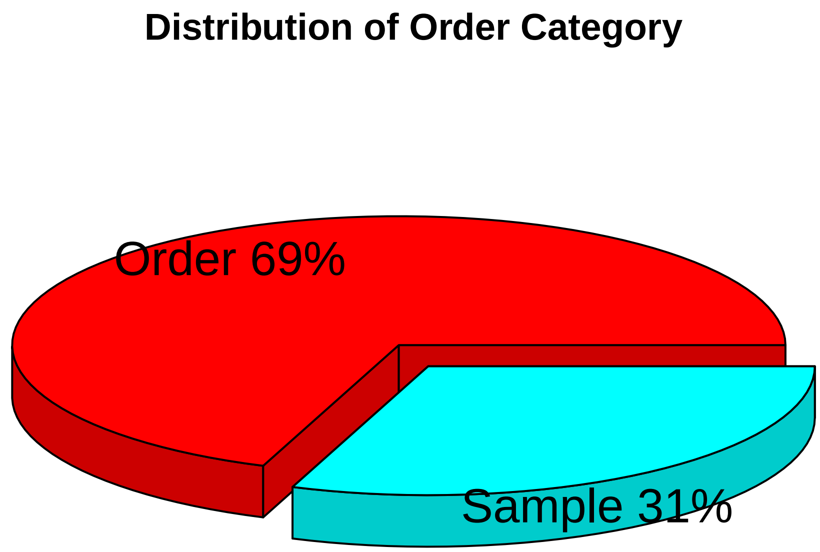
3. CountryName v/s OrderCategory

```
ggplot(ROS, aes(CountryName, OrderCategory, fill = OrderCategory)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Country Name v/s Order Category", x = "Country Name",
         y= "Order Category")
```



Pie chart for distribution of Order Category

```
pietable <- table(ROS$OrderCategory)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Order Category", radius = 1.5)
```



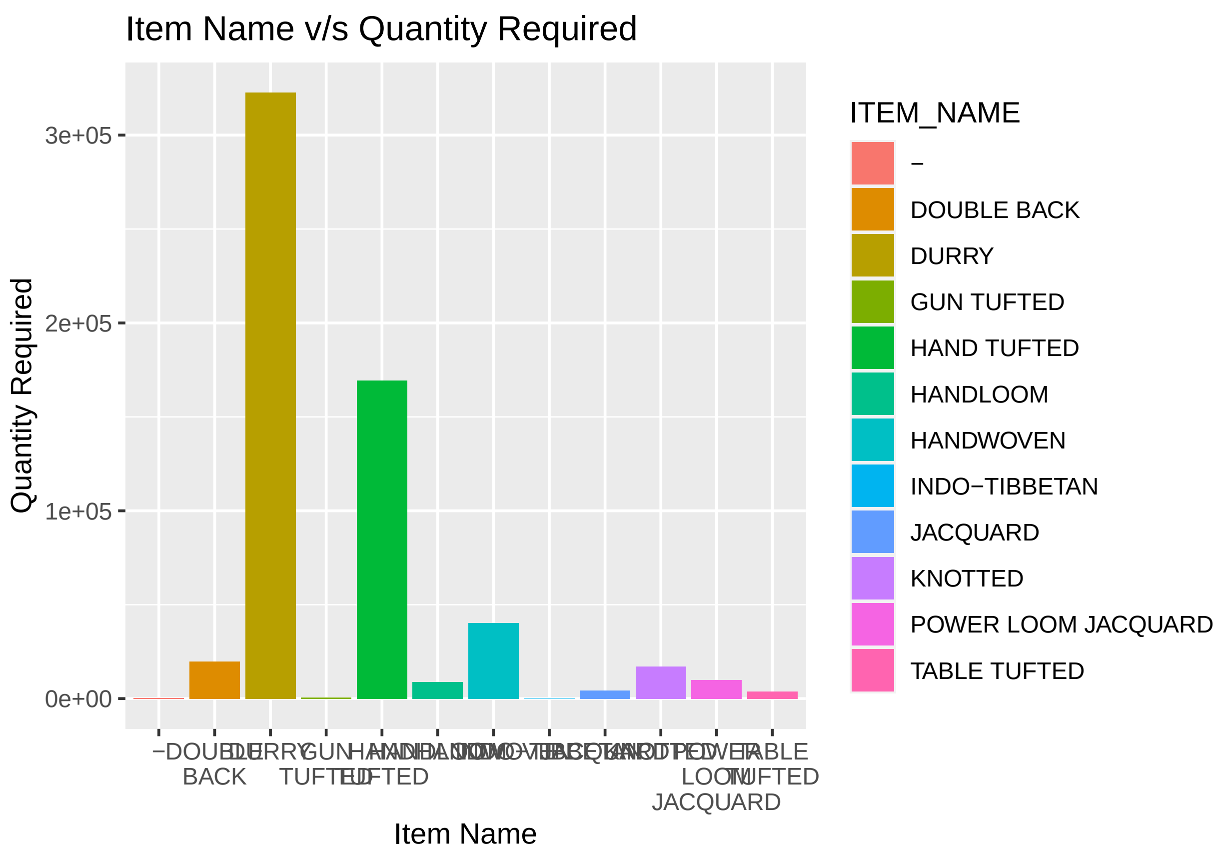
Analysis

From the first chart, we notice that **India** mainly requests **samples** from Champo v/s making an **order**. This is why in the above charts we saw India having the most requests that were not converting to sales, as they mainly request samples and then don't buy the product. **USA** on the other hand makes more order requests and less sample requests. **Australia**, **Brazil**, **Canada**, **China**, **Lebanon**, **Romania**, and **South Africa** make mostly order requests and little to none sales requests.

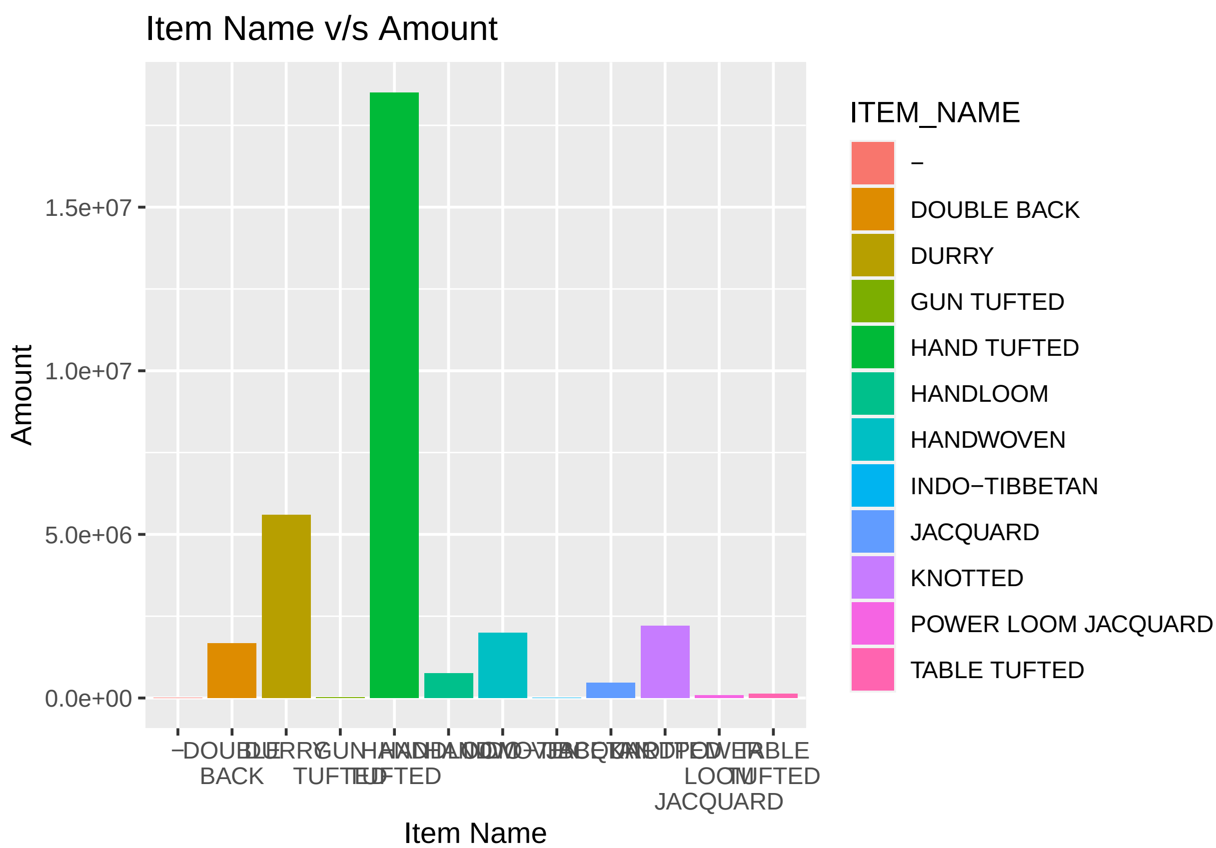
We also pull a pie chart of the distribution of order types. The pie chart shows us that **Orders** occupy **69%** of the distribution, while **samples** only form **31%** of the distribution.

4. **ITEM_NAME v/s QtyRequired & ITEM_NAME v/s Amount**

```
ggplot(ROS, aes(ITEM_NAME, QtyRequired, fill = ITEM_NAME)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Quantity Required", x = "Item Name",
         y= "Quantity Required")
```




```
ggplot(ROS, aes(ITEM_NAME, Amount, fill = ITEM_NAME)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Amount", x = "Item Name",
         y= "Amount")
```



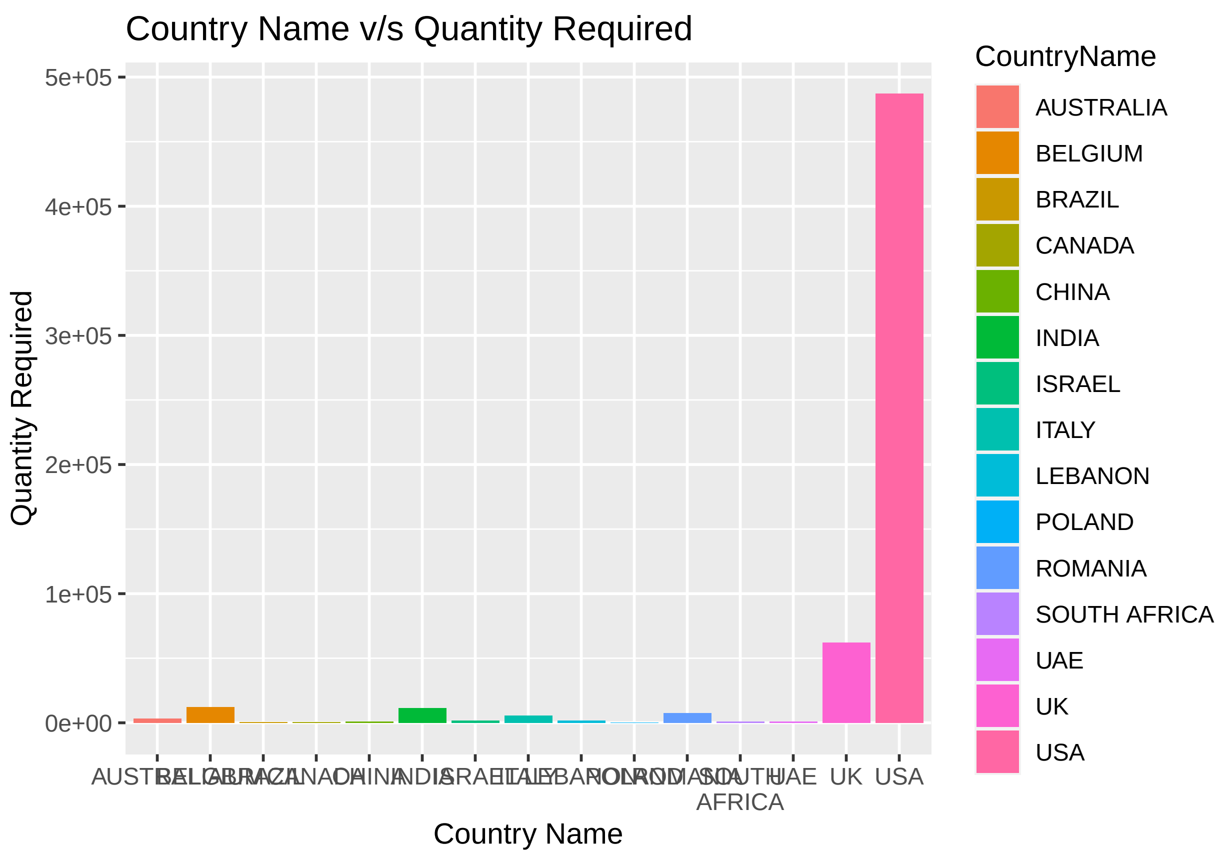
Analysis

From the first chart, we notice that **Durry** item type is ordered in the highest quantity compared to other item types. Next best is **Hand Tufted**. The least quantity required per the graph is for **Gun Tufted** while **Indo-Tibetan** was not sold at all in terms of quantity.

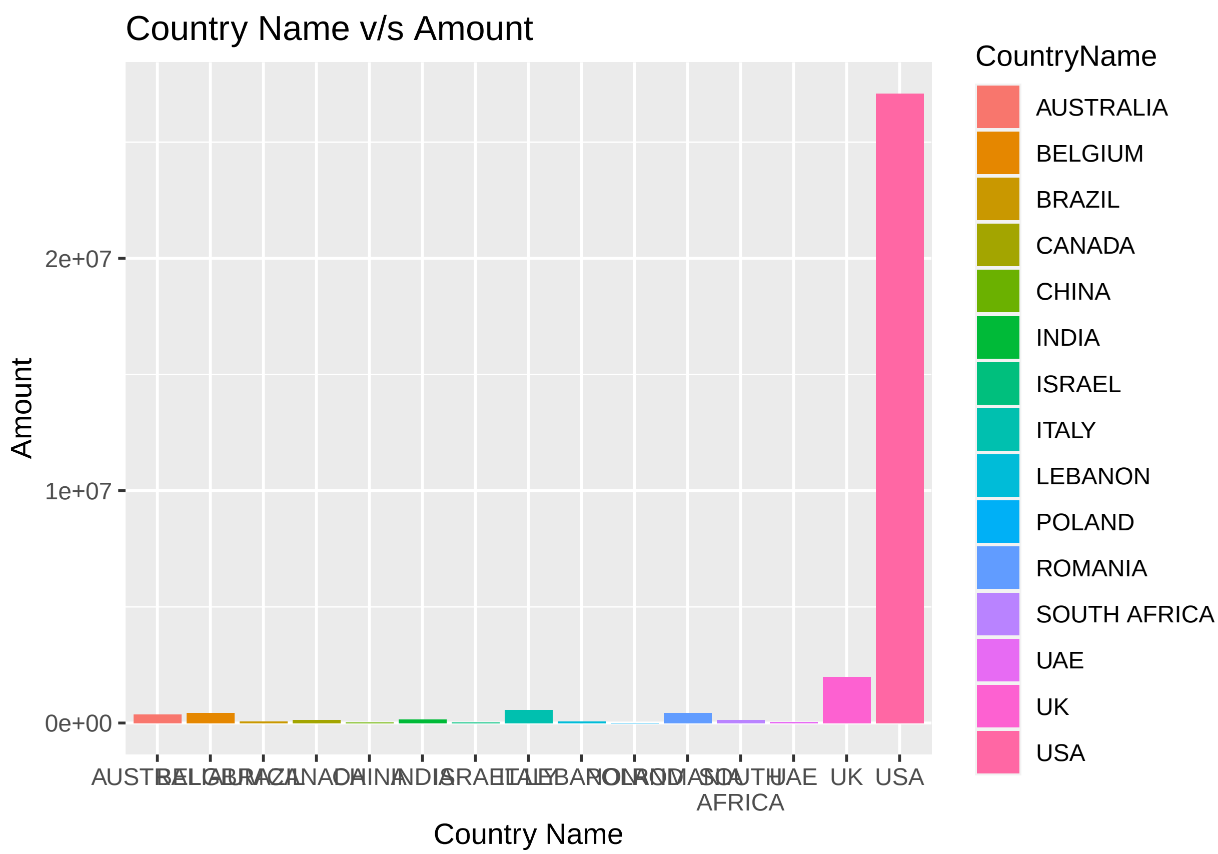
From the second chart, we can see that **Hand Tufted** is the most expensive item type followed by **Durry**, **Knotted**, and **Handwoven**. If we compare the findings with the above graphs, we find that since Hand Tufted carpets are the most expensive, they tend to sell in lower quantities compared to Durries, which are second most expensive carpet types.

5. CountryName v/s QtyRequired & CountryName v/s Amount

```
ggplot(ROS, aes(CountryName, QtyRequired, fill = CountryName)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Country Name v/s Quantity Required", x = "Country Name",
         y= "Quantity Required")
```




```
ggplot(ROS, aes(CountryName, Amount, fill = CountryName)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Country Name v/s Amount", x = "Country Name",
         y= "Amount")
```



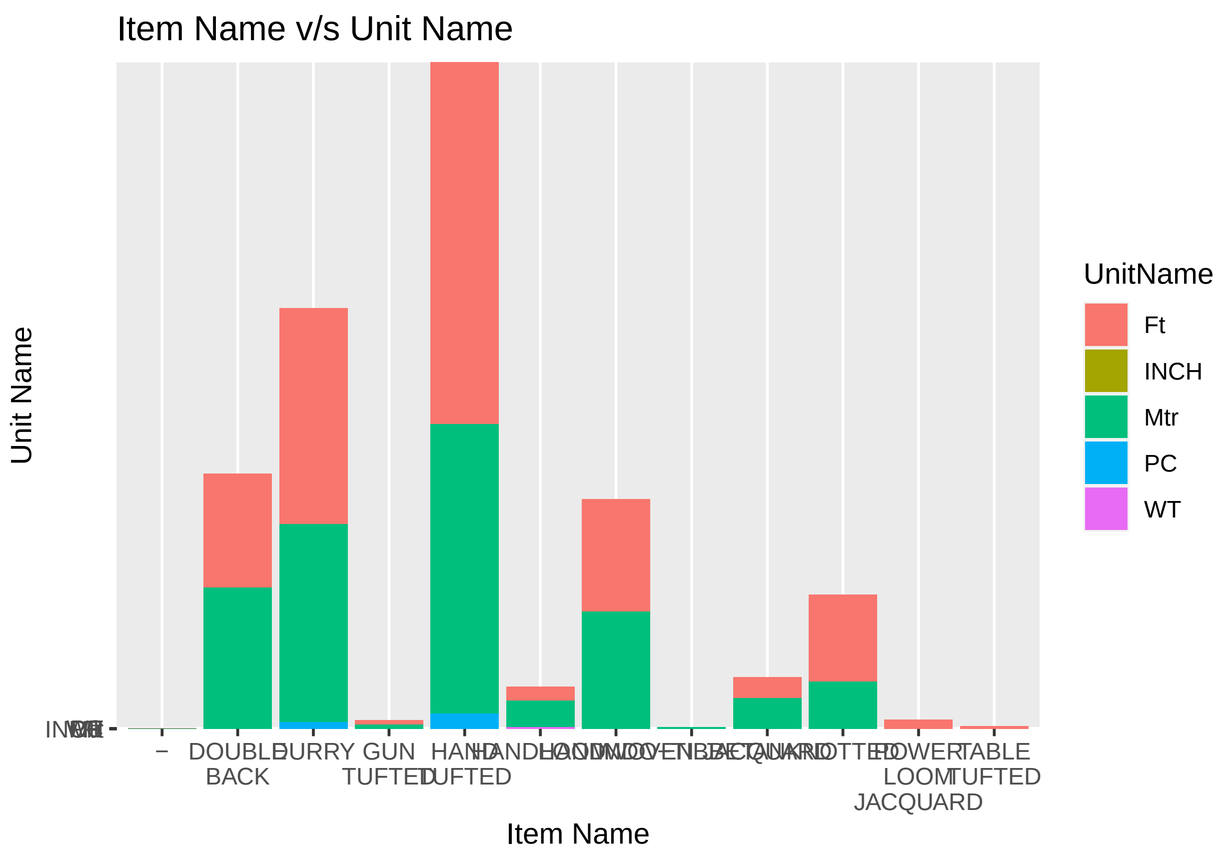
Analysis

From the first chart, we notice that **USA** requires the most quantity of carpets, followed by **UK**. This confirms are theory that **India** only orders samples but doesn't buy the actual product, as we saw in the above graphs.

From the second chart also we can see that **USA** spends the most amount on carpets, which is evident due to them ordering the most quantity as well. They're followed in the second spot by **UK**.

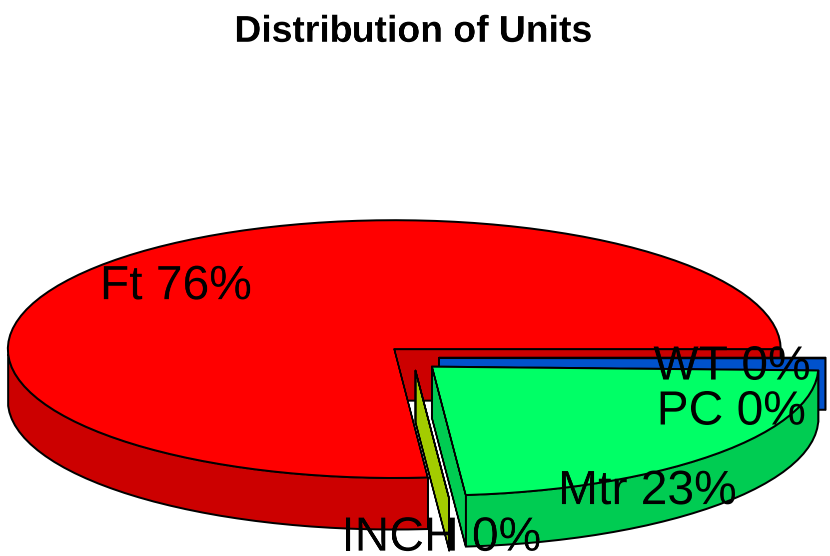
6. **ITEM_NAME v/s UnitName**

```
ggplot(ROS, aes(ITEM_NAME, UnitName, fill = UnitName)) +
    geom_bar(stat = "identity") + scale_x_discrete(labels = label_wrap(10)) +
    labs(title = "Item Name v/s Unit Name", x = "Item Name",
         y= "Unit Name")
```



Pie chart for distribution of Units

```
pietable <- table(ROS$UnitName)
percent <- round(pietable/sum(pietable)*100)
label1 <- paste(names(pietable),percent)
label2 <- paste(label1, "%", sep="")
pie3D(pietable, labels = label2, explode = 0.1,
      main="Distribution of Units", radius = 1.5)
```



Analysis

From the first chart, we notice that **Hand Tufted** carpets are sold a lot in terms of **feet**, **meters**, or **per piece**. They're followed by **Durry**, **Double Back**, **Handwoven**, and **Knotted**. However, Double Back, Handwoven, and Knotted are all sold in feet or meters only, not per piece. **Indo-Tibetan** is only sold in **meters**, while **Power Loom Jacquard** and **Table Tufted** carpets are sold per **foot**. **Handloom** is the only carpet type that is sold in terms of **Wt (weight)**.

We also pull a pie chart of the distribution of Units. The pie chart shows us that **Feet(ft)** occupies **76%** of the distribution, whereas **Meter(Mtr)** occupies **23%**. The rest are at less than **1%**.

Part B

Q. What kind of analytics and machine learning algorithms (e.g. classification, regression, clustering, recommender systems and etc) can be used by Champo Carpets to solve their problems, and in general for value creation? Justify your choices. Hint: This is just a conceptual question. You do not need to run any of these models for this question. Constructing models is done in the next question.

A. To solve Champo Carpets' problems and create value, we can use the following:

- Predictive analytics can be used to forecast demand and optimize inventory levels
- Machine learning algorithms such as Decision Trees and Random Forests can be used for classification and prediction
  - Decision tree can be used to identify the important attributes that determine the conversion of samples sent to the customers.
- Clustering algorithms like k-means can be used for customer segmentation
- Analytics techniques such as association rule mining and time series analysis can also be used to identify patterns and forecast trends

These techniques can help Champo Carpets make better decisions regarding inventory management, pricing, and marketing strategies.

Part C

Q. Develop ML models (e.g. logistic regression, decision trees, random forest, neural network, and boosting) to help identify features contributing to conversion (or non-conversion) of samples sent to customers. Hint: For each model, discuss how you select features and tune different parameters. How do you evaluate the performance of each model? How do you select the best model(s). Run all your models on both balanced and imbalanced data and check the difference. Please note that your binary target is the "order conversion" variable in the sample data. You can obtain this variable from the information provided in the raw data.

Decision Tree

Partitioning data into train and test data:

```
set.seed(123)
index_dt <- sample(2,nrow(DOS), replace=TRUE, prob=c(0.7,0.3))
train <- DOS[index_dt==1, ]
test <- DOS[index_dt==2, ]
```

To build a decision tree model we use the "rpart" function from "rpart" package:

```
library(rpart)
tree_model_champo <- rpart(target ~ ., train)
```

We can access the decision rules in the decision tree model using the print() functions:

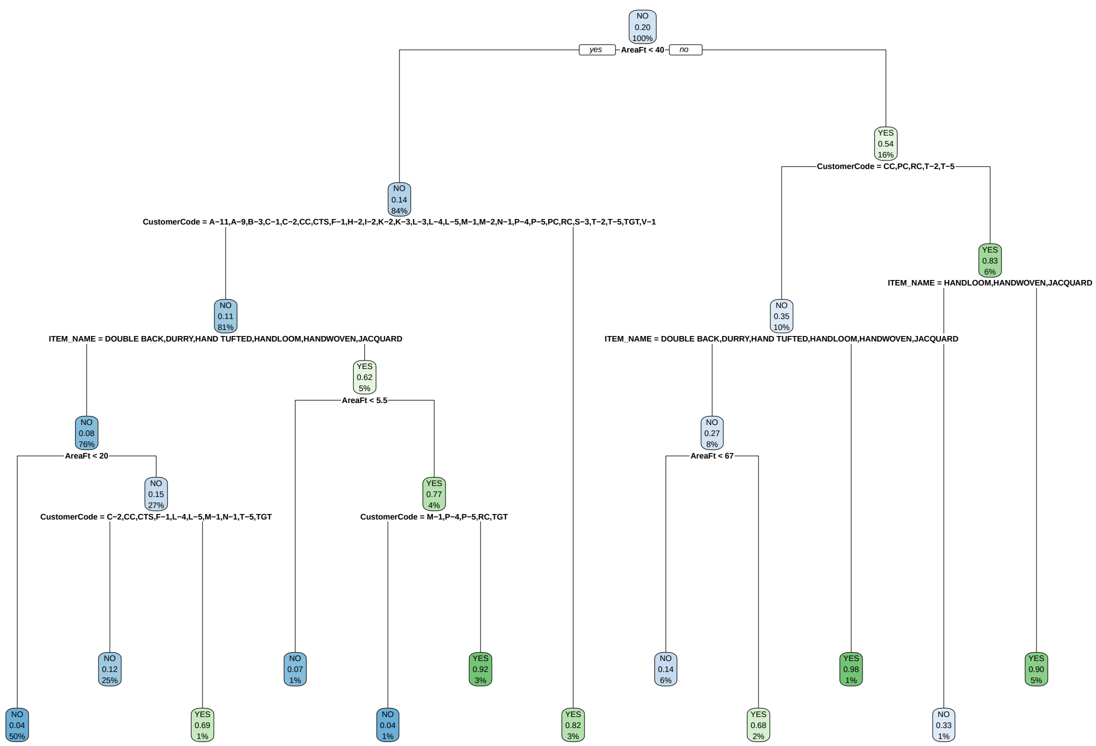
```
print(tree_model_champo)
```

```
## n= 4095
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 4095 819 NO (0.80000000 0.20000000)
##    2) AreaFt< 39.90625 3449 473 NO (0.86285880 0.13714120)
##      4) CustomerCode=A-11,A-9,B-3,C-1,C-2,CC,CTS,F-1,H-2,I-2,K-2,K-3,L-3,L-4,L-5,M-
1,M-2,N-1,P-4,P-5,PC,RC,S-3,T-2,T-5,TGT,V-1 3324 370 NO (0.88868833 0.11131167)
##        8) ITEM_NAME=DOUBLE BACK,DURRY,HAND TUFTED,HANDLOOM,HANDWOVEN,JACQUARD 3121 245 NO (0.9214995
##         16) AreaFt< 19.84375 2029  81 NO (0.96007886 0.03992114) *
##         17) AreaFt>=19.84375 1092 164 NO (0.84981685 0.15018315)
##           34) CustomerCode=C-2,CC,CTS,F-1,L-4,L-5,M-1,N-1,T-5,TGT 1040 128 NO (0.87692308 0.12307692
##           35) CustomerCode=A-9,C-1,H-2,I-2,M-2,P-4,P-5,RC,S-3 52  16 YES (0.30769231 0.69230769) *
##        9) ITEM_NAME=GUN TUFTED,INDO-TIBBETAN,KNOTTED,POWER LOOM JACQUARD,TABLE TUFTED 203  78 YES (0
##         18) AreaFt< 5.5 45   3 NO (0.93333333 0.06666667) *
##         19) AreaFt>=5.5 158  36 YES (0.22784810 0.77215190)
##           38) CustomerCode=M-1,P-4,P-5,RC,TGT 26   1 NO (0.96153846 0.03846154) *
##           39) CustomerCode=CC,L-3,S-3,T-2 132  11 YES (0.08333333 0.91666667) *
##      5) CustomerCode=E-2,F-2,F-6,JL,PD,T-4 125  22 YES (0.17600000 0.82400000) *
##    3) AreaFt>=39.90625 646 300 YES (0.46439628 0.53560372)
##      6) CustomerCode=CC,PC,RC,T-2,T-5 393 137 NO (0.65139949 0.34860051)
##       12) ITEM_NAME=DOUBLE BACK,DURRY,HAND TUFTED,HANDLOOM,HANDWOVEN,JACQUARD 348  93 NO (0.73275862
##         24) AreaFt< 66.93055 266  37 NO (0.86090226 0.13909774) *
```

```
##         25) AreaFt>=66.93055 82  26 YES (0.31707317 0.68292683) *
##        13) ITEM_NAME=GUN TUFTED,KNOTTED,POWER LOOM JACQUARD 45   1 YES (0.02222222 0.97777778) *
##      7) CustomerCode=A-9,C-1,C-2,F-1,H-2,JL,M-1,M-2,N-1,P-4,P-5,PD,S-3,TGT 253  44 YES (0.17391304 0
##        14) ITEM_NAME=HANDLOOM,HANDWOVEN,JACQUARD 33  11 NO (0.66666667 0.33333333) *
##        15) ITEM_NAME=DOUBLE BACK,DURRY,HAND TUFTED,KNOTTED,TABLE TUFTED 220  22 YES (0.10000000 0.900
```

To plot an rpart decision tree we can use the "rpart.plot()" function from "rpart.plot" package:

```
library(rpart.plot)
rpart.plot(tree_model_champo)
```



```
rpart.rules(tree_model_champo)
```

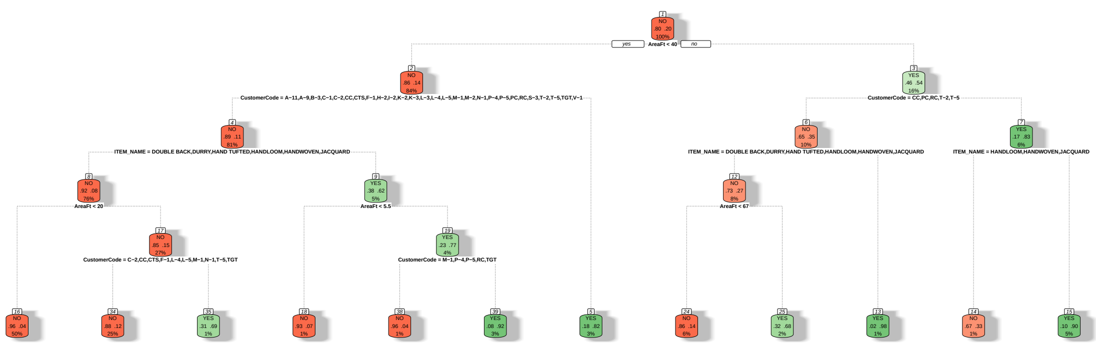
```
##  target
##    0.04 when AreaFt is  5.5 to 39.9 & CustomerCode is
1 or P-4 or P-5 or RC or TGT & ITEM_NAME is GUN TUFTED or INDO-TIBBETAN or KNOTTED or POWER LOOM JACQUAR
##    0.04 when AreaFt <  19.8         & CustomerCode is A-11 or A-9 or B-3 or C-1 or C-
2 or CC or CTS or F-1 or H-2 or I-2 or K-2 or K-3 or L-3 or L-4 or L-5 or M-1 or M-
2 or N-1 or P-4 or P-5 or PC or RC or S-3 or T-2 or T-5 or TGT or V-1 & ITEM_NAME is      DOUBLE BACK or
##    0.07 when AreaFt <   5.5         & CustomerCode is A-11 or A-9 or B-3 or C-1 or C-
2 or CC or CTS or F-1 or H-2 or I-2 or K-2 or K-3 or L-3 or L-4 or L-5 or M-1 or M-
2 or N-1 or P-4 or P-5 or PC or RC or S-3 or T-2 or T-5 or TGT or V-1 & ITEM_NAME is GUN TUFTED or INDO-
TIBBETAN or KNOTTED or POWER LOOM JACQUARD or TABLE TUFTED
##    0.12 when AreaFt is 19.8 to 39.9 & CustomerCode is
2 or CC or CTS or F-1 or L-4 or L-5 or M-1 or N-1 or T-5 or TGT & ITEM_NAME is      DOUBLE BACK or DURRY
##    0.14 when AreaFt is 39.9 to 66.9 & CustomerCode is
2 or T-5 & ITEM_NAME is      DOUBLE BACK or DURRY or HAND TUFTED or HANDLOOM or HANDWOVEN or JACQUARD
##    0.33 when AreaFt >=         39.9 & CustomerCode is
9 or C-1 or C-2 or F-1 or H-2 or JL or M-1 or M-2 or N-1 or P-4 or P-5 or PD or S-3 or TGT & ITEM_NAME i
##    0.68 when AreaFt >=         66.9 & CustomerCode is
2 or T-5 & ITEM_NAME is      DOUBLE BACK or DURRY or HAND TUFTED or HANDLOOM or HANDWOVEN or JACQUARD
```

```
##    0.69 when AreaFt is 19.8 to 39.9 & CustomerCode is
9 or C-1 or H-2 or I-2 or M-2 or P-4 or P-5 or RC or S-3 & ITEM_NAME is      DOUBLE BACK or DURRY or HA
##    0.82 when AreaFt <  39.9         & CustomerCode is
2 or F-2 or F-6 or JL or PD or T-4
##    0.90 when AreaFt >=         39.9 & CustomerCode is
9 or C-1 or C-2 or F-1 or H-2 or JL or M-1 or M-2 or N-1 or P-4 or P-5 or PD or S-3 or TGT & ITEM_NAME 
##    0.92 when AreaFt is  5.5 to 39.9 & CustomerCode is
3 or S-3 or T-2 & ITEM_NAME is GUN TUFTED or INDO-TIBBETAN or KNOTTED or POWER LOOM JACQUARD or TABLE T
##    0.98 when AreaFt >=         39.9 & CustomerCode is
2 or T-5 & ITEM_NAME is                                  GUN TUFTED or KNOTTED or POWER LOOM JACQUARD
```

According to our decision tree mode, the root node splits on **"AreaFt"** variable and if it's less than or greater than 40 feet. If yes, it further splits into **CustomerCode**, followed by **"ITEM_NAME"**.

In order to see a more fancier version of rpart.plot, we also have the option of fancyRpartPlot() function, which is part of the rattle library. It can be run as follows:

```
library(rattle)
fancyRpartPlot(tree_model_champo, palettes=c("Reds", "Greens"), sub="")
```



To obtain the predicted classes or predicted probabilities we can use the "predict" function:

```
tree_pred_prob_champo <- predict(tree_model_champo, train)
tree_pred_prob_champo <- predict(tree_model_champo, train, type = "prob")
tree_pred_class_champo <- predict(tree_model_champo, train, type = "class")
```

Error rate of the decision tree model on training data:

```
mean(tree_pred_class_champo != train$target)
```

```
## [1] 0.08766789
```

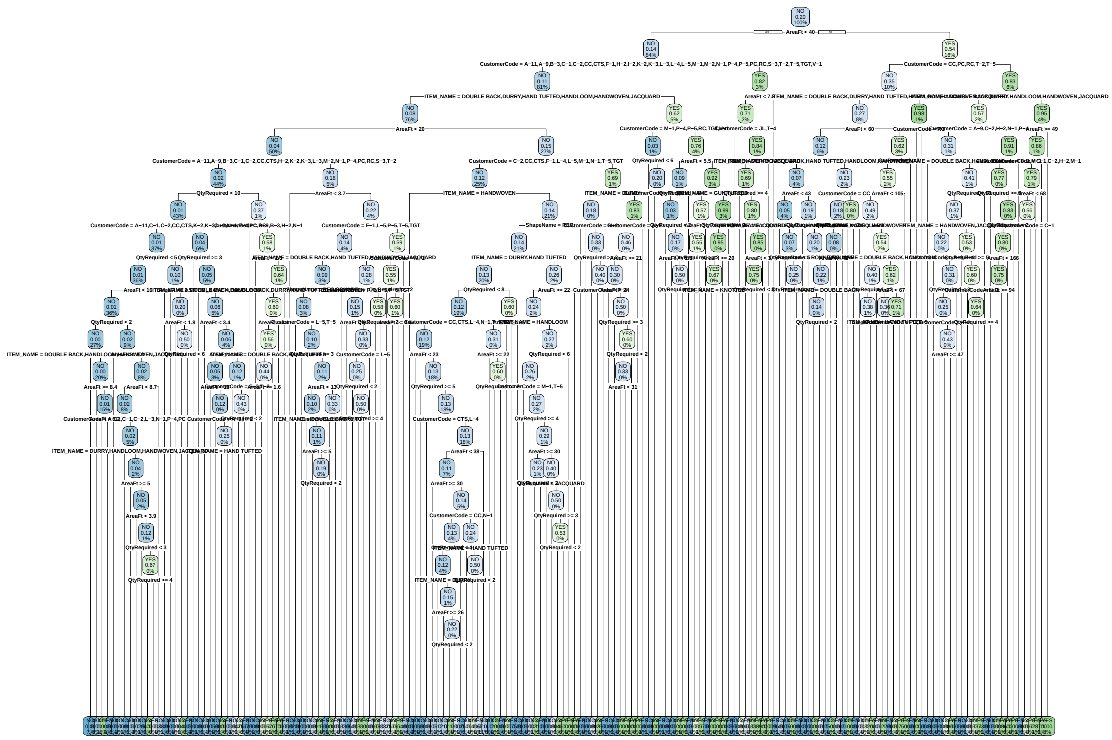
Error rate of the decision tree model on test data:

```
tree_pred_test_champo <- predict(tree_model_champo, test, type = "class")
base_error_champo <- mean(tree_pred_test_champo != test$target)
base_error_champo
```

```
## [1] 0.09217391
```

Changing the parameters of rpart. parms = list(split = "information") or parms = list(split = "gini"). We can also modify the pre-pruning options in the rpart.control:

```
tree_model_champo2 <- rpart(target ~ ., train,
                  parms = list(split = "information"),
                  control = rpart.control(minbucket = 0, minsplit = 0, cp = 0))

rpart.plot(tree_model_champo2)
```



```
pred_test_champo <- predict(tree_model_champo2, test, type = "class")
error_preprun_champo <- mean(pred_test_champo != test$target)
mincp_i <- which.min(tree_model_champo2$cptable[, 'xerror'])
```

We can select the best cp in two different approach:

```
optCP <- tree_model_champo2$cptable[mincp_i, "CP"]
```

```
#The optimal xerror is the min_xError + xstd
optError <- tree_model_champo2$cptable[mincp_i, "xerror"] +
  tree_model_champo2$cptable[mincp_i, "xstd"]

#the row(index) of the xerror value which is closest to optError
optCP_i <- which.min(abs(tree_model_champo2$cptable[,"xerror"] - optError))
```

```r
#finally, get the best CP value corresponding to optCP_i
optCP <- tree_model_champo2$cptable[optCP_i, "CP"]
optCP
```

```
## [1] 0.0003052503
```

Now we can prune the tree based on this best CP value:

```r
model_pruned_champo <- prune(tree_model_champo2, cp = optCP)
```

Computing the accuracy of the pruned tree:

```r
library(knitr)
test$pred <- predict(model_pruned_champo, test, type = "class")
error_postprun_champo <- mean(test$pred != test$target)
df <- data.frame(base_error_champo, error_preprun_champo, error_postprun_champo)

base_error_champo_pct <-
paste(round(base_error_champo*100, 3), "%", sep = "")

error_preprun_champo_pct <-
paste(round(error_preprun_champo*100, 3), "%", sep = "")

error_postprun_champo_pct <-
paste(round(error_postprun_champo*100, 3), "%", sep = "")

df_pct <-
data.frame(base_error_champo_pct,
error_preprun_champo_pct, error_postprun_champo_pct)

kable(df)
```

base_error_champo	error_preprun_champo	error_postprun_champo
0.0921739	0.0707246	0.0713043

```r
kable(df_pct)
```

base_error_champo_pct	error_preprun_champo_pct	error_postprun_champo_pct
9.217%	7.072%	7.13%

The summary is that our model's base error rate is **9.22%** on the training data set. However, before pruning, the error rate on the test data set is **7.07%**. Lastly, after we prune our data set, the model's error rate becomes **7.13%**, which increased a little compared to preprune.

Random Forest

The main input arguments of the randomForest() function are: Formula that determines which variable is the target and which variables are inputs; data that indicates the training data; ntree that denotes the number of trees considered in the random forest. This should not be set to a too small number; and mtry that indicates the number of variables randomly sampled as candidates at each split. If you need to obtain the proximity matrix or important variables suggested by this model you can use "proximity = TRUE" and "importance = TRUE"

```
library(randomForest)

rf_champo <- randomForest(target ~ DOS$CustomerCode + DOS$CountryName +
DOS$USA + DOS$UK + DOS$Italy + DOS$Belgium + DOS$Romania + DOS$Australia +
DOS$India + DOS$QtyRequired + DOS$ITEM_NAME + DOS$`Hand Tufted` + DOS$Durry +
DOS$`Double Back` + DOS$`Hand Woven` + DOS$Knotted + DOS$Jacquard +
DOS$Handloom + DOS$Other + DOS$ShapeName + DOS$REC + DOS$Round + DOS$Square +
DOS$AreaFt, data = DOS, mtry = sqrt(ncol(DOS)-1),
ntree = 300, proximity = T, importance = T)
```

We can print the model, attributes, and also plot the error rates with various number of trees:

```
print(rf_champo)
```

```
##
## Call:
##  randomForest(formula = target ~ DOS$CustomerCode + DOS$CountryName +      DOS$USA + DOS$UK + DOS$Ita
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 7.39%
## Confusion matrix:
##       NO YES class.error
## NO  4555  96  0.02064072
## YES  334 835  0.28571429
```

```
names(rf_champo)
```

```
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
```

The OOB error rate for our random forest model is **0.0739** or **7.39%**

Attributes

To view all the attributes we can call upon using our model, we utilize the attributes() function as follows:

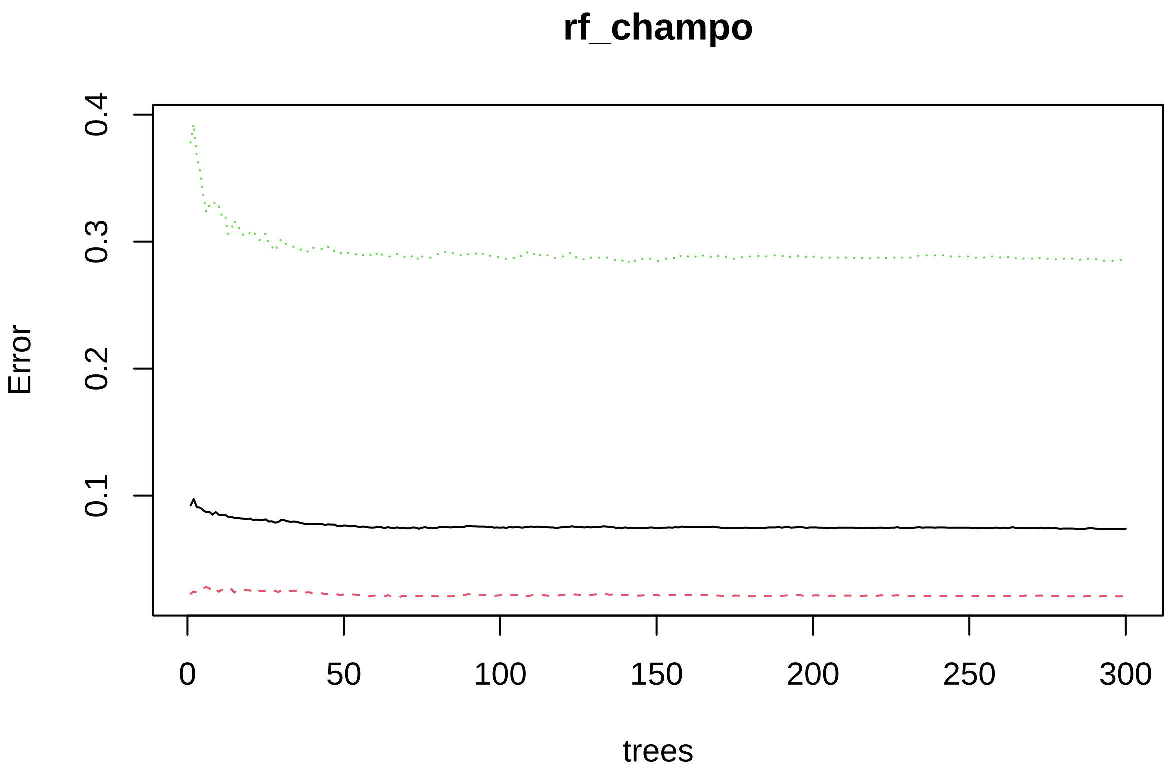
```
attributes(rf_champo)
```

```
## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

Plot

We plot the error rates with various number of trees using the plot() function. In this result, we'll see a red curve, which is the error rate for the positive class that is **"YES"** in our case, green curve is for the negative class that is **"NO"**, and the black curve indicates the error rate on OOB.

```
plot(rf_champo)
```



MeanDecreaseAccuracy & MeanDecreaseGini

Get the importance of variables by the function "importance()". Include type = 1 in the importance function is to get the important variables based on MeanDecreaseAccuracy. Type=2 is for MeanDecreaseGini. Just selecting a subset of results, where the value is greater than 10, so we don't get many variables back.

```
imp1 <- importance(rf_champo, type = 1)
imp2 <- importance(rf_champo, type = 2)

imp1
```

```
##                      MeanDecreaseAccuracy
## DOS$CustomerCode                31.057285
## DOS$CountryName                 17.954372
## DOS$USA                         11.970046
## DOS$UK                          11.452146
## DOS$Italy                        4.617332
## DOS$Belgium                     10.546275
## DOS$Romania                      5.109034
## DOS$Australia                    3.152470
## DOS$India                       13.257045
## DOS$QtyRequired                 36.736141
## DOS$ITEM_NAME                   29.589208
## DOS$`Hand Tufted`               10.948977
## DOS$Durry                        6.907514
## DOS$`Double Back`                5.908045
## DOS$`Hand Woven`                 7.646960
## DOS$Knotted                     14.025659
## DOS$Jacquard                     6.746856
## DOS$Handloom                    11.767232
## DOS$Other                       14.549443
## DOS$ShapeName                   10.151401
## DOS$REC                          9.365213
## DOS$Round                        4.476112
## DOS$Square                       6.535739
## DOS$AreaFt                      73.352373
```

```
imp2
```

```
##                      MeanDecreaseGini
## DOS$CustomerCode          197.2615998
## DOS$CountryName            94.0178654
## DOS$USA                    33.0311709
## DOS$UK                      8.5855733
## DOS$Italy                   0.8389421
## DOS$Belgium                38.2955204
## DOS$Romania                 1.0731408
## DOS$Australia               0.8075822
## DOS$India                  33.5083757
## DOS$QtyRequired            55.8010553
## DOS$ITEM_NAME             181.6432065
## DOS$`Hand Tufted`          15.6955995
## DOS$Durry                  10.7415275
```

```
## DOS$`Double Back`        4.5202980
## DOS$`Hand Woven`         8.4710932
## DOS$Knotted             41.6649436
## DOS$Jacquard             2.1544254
## DOS$Handloom             7.3699505
## DOS$Other               55.4396524
## DOS$ShapeName            3.4607528
## DOS$REC                  2.8127294
## DOS$Round                1.4053846
## DOS$Square               1.7471613
## DOS$AreaFt             342.5833140
```

To see just a subset of the important variables, we can set a threshold on MeanDecreaseAccuracy and MeanDecreaseGini > 10

```
subset(imp1, imp1[] >  10)
```

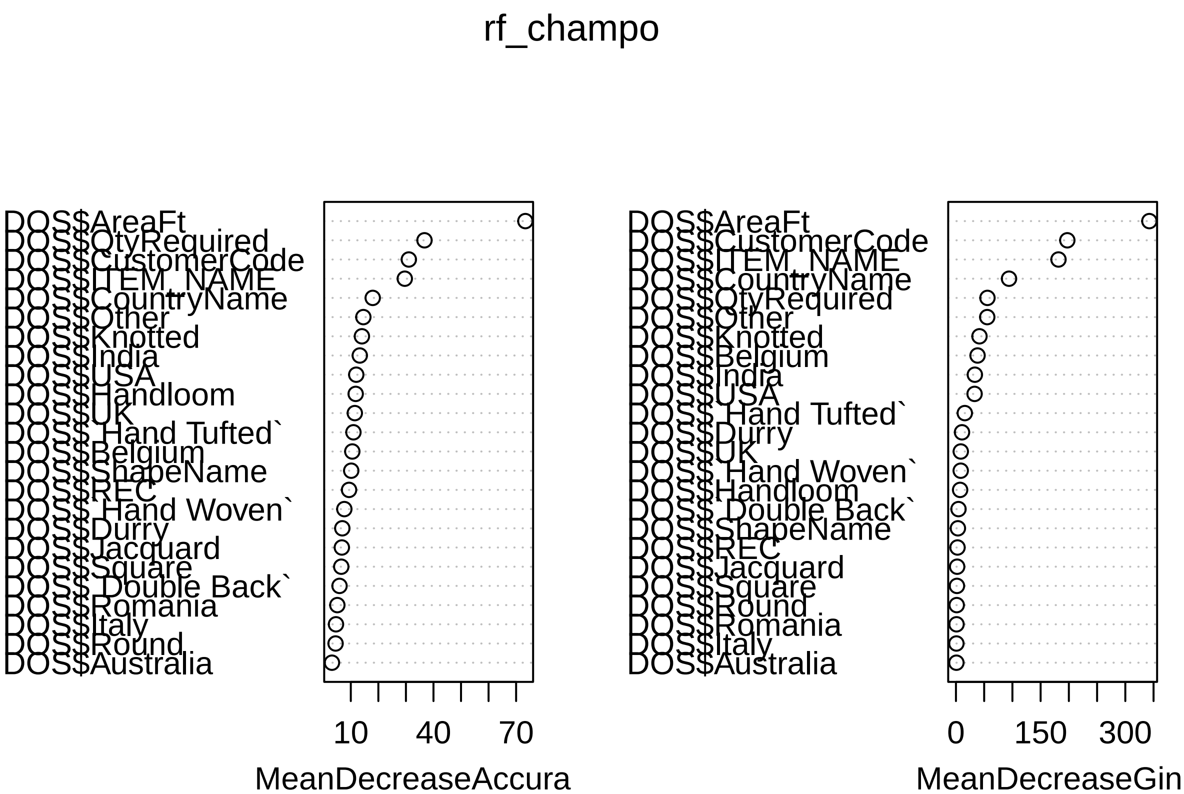
```
##                    MeanDecreaseAccuracy
## DOS$CustomerCode               31.05729
## DOS$CountryName                17.95437
## DOS$USA                        11.97005
## DOS$UK                         11.45215
## DOS$Belgium                    10.54628
## DOS$India                      13.25705
## DOS$QtyRequired                36.73614
## DOS$ITEM_NAME                  29.58921
## DOS$`Hand Tufted`              10.94898
## DOS$Knotted                    14.02566
## DOS$Handloom                   11.76723
## DOS$Other                      14.54944
## DOS$ShapeName                  10.15140
## DOS$AreaFt                     73.35237
```

```
subset(imp2, imp2[] >  10)
```

```
##                    MeanDecreaseGini
## DOS$CustomerCode          197.26160
## DOS$CountryName            94.01787
## DOS$USA                    33.03117
## DOS$Belgium                38.29552
## DOS$India                  33.50838
## DOS$QtyRequired            55.80106
## DOS$ITEM_NAME             181.64321
## DOS$`Hand Tufted`          15.69560
## DOS$Durry                  10.74153
## DOS$Knotted                41.66494
## DOS$Other                  55.43965
## DOS$AreaFt                342.58331
```

Importance Plot

```
varImpPlot(rf_champo)
```



Summary

The most important variable according to the MeanDecreaseAccuracy graph is **"QtyRequired"**, followed by **"CustomerCode"** and **"ITEM_NAME"** at third position. According to MeanDecreaseGini model, the most important variable is **"AreaFt"**, followed by **"CustomerCode"** and **"ITEM_NAME"**, just like the first model. These make sense, as quantity required, area in feet, and item names are crucial variables for buying Champo Carpets, whereas customer code is also important to ensure we're keeping the customers in mind.

Predicted Classes & Probablities

We can also obtain the predicted classes and predicted probabilities using the following codes:

```
head(rf_champo$predicted)
```

```
##   1   2   3   4   5   6
## YES YES YES YES YES  NO
## Levels: NO YES
```

```
head(rf_champo$votes)
```

```
##           NO       YES
## 1 0.16822430 0.8317757
## 2 0.04255319 0.9574468
## 3 0.07216495 0.9278351
## 4 0.07500000 0.9250000
## 5 0.06194690 0.9380531
## 6 0.85470085 0.1452991
```

Confusion Matrix

To obtain a confusion matrix, we can also use the confusionMatrix() function from the "caret" package. Similar to the table() function, confusionMatrix() also receives the predicted and actual labels as inputs.

```
library(caret)
confusionMatrix(rf_champo$predicted, DOS$target, positive = "YES")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NO  YES
##        NO  4555  334
##        YES   96  835
##
##                Accuracy : 0.9261
##                  95% CI : (0.9191, 0.9327)
##     No Information Rate : 0.7991
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7509
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7143
##             Specificity : 0.9794
##          Pos Pred Value : 0.8969
##          Neg Pred Value : 0.9317
##              Prevalence : 0.2009
##          Detection Rate : 0.1435
##    Detection Prevalence : 0.1600
##       Balanced Accuracy : 0.8468
##
##        'Positive' Class : YES
##
```

Evaluation Charts

To draw the evaluation charts we use "ROCR" package. There are two function in this package that we require to draw all different charts discussed in class: prediction and performance. The prediction() function receives two inputs:

1. The predicted probability of the positive class and
2. The true labels

The output of the prediction function will be given to the performance() function to draw the charts

```
library(ROCR)
score <- rf_champo$votes[, 2]
pred <- prediction(score, DOS$target)

pred
```

```
## A prediction instance
##   with 5820 data points
```

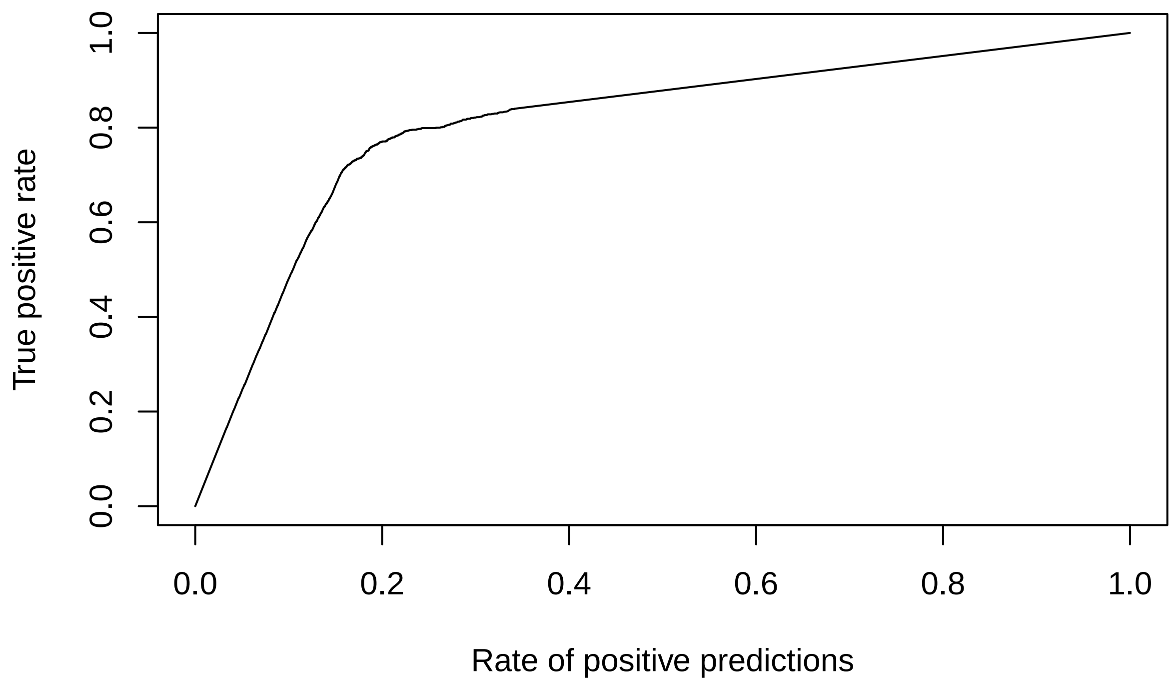
Gain chart

The gain chart for our model is:

```
perf <- performance(pred, "tpr", "rpp")

perf
```

```
## A performance instance
##   'Rate of positive predictions' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 1120 data points
```

```
plot(perf)
```



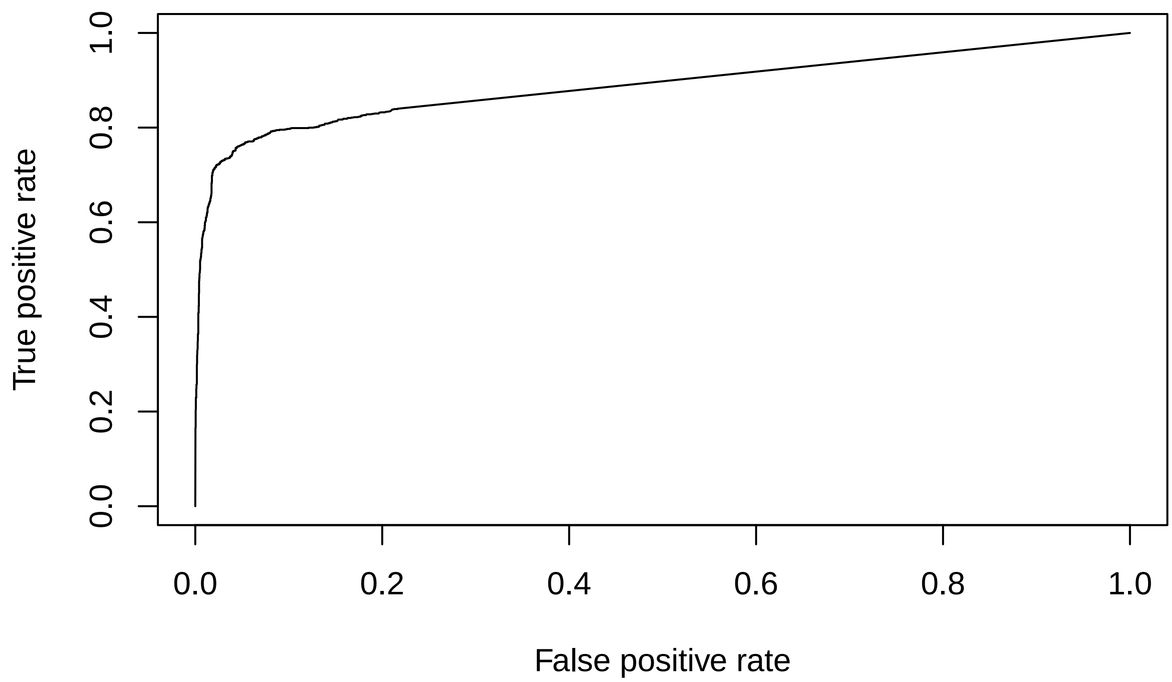
ROC Curve

The ROC curve for our model is:

```
perf1 <- performance(pred, "tpr", "fpr")

perf1
```

```
## A performance instance
##   'False positive rate' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 1120 data points
```

```
plot(perf1)
```



Area under the curve

The area under the curve of our ROC curve is:

```
auc <- unlist(slot(performance(pred, "auc"), "y.values"))
auc
```

```
## [1] 0.8881341
```

Determining the best cut-off point

The performance() function for ROC curve returns tpr, fpr and alpha-values (cut-off points). We can use the following code to write a function that received these and return the best cut-off point as the point closest to the corner [0, 1]. The input argument to this function is perf (the output of the performance() function).

The mapply function applies the function FUN to all **perf@x.values**, **perf@y.values**, and **perf@alpha.values**. In the function FUN(), we first compute the distance of all the points on the ROC curve from the corner point [0,1]. These distance values are stored in the vector "d". We then find the index of the point that is the closest point to the corner. This index is stored in the variable named "ind". The output of this function is then the tpr, fpr and the probability threshold corresponding to this index.

```
cut.ind <- mapply(FUN = function(x,y,p) {
  d=(x-0)^2+(y-1)^2
  ind<- which(d==min(d))
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, perf@alpha.values)

cut.ind
```

```
##                  [,1]
## recall      0.7690334
## specificity 0.8027491
## cutoff      0.1981132
```

Per the values above, we can see that the recall value is **76.9%**, whereas the cutoff is at **0.19811**.

Naive Bayes Model

In particular, Naive Bayes is often used with nominal (categorical) variables. If numeric variables are to be used, they have to be:

- represented as probabilities based on the appropriate probability distribution (typically Normal dist.) or,
- discretized, that is, have their range of values split into several segments, and thus be turned into discrete set of values.

We check if our numeric variables are normally distributed or not. For this, we use the Shapiro test. Our numeric variables are: **"USA", "UK", "Italy", "Belgium", "Romania", "Australia", "India", "QtyRequired", and "AreaFt".

```
#creating a variable to hold 5000 values, as Shapiro test fails on large data

print(var_usa <- shapiro.test(DOS$USA[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$USA[0:5000]
## W = 0.53495, p-value < 2.2e-16
```

```
print(var_uk <- shapiro.test(DOS$UK[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$UK[0:5000]
## W = 0.18748, p-value < 2.2e-16
```

```
print(var_italy <- shapiro.test(DOS$Italy[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$Italy[0:5000]
## W = 0.035255, p-value < 2.2e-16
```

```
print(var_belgium <- shapiro.test(DOS$Belgium[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$Belgium[0:5000]
## W = 0.11958, p-value < 2.2e-16
```

```r
print(var_romania <- shapiro.test(DOS$Romania[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$Romania[0:5000]
## W = 0.035255, p-value < 2.2e-16
```

```r
print(var_australia <- shapiro.test(DOS$Australia[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$Australia[0:5000]
## W = 0.013301, p-value < 2.2e-16
```

```r
print(var_india <- shapiro.test(DOS$India[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$India[0:5000]
## W = 0.58307, p-value < 2.2e-16
```

```r
print(var_qty <- shapiro.test(DOS$QtyRequired[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$QtyRequired[0:5000]
## W = 0.14506, p-value < 2.2e-16
```

```r
print(var_area <- shapiro.test(DOS$AreaFt[0:5000]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  DOS$AreaFt[0:5000]
## W = 0.78537, p-value < 2.2e-16
```

Since the shapiro test reveals that all are numeric variables are not normally distributed and they fail the hypothesis, we remove them from our data set. For this, we create a copy of our DOS dataset called DOS_nb:

```
DOS_nb <- DOS
DOS_nb <- within(DOS_nb, rm('USA', 'UK','Italy','Belgium', 'Romania',
                            'Australia', "India", "QtyRequired", "AreaFt"))
```

Now that we have prepared the data, we can proceed to create sets for training and testing:

```
index_nb <- sample(2, nrow(DOS_nb), replace = T, prob = c(0.8, 0.2))
train_nb <- DOS_nb[index_nb == 1, ]
test_nb <- DOS_nb[index_nb == 2, ]
```

To build a NB model, we will use the naiveBayes() function from the e1071 package. First, we'll build a model using all the variables:

```
library(e1071)
nb1 <- naiveBayes(target ~ ., data = train_nb)
print(nb1)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        NO       YES
## 0.7986275 0.2013725
##
## Conditional probabilities:
##      CustomerCode
## Y             A-11          A-9          B-2          B-3          C-1
##   NO  0.0002685285 0.0295381310 0.0000000000 0.0002685285 0.0048335124
##   YES 0.0000000000 0.0830670927 0.0000000000 0.0000000000 0.0117145900
##      CustomerCode
## Y              C-2           CC          CTS          E-2          F-1
##   NO  0.0085929108 0.7303974221 0.0024167562 0.0005370569 0.0067132116
##   YES 0.0202342918 0.4664536741 0.0000000000 0.0106496273 0.0127795527
##      CustomerCode
## Y              F-2          F-6          H-2          I-2           JL
##   NO  0.0002685285 0.0000000000 0.0249731472 0.0024167562 0.0040279270
##   YES 0.0010649627 0.0031948882 0.0607028754 0.0074547391 0.0351437700
##      CustomerCode
## Y              K-2          K-3          L-3          L-4          L-5
##   NO  0.0013426423 0.0002685285 0.0024167562 0.0005370569 0.0051020408
##   YES 0.0000000000 0.0000000000 0.0010649627 0.0000000000 0.0021299255
##      CustomerCode
## Y              M-1          M-2          N-1          P-4          P-5
##   NO  0.0155746509 0.0067132116 0.0464554243 0.0093984962 0.0059076262
##   YES 0.0351437700 0.0149094782 0.0095846645 0.0106496273 0.0330138445
##      CustomerCode
```

```
## Y              PC           PD           RC          S-3          T-2
##   NO  0.0008055854 0.0021482277 0.0016111708 0.0190655209 0.0077873255
##   YES 0.0000000000 0.1043663472 0.0031948882 0.0149094782 0.0085197018
##      CustomerCode
## Y             T-4          T-5          TGT          V-1
##   NO  0.0002685285 0.0249731472 0.0338345865 0.0005370569
##   YES 0.0000000000 0.0276890309 0.0223642173 0.0000000000
##
##      CountryName
## Y        AUSTRALIA      BELGIUM       BRAZIL       CANADA        CHINA
##   NO  0.0016111708 0.0021482277 0.0013426423 0.0005370569 0.0000000000
##   YES 0.0031948882 0.1043663472 0.0000000000 0.0106496273 0.0000000000
##      CountryName
## Y            INDIA       ISRAEL        ITALY       POLAND      ROMANIA
##   NO  0.7303974221 0.0000000000 0.0077873255 0.0002685285 0.0024167562
##   YES 0.4664536741 0.0031948882 0.0085197018 0.0010649627 0.0074547391
##      CountryName
## Y     SOUTH AFRICA          UAE           UK          USA
##   NO  0.0024167562 0.0005370569 0.0290010741 0.2215359828
##   YES 0.0010649627 0.0000000000 0.0628328009 0.3312034079
##
##      ITEM_NAME
## Y     DOUBLE BACK       DURRY  GUN TUFTED HAND TUFTED    HANDLOOM   HANDWOVEN
##   NO  0.102846402 0.284103115 0.004296455 0.421589689 0.017722879 0.133727175
##   YES 0.068157614 0.202342918 0.014909478 0.380191693 0.021299255 0.079872204
##      ITEM_NAME
## Y     INDO-TIBBETAN    JACQUARD     KNOTTED POWER LOOM JACQUARD TABLE TUFTED
##   NO    0.000000000 0.014500537 0.017991407         0.001611171  0.001611171
##   YES   0.002129925 0.014909478 0.117145900         0.085197018  0.013844515
##
##      Hand Tufted
## Y             0         1
##   NO  0.5784103 0.4215897
##   YES 0.6198083 0.3801917
##
##      Durry
## Y             0         1
##   NO  0.7158969 0.2841031
##   YES 0.7976571 0.2023429
##
##      Double Back
## Y              0          1
##   NO  0.89715360 0.10284640
##   YES 0.93184239 0.06815761
##
##      Hand Woven
## Y             0         1
##   NO  0.8662728 0.1337272
##   YES 0.9201278 0.0798722
##
##      Knotted
## Y              0          1
##   NO  0.98200859 0.01799141
##   YES 0.88285410 0.11714590
```

```
##
##      Jacquard
## Y              0          1
##   NO  0.98549946 0.01450054
##   YES 0.98509052 0.01490948
##
##      Handloom
## Y              0          1
##   NO  0.98227712 0.01772288
##   YES 0.97870075 0.02129925
##
##      Other
## Y               0           1
##   NO  0.992481203 0.007518797
##   YES 0.883919063 0.116080937
##
##      ShapeName
## Y             REC       ROUND      SQUARE
##   NO  0.988721805 0.008324382 0.002953813
##   YES 0.976570820 0.017039404 0.006389776
##
##      REC
## Y              0          1
##   NO  0.01127820 0.98872180
##   YES 0.02342918 0.97657082
##
##      Round
## Y               0           1
##   NO  0.991675618 0.008324382
##   YES 0.982960596 0.017039404
##
##      Square
## Y               0           1
##   NO  0.997046187 0.002953813
##   YES 0.993610224 0.006389776
```

We then evaluate our model on the Test set:

```r
nb1.pred <- predict(nb1, newdata = test_nb, type = 'class')
head(nb1.pred)
```

```
## [1] NO  NO  NO  YES YES YES
## Levels: NO YES
```

We then create the confusion matrix:

```r
nb1.cm <- table(true = test_nb$target, predicted = nb1.pred)
nb1.cm
```

```
##      predicted
## true   NO YES
##   NO  885  42
##   YES 148  82
```

Summary

The diagonal cells of the table indicate the number of correct predictions, while the off-diagonal cells indicate the number of incorrect predictions. In the table, **885** instances with a true category of order conversion “not happening” were correctly predicted as not happening, and **82** instances with a true category of order conversion “happening” were correctly predicted as happening. However, **42** instances where the true category was order conversion “not happening” were incorrectly predicted as happening, and **148** instances where the true category was “happening” were incorrectly predicted as not happening.

Logistic Regression

The original data is first divided into a test set and a training set based on a ratio of 80:20

```
DOS_logit <- DOS#creating a copy

set.seed(256)
index_logit <- sample(2, nrow(DOS_logit), replace = T, prob = c(0.8, 0.2))
train_logit <- DOS_logit[index_logit == 1, ]
test_logit <- DOS_logit[index_logit == 2, ]
```

Use the glm() function to create a logistic regression model. The predicted variable is 'target'. 'family = "binomial" ' specifies the type of probability distribution used in the logistic regression model, in this case binomial, which is suitable for binary classification problems. Here, we're excluding **CustomerCode** and **CountryName**, as they both introduce new levels post running the glm function, which skews the data model:

```
logitModel <- glm(target ~ ., data = train_logit[, !colnames(train_logit) %in%
                    c("CustomerCode", "CountryName")], family = "binomial")

summary(logitModel)
```

```
##
## Call:
## glm(formula = target ~ ., family = "binomial", data = train_logit[,
##     !colnames(train_logit) %in% c("CustomerCode", "CountryName")])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1821  -0.5809  -0.2811  -0.1951   2.9272
##
## Coefficients: (12 not defined because of singularities)
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -4.041165   0.206951 -19.527  < 2e-16 ***
## USA                          1.475512   0.108047  13.656  < 2e-16 ***
## UK                           1.794512   0.190534   9.418  < 2e-16 ***
## Italy                       -0.106178   0.573622  -0.185  0.85315
## Belgium                      5.737831   0.404424  14.188  < 2e-16 ***
## Romania                      3.382529   0.566457   5.971 2.35e-09 ***
## Australia                   -0.576198   0.895232  -0.644  0.51982
## India                              NA         NA      NA       NA
## QtyRequired                  0.009723   0.006914   1.406  0.15963
## ITEM_NAMEDURRY               0.324345   0.205874   1.575  0.11515
## ITEM_NAMEGUN TUFTED          2.691611   0.428164   6.286 3.25e-10 ***
## ITEM_NAMEHAND TUFTED         0.081567   0.191881   0.425  0.67077
## ITEM_NAMEHANDLOOM            0.415836   0.354444   1.173  0.24071
## ITEM_NAMEHANDWOVEN          -0.478026   0.253828  -1.883  0.05966 .
## ITEM_NAMEINDO-TIBBETAN      16.244106 309.121056   0.053  0.95809
## ITEM_NAMEJACQUARD            0.650279   0.368719   1.764  0.07780 .
## ITEM_NAMEKNOTTED             2.762514   0.259651  10.639  < 2e-16 ***
## ITEM_NAMEPOWER LOOM JACQUARD 5.547777   0.471715  11.761  < 2e-16 ***
## ITEM_NAMETABLE TUFTED        3.302655   0.524286   6.299 2.99e-10 ***
## `Hand Tufted`1                     NA         NA      NA       NA
```

```
## Durry1                              NA         NA      NA       NA
## `Double Back`1                      NA         NA      NA       NA
## `Hand Woven`1                       NA         NA      NA       NA
## Knotted1                            NA         NA      NA       NA
## Jacquard1                           NA         NA      NA       NA
## Handloom1                           NA         NA      NA       NA
## Other1                              NA         NA      NA       NA
## ShapeNameROUND                0.949585   0.356673   2.662  0.00776 **
## ShapeNameSQUARE               1.067743   0.649430   1.644  0.10015
## REC1                                NA         NA      NA       NA
## Round1                              NA         NA      NA       NA
## Square1                             NA         NA      NA       NA
## AreaFt                        0.058715   0.002635  22.281  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4734.9  on 4630  degrees of freedom
## Residual deviance: 3075.6  on 4610  degrees of freedom
## AIC: 3117.6
##
## Number of Fisher Scoring iterations: 12
```

Post that, we calculate the residual deviance of the logistic regression model:

```
rd <- summary(logitModel)$deviance
1-pchisq(rd, 10)
```

```
## [1] 0
```

After that, we move onto predicting our model:

```
Pred <- predict(logitModel, newdata = test_logit, type = "response")
Pred
```

```
##          1          2          3          4          5          6          7
## 0.66054477 0.07883188 0.07883188 0.14808382 0.08881867 0.02462272 0.09669362
##          8          9         10         11         12         13         14
## 0.07305200 0.90227847 0.43100132 0.26258417 0.09754617 0.53442959 0.57313556
##         15         16         17         18         19         20         21
## 0.67858781 0.72140982 0.99855353 0.12010380 0.24374745 0.02712115 0.03373801
##         22         23         24         25         26         27         28
## 0.67858781 0.98748591 0.96857545 0.38032177 0.06833381 0.06833381 0.12856829
##         29         30         31         32         33         34         35
## 0.22480939 0.83293638 0.32164678 0.40121660 0.73277565 0.53449123 0.04439758
##         36         37         38         39         40         41         42
## 0.03290257 0.20269679 0.24880922 0.67858781 0.91764282 0.91837463 0.90972694
##         43         44         45         46         47         48         49
## 0.90565430 0.99998859 0.67858781 0.04162326 0.05865791 0.02377596 0.02195450
##         50         51         52         53         54         55         56
## 0.16626926 0.16626926 0.16780098 0.16780098 0.04162326 0.13179924 0.02400267
##         57         58         59         60         61         62         63
## 0.13179924 0.13179924 0.30524681 0.02069458 0.02069458 0.01142342 0.01142342
##         64         65         66         67         68         69         70
## 0.16626926 0.02377596 0.02377596 0.02059306 0.02059306 0.02059306 0.02059306
##         71         72         73         74         75         76         77
## 0.26418152 0.90967972 0.09795409 0.16626926 0.15526775 0.15526775 0.16626926
##         78         79         80         81         82         83         84
## 0.16626926 0.20448179 0.02666475 0.20448179 0.09129306 0.09129306 0.05926404
##         85         86         87         88         89         90         91
## 0.05926404 0.69736754 0.16780098 0.03373801 0.20269679 0.26045804 0.05764834
##         92         93         94         95         96         97         98
## 0.26258417 0.20269679 0.03011252 0.03011252 0.74433926 0.02665951 0.16626926
##         99        100        101        102        103        104        105
## 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926
##        106        107        108        109        110        111        112
## 0.02195450 0.09129306 0.02566235 0.02665951 0.02216425 0.57313556 0.57551251
##        113        114        115        116        117        118        119
## 0.05100301 0.16626926 0.16780098 0.02665951 0.16626926 0.16626926 0.37901920
##        120        121        122        123        124        125        126
## 0.16626926 0.02665951 0.17641134 0.02665951 0.13179924 0.08028351 0.13179924
##        127        128        129        130        131        132        133
## 0.13179924 0.30689087 0.03118540 0.16626926 0.02020450 0.16626926 0.16626926
##        134        135        136        137        138        139        140
## 0.10552019 0.08651615 0.08651615 0.25331662 0.55822429 0.06771741 0.28564290
##        141        142        143        144        145        146        147
```

```
## 0.74642861 0.09129306 0.09129306 0.05643071 0.82119156 0.03187545 0.03187545
##        148        149        150        151        152        153        154
## 0.08651615 0.15869041 0.16626926 0.16626926 0.16626926 0.02553179 0.15526775
##        155        156        157        158        159        160        161
## 0.05643071 0.20427262 0.16626926 0.08028351 0.09129306 0.74642861 0.16626926
##        162        163        164        165        166        167        168
## 0.16626926 0.16626926 0.16626926 0.02377596 0.02377596 0.02377596 0.16626926
##        169        170        171        172        173        174        175
## 0.08925134 0.08846417 0.08846417 0.08846417 0.16626926 0.16626926 0.59920962
##        176        177        178        179        180        181        182
## 0.09667403 0.20448179 0.67858781 0.16780098 0.16780098 0.67858781 0.02411460
##        183        184        185        186        187        188        189
## 0.02411460 0.02411460 0.09582829 0.09498917 0.16780098 0.09667403 0.16780098
##        190        191        192        193        194        195        196
## 0.09796350 0.10331678 0.09625856 0.10331678 0.09625856 0.10331678 0.11500639
##        197        198        199        200        201        202        203
## 0.08939275 0.11599968 0.11500639 0.13347680 0.11599968 0.74642861 0.02195450
##        204        205        206        207        208        209        210
## 0.97265098 0.02462272 0.09625856 0.09625856 0.09625856 0.04656808 0.32322122
##        211        212        213        214        215        216        217
## 0.37901920 0.60962107 0.55056197 0.55056197 0.46859610 0.46859610 0.04309410
##        218        219        220        221        222        223        224
## 0.09129306 0.02195450 0.26229588 0.16780098 0.09625856 0.89730276 0.09625856
##        225        226        227        228        229        230        231
## 0.09625856 0.09625856 0.09625856 0.09625856 0.08939275 0.09625856 0.13179924
##        232        233        234        235        236        237        238
## 0.54815499 0.16626926 0.14147701 0.12057412 0.03011252 0.03011252 0.02691296
##        239        240        241        242        243        244        245
## 0.09625856 0.02462272 0.09625856 0.09625856 0.09625856 0.09710769 0.02665951
##        246        247        248        249        250        251        252
## 0.02377596 0.02377596 0.87484088 0.16626926 0.21415283 0.02234311 0.28564290
##        253        254        255        256        257        258        259
## 0.02234311 0.02234311 0.02234311 0.85807527 0.08939275 0.21788522 0.09625856
##        260        261        262        263        264        265        266
## 0.16780098 0.46859610 0.09625856 0.53499807 0.04229173 0.20448179 0.16780098
##        267        268        269        270        271        272        273
## 0.02150421 0.46859610 0.04349681 0.16780098 0.46859610 0.02234311 0.02234311
##        274        275        276        277        278        279        280
## 0.02234311 0.29355310 0.29355310 0.01950709 0.09625856 0.09129306 0.60860262
##        281        282        283        284        285        286        287
## 0.09625856 0.12951695 0.16780098 0.04309410 0.04309410 0.74433926 0.74433926
##        288        289        290        291        292        293        294
## 0.10230287 0.10230287 0.21788522 0.97946584 0.15526775 0.03348077 0.09129306
##        295        296        297        298        299        300        301
## 0.04309410 0.04309410 0.20269679 0.16780098 0.12951695 0.78566968 0.74433926
##        302        303        304        305        306        307        308
## 0.46859610 0.46859610 0.32718314 0.31657290 0.09018736 0.16626926 0.13347680
##        309        310        311        312        313        314        315
## 0.05790099 0.22832142 0.05790099 0.09796924 0.10331833 0.16780098 0.46636182
##        316        317        318        319        320        321        322
## 0.74433926 0.16626926 0.09625856 0.16780098 0.05737291 0.05737291 0.16780098
##        323        324        325        326        327        328        329
## 0.20606791 0.09625856 0.09210285 0.09129306 0.72934563 0.72742210 0.02509407
##        330        331        332        333        334        335        336
```

```
## 0.29716058 0.09428885 0.94183682 0.02898655 0.10214658 0.05865791 0.05865791
##        337        338        339        340        341        342        343
## 0.26258417 0.16626926 0.23658574 0.16626926 0.16626926 0.02216425 0.93242645
##        344        345        346        347        348        349        350
## 0.04309410 0.04309410 0.04309410 0.10326592 0.03092585 0.04481191 0.04481191
##        351        352        353        354        355        356        357
## 0.05865791 0.66054477 0.02665951 0.02665951 0.03163195 0.31447311 0.91694315
##        358        359        360        361        362        363        364
## 0.15526775 0.03379683 0.03379683 0.04309410 0.04349681 0.04349681 0.67858781
##        365        366        367        368        369        370        371
## 0.49872030 0.02400267 0.04309410 0.04309410 0.06406371 0.03011252 0.03011252
##        372        373        374        375        376        377        378
## 0.96584266 0.63843868 0.02101434 0.03373801 0.03373801 0.01541042 0.01541042
##        379        380        381        382        383        384        385
## 0.03373801 0.02001293 0.02195450 0.02195450 0.04309410 0.28564290 0.04349681
##        386        387        388        389        390        391        392
## 0.02001293 0.01541042 0.01541042 0.01541042 0.10696957 0.02377596 0.02377596
##        393        394        395        396        397        398        399
## 0.20766312 0.02377596 0.02377596 0.95290903 0.03405641 0.26229588 0.10696957
##        400        401        402        403        404        405        406
## 0.02665951 0.10696957 0.03373801 0.02195450 0.02195450 0.10696957 0.10790190
##        407        408        409        410        411        412        413
## 0.10696957 0.10884137 0.02400267 0.02400267 0.16780098 0.01541042 0.01541042
##        414        415        416        417        418        419        420
## 0.10790190 0.10696957 0.05695065 0.10790190 0.03321336 0.16626926 0.86629879
##        421        422        423        424        425        426        427
## 0.86629879 0.86629879 0.03290257 0.16626926 0.21976710 0.13359085 0.16780098
##        428        429        430        431        432        433        434
## 0.03373801 0.20606791 0.09129306 0.09129306 0.09129306 0.09129306 0.04309410
##        435        436        437        438        439        440        441
## 0.09129306 0.09129306 0.09129306 0.09129306 0.16780098 0.16916308 0.16626926
##        442        443        444        445        446        447        448
## 0.16626926 0.16780098 0.01956962 0.01956962 0.01956962 0.07305200 0.07305200
##        449        450        451        452        453        454        455
## 0.16780098 0.16780098 0.16780098 0.01941082 0.02665951 0.16626926 0.13359085
##        456        457        458        459        460        461        462
## 0.09210285 0.09129306 0.09129306 0.03011252 0.04309410 0.04309410 0.16626926
##        463        464        465        466        467        468        469
## 0.16626926 0.16626926 0.16626926 0.16626926 0.16780098 0.07128478 0.16626926
##        470        471        472        473        474        475        476
## 0.16780098 0.16626926 0.16780098 0.10884137 0.10884137 0.10884137 0.10884137
##        477        478        479        480        481        482        483
## 0.10884137 0.92100969 0.20269679 0.54678683 0.10331833 0.01372643 0.09667403
##        484        485        486        487        488        489        490
## 0.09667403 0.09667403 0.22901907 0.46859610 0.16780098 0.90227847 0.02377596
##        491        492        493        494        495        496        497
## 0.06464915 0.03373801 0.20448179 0.94909194 0.16626926 0.06464915 0.06464915
##        498        499        500        501        502        503        504
## 0.20448179 0.16780098 0.04439758 0.16780098 0.46859610 0.63843868 0.02400267
##        505        506        507        508        509        510        511
## 0.03872805 0.02665951 0.03373801 0.20269679 0.02629611 0.46859610 0.02665951
##        512        513        514        515        516        517        518
## 0.08414581 0.02366120 0.06406371 0.03373801 0.03373801 0.01865143 0.66054477
##        519        520        521        522        523        524        525
```

```
## 0.31657290 0.20427262 0.20269679 0.15526775 0.05591524 0.09752642 0.09752642
##        526        527        528        529        530        531        532
## 0.09752642 0.03373801 0.20269679 0.03997649 0.16626926 0.03348077 0.32164678
##        533        534        535        536        537        538        539
## 0.47029198 0.47029198 0.47029198 0.16626926 0.09129306 0.04229173 0.01248004
##        540        541        542        543        544        545        546
## 0.03405641 0.03405641 0.03373801 0.03373801 0.15671718 0.03057572 0.03057572
##        547        548        549        550        551        552        553
## 0.02915816 0.20606791 0.02377596 0.20269679 0.16626926 0.02400267 0.02140017
##        554        555        556        557        558        559        560
## 0.02377596 0.28564290 0.02377596 0.72910129 0.02063071 0.01322774 0.03405641
##        561        562        563        564        565        566        567
## 0.03405641 0.46859610 0.46859610 0.03011252 0.03011252 0.03011252 0.03011252
##        568        569        570        571        572        573        574
## 0.03011252 0.03011252 0.03405641 0.24455647 0.10117855 0.02377596 0.05591524
##        575        576        577        578        579        580        581
## 0.03011252 0.02693724 0.02462272 0.10230287 0.13574194 0.11700041 0.09645527
##        582        583        584        585        586        587        588
## 0.09477382 0.09477382 0.03205002 0.16626926 0.02377596 0.02665951 0.02686579
##        589        590        591        592        593        594        595
## 0.03039778 0.13069067 0.05105872 0.16626926 0.16626926 0.05865791 0.13359085
##        596        597        598        599        600        601        602
## 0.20585754 0.13359085 0.53499807 0.10696957 0.06771741 0.02377596 0.02665951
##        603        604        605        606        607        608        609
## 0.10696957 0.79342040 0.16626926 0.20448179 0.02195450 0.02377596 0.03872805
##        610        611        612        613        614        615        616
## 0.15869041 0.10696957 0.49566832 0.16626926 0.16626926 0.46859610 0.02665951
##        617        618        619        620        621        622        623
## 0.14777669 0.12247088 0.01956962 0.01956962 0.02195450 0.03373801 0.03373801
##        624        625        626        627        628        629        630
## 0.03373801 0.02771242 0.02195450 0.03405641 0.27267905 0.02377596 0.02377596
##        631        632        633        634        635        636        637
## 0.02377596 0.38859044 0.13347680 0.13347680 0.13347680 0.09477382 0.12247088
##        638        639        640        641        642        643        644
## 0.12142979 0.07319931 0.31657290 0.14979862 0.20269679 0.20269679 0.20269679
##        645        646        647        648        649        650        651
## 0.15104111 0.14979862 0.15738099 0.24455647 0.66381946 0.15104111 0.03470192
##        652        653        654        655        656        657        658
## 0.03470192 0.03373801 0.03373801 0.09129306 0.03470192 0.23489831 0.16626926
##        659        660        661        662        663        664        665
## 0.02665951 0.02377596 0.02462272 0.46859610 0.14979862 0.14979862 0.66381946
##        666        667        668        669        670        671        672
## 0.01815173 0.01815173 0.01889294 0.01889294 0.01889294 0.01889294 0.01815173
##        673        674        675        676        677        678        679
## 0.01815173 0.14979862 0.74642861 0.03373801 0.10230287 0.02377596 0.02377596
##        680        681        682        683        684        685        686
## 0.07319931 0.12340091 0.12340091 0.12340091 0.12340091 0.12340091 0.46859610
##        687        688        689        690        691        692        693
## 0.12340091 0.12340091 0.12340091 0.12340091 0.03373801 0.03373801 0.03373801
##        694        695        696        697        698        699        700
## 0.03373801 0.15532506 0.03011252 0.03011252 0.03011252 0.03011252 0.03011252
##        701        702        703        704        705        706        707
## 0.03011252 0.03011252 0.03373801 0.01385868 0.13156199 0.01541042 0.01541042
##        708        709        710        711        712        713        714
```

```
## 0.20269679 0.13156199 0.13156199 0.13267683 0.13156199 0.94855784 0.03373801
##        715        716        717        718        719        720        721
## 0.03373801 0.06771741 0.06771741 0.66054477 0.66054477 0.66054477 0.03373801
##        722        723        724        725        726        727        728
## 0.02561421 0.86742092 0.13267683 0.03373801 0.03373801 0.03373801 0.03405641
##        729        730        731        732        733        734        735
## 0.03405641 0.03872805 0.13156199 0.03373801 0.13156199 0.16780098 0.13156199
##        736        737        738        739        740        741        742
## 0.21788522 0.21788522 0.13156199 0.16626926 0.46859610 0.07967206 0.07967206
##        743        744        745        746        747        748        749
## 0.09129306 0.03740384 0.16626926 0.07305200 0.13156199 0.10331833 0.47586548
##        750        751        752        753        754        755        756
## 0.68863158 0.04439758 0.32377184 0.10810155 0.02036203 0.02377596 0.10230287
##        757        758        759        760        761        762        763
## 0.03373801 0.03373801 0.01832583 0.18793318 0.01541042 0.16186183 0.03405641
##        764        765        766        767        768        769        770
## 0.03373801 0.02377596 0.54542686 0.09129306 0.16186183 0.90813459 0.10230287
##        771        772        773        774        775        776        777
## 0.16186183 0.07967206 0.04309410 0.90227847 0.02462272 0.46859610 0.09129306
##        778        779        780        781        782        783        784
## 0.20269679 0.16626926 0.16780098 0.01541042 0.03011252 0.16626926 0.13347680
##        785        786        787        788        789        790        791
## 0.04134283 0.03909163 0.03011252 0.03011252 0.03011252 0.03373801 0.16626926
##        792        793        794        795        796        797        798
## 0.20572865 0.20572865 0.02195450 0.16626926 0.03373801 0.03373801 0.03373801
##        799        800        801        802        803        804        805
## 0.03373801 0.16626926 0.47101790 0.09129306 0.16626926 0.16626926 0.16626926
##        806        807        808        809        810        811        812
## 0.03373801 0.16626926 0.03373801 0.09129306 0.46859610 0.20572865 0.16887166
##        813        814        815        816        817        818        819
## 0.16887166 0.04309410 0.16887166 0.19663520 0.19663520 0.03872805 0.20269679
##        820        821        822        823        824        825        826
## 0.28763095 0.06561720 0.09129306 0.09129306 0.03373801 0.03373801 0.03373801
##        827        828        829        830        831        832        833
## 0.03373801 0.04309410 0.46859610 0.20572865 0.16887166 0.16887166 0.16887166
##        834        835        836        837        838        839        840
## 0.16780098 0.16626926 0.06783378 0.03373801 0.16626926 0.16626926 0.16626926
##        841        842        843        844        845        846        847
## 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926 0.16626926
##        848        849        850        851        852        853        854
## 0.46859610 0.16887166 0.01790968 0.01790968 0.10230287 0.16626926 0.16626926
##        855        856        857        858        859        860        861
## 0.16626926 0.01541042 0.16887166 0.01541042 0.20732193 0.10677924 0.01773947
##        862        863        864        865        866        867        868
## 0.97961909 0.21415283 0.01773947 0.01773947 0.20572865 0.02001293 0.04439758
##        869        870        871        872        873        874        875
## 0.97961909 0.10402900 0.01773947 0.01773947 0.46859610 0.20269724 0.44754957
##        876        877        878        879        880        881        882
## 0.02446245 0.04309410 0.01773947 0.20074589 0.24166673 0.20269679 0.13069067
##        883        884        885        886        887        888        889
## 0.02665951 0.40697127 0.33141831 0.42619031 0.54678683 0.19510384 0.10331833
##        890        891        892        893        894        895        896
## 0.16626926 0.09129306 0.01541042 0.01541042 0.04309410 0.04309410 0.04309410
##        897        898        899        900        901        902        903
```

```
## 0.30524681 0.02665951 0.05865791 0.24107757 0.31146740 0.03502908 0.09129306
##        904        905        906        907        908        909        910
## 0.03373801 0.47493255 0.10331833 0.10331833 0.01985325 0.25631428 0.83420588
##        911        912        913        914        915        916        917
## 0.88202411 0.88202411 0.02377596 0.88303208 0.16588388 0.01541042 0.03373801
##        918        919        920        921        922        923        924
## 0.98346729 0.90155191 0.46859610 0.47101790 0.32859224 0.10331833 0.33074081
##        925        926        927        928        929        930        931
## 0.33074081 0.33074081 0.10422256 0.32859224 0.21415283 0.03909163 0.33289647
##        932        933        934        935        936        937        938
## 0.01541042 0.16626926 0.01372643 0.28564290 0.02665951 0.03011252 0.33074081
##        939        940        941        942        943        944        945
## 0.33289647 0.01555864 0.39667135 0.10230287 0.39667135 0.04309410 0.40601516
##        946        947        948        949        950        951        952
## 0.02822617 0.02150421 0.02150421 0.02623349 0.02623349 0.16626926 0.97250673
##        953        954        955        956        957        958        959
## 0.80741428 0.79660950 0.80892160 0.82929652 0.79660950 0.74642861 0.10331833
##        960        961        962        963        964        965        966
## 0.16626926 0.39667135 0.54815499 0.32164678 0.54815499 0.03373801 0.97276550
##        967        968        969        970        971        972        973
## 0.83066849 0.79502967 0.83066849 0.83066849 0.02665951 0.03373801 0.16626926
##        974        975        976        977        978        979        980
## 0.03373801 0.80741428 0.03872805 0.03872805 0.03909163 0.03872805 0.03872805
##        981        982        983        984        985        986        987
## 0.03872805 0.17943661 0.03470192 0.03872805 0.03872805 0.03872805 0.03872805
##        988        989        990        991        992        993        994
## 0.91627229 0.16626926 0.16626926 0.39900053 0.03039778 0.03872805 0.39667135
##        995        996        997        998        999       1000       1001
## 0.39667135 0.39667135 0.39667135 0.39667135 0.91660234 0.39667135 0.83203167
##       1002       1003       1004       1005       1006       1007       1008
## 0.83203167 0.43656808 0.97062337 0.39667135 0.39667135 0.42490199 0.39667135
##       1009       1010       1011       1012       1013       1014       1015
## 0.02358698 0.02358698 0.20269679 0.91660234 0.91660234 0.26045804 0.91660234
##       1016       1017       1018       1019       1020       1021       1022
## 0.39667135 0.01541042 0.03373801 0.91660234 0.28564290 0.03872805 0.03872805
##       1023       1024       1025       1026       1027       1028       1029
## 0.20269679 0.20269679 0.10230287 0.03872805 0.03872805 0.39667135 0.27267905
##       1030       1031       1032       1033       1034       1035       1036
## 0.15526775 0.03872805 0.03872805 0.03872805 0.03872805 0.39900053 0.02665951
##       1037       1038       1039       1040       1041       1042       1043
## 0.16626926 0.03872805 0.03872805 0.03872805 0.03872805 0.26045804 0.15671718
##       1044       1045       1046       1047       1048       1049       1050
## 0.46859610 0.43656808 0.10310894 0.39900053 0.10230287 0.03373801 0.03373801
##       1051       1052       1053       1054       1055       1056       1057
## 0.05289538 0.16082700 0.05289538 0.46859610 0.39900053 0.03373801 0.40133429
##       1058       1059       1060       1061       1062       1063       1064
## 0.16626926 0.46079801 0.04242684 0.02665951 0.02665951 0.02665951 0.02665951
##       1065       1066       1067       1068       1069       1070       1071
## 0.02665951 0.10230287 0.03930241 0.46563299 0.03872805 0.03872805 0.02485731
##       1072       1073       1074       1075       1076       1077       1078
## 0.91660234 0.91660234 0.91660234 0.05591524 0.03872805 0.03872805 0.04086468
##       1079       1080       1081       1082       1083       1084       1085
## 0.20269679 0.16626926 0.16626926 0.20269679 0.03872805 0.03872805 0.03872805
##       1086       1087       1088       1089       1090       1091       1092
```

```
## 0.46585569 0.46585569 0.46585569 0.21415283 0.92791983 0.54854122 0.53163759
##       1093       1094       1095       1096       1097       1098       1099
## 0.16626926 0.20606791 0.16626926 0.03872805 0.10230287 0.55193395 0.16626926
##       1100       1101       1102       1103       1104       1105       1106
## 0.16626926 0.16626926 0.01808150 0.01773947 0.01808150 0.05695065 0.91660234
##       1107       1108       1109       1110       1111       1112       1113
## 0.03872805 0.03872805 0.16626926 0.01773947 0.09129306 0.01773947 0.07359011
##       1114       1115       1116       1117       1118       1119       1120
## 0.55193395 0.16780098 0.91660234 0.91660234 0.94909194 0.94909194 0.02665951
##       1121       1122       1123       1124       1125       1126       1127
## 0.55193395 0.55193395 0.01773947 0.01773947 0.03011252 0.04095920 0.91660234
##       1128       1129       1130       1131       1132       1133       1134
## 0.91660234 0.32164678 0.02001293 0.02001293 0.16626926 0.09752642 0.63373963
##       1135       1136       1137       1138       1139       1140       1141
## 0.44834797 0.44834797 0.01222418 0.74642861 0.01773947 0.01555864 0.89730276
##       1142       1143       1144       1145       1146       1147       1148
## 0.03348077 0.02377596 0.01997774 0.52921593 0.01773947 0.01773947 0.45075389
##       1149       1150       1151       1152       1153       1154       1155
## 0.53647634 0.53163759 0.02423150 0.52921593 0.03872805 0.33940903 0.60895734
##       1156       1157       1158       1159       1160       1161       1162
## 0.03872805 0.37896834 0.37896834 0.02586271 0.10331833 0.03321336 0.52921593
##       1163       1164       1165       1166       1167       1168       1169
## 0.21586625 0.92791983 0.52921593 0.04439758 0.33723252 0.33723252 0.53405777
##       1170       1171       1172       1173       1174       1175       1176
## 0.52921593 0.33506289 0.95669874 0.03872805 0.66734977 0.02691296 0.09129306
##       1177       1178       1179       1180       1181       1182       1183
## 0.54815499 0.89730276 0.01773947 0.13069067 0.83826545 0.16780098 0.99263591
##       1184       1185       1186       1187       1188       1189
## 0.94728704 0.16626926 0.16626926 0.98538236 0.99657708 0.01997774
```

Lastly, we predict using the predicate that above calculated value of **Pred** is greater than or equal to 0.5

```
Class <- ifelse(Pred >= 0.5, "YES", "NO")
Class
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO" "YES"
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO"
##    27    28    29    30    31    32    33    34    35    36    37    38    39
##  "NO"  "NO"  "NO" "YES"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES"
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## "YES" "YES" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##    53    54    55    56    57    58    59    60    61    62    63    64    65
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##    66    67    68    69    70    71    72    73    74    75    76    77    78
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##    79    80    81    82    83    84    85    86    87    88    89    90    91
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"
##    92    93    94    95    96    97    98    99   100   101   102   103   104
##  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   105   106   107   108   109   110   111   112   113   114   115   116   117
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"
##   118   119   120   121   122   123   124   125   126   127   128   129   130
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   131   132   133   134   135   136   137   138   139   140   141   142   143
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO" "YES"  "NO"  "NO"
##   144   145   146   147   148   149   150   151   152   153   154   155   156
##  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   157   158   159   160   161   162   163   164   165   166   167   168   169
##  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   170   171   172   173   174   175   176   177   178   179   180   181   182
##  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO" "YES"  "NO"  "NO" "YES"  "NO"
##   183   184   185   186   187   188   189   190   191   192   193   194   195
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   196   197   198   199   200   201   202   203   204   205   206   207   208
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"
##   209   210   211   212   213   214   215   216   217   218   219   220   221
##  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   222   223   224   225   226   227   228   229   230   231   232   233   234
##  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"
##   235   236   237   238   239   240   241   242   243   244   245   246   247
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   248   249   250   251   252   253   254   255   256   257   258   259   260
## "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"
##   261   262   263   264   265   266   267   268   269   270   271   272   273
##  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   274   275   276   277   278   279   280   281   282   283   284   285   286
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"
##   287   288   289   290   291   292   293   294   295   296   297   298   299
## "YES"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   300   301   302   303   304   305   306   307   308   309   310   311   312
## "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   313   314   315   316   317   318   319   320   321   322   323   324   325
```

```
##  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   326   327   328   329   330   331   332   333   334   335   336   337   338
##  "NO" "YES" "YES"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   339   340   341   342   343   344   345   346   347   348   349   350   351
##  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   352   353   354   355   356   357   358   359   360   361   362   363   364
## "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"
##   365   366   367   368   369   370   371   372   373   374   375   376   377
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO"
##   378   379   380   381   382   383   384   385   386   387   388   389   390
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   391   392   393   394   395   396   397   398   399   400   401   402   403
##  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   404   405   406   407   408   409   410   411   412   413   414   415   416
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   417   418   419   420   421   422   423   424   425   426   427   428   429
##  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   430   431   432   433   434   435   436   437   438   439   440   441   442
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   443   444   445   446   447   448   449   450   451   452   453   454   455
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   456   457   458   459   460   461   462   463   464   465   466   467   468
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   469   470   471   472   473   474   475   476   477   478   479   480   481
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES"  "NO"
##   482   483   484   485   486   487   488   489   490   491   492   493   494
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO" "YES"
##   495   496   497   498   499   500   501   502   503   504   505   506   507
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"
##   508   509   510   511   512   513   514   515   516   517   518   519   520
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"
##   521   522   523   524   525   526   527   528   529   530   531   532   533
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   534   535   536   537   538   539   540   541   542   543   544   545   546
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   547   548   549   550   551   552   553   554   555   556   557   558   559
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"
##   560   561   562   563   564   565   566   567   568   569   570   571   572
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   573   574   575   576   577   578   579   580   581   582   583   584   585
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   586   587   588   589   590   591   592   593   594   595   596   597   598
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"
##   599   600   601   602   603   604   605   606   607   608   609   610   611
##  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   612   613   614   615   616   617   618   619   620   621   622   623   624
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   625   626   627   628   629   630   631   632   633   634   635   636   637
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   638   639   640   641   642   643   644   645   646   647   648   649   650
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"
##   651   652   653   654   655   656   657   658   659   660   661   662   663
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##   664   665   666   667   668   669   670   671   672   673   674   675   676
```

```
##  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" 
##   677   678   679   680   681   682   683   684   685   686   687   688   689 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   690   691   692   693   694   695   696   697   698   699   700   701   702 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   703   704   705   706   707   708   709   710   711   712   713   714   715 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO" 
##   716   717   718   719   720   721   722   723   724   725   726   727   728 
##  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   729   730   731   732   733   734   735   736   737   738   739   740   741 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   742   743   744   745   746   747   748   749   750   751   752   753   754 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO" 
##   755   756   757   758   759   760   761   762   763   764   765   766   767 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" 
##   768   769   770   771   772   773   774   775   776   777   778   779   780 
##  "NO" "YES"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   781   782   783   784   785   786   787   788   789   790   791   792   793 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   794   795   796   797   798   799   800   801   802   803   804   805   806 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   807   808   809   810   811   812   813   814   815   816   817   818   819 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   820   821   822   823   824   825   826   827   828   829   830   831   832 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   833   834   835   836   837   838   839   840   841   842   843   844   845 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   846   847   848   849   850   851   852   853   854   855   856   857   858 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   859   860   861   862   863   864   865   866   867   868   869   870   871 
##  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO" 
##   872   873   874   875   876   877   878   879   880   881   882   883   884 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   885   886   887   888   889   890   891   892   893   894   895   896   897 
##  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   898   899   900   901   902   903   904   905   906   907   908   909   910 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" 
##   911   912   913   914   915   916   917   918   919   920   921   922   923 
## "YES" "YES"  "NO" "YES"  "NO"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO" 
##   924   925   926   927   928   929   930   931   932   933   934   935   936 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   937   938   939   940   941   942   943   944   945   946   947   948   949 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" 
##   950   951   952   953   954   955   956   957   958   959   960   961   962 
##  "NO"  "NO" "YES" "YES" "YES" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO" "YES" 
##   963   964   965   966   967   968   969   970   971   972   973   974   975 
##  "NO" "YES"  "NO" "YES" "YES" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES" 
##   976   977   978   979   980   981   982   983   984   985   986   987   988 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" 
##   989   990   991   992   993   994   995   996   997   998   999  1000  1001 
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES" 
##  1002  1003  1004  1005  1006  1007  1008  1009  1010  1011  1012  1013  1014 
## "YES"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO" 
##  1015  1016  1017  1018  1019  1020  1021  1022  1023  1024  1025  1026  1027 
```

```
## "YES"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1028  1029  1030  1031  1032  1033  1034  1035  1036  1037  1038  1039  1040
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1041  1042  1043  1044  1045  1046  1047  1048  1049  1050  1051  1052  1053
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1054  1055  1056  1057  1058  1059  1060  1061  1062  1063  1064  1065  1066
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1067  1068  1069  1070  1071  1072  1073  1074  1075  1076  1077  1078  1079
##  "NO"  "NO"  "NO"  "NO"  "NO" "YES" "YES" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1080  1081  1082  1083  1084  1085  1086  1087  1088  1089  1090  1091  1092
##  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES" "YES" "YES"
##  1093  1094  1095  1096  1097  1098  1099  1100  1101  1102  1103  1104  1105
##  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"
##  1106  1107  1108  1109  1110  1111  1112  1113  1114  1115  1116  1117  1118
## "YES"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES" "YES" "YES"
##  1119  1120  1121  1122  1123  1124  1125  1126  1127  1128  1129  1130  1131
## "YES"  "NO" "YES" "YES"  "NO"  "NO"  "NO"  "NO" "YES" "YES"  "NO"  "NO"  "NO"
##  1132  1133  1134  1135  1136  1137  1138  1139  1140  1141  1142  1143  1144
##  "NO"  "NO" "YES"  "NO"  "NO"  "NO" "YES"  "NO"  "NO" "YES"  "NO"  "NO"  "NO"
##  1145  1146  1147  1148  1149  1150  1151  1152  1153  1154  1155  1156  1157
## "YES"  "NO"  "NO"  "NO" "YES" "YES"  "NO" "YES"  "NO"  "NO" "YES"  "NO"  "NO"
##  1158  1159  1160  1161  1162  1163  1164  1165  1166  1167  1168  1169  1170
##  "NO"  "NO"  "NO"  "NO" "YES"  "NO" "YES" "YES"  "NO"  "NO"  "NO" "YES" "YES"
##  1171  1172  1173  1174  1175  1176  1177  1178  1179  1180  1181  1182  1183
##  "NO" "YES"  "NO" "YES"  "NO"  "NO" "YES" "YES"  "NO"  "NO" "YES"  "NO" "YES"
##  1184  1185  1186  1187  1188  1189
## "YES"  "NO"  "NO" "YES" "YES"  "NO"
```

Summary

The Logistic Regression model suggests that the important attributes are: * **ITEM_NAME - Gun Tufted**, **Knotted**, **Table Tufted**, and **Power Loom Jacquard** * **Shape Name (Round)**, and * **AreaFt**

Neural Network

It is important to normalize data before training a neural network. Otherwise, the neural network may have difficulty converging before the maximum number of iterations. We will use the following code to accomplish this task. In the below code, the myscale() function uses the min-max transformation to normalize the variable x. Using mutate_if() from the dplyr package, we can easily apply this function to all numerical variables and normalize them:

```
library(dplyr)

DOS_NN <- DOS#creating a copy

myscale <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

DOS_NN <- DOS_NN %>% mutate_if(is.numeric, myscale)
```

Now, let's split our normalized data into a training set and a test set. We will run our neural network on the training set and then check its performance on the test set:

```
set.seed(1234)
index_NN <- sample(2, nrow(DOS_NN), replace = T, prob = c(0.7, 0.3))
train_NN <- DOS_NN[index_NN == 1, ]
test_NN <- DOS_NN[index_NN == 2, ]
```

We use the "nnet()" function from the "nnet" package to build a neural network model. Here we notice the following arguments:

- **"size"**: this argument specifies how many nodes to have in the hidden layer, skip indicates that the input
- **layer** has a direct connection to the output layer
- **"linout"** specifies the simple identity activation function. This is mainly used if the target variable is numerical. When the linout is False (the target variable is nominal), the entropy function is used as the activation function in the output layer
- **"decay"** is the regularization parameter to avoid over-fitting
- **"maxit"** is the maximum number of iterations for the algorithm that finds the network weights

The below function call will build a neural network with a single hidden layer, formed by ten hidden units. Moreover, the weights will be learned with a weight updating rate of 0.01 (the parameter decay). The parameter linout = FALSE indicates that the target variable is not continuous.

```
library(nnet)
nnModel <- nnet(target ~ ., data = train, linout = FALSE,
                size = 3, decay = 0.01, maxit = 1000)
```

```
## # weights:  241
## initial  value 2911.115425
## iter  10 value 2001.503436
## iter  20 value 1687.970221
## iter  30 value 1374.225686
## iter  40 value 1171.912010
```

```
## iter  50 value 1104.379800
## iter  60 value 1087.959661
## iter  70 value 1072.866516
## iter  80 value 1056.138059
## iter  90 value 1050.159111
## iter 100 value 1029.386498
## iter 110 value 1015.455822
## iter 120 value 1004.988993
## iter 130 value 1001.488875
## iter 140 value 988.148132
## iter 150 value 982.043931
## iter 160 value 975.815167
## iter 170 value 974.470948
## iter 180 value 972.338034
## iter 190 value 971.691799
## iter 200 value 971.663755
## iter 210 value 970.962179
## iter 220 value 965.132076
## iter 230 value 964.491613
## iter 240 value 964.419885
## iter 250 value 964.398632
## iter 260 value 964.362775
## iter 270 value 964.340027
## iter 280 value 964.339371
## final  value 964.338739
## converged
```

The nnet() function uses the back-propagation algorithm as the basis of an iterative process of updating the neural network weights, p, to a maximum of "maxit" cycles. This iterative process may take a long time to compute for large data sets.

The output of the above function is a neural network with **78** input units (predictor variables) connected to **3** hidden units, which will then be linked to a **single** output unit. We can see the final weights (**241**) of these connections by looking at the summary and the neural network plot.

```
summary(nnModel)
```

```
## a 78-3-1 network with 241 weights
## options were - entropy fitting  decay=0.01
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1
##    0.35    2.97    0.00    0.00   -0.64   -5.71    2.42   -0.66    0.78   -1.13
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1 i18->h1 i19->h1
##    0.00    0.90   -2.36   -2.26   -4.03   -1.03   -0.01    0.12    0.00    6.81
## i20->h1 i21->h1 i22->h1 i23->h1 i24->h1 i25->h1 i26->h1 i27->h1 i28->h1 i29->h1
##    0.23   -4.45   -8.21   -5.44   -2.06   -0.54    1.14   -1.40    8.06    0.82
## i30->h1 i31->h1 i32->h1 i33->h1 i34->h1 i35->h1 i36->h1 i37->h1 i38->h1 i39->h1
##   -0.39    4.90    3.84    7.68    1.14   -1.03    0.78    0.00    2.42    0.90
## i40->h1 i41->h1 i42->h1 i43->h1 i44->h1 i45->h1 i46->h1 i47->h1 i48->h1 i49->h1
##    0.82    0.00   -2.26    0.12    0.00    0.87   -2.00   -3.40    0.87    0.82
## i50->h1 i51->h1 i52->h1 i53->h1 i54->h1 i55->h1 i56->h1 i57->h1 i58->h1 i59->h1
##    1.14   -2.26   -1.40    3.19   -0.21   -2.50    0.16    1.05    2.01    3.06
## i60->h1 i61->h1 i62->h1 i63->h1 i64->h1 i65->h1 i66->h1 i67->h1 i68->h1 i69->h1
##    0.22   -0.03   -3.72   -0.20   -1.49    1.05   -2.50    1.79    3.06   -3.72
## i70->h1 i71->h1 i72->h1 i73->h1 i74->h1 i75->h1 i76->h1 i77->h1 i78->h1
##   -0.03    2.01   -1.31   -1.84    0.45    1.73   -1.84    0.45    0.39
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2
##    1.36   -0.75    0.00    0.00    0.09    0.02    5.13    0.04   -1.17   -5.54
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2 i18->h2 i19->h2
##    0.00   -1.70   -2.36   -0.37    1.54    0.66    0.00    0.08    0.00    0.04
## i20->h2 i21->h2 i22->h2 i23->h2 i24->h2 i25->h2 i26->h2 i27->h2 i28->h2 i29->h2
##    1.19    0.01    0.02    3.67   -0.08    0.00   -2.70    0.04    3.04    1.04
## i30->h2 i31->h2 i32->h2 i33->h2 i34->h2 i35->h2 i36->h2 i37->h2 i38->h2 i39->h2
##    0.02   -0.04   -0.52   -0.02   -2.70    0.66   -1.17    0.00    5.13   -1.70
## i40->h2 i41->h2 i42->h2 i43->h2 i44->h2 i45->h2 i46->h2 i47->h2 i48->h2 i49->h2
##    1.04    0.00   -0.37    0.08    0.00    1.50   -1.15   -1.11    1.50    1.04
## i50->h2 i51->h2 i52->h2 i53->h2 i54->h2 i55->h2 i56->h2 i57->h2 i58->h2 i59->h2
##   -2.70   -0.37    0.04    3.00   -7.01    1.79    0.51    1.76    0.42    5.53
## i60->h2 i61->h2 i62->h2 i63->h2 i64->h2 i65->h2 i66->h2 i67->h2 i68->h2 i69->h2
##    0.00    0.74   -3.75   -6.04    2.87    1.76    1.79   -2.48    5.53   -3.75
## i70->h2 i71->h2 i72->h2 i73->h2 i74->h2 i75->h2 i76->h2 i77->h2 i78->h2
##    0.74    0.42   -2.66   -3.59    0.81    4.15   -3.59    0.81   -0.94
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3
##   -0.37    0.97    0.00    0.00   -0.70   -3.86    3.81    5.04   -5.43    0.24
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3 i18->h3 i19->h3
##    0.37   -2.45   -2.14   -2.65   -7.95    0.70    0.00    1.21    4.61    6.93
## i20->h3 i21->h3 i22->h3 i23->h3 i24->h3 i25->h3 i26->h3 i27->h3 i28->h3 i29->h3
##   -7.05   -2.09    2.67   -5.06   -9.38   12.54   -4.71   -0.02    0.09    3.23
## i30->h3 i31->h3 i32->h3 i33->h3 i34->h3 i35->h3 i36->h3 i37->h3 i38->h3 i39->h3
##   -0.67    7.55    5.63   -2.25   -4.71    0.70   -5.43    0.00    3.81   -2.45
## i40->h3 i41->h3 i42->h3 i43->h3 i44->h3 i45->h3 i46->h3 i47->h3 i48->h3 i49->h3
##    3.23    0.37   -2.65    1.21    4.61   -0.40    1.36    1.35   -0.40    3.23
```

```
## i50->h3 i51->h3 i52->h3 i53->h3 i54->h3 i55->h3 i56->h3 i57->h3 i58->h3 i59->h3
##   -4.71   -2.65   -0.02    2.82   -0.41   -1.59    0.46    0.22    3.31    5.34
## i60->h3 i61->h3 i62->h3 i63->h3 i64->h3 i65->h3 i66->h3 i67->h3 i68->h3 i69->h3
##   -0.39    4.26   -5.44   -5.53   -3.17    0.22   -1.59    2.15    5.34   -5.44
## i70->h3 i71->h3 i72->h3 i73->h3 i74->h3 i75->h3 i76->h3 i77->h3 i78->h3
##    4.26    3.31   -8.62   -0.01   -1.54    1.18   -0.01   -1.54   -0.14
##  b->o h1->o h2->o h3->o
## -3.37  6.35 -3.76 -5.02
```

We can use wts to get the best weights found and fitted.values to get the fitted values on training data:

```
options(scipen = 5)
nnModel$wts
```

```
##   [1]  0.34949399376  2.97426179244  0.00011695303  0.00001750815 -0.63667801913
##   [6] -5.71083765808  2.42307336717 -0.65606048806  0.77501927300 -1.12555685595
##  [11] -0.00010241766  0.89873956724 -2.35909443398 -2.26140304808 -4.03408168816
##  [16] -1.03049889765 -0.01369093776  0.11858505463  0.00014542062  6.80781664960
##  [21]  0.22725940332 -4.44500669343 -8.21094891581 -5.44056041301 -2.06208648280
##  [26] -0.54363927621  1.14192711697 -1.40044502191  8.06472397987  0.81819337859
##  [31] -0.38968637424  4.90016864006  3.83589098721  7.68488527907  1.14192416572
##  [36] -1.03072446824  0.77503482301 -0.00015756162  2.42313764457  0.89884302946
##  [41]  0.81790417629  0.00015883586 -2.26143659950  0.11862772130  0.00008999574
##  [46]  0.86615353266 -2.00006971740 -3.40032932567  0.86628443809  0.81817095143
##  [51]  1.14209058679 -2.26136493655 -1.40045198536  3.18501671527 -0.20647709746
##  [56] -2.49572397471  0.15873475884  1.04904432648  2.01192137306  3.06279088101
##  [61]  0.22294629808 -0.03282338969 -3.71849804740 -0.19949859633 -1.49454700597
##  [66]  1.04910200731 -2.49572597726  1.78501555345  3.06260225065 -3.71864057178
##  [71] -0.03275802766  2.01180171565 -1.31206707401 -1.83602115460  0.45384217639
##  [76]  1.73185652745 -1.83587713545  0.45362777014  0.38745281340  1.36430920830
##  [81] -0.75366875298 -0.00015636491 -0.00005747023  0.09343319474  0.01721485460
##  [86]  5.13484784876  0.03734083152 -1.17335621460 -5.54398257118 -0.00018695775
##  [91] -1.70447031656 -2.36176495878 -0.37351781378  1.54362276792  0.66241819201
##  [96] -0.00001758049  0.08202648643 -0.00004905640  0.03512550906  1.18873216534
## [101]  0.00819740522  0.01632044601  3.67001134507 -0.08419495264 -0.00008045941
## [106] -2.69835884273  0.03918572444  3.04446461390  1.04424388935  0.02120546243
## [111] -0.04043050520 -0.52197688344 -0.01840824973 -2.69826108636  0.66245927843
## [116] -1.17334229247 -0.00008671342  5.13484423677 -1.70469733772  1.04418681950
## [121] -0.00005724561 -0.37350567804  0.08200733862  0.00006575766  1.50305935606
## [126] -1.15175003704 -1.11275736592  1.50324496369  1.04429116196 -2.69819491504
## [131] -0.37370987432  0.03924333964  3.00123903266 -7.00977337439  1.79223391895
## [136]  0.51374812194  1.76274136473  0.42093923250  5.53092281424 -0.00137188430
## [141]  0.74069818005 -3.74758981286 -6.04004184108  2.87099592933  1.76246871799
## [146]  1.79228887008 -2.47813491634  5.53106294043 -3.74776350200  0.74060346821
## [151]  0.42087848892 -2.65698410096 -3.58998039148  0.80684760836  4.14760039999
## [156] -3.59017831075  0.80673531700 -0.94490645703 -0.36996168011  0.96618237042
## [161]  0.00013087966  0.00266647778 -0.69518208068 -3.86356203574  3.80885518029
## [166]  5.04423161759 -5.42642949065  0.23650949446  0.37253502896 -2.45265328532
## [171] -2.14021797611 -2.65464704822 -7.95398667845  0.69858822425  0.00474117533
## [176]  1.21464350768  4.60512463150  6.93305106134 -7.04965844797 -2.09249882133
## [181]  2.66595507659 -5.05820057980 -9.37612636950 12.53836736671 -4.71287028816
## [186] -0.01530255379  0.08785313474  3.23132320700 -0.67446748103  7.55419610128
## [191]  5.62966973502 -2.25157535364 -4.71269678916  0.69872123648 -5.42661807321
## [196]  0.00012065351  3.80895434571 -2.45249717772  3.23138516735  0.37250071268
## [201] -2.65449965629  1.21475642478  4.60501918588 -0.39990652816  1.36028763946
## [206]  1.34511658396 -0.39998393723  3.23138223519 -4.71275609585 -2.65447994932
## [211] -0.01544642423  2.82063776808 -0.41152843301 -1.58855236697  0.46470143427
## [216]  0.21696382139  3.31165420564  5.33779925519 -0.38576714382  4.25880188610
## [221] -5.43646387032 -5.53468903072 -3.16600336958  0.21712772825 -1.58830281716
## [226]  2.15135870615  5.33752385798 -5.43615535899  4.25873738628  3.31172146346
## [231] -8.62182728236 -0.00643130397 -1.54238857157  1.17880345480 -0.00647922327
## [236] -1.54217351284 -0.14431252846 -3.37447252235  6.35265198206 -3.76498064519
## [241] -5.01716534528
```

```r
nnModel$fitted.values
```

```
##              [,1]
## 1    0.69265351779
## 2    0.95157855552
## 3    0.25357889024
## 4    0.25357889024
## 5    0.95157855552
## 6    0.13140430360
## 7    0.29738210788
## 8    0.95157855552
## 9    0.95156412075
## 10   0.03311295319
## 11   0.03311295319
## 12   0.01878404145
## 13   0.01878404145
## 14   0.11519333731
## 15   0.01878404145
## 16   0.03309507786
## 17   0.03309507786
## 18   0.00027461640
## 19   0.11653900560
## 20   0.11517059949
## 21   0.03294772486
## 22   0.95157788062
## 23   0.95157855552
## 24   0.03502124219
## 25   0.00052017781
## 26   0.00331405746
## 27   0.00331405746
## 28   0.03366881872
## 29   0.11513888229
## 30   0.80287206140
## 31   0.95157855552
## 32   0.28674516227
## 33   0.11600514627
## 34   0.12270921790
## 35   0.17391515180
## 36   0.69265351779
## 37   0.03379916463
## 38   0.11517068250
## 39   0.95157706327
## 40   0.00300769057
## 41   0.03366881872
## 42   0.00300650332
## 43   0.03356293968
## 44   0.00313245541
## 45   0.59743375669
## 46   0.04637720433
## 47   0.06128884303
## 48   0.12270921790
## 49   0.11522644801
## 50   0.95129901030
```

```
## 51   0.95129901030
## 52   0.50301546359
## 53   0.95157706327
## 54   0.95129901030
## 55   0.00305211043
## 56   0.25357889024
## 57   0.05721281734
## 58   0.06431685156
## 59   0.11758143523
## 60   0.78559441712
## 61   0.95154755970
## 62   0.95136882952
## 63   0.11523179750
## 64   0.00068772356
## 65   0.93282140864
## 66   0.95157802728
## 67   0.11517290767
## 68   0.07965200349
## 69   0.95157802728
## 70   0.95157802728
## 71   0.03310608945
## 72   0.14081865012
## 73   0.85396698031
## 74   0.94721549972
## 75   0.95157855552
## 76   0.85570265029
## 77   0.70394985691
## 78   0.11948408772
## 79   0.03310285724
## 80   0.11519232710
## 81   0.11191423923
## 82   0.11191423923
## 83   0.53347278953
## 84   0.11191423923
## 85   0.53347278953
## 86   0.75265377808
## 87   0.00295999021
## 88   0.11191423923
## 89   0.11191423923
## 90   0.95088255192
## 91   0.11721665850
## 92   0.03366881872
## 93   0.95157855552
## 94   0.00301039593
## 95   0.00301039593
## 96   0.00383728043
## 97   0.00313245541
## 98   0.95157855552
## 99   0.69265351779
## 100  0.95157855552
## 101  0.95157821921
## 102  0.69265351779
## 103  0.01353527113
## 104  0.11465704633
```

```
## 105  0.01353527113
## 106  0.11465704633
## 107  0.24505290713
## 108  0.95157855552
## 109  0.13140430360
## 110  0.92248591303
## 111  0.95157639774
## 112  0.95157855552
## 113  0.12234961130
## 114  0.12234961130
## 115  0.12234961130
## 116  0.95157847846
## 117  0.12234961130
## 118  0.12234961130
## 119  0.11525517310
## 120  0.11529852847
## 121  0.11520682380
## 122  0.11522644801
## 123  0.11529852847
## 124  0.11520682380
## 125  0.11878538023
## 126  0.11517079481
## 127  0.11878538023
## 128  0.03310953191
## 129  0.11560213759
## 130  0.94019111082
## 131  0.10372786974
## 132  0.69265351779
## 133  0.78559441712
## 134  0.94715257759
## 135  0.95157740357
## 136  0.11518480669
## 137  0.11726194192
## 138  0.12357948477
## 139  0.11564452437
## 140  0.49891885282
## 141  0.49891885282
## 142  0.49891885282
## 143  0.49891885282
## 144  0.49891885282
## 145  0.49891885282
## 146  0.49891885282
## 147  0.49891885282
## 148  0.94051090878
## 149  0.55660401215
## 150  0.55660401215
## 151  0.55660401215
## 152  0.55660401215
## 153  0.55660401215
## 154  0.55660401215
## 155  0.55660401215
## 156  0.55660401215
## 157  0.55660401215
## 158  0.55660401215
```

```
## 159  0.00300693239
## 160  0.11517433350
## 161  0.77246654915
## 162  0.00753639509
## 163  0.00753639509
## 164  0.00753639509
## 165  0.94019111082
## 166  0.94019111082
## 167  0.94051090878
## 168  0.00305211043
## 169  0.00305211043
## 170  0.00305211043
## 171  0.00305211043
## 172  0.22829672827
## 173  0.06128884303
## 174  0.06128884303
## 175  0.00305211043
## 176  0.11520962248
## 177  0.00300693239
## 178  0.94227328009
## 179  0.19879909423
## 180  0.19879909423
## 181  0.00523735121
## 182  0.69265351779
## 183  0.94715257759
## 184  0.88797115252
## 185  0.69265351779
## 186  0.78559441712
## 187  0.69265351779
## 188  0.91192616981
## 189  0.78559441712
## 190  0.69265351779
## 191  0.95157855552
## 192  0.95157833267
## 193  0.95157681112
## 194  0.95157855552
## 195  0.00300693239
## 196  0.95101525884
## 197  0.10372786974
## 198  0.11727300429
## 199  0.69265351779
## 200  0.95154526130
## 201  0.10545877037
## 202  0.10545877037
## 203  0.10545877037
## 204  0.10545877037
## 205  0.11727300429
## 206  0.11727300429
## 207  0.11727300429
## 208  0.94715257759
## 209  0.95150794875
## 210  0.95157855552
## 211  0.11727300429
## 212  0.69265351779
```

```
## 213  0.95157855552
## 214  0.95157855552
## 215  0.00300769057
## 216  0.00300769057
## 217  0.06128884303
## 218  0.00305211043
## 219  0.00305211043
## 220  0.00305211043
## 221  0.00300769057
## 222  0.00300769057
## 223  0.00300769057
## 224  0.00300769057
## 225  0.00300769057
## 226  0.11721665850
## 227  0.11721665850
## 228  0.11727300429
## 229  0.00300693239
## 230  0.00313245541
## 231  0.00313245541
## 232  0.00300693239
## 233  0.00300693239
## 234  0.94941078760
## 235  0.00300693239
## 236  0.00300693239
## 237  0.00301458806
## 238  0.11721665850
## 239  0.11721665850
## 240  0.00300769057
## 241  0.00403355310
## 242  0.00403355310
## 243  0.00388005328
## 244  0.00300769057
## 245  0.00300769057
## 246  0.00300769057
## 247  0.11517066524
## 248  0.94862255668
## 249  0.11671085147
## 250  0.11671085147
## 251  0.11671085147
## 252  0.11671085147
## 253  0.00300769057
## 254  0.11671085147
## 255  0.11671085147
## 256  0.11671085147
## 257  0.00305211043
## 258  0.00305211043
## 259  0.00305211043
## 260  0.00305211043
## 261  0.00305211043
## 262  0.00305211043
## 263  0.00305211043
## 264  0.00305211043
## 265  0.00305211043
## 266  0.00305211043
```

```
## 267  0.11671085147
## 268  0.11671085147
## 269  0.11671085147
## 270  0.49007823682
## 271  0.00301760581
## 272  0.00293204828
## 273  0.00301760581
## 274  0.00301760581
## 275  0.00300650332
## 276  0.00300650332
## 277  0.00300650332
## 278  0.00548596019
## 279  0.00548596019
## 280  0.11671085147
## 281  0.11671085147
## 282  0.11671085147
## 283  0.11671085147
## 284  0.11671085147
## 285  0.11671085147
## 286  0.11517066524
## 287  0.11721665850
## 288  0.11721665850
## 289  0.00300693239
## 290  0.00300693239
## 291  0.00300693239
## 292  0.11470857701
## 293  0.11470857701
## 294  0.11470857701
## 295  0.11470857701
## 296  0.11470857701
## 297  0.11470857701
## 298  0.11470857701
## 299  0.11470857701
## 300  0.11470857701
## 301  0.11470857701
## 302  0.94683527875
## 303  0.81921922101
## 304  0.11721665850
## 305  0.03394005074
## 306  0.14282639280
## 307  0.25357889024
## 308  0.11520682380
## 309  0.00331405746
## 310  0.00331405746
## 311  0.11721665850
## 312  0.11517003596
## 313  0.00068772356
## 314  0.12270921790
## 315  0.12270921790
## 316  0.35616056849
## 317  0.14282639280
## 318  0.12270921790
## 319  0.11721665850
## 320  0.11516841335
```

```
## 321  0.11553641158
## 322  0.11553641158
## 323  0.11553641158
## 324  0.11553641158
## 325  0.11721665850
## 326  0.11721665850
## 327  0.11727300429
## 328  0.11769504254
## 329  0.92899528133
## 330  0.11721665850
## 331  0.20125788696
## 332  0.20125788696
## 333  0.20125788696
## 334  0.00291137805
## 335  0.10716149301
## 336  0.20125788696
## 337  0.24505290713
## 338  0.20125788696
## 339  0.10716149301
## 340  0.20125788696
## 341  0.11517068250
## 342  0.00326242303
## 343  0.06128884303
## 344  0.00300769057
## 345  0.11721665850
## 346  0.12270921790
## 347  0.12270921790
## 348  0.12270921790
## 349  0.12270921790
## 350  0.12270921790
## 351  0.12270921790
## 352  0.12270921790
## 353  0.12270921790
## 354  0.00342157646
## 355  0.89659031719
## 356  0.95157855552
## 357  0.95157833267
## 358  0.94715257759
## 359  0.11727300429
## 360  0.11727300429
## 361  0.14095883622
## 362  0.11727300429
## 363  0.95156950730
## 364  0.95157681112
## 365  0.94862255668
## 366  0.13140430360
## 367  0.00301039593
## 368  0.12270921790
## 369  0.11721665850
## 370  0.11517066524
## 371  0.11517068250
## 372  0.00326242303
## 373  0.11517068250
## 374  0.00326242303
```

```
## 375  0.13455210963
## 376  0.00302738675
## 377  0.19879909423
## 378  0.19879909423
## 379  0.19879909423
## 380  0.00298966614
## 381  0.00298966614
## 382  0.00298966614
## 383  0.11727300429
## 384  0.00301738039
## 385  0.00301738039
## 386  0.00301738039
## 387  0.11517036916
## 388  0.11721665850
## 389  0.11721665850
## 390  0.95088255192
## 391  0.25357889024
## 392  0.11520682380
## 393  0.11520682380
## 394  0.11721665850
## 395  0.11721665850
## 396  0.11721665850
## 397  0.11721665850
## 398  0.11721665850
## 399  0.11721665850
## 400  0.11721665850
## 401  0.11721665850
## 402  0.11721665850
## 403  0.11721665850
## 404  0.11721665850
## 405  0.00305211043
## 406  0.11531494851
## 407  0.00305211043
## 408  0.00305211043
## 409  0.00738200246
## 410  0.12270921790
## 411  0.00300729821
## 412  0.00300680618
## 413  0.20125788696
## 414  0.11721665850
## 415  0.11727300429
## 416  0.95157845767
## 417  0.11727300429
## 418  0.11721665850
## 419  0.11721665850
## 420  0.11721665850
## 421  0.11721665850
## 422  0.08914599110
## 423  0.03310929347
## 424  0.03310929347
## 425  0.01420532165
## 426  0.11721665850
## 427  0.11721665850
## 428  0.01420532165
```

```
## 429  0.00300693239
## 430  0.11727300429
## 431  0.00303154102
## 432  0.01420532165
## 433  0.00300693239
## 434  0.11721665850
## 435  0.11721665850
## 436  0.00305211043
## 437  0.00300769057
## 438  0.00300769057
## 439  0.00303154102
## 440  0.11721665850
## 441  0.11721665850
## 442  0.11721665850
## 443  0.89131793165
## 444  0.85545932262
## 445  0.94380000393
## 446  0.87720335837
## 447  0.94191676633
## 448  0.82341217115
## 449  0.11721665850
## 450  0.11721665850
## 451  0.11721665850
## 452  0.00300769057
## 453  0.00300769057
## 454  0.12270921790
## 455  0.00300693239
## 456  0.11727300429
## 457  0.11721665850
## 458  0.01420532165
## 459  0.13250586484
## 460  0.85275712806
## 461  0.12922352593
## 462  0.01391666720
## 463  0.00300769057
## 464  0.00300769057
## 465  0.11554646739
## 466  0.06128884303
## 467  0.11671085147
## 468  0.13250586484
## 469  0.13250586484
## 470  0.11671085147
## 471  0.11671085147
## 472  0.11671085147
## 473  0.11588309782
## 474  0.11619008240
## 475  0.11671085147
## 476  0.11671085147
## 477  0.11517562152
## 478  0.11517562152
## 479  0.11517562152
## 480  0.11517562152
## 481  0.11721665850
## 482  0.11721665850
```

```
## 483  0.00301717457
## 484  0.00301717457
## 485  0.00301717457
## 486  0.00301717457
## 487  0.00301717457
## 488  0.00301717457
## 489  0.00301717457
## 490  0.00301717457
## 491  0.00301717457
## 492  0.11721665850
## 493  0.11721665850
## 494  0.11721665850
## 495  0.11721665850
## 496  0.11721665850
## 497  0.11721665850
## 498  0.11721665850
## 499  0.11721665850
## 500  0.11588309782
## 501  0.00301404361
## 502  0.00301404361
## 503  0.00301404361
## 504  0.00301404361
## 505  0.11721665850
## 506  0.11721665850
## 507  0.11721665850
## 508  0.11721665850
## 509  0.00301986500
## 510  0.11588309782
## 511  0.11721665850
## 512  0.11721665850
## 513  0.00300693239
## 514  0.11721665850
## 515  0.11721665850
## 516  0.10141192736
## 517  0.10141192736
## 518  0.11834972807
## 519  0.20125788696
## 520  0.20125788696
## 521  0.11588309782
## 522  0.12270921790
## 523  0.00301039593
## 524  0.92208694051
## 525  0.11521719368
## 526  0.11636703347
## 527  0.11644151791
## 528  0.25357889024
## 529  0.11520682380
## 530  0.25357889024
## 531  0.11520682380
## 532  0.20125788696
## 533  0.00301039593
## 534  0.67455769004
## 535  0.67455769004
## 536  0.95088255192
```

```
## 537  0.67455769004
## 538  0.95088255192
## 539  0.67455769004
## 540  0.95088255192
## 541  0.95088255192
## 542  0.12270921790
## 543  0.00300769057
## 544  0.03549747240
## 545  0.03549747240
## 546  0.11721665850
## 547  0.11719112119
## 548  0.92208694051
## 549  0.00331405746
## 550  0.00331405746
## 551  0.12270921790
## 552  0.12270921790
## 553  0.12270921790
## 554  0.00312444862
## 555  0.00312444862
## 556  0.00300125198
## 557  0.00300125198
## 558  0.00300125198
## 559  0.05291174217
## 560  0.05291174217
## 561  0.93282140864
## 562  0.11520682380
## 563  0.25357889024
## 564  0.08914599110
## 565  0.00300693239
## 566  0.11727300429
## 567  0.00300769057
## 568  0.95086338793
## 569  0.00300769057
## 570  0.15770807721
## 571  0.11588309782
## 572  0.12270921790
## 573  0.12270921790
## 574  0.03466961141
## 575  0.11721665850
## 576  0.00300790631
## 577  0.11721665850
## 578  0.11588309782
## 579  0.95088255192
## 580  0.22487349451
## 581  0.22487349451
## 582  0.12270921790
## 583  0.12270921790
## 584  0.95156191729
## 585  0.12270921790
## 586  0.12270921790
## 587  0.12270921790
## 588  0.12270921790
## 589  0.12270921790
## 590  0.12270921790
```

```
## 591  0.11721665850
## 592  0.12270921790
## 593  0.00145641196
## 594  0.24884599669
## 595  0.60709042693
## 596  0.11721665850
## 597  0.11721665850
## 598  0.11721665850
## 599  0.11721665850
## 600  0.60709042693
## 601  0.11721665850
## 602  0.11721665850
## 603  0.19879909423
## 604  0.00301039593
## 605  0.11721665850
## 606  0.11721665850
## 607  0.11721665850
## 608  0.11721665850
## 609  0.11721665850
## 610  0.11721665850
## 611  0.60709042693
## 612  0.60709042693
## 613  0.00301039593
## 614  0.20125788696
## 615  0.11721665850
## 616  0.11721665850
## 617  0.11721665850
## 618  0.11553641158
## 619  0.11553641158
## 620  0.32055363515
## 621  0.00301039593
## 622  0.00301039593
## 623  0.11719112119
## 624  0.00738200246
## 625  0.19879909423
## 626  0.11721665850
## 627  0.95157699195
## 628  0.30005285140
## 629  0.00046843731
## 630  0.11552383600
## 631  0.94845789993
## 632  0.12270921790
## 633  0.12270921790
## 634  0.27351373593
## 635  0.11552383600
## 636  0.11552383600
## 637  0.11552383600
## 638  0.11721665850
## 639  0.11721665850
## 640  0.11721665850
## 641  0.00301388069
## 642  0.25831640174
## 643  0.22487349451
## 644  0.03343964494
```

```
## 645  0.00301986500
## 646  0.87720335837
## 647  0.94380000393
## 648  0.87720335837
## 649  0.00505920368
## 650  0.11721665850
## 651  0.11721665850
## 652  0.11721665850
## 653  0.11721665850
## 654  0.11721665850
## 655  0.11721665850
## 656  0.00142785832
## 657  0.47622722621
## 658  0.11721665850
## 659  0.11721665850
## 660  0.00300693239
## 661  0.00300693239
## 662  0.00300693239
## 663  0.00300693239
## 664  0.00300693239
## 665  0.00300693239
## 666  0.00300693239
## 667  0.00300693239
## 668  0.11721665850
## 669  0.00300693239
## 670  0.12245845884
## 671  0.11721665850
## 672  0.00142785832
## 673  0.00023793404
## 674  0.00023165141
## 675  0.00023165141
## 676  0.00023165141
## 677  0.00023165141
## 678  0.00023165141
## 679  0.00023165141
## 680  0.00023165141
## 681  0.11721665850
## 682  0.11721665850
## 683  0.11721665850
## 684  0.02061411766
## 685  0.00023165141
## 686  0.04264954533
## 687  0.04264954533
## 688  0.03310380585
## 689  0.03310968322
## 690  0.03310968322
## 691  0.03310968322
## 692  0.00023282546
## 693  0.29968295826
## 694  0.00023609278
## 695  0.00023609278
## 696  0.00023609278
## 697  0.00023609278
## 698  0.00310668522
```

```
## 699  0.00298966614
## 700  0.00023609278
## 701  0.20125788696
## 702  0.08740552571
## 703  0.00023609278
## 704  0.11517066524
## 705  0.00023609278
## 706  0.00023609278
## 707  0.00023609278
## 708  0.00023609278
## 709  0.11727300429
## 710  0.11727300429
## 711  0.69265351779
## 712  0.11727300429
## 713  0.11727300429
## 714  0.69265351779
## 715  0.69265351779
## 716  0.11517082711
## 717  0.11517082711
## 718  0.11517082711
## 719  0.11517082711
## 720  0.11517082711
## 721  0.00021219394
## 722  0.11517082711
## 723  0.11517082711
## 724  0.00021219394
## 725  0.11517082711
## 726  0.00021219394
## 727  0.11517082711
## 728  0.00021219394
## 729  0.11517082711
## 730  0.10317758272
## 731  0.10317758272
## 732  0.00023609278
## 733  0.00023609278
## 734  0.11619008240
## 735  0.95088255192
## 736  0.22487349451
## 737  0.22487349451
## 738  0.11566246533
## 739  0.11686909967
## 740  0.11727300429
## 741  0.11721665850
## 742  0.00023609278
## 743  0.00023609278
## 744  0.00023609278
## 745  0.00023609278
## 746  0.95157855552
## 747  0.03312713867
## 748  0.03321946911
## 749  0.03321946911
## 750  0.00277877587
## 751  0.00277877587
## 752  0.95157855552
```

```
## 753  0.95157855552
## 754  0.11802136488
## 755  0.11686909967
## 756  0.00277877587
## 757  0.00025450407
## 758  0.11686909967
## 759  0.11686909967
## 760  0.00300769057
## 761  0.00023799138
## 762  0.00024167102
## 763  0.00023410696
## 764  0.00023799138
## 765  0.08384620424
## 766  0.08384620424
## 767  0.00221415295
## 768  0.10002980213
## 769  0.00023410696
## 770  0.00024294902
## 771  0.00023799138
## 772  0.00025342200
## 773  0.00024167102
## 774  0.10002980213
## 775  0.00025342200
## 776  0.00023799138
## 777  0.10002980213
## 778  0.10002980213
## 779  0.00021219394
## 780  0.03321634996
## 781  0.03321634996
## 782  0.00331405746
## 783  0.00331405746
## 784  0.00331405746
## 785  0.10002980213
## 786  0.10002980213
## 787  0.10002980213
## 788  0.10002980213
## 789  0.10002980213
## 790  0.10002980213
## 791  0.10002980213
## 792  0.10002980213
## 793  0.00221415295
## 794  0.00305211043
## 795  0.86734768417
## 796  0.01156163066
## 797  0.11517136631
## 798  0.11727300429
## 799  0.00300769057
## 800  0.92286130627
## 801  0.92286130627
## 802  0.58530330738
## 803  0.92286130627
## 804  0.92286130627
## 805  0.25963868156
## 806  0.25963868156
```

```
## 807  0.95156950730
## 808  0.95157855552
## 809  0.95157458267
## 810  0.94862255668
## 811  0.94862255668
## 812  0.01680340092
## 813  0.01680340092
## 814  0.12270921790
## 815  0.94862255668
## 816  0.94862255668
## 817  0.94862255668
## 818  0.95072418248
## 819  0.94862255668
## 820  0.94862255668
## 821  0.95119957726
## 822  0.11517652150
## 823  0.11517066524
## 824  0.86734768417
## 825  0.00666112180
## 826  0.00666112180
## 827  0.12270921790
## 828  0.03312713867
## 829  0.12270921790
## 830  0.00305211043
## 831  0.11721665850
## 832  0.00666112180
## 833  0.19879909423
## 834  0.00298966614
## 835  0.00666112180
## 836  0.00666112180
## 837  0.11727300429
## 838  0.19879909423
## 839  0.00666112180
## 840  0.00221415295
## 841  0.00221415295
## 842  0.00277877587
## 843  0.69629138658
## 844  0.11513888229
## 845  0.60432046167
## 846  0.60432046167
## 847  0.60432046167
## 848  0.60432046167
## 849  0.60432046167
## 850  0.09549765120
## 851  0.00021614300
## 852  0.09549765120
## 853  0.09549765120
## 854  0.00356548855
## 855  0.09723205822
## 856  0.05486221223
## 857  0.05486221223
## 858  0.05486221223
## 859  0.09723205822
## 860  0.10125859271
```

```
## 861  0.10002980213
## 862  0.51470815821
## 863  0.11516798040
## 864  0.09549765120
## 865  0.09723205822
## 866  0.10002980213
## 867  0.36159635628
## 868  0.36159635628
## 869  0.36159635628
## 870  0.51470815821
## 871  0.51470815821
## 872  0.51470815821
## 873  0.51470815821
## 874  0.51470815821
## 875  0.51470815821
## 876  0.51470815821
## 877  0.51470815821
## 878  0.00021614300
## 879  0.51470815821
## 880  0.11721665850
## 881  0.19879909423
## 882  0.12175902099
## 883  0.12175902099
## 884  0.11671085147
## 885  0.11619008240
## 886  0.51470815821
## 887  0.11721665850
## 888  0.51470815821
## 889  0.94380000393
## 890  0.10002980213
## 891  0.94380000393
## 892  0.94191676633
## 893  0.20125788696
## 894  0.20827991389
## 895  0.03549747240
## 896  0.00300693239
## 897  0.00300693239
## 898  0.11516991395
## 899  0.00021219394
## 900  0.00021219394
## 901  0.11721665850
## 902  0.11721665850
## 903  0.11721665850
## 904  0.03549747240
## 905  0.03549747240
## 906  0.03549747240
## 907  0.03549747240
## 908  0.00939775898
## 909  0.00362852767
## 910  0.00362852767
## 911  0.00310668522
## 912  0.00310668522
## 913  0.00298966614
## 914  0.00298966614
```

```
## 915  0.11727300429
## 916  0.00362852767
## 917  0.00362852767
## 918  0.00362852767
## 919  0.86734768417
## 920  0.00331405746
## 921  0.00301039593
## 922  0.51470815821
## 923  0.51470815821
## 924  0.51470815821
## 925  0.51470815821
## 926  0.49341974966
## 927  0.00301039593
## 928  0.00300769057
## 929  0.00300769057
## 930  0.00300693239
## 931  0.00300769057
## 932  0.11517066524
## 933  0.11520888224
## 934  0.51470815821
## 935  0.94304528023
## 936  0.51470815821
## 937  0.94304528023
## 938  0.51470815821
## 939  0.49341974966
## 940  0.94304528023
## 941  0.75936768510
## 942  0.00300693239
## 943  0.00300693239
## 944  0.00300693239
## 945  0.00300693239
## 946  0.00300693239
## 947  0.00300693239
## 948  0.11517003596
## 949  0.11517003596
## 950  0.82341217115
## 951  0.11674253241
## 952  0.11517003596
## 953  0.07598406025
## 954  0.00300693239
## 955  0.00300769057
## 956  0.19879909423
## 957  0.00299355380
## 958  0.00299355380
## 959  0.00301039593
## 960  0.11190382654
## 961  0.11727300429
## 962  0.00300693239
## 963  0.00300665237
## 964  0.00300769057
## 965  0.95152506371
## 966  0.87398057481
## 967  0.94862255668
## 968  0.87398057481
```

```
## 969  0.87398057481
## 970  0.87398057481
## 971  0.87398057481
## 972  0.00300769057
## 973  0.13483873237
## 974  0.13483873237
## 975  0.14179586633
## 976  0.00300693239
## 977  0.00300693239
## 978  0.00300769057
## 979  0.11721665850
## 980  0.00306744690
## 981  0.00306744690
## 982  0.00306744690
## 983  0.00306744690
## 984  0.00306744690
## 985  0.11727300429
## 986  0.11516989030
## 987  0.11516989030
## 988  0.11516989030
## 989  0.11516989030
## 990  0.11516989030
## 991  0.11516989030
## 992  0.00300693239
## 993  0.95086338793
## 994  0.95086338793
## 995  0.12571881318
## 996  0.13455210963
## 997  0.13455210963
## 998  0.13455210963
## 999  0.13455210963
## 1000 0.13455210963
## 1001 0.11619008240
## 1002 0.11517068250
## 1003 0.11553641158
## 1004 0.00306744690
## 1005 0.00306744690
## 1006 0.00306744690
## 1007 0.00300693239
## 1008 0.00310668522
## 1009 0.11894029096
## 1010 0.91885593617
## 1011 0.89853396805
## 1012 0.95147988878
## 1013 0.11721665850
## 1014 0.11894029096
## 1015 0.11894029096
## 1016 0.11517421878
## 1017 0.05486221223
## 1018 0.05486221223
## 1019 0.11727300429
## 1020 0.11727300429
## 1021 0.11727300429
## 1022 0.11727300429
```

```
## 1023 0.11727300429
## 1024 0.86734768417
## 1025 0.86734768417
## 1026 0.95157855552
## 1027 0.94433435899
## 1028 0.45819733112
## 1029 0.45819733112
## 1030 0.00306470954
## 1031 0.20125788696
## 1032 0.11727300429
## 1033 0.19879909423
## 1034 0.00300674141
## 1035 0.00300674141
## 1036 0.11727300429
## 1037 0.00300674141
## 1038 0.93843149301
## 1039 0.93843149301
## 1040 0.93843149301
## 1041 0.01680340092
## 1042 0.01680340092
## 1043 0.11721665850
## 1044 0.86734768417
## 1045 0.05486221223
## 1046 0.00457686608
## 1047 0.00457686608
## 1048 0.05486221223
## 1049 0.12571881318
## 1050 0.00306744690
## 1051 0.00306744690
## 1052 0.00306744690
## 1053 0.00306744690
## 1054 0.00306744690
## 1055 0.00306744690
## 1056 0.00306744690
## 1057 0.00306744690
## 1058 0.00306744690
## 1059 0.00306744690
## 1060 0.00306744690
## 1061 0.00306744690
## 1062 0.00306744690
## 1063 0.00306744690
## 1064 0.95086338793
## 1065 0.95086338793
## 1066 0.05486221223
## 1067 0.05486221223
## 1068 0.11727300429
## 1069 0.11517101166
## 1070 0.12501820080
## 1071 0.12501820080
## 1072 0.12501820080
## 1073 0.12501820080
## 1074 0.12501820080
## 1075 0.12501820080
## 1076 0.12501820080
```

```
## 1077 0.95088255192
## 1078 0.11553641158
## 1079 0.00301298885
## 1080 0.00301298885
## 1081 0.00301298885
## 1082 0.11997418677
## 1083 0.13140430360
## 1084 0.86734768417
## 1085 0.05486221223
## 1086 0.00423959417
## 1087 0.05486221223
## 1088 0.05486221223
## 1089 0.86734768417
## 1090 0.12270921790
## 1091 0.11834972807
## 1092 0.12270921790
## 1093 0.00333040446
## 1094 0.00333040446
## 1095 0.00333040446
## 1096 0.11727300429
## 1097 0.01680340092
## 1098 0.01680340092
## 1099 0.01680340092
## 1100 0.95086338793
## 1101 0.95086338793
## 1102 0.00333040446
## 1103 0.00298966614
## 1104 0.95149508120
## 1105 0.11516989030
## 1106 0.11516989030
## 1107 0.11517068250
## 1108 0.11517068250
## 1109 0.11517421878
## 1110 0.11517421878
## 1111 0.95155256283
## 1112 0.11517421878
## 1113 0.11517421878
## 1114 0.11517421878
## 1115 0.11517066524
## 1116 0.11553641158
## 1117 0.11553641158
## 1118 0.00303993204
## 1119 0.00303993204
## 1120 0.00303993204
## 1121 0.12270921790
## 1122 0.01680340092
## 1123 0.01680340092
## 1124 0.01680340092
## 1125 0.11476560277
## 1126 0.01680340092
## 1127 0.01680340092
## 1128 0.11721665850
## 1129 0.00301298885
## 1130 0.00301298885
```

```
## 1131 0.00301298885
## 1132 0.19879909423
## 1133 0.00181988054
## 1134 0.00333040446
## 1135 0.00333040446
## 1136 0.00333040446
## 1137 0.11517068250
## 1138 0.95086338793
## 1139 0.95086338793
## 1140 0.94843942747
## 1141 0.19879909423
## 1142 0.24884599669
## 1143 0.94671139534
## 1144 0.94862255668
## 1145 0.11727300429
## 1146 0.95157855552
## 1147 0.94962867955
## 1148 0.94862255668
## 1149 0.95157855552
## 1150 0.33240056577
## 1151 0.13140430360
## 1152 0.13998134866
## 1153 0.94862255668
## 1154 0.11727300429
## 1155 0.69265351779
## 1156 0.95157855552
## 1157 0.06128884303
## 1158 0.00043259313
## 1159 0.05486221223
## 1160 0.58530330738
## 1161 0.00526263122
## 1162 0.00074669217
## 1163 0.11721665850
## 1164 0.12506931341
## 1165 0.11516940623
## 1166 0.00090035202
## 1167 0.08667310727
## 1168 0.00301717457
## 1169 0.00024609043
## 1170 0.00023347636
## 1171 0.00024609043
## 1172 0.00051139395
## 1173 0.00023347636
## 1174 0.02049172168
## 1175 0.03284254883
## 1176 0.00025120025
## 1177 0.00051139395
## 1178 0.00023347636
## 1179 0.00025120025
## 1180 0.00024609043
## 1181 0.00051139395
## 1182 0.00023347636
## 1183 0.00025120025
## 1184 0.00051139395
```

```
## 1185 0.11516981574
## 1186 0.11516991395
## 1187 0.11727300429
## 1188 0.11727300429
## 1189 0.05486221223
## 1190 0.03310071876
## 1191 0.07854548428
## 1192 0.00333656711
## 1193 0.11721665850
## 1194 0.00394418361
## 1195 0.05486221223
## 1196 0.05486221223
## 1197 0.05486221223
## 1198 0.11727300429
## 1199 0.94862255668
## 1200 0.94862255668
## 1201 0.11727300429
## 1202 0.94862255668
## 1203 0.11727300429
## 1204 0.11517433350
## 1205 0.11553641158
## 1206 0.11553641158
## 1207 0.48001829725
## 1208 0.48001829725
## 1209 0.00302236136
## 1210 0.00302236136
## 1211 0.48001829725
## 1212 0.24884599669
## 1213 0.00301073720
## 1214 0.00301073720
## 1215 0.00301073720
## 1216 0.46056195091
## 1217 0.46056195091
## 1218 0.46056195091
## 1219 0.46056195091
## 1220 0.07854548428
## 1221 0.05486221223
## 1222 0.11894029096
## 1223 0.00090035202
## 1224 0.00090035202
## 1225 0.11894029096
## 1226 0.11727300429
## 1227 0.11727300429
## 1228 0.00305211043
## 1229 0.00305211043
## 1230 0.11244292644
## 1231 0.00302236136
## 1232 0.12270921790
## 1233 0.12270921790
## 1234 0.12703124786
## 1235 0.19879909423
## 1236 0.12703124786
## 1237 0.12270921790
## 1238 0.12270921790
```

```
## 1239 0.19879909423
## 1240 0.00090035202
## 1241 0.03310599596
## 1242 0.03021067690
## 1243 0.10545877037
## 1244 0.87398057481
## 1245 0.10545877037
## 1246 0.10545877037
## 1247 0.10545877037
## 1248 0.05648262800
## 1249 0.11516382734
## 1250 0.00302236136
## 1251 0.00302236136
## 1252 0.11802136488
## 1253 0.00302236136
## 1254 0.10220523762
## 1255 0.11497185937
## 1256 0.11517861572
## 1257 0.11516382734
## 1258 0.11516382734
## 1259 0.52261571836
## 1260 0.52261571836
## 1261 0.52261571836
## 1262 0.11517940668
## 1263 0.00383728043
## 1264 0.95145929101
## 1265 0.06057747846
## 1266 0.00038910030
## 1267 0.11619008240
## 1268 0.00301717457
## 1269 0.10545877037
## 1270 0.10545877037
## 1271 0.87398057481
## 1272 0.87398057481
## 1273 0.87398057481
## 1274 0.87398057481
## 1275 0.87398057481
## 1276 0.10545877037
## 1277 0.87398057481
## 1278 0.10545877037
## 1279 0.10545877037
## 1280 0.10545877037
## 1281 0.11721665850
## 1282 0.11721665850
## 1283 0.11721665850
## 1284 0.11721665850
## 1285 0.12270921790
## 1286 0.00308632447
## 1287 0.00301495527
## 1288 0.00301495527
## 1289 0.95088255192
## 1290 0.10545877037
## 1291 0.11727300429
## 1292 0.10545877037
```

```
## 1293 0.11553641158
## 1294 0.11553641158
## 1295 0.00305211043
## 1296 0.11553641158
## 1297 0.00305211043
## 1298 0.11553641158
## 1299 0.86734768417
## 1300 0.11727300429
## 1301 0.00300693239
## 1302 0.11517068250
## 1303 0.11517068250
## 1304 0.12270921790
## 1305 0.12270921790
## 1306 0.11721665850
## 1307 0.01680340092
## 1308 0.01680340092
## 1309 0.01680340092
## 1310 0.12270921790
## 1311 0.67585126866
## 1312 0.67585126866
## 1313 0.95155256283
## 1314 0.00302418622
## 1315 0.01680340092
## 1316 0.00298966614
## 1317 0.11445709200
## 1318 0.11445709200
## 1319 0.11721665850
## 1320 0.11721665850
## 1321 0.11721665850
## 1322 0.11721665850
## 1323 0.11721665850
## 1324 0.11721665850
## 1325 0.06128884303
## 1326 0.75496030113
## 1327 0.03312713867
## 1328 0.62880244439
## 1329 0.00024076776
## 1330 0.03314811316
## 1331 0.10880302425
## 1332 0.01680340092
## 1333 0.01680340092
## 1334 0.11476560277
## 1335 0.01680340092
## 1336 0.01680340092
## 1337 0.01680340092
## 1338 0.01680340092
## 1339 0.01680340092
## 1340 0.01680340092
## 1341 0.24884599669
## 1342 0.01680340092
## 1343 0.01680340092
## 1344 0.39196422278
## 1345 0.03233714151
## 1346 0.01454293146
```

```
## 1347 0.11419546419
## 1348 0.11419546419
## 1349 0.11419546419
## 1350 0.95157739962
## 1351 0.10545877037
## 1352 0.00383728043
## 1353 0.22487349451
## 1354 0.25357889024
## 1355 0.00331405746
## 1356 0.00300769057
## 1357 0.00300769057
## 1358 0.00383728043
## 1359 0.00300769057
## 1360 0.00300769057
## 1361 0.95155256283
## 1362 0.00300693239
## 1363 0.00301738039
## 1364 0.00300769057
## 1365 0.00300769057
## 1366 0.00300769057
## 1367 0.13140430360
## 1368 0.13140430360
## 1369 0.11517861572
## 1370 0.10317758272
## 1371 0.95157855552
## 1372 0.95157855552
## 1373 0.10917332636
## 1374 0.10917332636
## 1375 0.11516989030
## 1376 0.11516989030
## 1377 0.11516989030
## 1378 0.11516989030
## 1379 0.00300769057
## 1380 0.00300769057
## 1381 0.11834972807
## 1382 0.00300769057
## 1383 0.13140430360
## 1384 0.10917332636
## 1385 0.50301546359
## 1386 0.00322647325
## 1387 0.00300769057
## 1388 0.05036546684
## 1389 0.05036546684
## 1390 0.11476560277
## 1391 0.01680340092
## 1392 0.01680340092
## 1393 0.11476560277
## 1394 0.11476560277
## 1395 0.11476560277
## 1396 0.11476560277
## 1397 0.11476560277
## 1398 0.11476560277
## 1399 0.69265351779
## 1400 0.63433986775
```

```
## 1401 0.00388005328
## 1402 0.00300693239
## 1403 0.00300693239
## 1404 0.00313245541
## 1405 0.00300693239
## 1406 0.95148656384
## 1407 0.00313245541
## 1408 0.00300693239
## 1409 0.01680340092
## 1410 0.01680340092
## 1411 0.01680340092
## 1412 0.01680340092
## 1413 0.01680340092
## 1414 0.01680340092
## 1415 0.01680340092
## 1416 0.01680340092
## 1417 0.01680340092
## 1418 0.01680340092
## 1419 0.01680340092
## 1420 0.95148106261
## 1421 0.00475465129
## 1422 0.00475465129
## 1423 0.00298966614
## 1424 0.00298966614
## 1425 0.00298966614
## 1426 0.95154104162
## 1427 0.95154104162
## 1428 0.00475465129
## 1429 0.03310658612
## 1430 0.03310658612
## 1431 0.03310367008
## 1432 0.11527251521
## 1433 0.00298966614
## 1434 0.94671139534
## 1435 0.00301039593
## 1436 0.00301039593
## 1437 0.00301039593
## 1438 0.00300650332
## 1439 0.00300650332
## 1440 0.00301039593
## 1441 0.00301039593
## 1442 0.94671139534
## 1443 0.94671139534
## 1444 0.11476560277
## 1445 0.11536483732
## 1446 0.00300665237
## 1447 0.00301404361
## 1448 0.00300716985
## 1449 0.00305211043
## 1450 0.00305211043
## 1451 0.00305211043
## 1452 0.01680340092
## 1453 0.00301039593
## 1454 0.00301039593
```

```
## 1455 0.00301039593
## 1456 0.67455769004
## 1457 0.11859125700
## 1458 0.94845789993
## 1459 0.11476560277
## 1460 0.00300665237
## 1461 0.00300665237
## 1462 0.00300650332
## 1463 0.00300650332
## 1464 0.00300650332
## 1465 0.00300650332
## 1466 0.01641264264
## 1467 0.11458971168
## 1468 0.11721665850
## 1469 0.00300693239
## 1470 0.00300693239
## 1471 0.00300693239
## 1472 0.00301404361
## 1473 0.11513888229
## 1474 0.32055363515
## 1475 0.11721665850
## 1476 0.00300693239
## 1477 0.00300693239
## 1478 0.95136446221
## 1479 0.00383728043
## 1480 0.94433435899
## 1481 0.11458971168
## 1482 0.11721665850
## 1483 0.95157739962
## 1484 0.22487349451
## 1485 0.01471688214
## 1486 0.00300769057
## 1487 0.11458971168
## 1488 0.20125788696
## 1489 0.00301039593
## 1490 0.20125788696
## 1491 0.00301039593
## 1492 0.00301039593
## 1493 0.00301039593
## 1494 0.20125788696
## 1495 0.00305211043
## 1496 0.00305211043
## 1497 0.11458971168
## 1498 0.11458971168
## 1499 0.00300769057
## 1500 0.11458971168
## 1501 0.11458971168
## 1502 0.11458971168
## 1503 0.10860355764
## 1504 0.94845789993
## 1505 0.10860355764
## 1506 0.11554646739
## 1507 0.24884599669
## 1508 0.03282928888
```

```
## 1509 0.00488634082
## 1510 0.03310589046
## 1511 0.10917332636
## 1512 0.91887711141
## 1513 0.10917332636
## 1514 0.95137620024
## 1515 0.00162353565
## 1516 0.12466224214
## 1517 0.00300693239
## 1518 0.00313245541
## 1519 0.00313245541
## 1520 0.11727300429
## 1521 0.11721665850
## 1522 0.11721665850
## 1523 0.19879909423
## 1524 0.31497566989
## 1525 0.00300650332
## 1526 0.00300650332
## 1527 0.00300650332
## 1528 0.00300650332
## 1529 0.00300650332
## 1530 0.00300650332
## 1531 0.00300650332
## 1532 0.00300650332
## 1533 0.00300650332
## 1534 0.11859125700
## 1535 0.13478555890
## 1536 0.11458971168
## 1537 0.11859125700
## 1538 0.11859125700
## 1539 0.01902524755
## 1540 0.11458971168
## 1541 0.11997418677
## 1542 0.11859125700
## 1543 0.11859125700
## 1544 0.03314811316
## 1545 0.00301039593
## 1546 0.00505920368
## 1547 0.11721665850
## 1548 0.11727300429
## 1549 0.00681406290
## 1550 0.93345555447
## 1551 0.93345555447
## 1552 0.93345555447
## 1553 0.93345555447
## 1554 0.93345555447
## 1555 0.93345555447
## 1556 0.93345555447
## 1557 0.11517068250
## 1558 0.75265377808
## 1559 0.00300814568
## 1560 0.00505920368
## 1561 0.00505920368
## 1562 0.00300769057
```

```
## 1563 0.11859125700
## 1564 0.11998227296
## 1565 0.95157500112
## 1566 0.11516991395
## 1567 0.00301646996
## 1568 0.00300814568
## 1569 0.11721665850
## 1570 0.11721665850
## 1571 0.00300693239
## 1572 0.11517433350
## 1573 0.11517433350
## 1574 0.11517433350
## 1575 0.00300694176
## 1576 0.00300694176
## 1577 0.10880302425
## 1578 0.11834972807
## 1579 0.11727300429
## 1580 0.00301039593
## 1581 0.00301039593
## 1582 0.12270921790
## 1583 0.12270921790
## 1584 0.12270921790
## 1585 0.12270921790
## 1586 0.12270921790
## 1587 0.12270921790
## 1588 0.12270921790
## 1589 0.12270921790
## 1590 0.12270921790
## 1591 0.12270921790
## 1592 0.12270921790
## 1593 0.12270921790
## 1594 0.12270921790
## 1595 0.12270921790
## 1596 0.12270921790
## 1597 0.12270921790
## 1598 0.12270921790
## 1599 0.01680340092
## 1600 0.01680340092
## 1601 0.01680340092
## 1602 0.01680340092
## 1603 0.12270921790
## 1604 0.12270921790
## 1605 0.12270921790
## 1606 0.12270921790
## 1607 0.12270921790
## 1608 0.12270921790
## 1609 0.12270921790
## 1610 0.11834972807
## 1611 0.11727300429
## 1612 0.00300693239
## 1613 0.11721665850
## 1614 0.11721665850
## 1615 0.11721665850
## 1616 0.20125788696
```

```
## 1617 0.20125788696
## 1618 0.20125788696
## 1619 0.00298966614
## 1620 0.11721665850
## 1621 0.11721665850
## 1622 0.11721665850
## 1623 0.11721665850
## 1624 0.11721665850
## 1625 0.11721665850
## 1626 0.11721665850
## 1627 0.11727300429
## 1628 0.00301334244
## 1629 0.00301334244
## 1630 0.00301334244
## 1631 0.00301334244
## 1632 0.11513888229
## 1633 0.11513888229
## 1634 0.11727300429
## 1635 0.00300693239
## 1636 0.19879909423
## 1637 0.20125788696
## 1638 0.03224665826
## 1639 0.11721665850
## 1640 0.11727300429
## 1641 0.11727300429
## 1642 0.11727300429
## 1643 0.11727300429
## 1644 0.11727300429
## 1645 0.11727300429
## 1646 0.19879909423
## 1647 0.24505290713
## 1648 0.10134255504
## 1649 0.11727300429
## 1650 0.11517652150
## 1651 0.00300652066
## 1652 0.11727300429
## 1653 0.11859125700
## 1654 0.03313966743
## 1655 0.11721665850
## 1656 0.11727300429
## 1657 0.11727300429
## 1658 0.11727300429
## 1659 0.11721665850
## 1660 0.11721665850
## 1661 0.00300769057
## 1662 0.11721665850
## 1663 0.11721665850
## 1664 0.11480793866
## 1665 0.12270921790
## 1666 0.12270921790
## 1667 0.12270921790
## 1668 0.13324482777
## 1669 0.12270921790
## 1670 0.12270921790
```

```
## 1671 0.12270921790
## 1672 0.12270921790
## 1673 0.12270921790
## 1674 0.12270921790
## 1675 0.11721665850
## 1676 0.11721665850
## 1677 0.11721665850
## 1678 0.11721665850
## 1679 0.11721665850
## 1680 0.11721665850
## 1681 0.11721665850
## 1682 0.11721665850
## 1683 0.00298966614
## 1684 0.00301039593
## 1685 0.00301039593
## 1686 0.00301039593
## 1687 0.01680340092
## 1688 0.01680340092
## 1689 0.01680340092
## 1690 0.11721665850
## 1691 0.11721665850
## 1692 0.11721665850
## 1693 0.11721665850
## 1694 0.11721665850
## 1695 0.00300693239
## 1696 0.11721665850
## 1697 0.11721665850
## 1698 0.11721665850
## 1699 0.00301039593
## 1700 0.00301039593
## 1701 0.11721665850
## 1702 0.11721665850
## 1703 0.11721665850
## 1704 0.11721665850
## 1705 0.11721665850
## 1706 0.11721665850
## 1707 0.11721665850
## 1708 0.11727300429
## 1709 0.11727300429
## 1710 0.95075869342
## 1711 0.00331405746
## 1712 0.00331405746
## 1713 0.00331405746
## 1714 0.11480793866
## 1715 0.12466224214
## 1716 0.12106320552
## 1717 0.12106320552
## 1718 0.12106320552
## 1719 0.12106320552
## 1720 0.12106320552
## 1721 0.12106320552
## 1722 0.11721665850
## 1723 0.11721665850
## 1724 0.11721665850
```

```
## 1725 0.11721665850
## 1726 0.11721665850
## 1727 0.00301012849
## 1728 0.11727300429
## 1729 0.11554646739
## 1730 0.11554646739
## 1731 0.11554646739
## 1732 0.12466224214
## 1733 0.12466224214
## 1734 0.12466224214
## 1735 0.12466224214
## 1736 0.12466224214
## 1737 0.12466224214
## 1738 0.12466224214
## 1739 0.12466224214
## 1740 0.12466224214
## 1741 0.11480793866
## 1742 0.12270921790
## 1743 0.95157855552
## 1744 0.95157833267
## 1745 0.03282928888
## 1746 0.02252921071
## 1747 0.11727300429
## 1748 0.00300769057
## 1749 0.15214318450
## 1750 0.19879909423
## 1751 0.11519404985
## 1752 0.11516991395
## 1753 0.11519404985
## 1754 0.00300650332
## 1755 0.00488192388
## 1756 0.00300650332
## 1757 0.03310285724
## 1758 0.00024557546
## 1759 0.00023609278
## 1760 0.00024557546
## 1761 0.00024557546
## 1762 0.03310285724
## 1763 0.03310285724
## 1764 0.11516989030
## 1765 0.11516989030
## 1766 0.11516989030
## 1767 0.03283245139
## 1768 0.03283245139
## 1769 0.03283245139
## 1770 0.11727300429
## 1771 0.69265351779
## 1772 0.00300693239
## 1773 0.00101771481
## 1774 0.11727300429
## 1775 0.00300693239
## 1776 0.00331405746
## 1777 0.00301039593
## 1778 0.00301039593
```

```
## 1779 0.11721665850
## 1780 0.02245733518
## 1781 0.02245733518
## 1782 0.11554646739
## 1783 0.95131661850
## 1784 0.95131661850
## 1785 0.00753639509
## 1786 0.00753639509
## 1787 0.00753639509
## 1788 0.00124082659
## 1789 0.00124082659
## 1790 0.00007049385
## 1791 0.11517433350
## 1792 0.92775245562
## 1793 0.95149453596
## 1794 0.00124082659
## 1795 0.03314208436
## 1796 0.03314208436
## 1797 0.93079875326
## 1798 0.11727300429
## 1799 0.86734768417
## 1800 0.00753639509
## 1801 0.03314208436
## 1802 0.00648882205
## 1803 0.00648882205
## 1804 0.11727300429
## 1805 0.86734768417
## 1806 0.03310658612
## 1807 0.03310658612
## 1808 0.00313245541
## 1809 0.00313245541
## 1810 0.02376584397
## 1811 0.13140430360
## 1812 0.00303223651
## 1813 0.00738200246
## 1814 0.00738200246
## 1815 0.11727300429
## 1816 0.00300769057
## 1817 0.00301039593
## 1818 0.00301039593
## 1819 0.00301039593
## 1820 0.00301039593
## 1821 0.00301039593
## 1822 0.11516382734
## 1823 0.00331405746
## 1824 0.00312444862
## 1825 0.86734768417
## 1826 0.00301039593
## 1827 0.11627118989
## 1828 0.00101771481
## 1829 0.19879909423
## 1830 0.19879909423
## 1831 0.00294201920
## 1832 0.00294201920
```

```
## 1833 0.00477804025
## 1834 0.09519374057
## 1835 0.86734768417
## 1836 0.86734768417
## 1837 0.86734768417
## 1838 0.86734768417
## 1839 0.00303993204
## 1840 0.11708121375
## 1841 0.11627118989
## 1842 0.11835508213
## 1843 0.00300769057
## 1844 0.11997373935
## 1845 0.12442650310
## 1846 0.12270921790
## 1847 0.10880302425
## 1848 0.11517421878
## 1849 0.11535961056
## 1850 0.86734768417
## 1851 0.09519374057
## 1852 0.00310583917
## 1853 0.09519374057
## 1854 0.09519374057
## 1855 0.00301039593
## 1856 0.00301039593
## 1857 0.00301039593
## 1858 0.00301039593
## 1859 0.24884599669
## 1860 0.19879909423
## 1861 0.00738200246
## 1862 0.11283167580
## 1863 0.00966687571
## 1864 0.00966687571
## 1865 0.25357889024
## 1866 0.00654247421
## 1867 0.00654247421
## 1868 0.11834972807
## 1869 0.13998134866
## 1870 0.11516865835
## 1871 0.11517433350
## 1872 0.09722123738
## 1873 0.19754086065
## 1874 0.24505290713
## 1875 0.00383728043
## 1876 0.00301039593
## 1877 0.95156133264
## 1878 0.11407157911
## 1879 0.11553641158
## 1880 0.11581577118
## 1881 0.11517003596
## 1882 0.00549585391
## 1883 0.40527256715
## 1884 0.00325407608
## 1885 0.09722123738
## 1886 0.00300715441
```

```
## 1887 0.00300665237
## 1888 0.01680340092
## 1889 0.00024076776
## 1890 0.00024076776
## 1891 0.00024076776
## 1892 0.00024076776
## 1893 0.00024076776
## 1894 0.00024076776
## 1895 0.00024076776
## 1896 0.03310285724
## 1897 0.03310285724
## 1898 0.00025908947
## 1899 0.03310285724
## 1900 0.00029039751
## 1901 0.20125788696
## 1902 0.20125788696
## 1903 0.20125788696
## 1904 0.20125788696
## 1905 0.00301039593
## 1906 0.95114923725
## 1907 0.00304028064
## 1908 0.00304028064
## 1909 0.19879909423
## 1910 0.00358487440
## 1911 0.11721665850
## 1912 0.00302204425
## 1913 0.00302204425
## 1914 0.00302204425
## 1915 0.11721665850
## 1916 0.11721665850
## 1917 0.11721665850
## 1918 0.00303993204
## 1919 0.00303993204
## 1920 0.11727300429
## 1921 0.00303993204
## 1922 0.01680340092
## 1923 0.00101771481
## 1924 0.02376584397
## 1925 0.15836499215
## 1926 0.11513888229
## 1927 0.12357948477
## 1928 0.12357948477
## 1929 0.12357948477
## 1930 0.58530330738
## 1931 0.15836499215
## 1932 0.00300693239
## 1933 0.00331405746
## 1934 0.15905679286
## 1935 0.15905679286
## 1936 0.15905679286
## 1937 0.05656043864
## 1938 0.11553434908
## 1939 0.11721665850
## 1940 0.12270921790
```

```
## 1941 0.00306470954
## 1942 0.00306470954
## 1943 0.00306470954
## 1944 0.00306470954
## 1945 0.00488192388
## 1946 0.00488192388
## 1947 0.00383728043
## 1948 0.00383728043
## 1949 0.00383728043
## 1950 0.00383728043
## 1951 0.00383728043
## 1952 0.00383728043
## 1953 0.00383728043
## 1954 0.00301039593
## 1955 0.00301039593
## 1956 0.00301039593
## 1957 0.00301039593
## 1958 0.11554646739
## 1959 0.11922732438
## 1960 0.11922732438
## 1961 0.12703124786
## 1962 0.24884599669
## 1963 0.00300693239
## 1964 0.00300693239
## 1965 0.03315049532
## 1966 0.00300693239
## 1967 0.00300693239
## 1968 0.00300693239
## 1969 0.00300693239
## 1970 0.00300693239
## 1971 0.00300693239
## 1972 0.00300693239
## 1973 0.11517003596
## 1974 0.19879909423
## 1975 0.11553641158
## 1976 0.92698449126
## 1977 0.00024076776
## 1978 0.00024076776
## 1979 0.11516989030
## 1980 0.00313245541
## 1981 0.86734768417
## 1982 0.11516090348
## 1983 0.11553641158
## 1984 0.00300693239
## 1985 0.00302562865
## 1986 0.00300672254
## 1987 0.00300693239
## 1988 0.00300693239
## 1989 0.00300672254
## 1990 0.00300693239
## 1991 0.67455769004
## 1992 0.11727300429
## 1993 0.00300693239
## 1994 0.00300693239
```

```
## 1995 0.00300693239
## 1996 0.00300665237
## 1997 0.11516974984
## 1998 0.11516974984
## 1999 0.11516974984
## 2000 0.10554222470
## 2001 0.02376584397
## 2002 0.00300769057
## 2003 0.11542458425
## 2004 0.11542458425
## 2005 0.94671139534
## 2006 0.00301039593
## 2007 0.00301039593
## 2008 0.12473784786
## 2009 0.09519374057
## 2010 0.11518212106
## 2011 0.11516665750
## 2012 0.93500003976
## 2013 0.93500003976
## 2014 0.00310668522
## 2015 0.00310668522
## 2016 0.00301039593
## 2017 0.00300650332
## 2018 0.11516989030
## 2019 0.00383728043
## 2020 0.00383728043
## 2021 0.11655445798
## 2022 0.20827991389
## 2023 0.86734768417
## 2024 0.00841474407
## 2025 0.86734768417
## 2026 0.91887711141
## 2027 0.00545776352
## 2028 0.00841474407
## 2029 0.86734768417
## 2030 0.86734768417
## 2031 0.11400389800
## 2032 0.00298966614
## 2033 0.00298966614
## 2034 0.00298966614
## 2035 0.00298966614
## 2036 0.00298966614
## 2037 0.00298966614
## 2038 0.00383728043
## 2039 0.20125788696
## 2040 0.20125788696
## 2041 0.19879909423
## 2042 0.15836499215
## 2043 0.11842877813
## 2044 0.15836499215
## 2045 0.07200405868
## 2046 0.07200405868
## 2047 0.00331405746
## 2048 0.04467362678
```

```
## 2049 0.04467362678
## 2050 0.04467362678
## 2051 0.04467362678
## 2052 0.04467362678
## 2053 0.04467362678
## 2054 0.04467362678
## 2055 0.00301921919
## 2056 0.00301921919
## 2057 0.11598923057
## 2058 0.11598923057
## 2059 0.00038251272
## 2060 0.00300693239
## 2061 0.00300693239
## 2062 0.15836499215
## 2063 0.94088796673
## 2064 0.00300693239
## 2065 0.00738200246
## 2066 0.00738200246
## 2067 0.00738200246
## 2068 0.00738200246
## 2069 0.00298966614
## 2070 0.00298966614
## 2071 0.02445316748
## 2072 0.02445316748
## 2073 0.02445316748
## 2074 0.00331405746
## 2075 0.00331405746
## 2076 0.00331405746
## 2077 0.00331405746
## 2078 0.10880302425
## 2079 0.00383728043
## 2080 0.00383728043
## 2081 0.11619008240
## 2082 0.05291174217
## 2083 0.11516989030
## 2084 0.11516989030
## 2085 0.00040854532
## 2086 0.11517068250
## 2087 0.15516258668
## 2088 0.00301986500
## 2089 0.11400389800
## 2090 0.00023282546
## 2091 0.00023282546
## 2092 0.00026527022
## 2093 0.00026527022
## 2094 0.00026527022
## 2095 0.00023799138
## 2096 0.00023799138
## 2097 0.00023799138
## 2098 0.00023282546
## 2099 0.00023282546
## 2100 0.00303193202
## 2101 0.00024033289
## 2102 0.00023282546
```

```
## 2103 0.00023799138
## 2104 0.00026527022
## 2105 0.00023799138
## 2106 0.00023799138
## 2107 0.00023282546
## 2108 0.00023799138
## 2109 0.00024167102
## 2110 0.11727300429
## 2111 0.19861233900
## 2112 0.00301039593
## 2113 0.11721665850
## 2114 0.00300650332
## 2115 0.11727300429
## 2116 0.00300693239
## 2117 0.11751335438
## 2118 0.11751335438
## 2119 0.11085885032
## 2120 0.02445316748
## 2121 0.00310668522
## 2122 0.00310668522
## 2123 0.00295231073
## 2124 0.86734768417
## 2125 0.00038251272
## 2126 0.05291174217
## 2127 0.11400389800
## 2128 0.00300693239
## 2129 0.11721665850
## 2130 0.11721665850
## 2131 0.00383728043
## 2132 0.10545877037
## 2133 0.11842877813
## 2134 0.00038251272
## 2135 0.15214318450
## 2136 0.00300693239
## 2137 0.00300693239
## 2138 0.00038251272
## 2139 0.01789247436
## 2140 0.01641264264
## 2141 0.01641264264
## 2142 0.01680340092
## 2143 0.01680340092
## 2144 0.25357889024
## 2145 0.11520682380
## 2146 0.00300693239
## 2147 0.00300693239
## 2148 0.00300693239
## 2149 0.03311665413
## 2150 0.20125788696
## 2151 0.00300769057
## 2152 0.10545877037
## 2153 0.01641264264
## 2154 0.11458971168
## 2155 0.95132735875
## 2156 0.10545877037
```

```
## 2157 0.11721665850
## 2158 0.11721665850
## 2159 0.25357889024
## 2160 0.20125788696
## 2161 0.20125788696
## 2162 0.03311547609
## 2163 0.10317758272
## 2164 0.10317758272
## 2165 0.12001718856
## 2166 0.95076994460
## 2167 0.11721665850
## 2168 0.00300650332
## 2169 0.00300693239
## 2170 0.00303223651
## 2171 0.00300650332
## 2172 0.11721665850
## 2173 0.00300650332
## 2174 0.00488634082
## 2175 0.00341400897
## 2176 0.03311194036
## 2177 0.11859125700
## 2178 0.11458971168
## 2179 0.00300769057
## 2180 0.11721665850
## 2181 0.00301039593
## 2182 0.11721665850
## 2183 0.95136882952
## 2184 0.02638394391
## 2185 0.00300769057
## 2186 0.00301039593
## 2187 0.00300769057
## 2188 0.00300769057
## 2189 0.00300769057
## 2190 0.11721665850
## 2191 0.11721665850
## 2192 0.11859125700
## 2193 0.11842877813
## 2194 0.10979687248
## 2195 0.01265950688
## 2196 0.00501139824
## 2197 0.01090212029
## 2198 0.94873748419
## 2199 0.00301646996
## 2200 0.00301334244
## 2201 0.00301334244
## 2202 0.00301334244
## 2203 0.00305211043
## 2204 0.00300769057
## 2205 0.00388005328
## 2206 0.00301039593
## 2207 0.00301039593
## 2208 0.00301039593
## 2209 0.00301039593
## 2210 0.00301039593
```

```
## 2211 0.00301039593
## 2212 0.00301039593
## 2213 0.00301039593
## 2214 0.00301039593
## 2215 0.00301039593
## 2216 0.11085885032
## 2217 0.11129113170
## 2218 0.00305211043
## 2219 0.00305211043
## 2220 0.00305211043
## 2221 0.00305211043
## 2222 0.00305211043
## 2223 0.00501139824
## 2224 0.01128907984
## 2225 0.00383728043
## 2226 0.11740028151
## 2227 0.20125788696
## 2228 0.00301039593
## 2229 0.11721665850
## 2230 0.11986133760
## 2231 0.11986133760
## 2232 0.00388005328
## 2233 0.02851185933
## 2234 0.00307288848
## 2235 0.00300693239
## 2236 0.00300693239
## 2237 0.05656043864
## 2238 0.05656043864
## 2239 0.05656043864
## 2240 0.05656043864
## 2241 0.05656043864
## 2242 0.11517433350
## 2243 0.11912157316
## 2244 0.11721665850
## 2245 0.95131661850
## 2246 0.02851185933
## 2247 0.02851185933
## 2248 0.02851185933
## 2249 0.00501139824
## 2250 0.00023282546
## 2251 0.00024609043
## 2252 0.00023282546
## 2253 0.00023282546
## 2254 0.11517433350
## 2255 0.14282639280
## 2256 0.27351373593
## 2257 0.12270921790
## 2258 0.02956086323
## 2259 0.02956086323
## 2260 0.00501139824
## 2261 0.03299836792
## 2262 0.03299836792
## 2263 0.03299836792
## 2264 0.00501139824
```

```
## 2265 0.11721665850
## 2266 0.94873748419
## 2267 0.08347857805
## 2268 0.08347857805
## 2269 0.08347857805
## 2270 0.19879909423
## 2271 0.19879909423
## 2272 0.19879909423
## 2273 0.19879909423
## 2274 0.19879909423
## 2275 0.19879909423
## 2276 0.02997058525
## 2277 0.02851185933
## 2278 0.01090212029
## 2279 0.00300693239
## 2280 0.15836499215
## 2281 0.11842877813
## 2282 0.11842877813
## 2283 0.11721665850
## 2284 0.03501090415
## 2285 0.00301039593
## 2286 0.00301039593
## 2287 0.11627118989
## 2288 0.11627118989
## 2289 0.11627118989
## 2290 0.11627118989
## 2291 0.00301039593
## 2292 0.00301039593
## 2293 0.00301039593
## 2294 0.00301039593
## 2295 0.14282639280
## 2296 0.12270921790
## 2297 0.12270921790
## 2298 0.02851185933
## 2299 0.02851185933
## 2300 0.01680340092
## 2301 0.11516989030
## 2302 0.01680340092
## 2303 0.00313245541
## 2304 0.11721665850
## 2305 0.11721665850
## 2306 0.00300769057
## 2307 0.00303993204
## 2308 0.00300693239
## 2309 0.11842877813
## 2310 0.00300693239
## 2311 0.00300693239
## 2312 0.00300769057
## 2313 0.00331405746
## 2314 0.11721665850
## 2315 0.93843149301
## 2316 0.93843149301
## 2317 0.93843149301
## 2318 0.93843149301
```

```
## 2319 0.03099051438
## 2320 0.02851185933
## 2321 0.02851185933
## 2322 0.02997058525
## 2323 0.02851185933
## 2324 0.02851185933
## 2325 0.11721665850
## 2326 0.94873748419
## 2327 0.00301030591
## 2328 0.00300841854
## 2329 0.00301030591
## 2330 0.00301030591
## 2331 0.00300841854
## 2332 0.00300841854
## 2333 0.00301030591
## 2334 0.00301030591
## 2335 0.00301030591
## 2336 0.00301030591
## 2337 0.00301030591
## 2338 0.00301030591
## 2339 0.00300841854
## 2340 0.00300841854
## 2341 0.00300841854
## 2342 0.00300841854
## 2343 0.00300841854
## 2344 0.00310668522
## 2345 0.00312444862
## 2346 0.00305211043
## 2347 0.00305211043
## 2348 0.00312444862
## 2349 0.00300650721
## 2350 0.00300650721
## 2351 0.00310668522
## 2352 0.02851185933
## 2353 0.95088255192
## 2354 0.95088255192
## 2355 0.95088255192
## 2356 0.22487349451
## 2357 0.20125788696
## 2358 0.00292154286
## 2359 0.00300769057
## 2360 0.01680340092
## 2361 0.01680340092
## 2362 0.00301039593
## 2363 0.00301039593
## 2364 0.15516258668
## 2365 0.11516989030
## 2366 0.12270921790
## 2367 0.01680340092
## 2368 0.00300693239
## 2369 0.00300693239
## 2370 0.00300693239
## 2371 0.00300693239
## 2372 0.00300693239
```

```
## 2373 0.95131661850
## 2374 0.95131661850
## 2375 0.12270921790
## 2376 0.00300650332
## 2377 0.00300650332
## 2378 0.00300650332
## 2379 0.00300650332
## 2380 0.94873748419
## 2381 0.02997058525
## 2382 0.02851185933
## 2383 0.01680340092
## 2384 0.09413598520
## 2385 0.08347857805
## 2386 0.11336859075
## 2387 0.11336859075
## 2388 0.11336859075
## 2389 0.11336859075
## 2390 0.67455769004
## 2391 0.11517003596
## 2392 0.11336859075
## 2393 0.94845789993
## 2394 0.94845789993
## 2395 0.92698449126
## 2396 0.11336859075
## 2397 0.11336859075
## 2398 0.11336859075
## 2399 0.11336859075
## 2400 0.11336859075
## 2401 0.11336859075
## 2402 0.00301039593
## 2403 0.00301039593
## 2404 0.00301039593
## 2405 0.00301039593
## 2406 0.00301039593
## 2407 0.00301039593
## 2408 0.00301039593
## 2409 0.00301039593
## 2410 0.00301039593
## 2411 0.00301039593
## 2412 0.00301039593
## 2413 0.00301039593
## 2414 0.00301039593
## 2415 0.00301039593
## 2416 0.00301039593
## 2417 0.00301039593
## 2418 0.00301039593
## 2419 0.00301039593
## 2420 0.00301039593
## 2421 0.00301039593
## 2422 0.00301039593
## 2423 0.00383728043
## 2424 0.00383728043
## 2425 0.11336859075
## 2426 0.11336859075
```

```
## 2427 0.11336859075
## 2428 0.00301039593
## 2429 0.00301039593
## 2430 0.12270921790
## 2431 0.00098464754
## 2432 0.00298966614
## 2433 0.00298966614
## 2434 0.00298966614
## 2435 0.00298966614
## 2436 0.00298966614
## 2437 0.00298966614
## 2438 0.00298966614
## 2439 0.00298966614
## 2440 0.00298966614
## 2441 0.00298966614
## 2442 0.00298966614
## 2443 0.00298966614
## 2444 0.00298966614
## 2445 0.00298966614
## 2446 0.00298966614
## 2447 0.00298966614
## 2448 0.00298966614
## 2449 0.00298966614
## 2450 0.00298966614
## 2451 0.00298966614
## 2452 0.00298966614
## 2453 0.19879909423
## 2454 0.00301039593
## 2455 0.00301039593
## 2456 0.00098464754
## 2457 0.13267735982
## 2458 0.03153726530
## 2459 0.13677704548
## 2460 0.00112179316
## 2461 0.86734768417
## 2462 0.12579769043
## 2463 0.00300656957
## 2464 0.07598406025
## 2465 0.00301039593
## 2466 0.00301039593
## 2467 0.00301039593
## 2468 0.00301039593
## 2469 0.00590465120
## 2470 0.03334224631
## 2471 0.13677704548
## 2472 0.13677704548
## 2473 0.00300650332
## 2474 0.00300650332
## 2475 0.00300650332
## 2476 0.00300650332
## 2477 0.00300650332
## 2478 0.13677704548
## 2479 0.20125788696
## 2480 0.00383728043
```

```
## 2481 0.00301039593
## 2482 0.00301039593
## 2483 0.00301039593
## 2484 0.00301039593
## 2485 0.13677704548
## 2486 0.04901725641
## 2487 0.04901725641
## 2488 0.04901725641
## 2489 0.09619214169
## 2490 0.22487349451
## 2491 0.22487349451
## 2492 0.95157092737
## 2493 0.95157092737
## 2494 0.95157092737
## 2495 0.95157092737
## 2496 0.00383728043
## 2497 0.00303223651
## 2498 0.00301039593
## 2499 0.00301039593
## 2500 0.00301039593
## 2501 0.00301039593
## 2502 0.25357889024
## 2503 0.11520682380
## 2504 0.11520682380
## 2505 0.25357889024
## 2506 0.11520682380
## 2507 0.25357889024
## 2508 0.11520682380
## 2509 0.00301039593
## 2510 0.00301039593
## 2511 0.00291137805
## 2512 0.00313245541
## 2513 0.00300650332
## 2514 0.00300650332
## 2515 0.00300650332
## 2516 0.02041264652
## 2517 0.00890834422
## 2518 0.13677704548
## 2519 0.00301039593
## 2520 0.13677704548
## 2521 0.94920041000
## 2522 0.94920041000
## 2523 0.94920041000
## 2524 0.94920041000
## 2525 0.94920041000
## 2526 0.13677704548
## 2527 0.14909301290
## 2528 0.00301039593
## 2529 0.00301039593
## 2530 0.00301039593
## 2531 0.00301039593
## 2532 0.00301039593
## 2533 0.00301039593
## 2534 0.00301039593
```

```
## 2535 0.00301039593
## 2536 0.00301039593
## 2537 0.00301039593
## 2538 0.00301039593
## 2539 0.94920041000
## 2540 0.11721665850
## 2541 0.11467493569
## 2542 0.11467493569
## 2543 0.13677704548
## 2544 0.11721665850
## 2545 0.94920041000
## 2546 0.94920041000
## 2547 0.00301039593
## 2548 0.00301039593
## 2549 0.00301039593
## 2550 0.00301039593
## 2551 0.00301039593
## 2552 0.11721665850
## 2553 0.11721665850
## 2554 0.11721665850
## 2555 0.11721665850
## 2556 0.12347470273
## 2557 0.00301039593
## 2558 0.00383728043
## 2559 0.00383728043
## 2560 0.00301039593
## 2561 0.00303223651
## 2562 0.00303223651
## 2563 0.00303223651
## 2564 0.00301039593
## 2565 0.00383728043
## 2566 0.00301039593
## 2567 0.00301039593
## 2568 0.00301039593
## 2569 0.00301039593
## 2570 0.11721665850
## 2571 0.11721665850
## 2572 0.17723124426
## 2573 0.00890834422
## 2574 0.11516989030
## 2575 0.00301039593
## 2576 0.00301039593
## 2577 0.00303223651
## 2578 0.00890834422
## 2579 0.12270921790
## 2580 0.05644795096
## 2581 0.05644795096
## 2582 0.00313245541
## 2583 0.05644795096
## 2584 0.11727300429
## 2585 0.40068871586
## 2586 0.17723124426
## 2587 0.11517421878
## 2588 0.17723124426
```

```
## 2589 0.17723124426
## 2590 0.11721665850
## 2591 0.11721665850
## 2592 0.11721665850
## 2593 0.11721665850
## 2594 0.11721665850
## 2595 0.17723124426
## 2596 0.86734768417
## 2597 0.86734768417
## 2598 0.00890834422
## 2599 0.00890834422
## 2600 0.00325407608
## 2601 0.00325407608
## 2602 0.12270921790
## 2603 0.00301039593
## 2604 0.00301039593
## 2605 0.17723124426
## 2606 0.95155692536
## 2607 0.11721665850
## 2608 0.12270921790
## 2609 0.12270921790
## 2610 0.00302418622
## 2611 0.11513888229
## 2612 0.00383728043
## 2613 0.00383728043
## 2614 0.00301039593
## 2615 0.11516665750
## 2616 0.17723124426
## 2617 0.17723124426
## 2618 0.12569448872
## 2619 0.12569448872
## 2620 0.11516991395
## 2621 0.11516991395
## 2622 0.11516989030
## 2623 0.12569448872
## 2624 0.12569448872
## 2625 0.12569448872
## 2626 0.14909301290
## 2627 0.91887711141
## 2628 0.17723124426
## 2629 0.91887711141
## 2630 0.17723124426
## 2631 0.77551916145
## 2632 0.00753639509
## 2633 0.00753639509
## 2634 0.00753639509
## 2635 0.12795169390
## 2636 0.35883430078
## 2637 0.35883430078
## 2638 0.11912157316
## 2639 0.00300693239
## 2640 0.00300693239
## 2641 0.00300693239
## 2642 0.00301039593
```

```
## 2643 0.00301039593
## 2644 0.05349814222
## 2645 0.11581577118
## 2646 0.05349814222
## 2647 0.11581577118
## 2648 0.00300650332
## 2649 0.00300650332
## 2650 0.11058989830
## 2651 0.00890834422
## 2652 0.00301039593
## 2653 0.00295231073
## 2654 0.02945791421
## 2655 0.02945791421
## 2656 0.91887711141
## 2657 0.02945791421
## 2658 0.00383728043
## 2659 0.00301039593
## 2660 0.12270921790
## 2661 0.11405535201
## 2662 0.11518679201
## 2663 0.11405535201
## 2664 0.11405535201
## 2665 0.11405535201
## 2666 0.00383728043
## 2667 0.11649597535
## 2668 0.95135973199
## 2669 0.95157639774
## 2670 0.11517433350
## 2671 0.11565059187
## 2672 0.11517059949
## 2673 0.11581577118
## 2674 0.00300650332
## 2675 0.00300650332
## 2676 0.00301039593
## 2677 0.00301039593
## 2678 0.00301039593
## 2679 0.00301039593
## 2680 0.11721665850
## 2681 0.00300693239
## 2682 0.00313245541
## 2683 0.00300693239
## 2684 0.00300693239
## 2685 0.11721665850
## 2686 0.20561938588
## 2687 0.02945791421
## 2688 0.12270921790
## 2689 0.02945791421
## 2690 0.95153204526
## 2691 0.11721665850
## 2692 0.11516989030
## 2693 0.11516989030
## 2694 0.02945791421
## 2695 0.00890834422
## 2696 0.00890834422
```

```
## 2697 0.13683011879
## 2698 0.01391666720
## 2699 0.01680340092
## 2700 0.91887711141
## 2701 0.12270921790
## 2702 0.12270921790
## 2703 0.89941079491
## 2704 0.78575177345
## 2705 0.00331405746
## 2706 0.00300650332
## 2707 0.00300650332
## 2708 0.00026527022
## 2709 0.12270921790
## 2710 0.12690128676
## 2711 0.12254277409
## 2712 0.12254277409
## 2713 0.00300769057
## 2714 0.00300769057
## 2715 0.12236833769
## 2716 0.12236833769
## 2717 0.11727300429
## 2718 0.86734768417
## 2719 0.00300769057
## 2720 0.00026527022
## 2721 0.00026527022
## 2722 0.86734768417
## 2723 0.00026527022
## 2724 0.00026527022
## 2725 0.11721665850
## 2726 0.11554646739
## 2727 0.11535291446
## 2728 0.11721665850
## 2729 0.11727300429
## 2730 0.00300650332
## 2731 0.00300650332
## 2732 0.11721665850
## 2733 0.00298966614
## 2734 0.00298966614
## 2735 0.00298966614
## 2736 0.00298966614
## 2737 0.11517048205
## 2738 0.11517048205
## 2739 0.11721665850
## 2740 0.19879909423
## 2741 0.11721665850
## 2742 0.00025342200
## 2743 0.00025342200
## 2744 0.00025342200
## 2745 0.00024609043
## 2746 0.00024995434
## 2747 0.00024995434
## 2748 0.00398516753
## 2749 0.00398516753
## 2750 0.13669357040
```

```
## 2751 0.13947270953
## 2752 0.88824815851
## 2753 0.11721665850
## 2754 0.12922352593
## 2755 0.01391666720
## 2756 0.12922352593
## 2757 0.01391666720
## 2758 0.12270921790
## 2759 0.12270921790
## 2760 0.00298966614
## 2761 0.00298966614
## 2762 0.00298966614
## 2763 0.00298966614
## 2764 0.12270921790
## 2765 0.00300693239
## 2766 0.11721665850
## 2767 0.00383728043
## 2768 0.00301039593
## 2769 0.00301039593
## 2770 0.11721665850
## 2771 0.11721665850
## 2772 0.11721665850
## 2773 0.11721665850
## 2774 0.03239848872
## 2775 0.00305211043
## 2776 0.03239848872
## 2777 0.11979831435
## 2778 0.11516991395
## 2779 0.11721665850
## 2780 0.11721665850
## 2781 0.11721665850
## 2782 0.11721665850
## 2783 0.11979831435
## 2784 0.11979831435
## 2785 0.11979831435
## 2786 0.11979831435
## 2787 0.11979831435
## 2788 0.31497566989
## 2789 0.00301039593
## 2790 0.00301039593
## 2791 0.00301039593
## 2792 0.00301039593
## 2793 0.00301039593
## 2794 0.00301039593
## 2795 0.11721665850
## 2796 0.11721665850
## 2797 0.11721665850
## 2798 0.11721665850
## 2799 0.11721665850
## 2800 0.89941079491
## 2801 0.78575177345
## 2802 0.11721665850
## 2803 0.11721665850
## 2804 0.11721665850
```

```
## 2805 0.11721665850
## 2806 0.00301039593
## 2807 0.00301039593
## 2808 0.11721665850
## 2809 0.11513903668
## 2810 0.03239848872
## 2811 0.11721665850
## 2812 0.00301039593
## 2813 0.00301039593
## 2814 0.00301039593
## 2815 0.00301039593
## 2816 0.00301039593
## 2817 0.00305211043
## 2818 0.00305211043
## 2819 0.12270921790
## 2820 0.11721665850
## 2821 0.12270921790
## 2822 0.20561938588
## 2823 0.00295730749
## 2824 0.00295730749
## 2825 0.12270921790
## 2826 0.03239848872
## 2827 0.19754086065
## 2828 0.92083294469
## 2829 0.92083294469
## 2830 0.92083294469
## 2831 0.92083294469
## 2832 0.92083294469
## 2833 0.92083294469
## 2834 0.03239848872
## 2835 0.03239848872
## 2836 0.03239848872
## 2837 0.03282064310
## 2838 0.03239848872
## 2839 0.03239848872
## 2840 0.16776930623
## 2841 0.16776930623
## 2842 0.16776930623
## 2843 0.03332366327
## 2844 0.16776930623
## 2845 0.16776930623
## 2846 0.16776930623
## 2847 0.01680340092
## 2848 0.16776930623
## 2849 0.01680340092
## 2850 0.16776930623
## 2851 0.15836499215
## 2852 0.11721665850
## 2853 0.11721665850
## 2854 0.16776930623
## 2855 0.16776930623
## 2856 0.11596300384
## 2857 0.16667148828
## 2858 0.00303223651
```

```
## 2859 0.16667148828
## 2860 0.16667148828
## 2861 0.00303223651
## 2862 0.11516989030
## 2863 0.01680340092
## 2864 0.95157855552
## 2865 0.95155571817
## 2866 0.95155571817
## 2867 0.77458169666
## 2868 0.00455511461
## 2869 0.22997518705
## 2870 0.16776930623
## 2871 0.00455511461
## 2872 0.00455511461
## 2873 0.00455511461
## 2874 0.00455511461
## 2875 0.00455511461
## 2876 0.00455511461
## 2877 0.12270921790
## 2878 0.12270921790
## 2879 0.00301039593
## 2880 0.00301039593
## 2881 0.00301039593
## 2882 0.00301039593
## 2883 0.00301039593
## 2884 0.00301039593
## 2885 0.00301039593
## 2886 0.00301039593
## 2887 0.01680340092
## 2888 0.94845789993
## 2889 0.94845789993
## 2890 0.11151881780
## 2891 0.10940788021
## 2892 0.19754086065
## 2893 0.01680340092
## 2894 0.10940788021
## 2895 0.00301039593
## 2896 0.11151881780
## 2897 0.10940788021
## 2898 0.11151881780
## 2899 0.10940788021
## 2900 0.16776930623
## 2901 0.16776930623
## 2902 0.12603072128
## 2903 0.12603072128
## 2904 0.11727300429
## 2905 0.11727300429
## 2906 0.11727300429
## 2907 0.11721665850
## 2908 0.20125788696
## 2909 0.10545449731
## 2910 0.10545449731
## 2911 0.12270921790
## 2912 0.00301039593
```

```
## 2913 0.00301039593
## 2914 0.00300650332
## 2915 0.94873748419
## 2916 0.00301039593
## 2917 0.00301039593
## 2918 0.00301039593
## 2919 0.00301039593
## 2920 0.00301039593
## 2921 0.00301039593
## 2922 0.00301039593
## 2923 0.12270921790
## 2924 0.12270921790
## 2925 0.12270921790
## 2926 0.11721665850
## 2927 0.11721665850
## 2928 0.11721665850
## 2929 0.11721665850
## 2930 0.11721665850
## 2931 0.11721665850
## 2932 0.11721665850
## 2933 0.12603072128
## 2934 0.11721665850
## 2935 0.20561938588
## 2936 0.20561938588
## 2937 0.11721665850
## 2938 0.11721665850
## 2939 0.11721665850
## 2940 0.00300650332
## 2941 0.11721665850
## 2942 0.11721665850
## 2943 0.11721665850
## 2944 0.11721665850
## 2945 0.11721665850
## 2946 0.11721665850
## 2947 0.11721665850
## 2948 0.11721665850
## 2949 0.11721665850
## 2950 0.11721665850
## 2951 0.11721665850
## 2952 0.11721665850
## 2953 0.01680340092
## 2954 0.00300650332
## 2955 0.86734768417
## 2956 0.86734768417
## 2957 0.02777656795
## 2958 0.02777656795
## 2959 0.02777656795
## 2960 0.00300650332
## 2961 0.00300650332
## 2962 0.12603072128
## 2963 0.00300650332
## 2964 0.00300650332
## 2965 0.12603072128
## 2966 0.11516989030
```

```
## 2967 0.00927642829
## 2968 0.00927642829
## 2969 0.00301089175
## 2970 0.00301089175
## 2971 0.00301089175
## 2972 0.11516989030
## 2973 0.11721665850
## 2974 0.22997518705
## 2975 0.11721665850
## 2976 0.11721665850
## 2977 0.01680340092
## 2978 0.00300650332
## 2979 0.00300650332
## 2980 0.00300650332
## 2981 0.00300650332
## 2982 0.00300650332
## 2983 0.12603072128
## 2984 0.00300650332
## 2985 0.12603072128
## 2986 0.10993419758
## 2987 0.00300650332
## 2988 0.00300650332
## 2989 0.15836499215
## 2990 0.00281282597
## 2991 0.03468291114
## 2992 0.00301089175
## 2993 0.00300650728
## 2994 0.00300650728
## 2995 0.00300650728
## 2996 0.03505191235
## 2997 0.63386307586
## 2998 0.00339048627
## 2999 0.00300650728
## 3000 0.00300650728
## 3001 0.00300650728
## 3002 0.00300650728
## 3003 0.00300650728
## 3004 0.00300650728
## 3005 0.00661595748
## 3006 0.00312596167
## 3007 0.00312596167
## 3008 0.00312596167
## 3009 0.00312596167
## 3010 0.00312596167
## 3011 0.03239848872
## 3012 0.13493868435
## 3013 0.00300665237
## 3014 0.00300665237
## 3015 0.03414310642
## 3016 0.11366168025
## 3017 0.00753639509
## 3018 0.03308272732
## 3019 0.03532743906
## 3020 0.00300650728
```

```
## 3021 0.00300650728
## 3022 0.92083294469
## 3023 0.18199919035
## 3024 0.15836499215
## 3025 0.15836499215
## 3026 0.76408695420
## 3027 0.92635634699
## 3028 0.11589419107
## 3029 0.41938525392
## 3030 0.92083294469
## 3031 0.11516989030
## 3032 0.11721665850
## 3033 0.11721665850
## 3034 0.11517491162
## 3035 0.12270921790
## 3036 0.11516989030
## 3037 0.11516989030
## 3038 0.19301788910
## 3039 0.28267116800
## 3040 0.03891767774
## 3041 0.03891767774
## 3042 0.11517997033
## 3043 0.95088255192
## 3044 0.03891767774
## 3045 0.95098895813
## 3046 0.95098895813
## 3047 0.95098895813
## 3048 0.95098895813
## 3049 0.00301986500
## 3050 0.95098895813
## 3051 0.95098895813
## 3052 0.01680340092
## 3053 0.01680340092
## 3054 0.00312596167
## 3055 0.00312596167
## 3056 0.00300650728
## 3057 0.94548494500
## 3058 0.03322020008
## 3059 0.03324119475
## 3060 0.28352335515
## 3061 0.28352335515
## 3062 0.19879909423
## 3063 0.19879909423
## 3064 0.15836499215
## 3065 0.00300650728
## 3066 0.00300650728
## 3067 0.11516991395
## 3068 0.00312596167
## 3069 0.11721665850
## 3070 0.18522596317
## 3071 0.95111824039
## 3072 0.28352335515
## 3073 0.28352335515
## 3074 0.58895400476
```

```
## 3075 0.03394005074
## 3076 0.03310346112
## 3077 0.01680340092
## 3078 0.01680340092
## 3079 0.11519404985
## 3080 0.11519404985
## 3081 0.12571881318
## 3082 0.01680340092
## 3083 0.01680340092
## 3084 0.11516989030
## 3085 0.11516991395
## 3086 0.11721665850
## 3087 0.12270921790
## 3088 0.09078779334
## 3089 0.00300650332
## 3090 0.00300650332
## 3091 0.00300650332
## 3092 0.01680340092
## 3093 0.01680340092
## 3094 0.01680340092
## 3095 0.01680340092
## 3096 0.01680340092
## 3097 0.01680340092
## 3098 0.09078779334
## 3099 0.10393553130
## 3100 0.92635634699
## 3101 0.94845789993
## 3102 0.03503487163
## 3103 0.03503487163
## 3104 0.03503487163
## 3105 0.00300769057
## 3106 0.00300769057
## 3107 0.10545877037
## 3108 0.10545877037
## 3109 0.00301458806
## 3110 0.00300650332
## 3111 0.03503487163
## 3112 0.10393553130
## 3113 0.95024840895
## 3114 0.95000848468
## 3115 0.95024840895
## 3116 0.03387147209
## 3117 0.41503321258
## 3118 0.00300650332
## 3119 0.10393553130
## 3120 0.00301039593
## 3121 0.11708121375
## 3122 0.10393553130
## 3123 0.00300650332
## 3124 0.12270921790
## 3125 0.12270921790
## 3126 0.12270921790
## 3127 0.12270921790
## 3128 0.12270921790
```

```
## 3129 0.01680340092
## 3130 0.00301039593
## 3131 0.00301039593
## 3132 0.11721665850
## 3133 0.11516991395
## 3134 0.11516991395
## 3135 0.11516991395
## 3136 0.11516991395
## 3137 0.11721665850
## 3138 0.12571881318
## 3139 0.12571881318
## 3140 0.12571881318
## 3141 0.95127306362
## 3142 0.00301518183
## 3143 0.00301518183
## 3144 0.11997418677
## 3145 0.95127306362
## 3146 0.10393553130
## 3147 0.94845789993
## 3148 0.94770452255
## 3149 0.94770452255
## 3150 0.94770452255
## 3151 0.94770452255
## 3152 0.94770452255
## 3153 0.94770452255
## 3154 0.94770452255
## 3155 0.00300693239
## 3156 0.00300693239
## 3157 0.00300693239
## 3158 0.00300693239
## 3159 0.10393553130
## 3160 0.00300693239
## 3161 0.00300674141
## 3162 0.00300693239
## 3163 0.94992187480
## 3164 0.00300650332
## 3165 0.46056195091
## 3166 0.12270921790
## 3167 0.12270921790
## 3168 0.00300650332
## 3169 0.95024840895
## 3170 0.11475544951
## 3171 0.11475544951
## 3172 0.11475544951
## 3173 0.11475544951
## 3174 0.11516984893
## 3175 0.21364497971
## 3176 0.11517003596
## 3177 0.11516991395
## 3178 0.11516991395
## 3179 0.00300650332
## 3180 0.00434186628
## 3181 0.00434186628
## 3182 0.00434186628
```

```
## 3183 0.11513888229
## 3184 0.11721665850
## 3185 0.95024388448
## 3186 0.13681231833
## 3187 0.01680340092
## 3188 0.95157827840
## 3189 0.00300650332
## 3190 0.06458767761
## 3191 0.92083294469
## 3192 0.39403661904
## 3193 0.20125788696
## 3194 0.20125788696
## 3195 0.00300650332
## 3196 0.00300650332
## 3197 0.00300650332
## 3198 0.95114923725
## 3199 0.74357932807
## 3200 0.11669917509
## 3201 0.10454557052
## 3202 0.10454557052
## 3203 0.00300650332
## 3204 0.00300650332
## 3205 0.00300650332
## 3206 0.00300650332
## 3207 0.00301986500
## 3208 0.13654201996
## 3209 0.11669917509
## 3210 0.95098895813
## 3211 0.11669917509
## 3212 0.11721665850
## 3213 0.24884599669
## 3214 0.10454557052
## 3215 0.00300769057
## 3216 0.63386307586
## 3217 0.01391666720
## 3218 0.01391666720
## 3219 0.11669917509
## 3220 0.10454557052
## 3221 0.11669917509
## 3222 0.11997418677
## 3223 0.10454557052
## 3224 0.13654201996
## 3225 0.00300694176
## 3226 0.95156906503
## 3227 0.11669917509
## 3228 0.00024609043
## 3229 0.10454557052
## 3230 0.10454557052
## 3231 0.11721665850
## 3232 0.11721665850
## 3233 0.00300769057
## 3234 0.00301072925
## 3235 0.00300650332
## 3236 0.13654201996
```

```
## 3237 0.11721665850
## 3238 0.13654201996
## 3239 0.13654201996
## 3240 0.00300650332
## 3241 0.00300650332
## 3242 0.00300650332
## 3243 0.11669917509
## 3244 0.13654201996
## 3245 0.00300650332
## 3246 0.28182563816
## 3247 0.12363656426
## 3248 0.11516989030
## 3249 0.01680340092
## 3250 0.01680340092
## 3251 0.12363656426
## 3252 0.12363656426
## 3253 0.12363656426
## 3254 0.01680340092
## 3255 0.11516989030
## 3256 0.11516989030
## 3257 0.95088255192
## 3258 0.12363656426
## 3259 0.01680340092
## 3260 0.00300650332
## 3261 0.00300650332
## 3262 0.01680340092
## 3263 0.00301039593
## 3264 0.16144723850
## 3265 0.16144723850
## 3266 0.00300893328
## 3267 0.16144723850
## 3268 0.12803901608
## 3269 0.16144723850
## 3270 0.00300893328
## 3271 0.00300893328
## 3272 0.00300674141
## 3273 0.95088255192
## 3274 0.95088255192
## 3275 0.00300674141
## 3276 0.00300674141
## 3277 0.00300674141
## 3278 0.00300674141
## 3279 0.00300893328
## 3280 0.00300893328
## 3281 0.00300893328
## 3282 0.00300672254
## 3283 0.11721665850
## 3284 0.11721665850
## 3285 0.00294078901
## 3286 0.00294078901
## 3287 0.00294078901
## 3288 0.00294078901
## 3289 0.00294078901
## 3290 0.00294078901
```

```
## 3291 0.00294078901
## 3292 0.00294078901
## 3293 0.00294078901
## 3294 0.00294078901
## 3295 0.00294078901
## 3296 0.00294078901
## 3297 0.11721665850
## 3298 0.00294078901
## 3299 0.00294078901
## 3300 0.00294078901
## 3301 0.00294078901
## 3302 0.00294078901
## 3303 0.08807224873
## 3304 0.11514021659
## 3305 0.11721665850
## 3306 0.11721665850
## 3307 0.11721665850
## 3308 0.11721665850
## 3309 0.00301039593
## 3310 0.00301039593
## 3311 0.00301039593
## 3312 0.95154408897
## 3313 0.95154408897
## 3314 0.95154408897
## 3315 0.95154408897
## 3316 0.94406723493
## 3317 0.94406723493
## 3318 0.94796171536
## 3319 0.95136995753
## 3320 0.95143502798
## 3321 0.94796171536
## 3322 0.11516989030
## 3323 0.11516989030
## 3324 0.11721665850
## 3325 0.00301039593
## 3326 0.00300650332
## 3327 0.00300650332
## 3328 0.95088255192
## 3329 0.95088255192
## 3330 0.11721665850
## 3331 0.00300769057
## 3332 0.11516991395
## 3333 0.00300650332
## 3334 0.02445781201
## 3335 0.02120345498
## 3336 0.16144723850
## 3337 0.24884599669
## 3338 0.16144723850
## 3339 0.13864356581
## 3340 0.00383728043
## 3341 0.13053641195
## 3342 0.13053641195
## 3343 0.92899528133
## 3344 0.01092288557
```

```
## 3345 0.20827991389
## 3346 0.12357948477
## 3347 0.20827991389
## 3348 0.00301039593
## 3349 0.00302418622
## 3350 0.00305211043
## 3351 0.13053641195
## 3352 0.95155571817
## 3353 0.94852208040
## 3354 0.94595603775
## 3355 0.94595603775
## 3356 0.94852208040
## 3357 0.94595603775
## 3358 0.95155571817
## 3359 0.94852208040
## 3360 0.94595603775
## 3361 0.94852208040
## 3362 0.95157855552
## 3363 0.13053641195
## 3364 0.13479386581
## 3365 0.00300769057
## 3366 0.00300769057
## 3367 0.00300769057
## 3368 0.00300769057
## 3369 0.00300769057
## 3370 0.14522439467
## 3371 0.00301039593
## 3372 0.00301039593
## 3373 0.00301039593
## 3374 0.00301039593
## 3375 0.19879909423
## 3376 0.11721665850
## 3377 0.11721665850
## 3378 0.11721665850
## 3379 0.11721665850
## 3380 0.00301039593
## 3381 0.00301039593
## 3382 0.95155571817
## 3383 0.12363656426
## 3384 0.95157855552
## 3385 0.00301039593
## 3386 0.00301039593
## 3387 0.00301039593
## 3388 0.94796171536
## 3389 0.95136995753
## 3390 0.69629138658
## 3391 0.94796171536
## 3392 0.95136995753
## 3393 0.69629138658
## 3394 0.00303223651
## 3395 0.00303223651
## 3396 0.00303223651
## 3397 0.00303223651
## 3398 0.00303223651
```

```
## 3399 0.00303223651
## 3400 0.11624768552
## 3401 0.00303223651
## 3402 0.00301039593
## 3403 0.00303223651
## 3404 0.00303223651
## 3405 0.00301039593
## 3406 0.00301039593
## 3407 0.67455769004
## 3408 0.67455769004
## 3409 0.67455769004
## 3410 0.01391666720
## 3411 0.11721665850
## 3412 0.95155571817
## 3413 0.95154408897
## 3414 0.95154408897
## 3415 0.95155571817
## 3416 0.92486187087
## 3417 0.92486187087
## 3418 0.11721665850
## 3419 0.00303223651
## 3420 0.00303223651
## 3421 0.00303223651
## 3422 0.11624768552
## 3423 0.00303223651
## 3424 0.00303223651
## 3425 0.00303223651
## 3426 0.00303223651
## 3427 0.00303223651
## 3428 0.00331405746
## 3429 0.00300769057
## 3430 0.00300769057
## 3431 0.00300769057
## 3432 0.11721665850
## 3433 0.00303223651
## 3434 0.00303223651
## 3435 0.00303223651
## 3436 0.00303223651
## 3437 0.12363656426
## 3438 0.12803901608
## 3439 0.30885381920
## 3440 0.11627118989
## 3441 0.00303223651
## 3442 0.00303223651
## 3443 0.00303223651
## 3444 0.12270921790
## 3445 0.00300650728
## 3446 0.00303223651
## 3447 0.00303223651
## 3448 0.00303223651
## 3449 0.95157855552
## 3450 0.92150923528
## 3451 0.11721665850
## 3452 0.11721665850
```

```
## 3453 0.11721665850
## 3454 0.11721665850
## 3455 0.11721665850
## 3456 0.11721665850
## 3457 0.11721665850
## 3458 0.11721665850
## 3459 0.11721665850
## 3460 0.11721665850
## 3461 0.11721665850
## 3462 0.92775245562
## 3463 0.11721665850
## 3464 0.11721665850
## 3465 0.00300674141
## 3466 0.00303223651
## 3467 0.08740552571
## 3468 0.00298966614
## 3469 0.08740552571
## 3470 0.00310668522
## 3471 0.08740552571
## 3472 0.00303223651
## 3473 0.11624768552
## 3474 0.00303223651
## 3475 0.00303223651
## 3476 0.12363656426
## 3477 0.00303223651
## 3478 0.12363656426
## 3479 0.12363656426
## 3480 0.12363656426
## 3481 0.12363656426
## 3482 0.12363656426
## 3483 0.00303223651
## 3484 0.00303223651
## 3485 0.00303223651
## 3486 0.00303223651
## 3487 0.00738200246
## 3488 0.12363656426
## 3489 0.95061204118
## 3490 0.95061204118
## 3491 0.95061204118
## 3492 0.12363656426
## 3493 0.12363656426
## 3494 0.12363656426
## 3495 0.12363656426
## 3496 0.95094089083
## 3497 0.12363656426
## 3498 0.12363656426
## 3499 0.94956560173
## 3500 0.94956560173
## 3501 0.12363656426
## 3502 0.12363656426
## 3503 0.95061204118
## 3504 0.95061204118
## 3505 0.12363656426
## 3506 0.12363656426
```

```
## 3507 0.12363656426
## 3508 0.94852208040
## 3509 0.11727300429
## 3510 0.12363656426
## 3511 0.12803901608
## 3512 0.12363656426
## 3513 0.12363656426
## 3514 0.12363656426
## 3515 0.95150406870
## 3516 0.29678949595
## 3517 0.95150406870
## 3518 0.12363656426
## 3519 0.12363656426
## 3520 0.12363656426
## 3521 0.12363656426
## 3522 0.12363656426
## 3523 0.12363656426
## 3524 0.11553641158
## 3525 0.11553641158
## 3526 0.00315622026
## 3527 0.00315622026
## 3528 0.00315622026
## 3529 0.19879909423
## 3530 0.95061204118
## 3531 0.11517066524
## 3532 0.11517066524
## 3533 0.11517066524
## 3534 0.95061204118
## 3535 0.95061204118
## 3536 0.12363656426
## 3537 0.00301039593
## 3538 0.67455769004
## 3539 0.86422103079
## 3540 0.67455769004
## 3541 0.67455769004
## 3542 0.00301012849
## 3543 0.00303223651
## 3544 0.19879909423
## 3545 0.19879909423
## 3546 0.11721665850
## 3547 0.11721665850
## 3548 0.11721665850
## 3549 0.12109319015
## 3550 0.11516989030
## 3551 0.00303223651
## 3552 0.00303223651
## 3553 0.00303223651
## 3554 0.12363656426
## 3555 0.36132082330
## 3556 0.03127888629
## 3557 0.27740487216
## 3558 0.95088255192
## 3559 0.00300769057
## 3560 0.11986133760
```

```
## 3561 0.00303223651
## 3562 0.11624768552
## 3563 0.00303223651
## 3564 0.11624768552
## 3565 0.00303223651
## 3566 0.00303223651
## 3567 0.00303223651
## 3568 0.00303223651
## 3569 0.00303223651
## 3570 0.00303223651
## 3571 0.00303223651
## 3572 0.00303223651
## 3573 0.00303223651
## 3574 0.95140507011
## 3575 0.92083294469
## 3576 0.36132082330
## 3577 0.11721665850
## 3578 0.95157268755
## 3579 0.11721665850
## 3580 0.11721665850
## 3581 0.95061204118
## 3582 0.11721665850
## 3583 0.00303223651
## 3584 0.11721665850
## 3585 0.11721665850
## 3586 0.95157268755
## 3587 0.12363656426
## 3588 0.11624768552
## 3589 0.00303223651
## 3590 0.00303223651
## 3591 0.00303223651
## 3592 0.00303223651
## 3593 0.00303223651
## 3594 0.00303223651
## 3595 0.00303223651
## 3596 0.00303223651
## 3597 0.00303223651
## 3598 0.19879909423
## 3599 0.11516991395
## 3600 0.00303223651
## 3601 0.00303223651
## 3602 0.12803901608
## 3603 0.20125788696
## 3604 0.11517066524
## 3605 0.11517066524
## 3606 0.19879909423
## 3607 0.19879909423
## 3608 0.19879909423
## 3609 0.00301039593
## 3610 0.11554646739
## 3611 0.11554646739
## 3612 0.86734768417
## 3613 0.12803901608
## 3614 0.11516989030
```

```
## 3615 0.19879909423
## 3616 0.19879909423
## 3617 0.17391515180
## 3618 0.29678949595
## 3619 0.29678949595
## 3620 0.18777448088
## 3621 0.18777448088
## 3622 0.18777448088
## 3623 0.12803901608
## 3624 0.11516989030
## 3625 0.11516989030
## 3626 0.00301039593
## 3627 0.00301039593
## 3628 0.00301039593
## 3629 0.00301039593
## 3630 0.00301039593
## 3631 0.00303223651
## 3632 0.00301039593
## 3633 0.19879909423
## 3634 0.19879909423
## 3635 0.15516258668
## 3636 0.00301039593
## 3637 0.00301039593
## 3638 0.00301039593
## 3639 0.00301039593
## 3640 0.00455895286
## 3641 0.00301039593
## 3642 0.00455895286
## 3643 0.00455895286
## 3644 0.00303223651
## 3645 0.86734768417
## 3646 0.00303223651
## 3647 0.00301039593
## 3648 0.29678949595
## 3649 0.27740487216
## 3650 0.67455769004
## 3651 0.86734768417
## 3652 0.46914189468
## 3653 0.11516989030
## 3654 0.11516989030
## 3655 0.11516989030
## 3656 0.11498483283
## 3657 0.19879909423
## 3658 0.46914189468
## 3659 0.46914189468
## 3660 0.03280703579
## 3661 0.69265351779
## 3662 0.00300769057
## 3663 0.95148106261
## 3664 0.69265351779
## 3665 0.11528789535
## 3666 0.11528789535
## 3667 0.11528789535
## 3668 0.00300769057
```

```
## 3669 0.00300769057
## 3670 0.00300769057
## 3671 0.00300769057
## 3672 0.00300769057
## 3673 0.00300769057
## 3674 0.00300769057
## 3675 0.00300769057
## 3676 0.00300769057
## 3677 0.00300769057
## 3678 0.11516989030
## 3679 0.11516989030
## 3680 0.00349791788
## 3681 0.11721665850
## 3682 0.11553641158
## 3683 0.11553641158
## 3684 0.67455769004
## 3685 0.56633522834
## 3686 0.56633522834
## 3687 0.11517045518
## 3688 0.00303223651
## 3689 0.19879909423
## 3690 0.00303223651
## 3691 0.00303223651
## 3692 0.00303223651
## 3693 0.75771394262
## 3694 0.10317758272
## 3695 0.10317758272
## 3696 0.00303223651
## 3697 0.00303223651
## 3698 0.00303223651
## 3699 0.00303223651
## 3700 0.11525864972
## 3701 0.95061204118
## 3702 0.95061204118
## 3703 0.95061204118
## 3704 0.95061204118
## 3705 0.95061204118
## 3706 0.95061204118
## 3707 0.00738200246
## 3708 0.00738200246
## 3709 0.00303223651
## 3710 0.00303223651
## 3711 0.00303223651
## 3712 0.00303223651
## 3713 0.00303223651
## 3714 0.00303223651
## 3715 0.11525864972
## 3716 0.19879909423
## 3717 0.94568408723
## 3718 0.94568408723
## 3719 0.19879909423
## 3720 0.19879909423
## 3721 0.00738200246
## 3722 0.00738200246
```

```
## 3723 0.00738200246
## 3724 0.00738200246
## 3725 0.11719112119
## 3726 0.11721665850
## 3727 0.11721665850
## 3728 0.11721665850
## 3729 0.11721665850
## 3730 0.11721665850
## 3731 0.11721665850
## 3732 0.11721665850
## 3733 0.19879909423
## 3734 0.11516989030
## 3735 0.11516989030
## 3736 0.11516989030
## 3737 0.12270921790
## 3738 0.00303223651
## 3739 0.00303223651
## 3740 0.00303223651
## 3741 0.00303223651
## 3742 0.00303223651
## 3743 0.00303223651
## 3744 0.00303223651
## 3745 0.00303223651
## 3746 0.00303223651
## 3747 0.00303223651
## 3748 0.95061204118
## 3749 0.95096940373
## 3750 0.13236355868
## 3751 0.13236355868
## 3752 0.95155632336
## 3753 0.91989151748
## 3754 0.91989151748
## 3755 0.94504320376
## 3756 0.94504320376
## 3757 0.95061204118
## 3758 0.11721665850
## 3759 0.11721665850
## 3760 0.00300650728
## 3761 0.00927642829
## 3762 0.00927642829
## 3763 0.95140222216
## 3764 0.93774770168
## 3765 0.92303247396
## 3766 0.31070493905
## 3767 0.20826222994
## 3768 0.11721665850
## 3769 0.11721665850
## 3770 0.11721665850
## 3771 0.11721665850
## 3772 0.00301039593
## 3773 0.00301039593
## 3774 0.00301039593
## 3775 0.00303223651
## 3776 0.11721665850
```

```
## 3777 0.20826222994
## 3778 0.11986133760
## 3779 0.11986133760
## 3780 0.00298966614
## 3781 0.00298966614
## 3782 0.11721665850
## 3783 0.11721665850
## 3784 0.11721665850
## 3785 0.11721665850
## 3786 0.11721665850
## 3787 0.11721665850
## 3788 0.11721665850
## 3789 0.11721665850
## 3790 0.11721665850
## 3791 0.11721665850
## 3792 0.00927642829
## 3793 0.00300650728
## 3794 0.00300650728
## 3795 0.00927642829
## 3796 0.00927642829
## 3797 0.11338362541
## 3798 0.00300845748
## 3799 0.11338362541
## 3800 0.95061204118
## 3801 0.95061204118
## 3802 0.11997418677
## 3803 0.00927642829
## 3804 0.00927642829
## 3805 0.00300650728
## 3806 0.11997418677
## 3807 0.11997418677
## 3808 0.11721665850
## 3809 0.00303223651
## 3810 0.00303223651
## 3811 0.00303223651
## 3812 0.11721665850
## 3813 0.11721665850
## 3814 0.11721665850
## 3815 0.11721665850
## 3816 0.00300650728
## 3817 0.12270921790
## 3818 0.95157855552
## 3819 0.95157855552
## 3820 0.95157855552
## 3821 0.95157855552
## 3822 0.00300650728
## 3823 0.20125788696
## 3824 0.10880302425
## 3825 0.12175902099
## 3826 0.86734768417
## 3827 0.20826222994
## 3828 0.11727300429
## 3829 0.95061204118
## 3830 0.95131661850
```

```
## 3831 0.95131661850
## 3832 0.00300769057
## 3833 0.00300693239
## 3834 0.20826222994
## 3835 0.20826222994
## 3836 0.00300650728
## 3837 0.00300650728
## 3838 0.00300650728
## 3839 0.00300650728
## 3840 0.00300650728
## 3841 0.22997518705
## 3842 0.20826222994
## 3843 0.91246160464
## 3844 0.11517003596
## 3845 0.20826222994
## 3846 0.95061204118
## 3847 0.95061204118
## 3848 0.11517003596
## 3849 0.20826222994
## 3850 0.95061204118
## 3851 0.95061204118
## 3852 0.95061204118
## 3853 0.95061204118
## 3854 0.11516989030
## 3855 0.11516989030
## 3856 0.95061204118
## 3857 0.95061204118
## 3858 0.87675764090
## 3859 0.69265351779
## 3860 0.12357948477
## 3861 0.12357948477
## 3862 0.12357948477
## 3863 0.20827991389
## 3864 0.20827991389
## 3865 0.00300769057
## 3866 0.12021981393
## 3867 0.00300665237
## 3868 0.00300665237
## 3869 0.00300665237
## 3870 0.00300665237
## 3871 0.00300665237
## 3872 0.00300665237
## 3873 0.00300665237
## 3874 0.00300665237
## 3875 0.00394701640
## 3876 0.00300693239
## 3877 0.00300665237
## 3878 0.00300665237
## 3879 0.00300665237
## 3880 0.87675764090
## 3881 0.91541702740
## 3882 0.11517066524
## 3883 0.11721665850
## 3884 0.11721665850
```

```
## 3885 0.11721665850
## 3886 0.11721665850
## 3887 0.11721665850
## 3888 0.11721665850
## 3889 0.11721665850
## 3890 0.47930611299
## 3891 0.47930611299
## 3892 0.00300769057
## 3893 0.00300769057
## 3894 0.00300769057
## 3895 0.00300769057
## 3896 0.00024076776
## 3897 0.21688042512
## 3898 0.21688042512
## 3899 0.11721665850
## 3900 0.47930611299
## 3901 0.69006863337
## 3902 0.92303247396
## 3903 0.00300650728
## 3904 0.93137609800
## 3905 0.93137609800
## 3906 0.00300650332
## 3907 0.00300650332
## 3908 0.00300650332
## 3909 0.95088255192
## 3910 0.95088255192
## 3911 0.00303223651
## 3912 0.95088255192
## 3913 0.00300769057
## 3914 0.00300769057
## 3915 0.00300769057
## 3916 0.00300650728
## 3917 0.69629138658
## 3918 0.69629138658
## 3919 0.69629138658
## 3920 0.69629138658
## 3921 0.69629138658
## 3922 0.75265377808
## 3923 0.11517059949
## 3924 0.95061204118
## 3925 0.00738200246
## 3926 0.00303993204
## 3927 0.00300693239
## 3928 0.00303993204
## 3929 0.00300693239
## 3930 0.00303993204
## 3931 0.00300693239
## 3932 0.94796171536
## 3933 0.00394701640
## 3934 0.00394701640
## 3935 0.94987161358
## 3936 0.94979191931
## 3937 0.94979191931
## 3938 0.94979191931
```

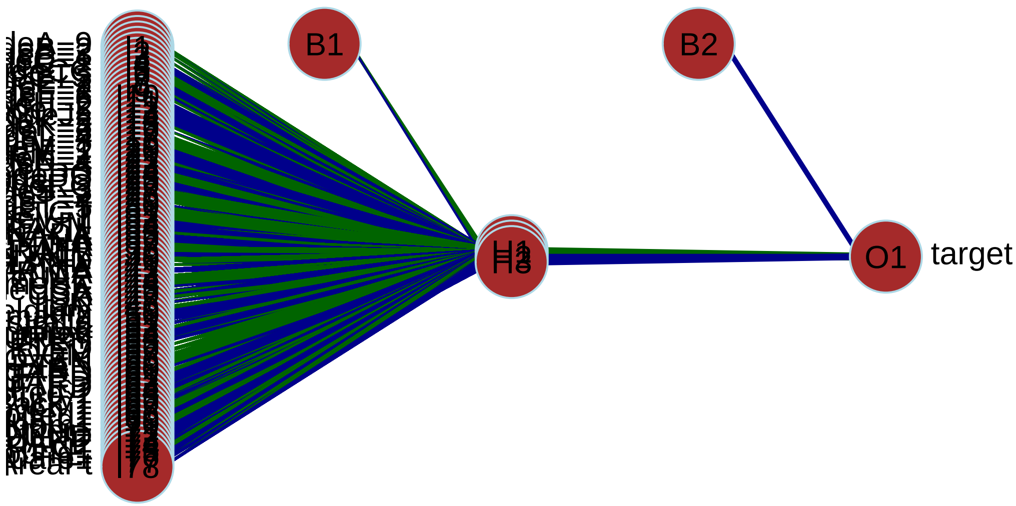
```
## 3939 0.00300650728
## 3940 0.00300650728
## 3941 0.00300650728
## 3942 0.00300650728
## 3943 0.00300650728
## 3944 0.00300650728
## 3945 0.00300650728
## 3946 0.69441813965
## 3947 0.95156782882
## 3948 0.95156782882
## 3949 0.11532630229
## 3950 0.69006863337
## 3951 0.19821741957
## 3952 0.79380511745
## 3953 0.94941078760
## 3954 0.62155887933
## 3955 0.82995883826
## 3956 0.00303223651
## 3957 0.00303993204
## 3958 0.69006863337
## 3959 0.94979191931
## 3960 0.00303223651
## 3961 0.00303223651
## 3962 0.00303223651
## 3963 0.00303223651
## 3964 0.00303223651
## 3965 0.94979191931
## 3966 0.11531789718
## 3967 0.11531789718
## 3968 0.11531789718
## 3969 0.12794220132
## 3970 0.92775245562
## 3971 0.95157855552
## 3972 0.95157855552
## 3973 0.92775245562
## 3974 0.00303223651
## 3975 0.00303223651
## 3976 0.00303223651
## 3977 0.00388005328
## 3978 0.12794220132
## 3979 0.12794220132
## 3980 0.12794220132
## 3981 0.00300650332
## 3982 0.15704103770
## 3983 0.15704103770
## 3984 0.36701512673
## 3985 0.00303223651
## 3986 0.00300845748
## 3987 0.00300845748
## 3988 0.00300845748
## 3989 0.11565059187
## 3990 0.00300650332
## 3991 0.00300650332
## 3992 0.00300650332
```

```
## 3993 0.00303223651
## 3994 0.91530991898
## 3995 0.91530991898
## 3996 0.91530991898
## 3997 0.94941078760
## 3998 0.94941078760
## 3999 0.11721665850
## 4000 0.94941078760
## 4001 0.69428241214
## 4002 0.94987161358
## 4003 0.11517028594
## 4004 0.94987161358
## 4005 0.94941078760
## 4006 0.94941078760
## 4007 0.94941078760
## 4008 0.92303247396
## 4009 0.92303247396
## 4010 0.12794220132
## 4011 0.12769277785
## 4012 0.12769277785
## 4013 0.12769277785
## 4014 0.95078319585
## 4015 0.94979191931
## 4016 0.79380511745
## 4017 0.12340971941
## 4018 0.12769277785
## 4019 0.94420527560
## 4020 0.94979191931
## 4021 0.00303223651
## 4022 0.84442391860
## 4023 0.95157571150
## 4024 0.95157571150
## 4025 0.95157571150
## 4026 0.35059768281
## 4027 0.94845678111
## 4028 0.95086338793
## 4029 0.00303223651
## 4030 0.95157571150
## 4031 0.95150942052
## 4032 0.11912157316
## 4033 0.11912157316
## 4034 0.95155797234
## 4035 0.12270921790
## 4036 0.95150942052
## 4037 0.20827991389
## 4038 0.95157571150
## 4039 0.21915693111
## 4040 0.21915693111
## 4041 0.95157742974
## 4042 0.87465619903
## 4043 0.82695342129
## 4044 0.11721665850
## 4045 0.93247113523
## 4046 0.69629138658
```

```
## 4047 0.00300650728
## 4048 0.11721665850
## 4049 0.11619008240
## 4050 0.11619008240
## 4051 0.00300693239
## 4052 0.95157855552
## 4053 0.95157855552
## 4054 0.95155632791
## 4055 0.95157855552
## 4056 0.11727300429
## 4057 0.93247113523
## 4058 0.87465619903
## 4059 0.90384850180
## 4060 0.82695342129
## 4061 0.87465619903
## 4062 0.87465619903
## 4063 0.94906905328
## 4064 0.94906905328
## 4065 0.95157855552
## 4066 0.95157855552
## 4067 0.11727300429
## 4068 0.95157855552
## 4069 0.95157855552
## 4070 0.95157818382
## 4071 0.11749858465
## 4072 0.11749858465
## 4073 0.95088255192
## 4074 0.95088255192
## 4075 0.11912157316
## 4076 0.13677503920
## 4077 0.95157839231
## 4078 0.93782026970
## 4079 0.00301039593
## 4080 0.95103426806
## 4081 0.11721665850
## 4082 0.95157855552
## 4083 0.95094089083
## 4084 0.11721665850
## 4085 0.11721665850
## 4086 0.11721665850
## 4087 0.95157855552
## 4088 0.95157855552
## 4089 0.95157625961
## 4090 0.95157855552
## 4091 0.00394701640
## 4092 0.00394701640
## 4093 0.00394701640
## 4094 0.95157855552
## 4095 0.11721665850
```

To draw a nnet model, we can use "plotnet()" function from "NeuralNetTools" package:

```
library(NeuralNetTools)
plotnet(nnModel, pos_col='darkgreen',neg_col='darkblue',
circle_cex=5,
circle_col='brown')
```



The neural network model can be used to make predictions for our test instances:

```
nn.preds = predict(nnModel, test)
```

We still have results between 0 and 1 that are more like probabilities of belonging to each class. To get the predicted classes, we can use change the type argument:

```
nn.preds = as.factor(predict(nnModel, test, type = "class"))
```

Now lets create a simple confusion matrix:

```
CM <- table(nn.preds, test$target, dnn = c("predicted","actual"))
print(CM)
```

```
##          actual
## predicted   NO  YES
##       NO  1346   99
##       YES   29  251
```

We can check the performance of the neural network model:

```
library(plyr)
error_metric = function(CM)
{
  TN = CM[1,1]
  TP = CM[2,2]
  FN = CM[1,2]
  FP = CM[2,1]
  recall = (TP)/(TP+FN)
  precision =(TP)/(TP+FP)
  falsePositiveRate = (FP)/(FP+TN)
  falseNegativeRate = (FN)/(FN+TP)
  error =(FP+FN)/(TP+TN+FP+FN)
  modelPerf <- list("precision" = precision,
                    "recall" = recall,
                    "falsepositiverate" = falsePositiveRate,
                    "falsenegativerate" = falseNegativeRate,
                    "error" = error)
  return(modelPerf)
}

outPutlist <- error_metric(CM)


df <- ldply(outPutlist, data.frame)
setNames(df, c("", "Values"))
```

```
##                         Values
## 1         precision 0.89642857
## 2            recall 0.71714286
## 3 falsepositiverate 0.02109091
## 4 falsenegativerate 0.28285714
## 5             error 0.07420290
```

Summary

The final result shows us that our model has a precision of **89.64%** and an error rate of **7.42%**. Also, according to our confusion matrix, **1346** instances with a true category of order conversion "not happening" were correctly predicted as not happening, and **251** instances with a true category of order conversion "happening" were correctly predicted as happening. However, **99** instances where the true category was order conversion "not happening" were incorrectly predicted as happening, and **29** instances where the true category was "happening" were incorrectly predicted as not happening.

Cross Validation

In order to perform cross validation on our data set, we do not consider two variables in our data set called **CustomerCode**, and **CountryName**, as these variables are causing level mismatches when running through Logistic Regression, which results in skewing of our analysis.

```
DOS_Cross <- DOS[sample(nrow(DOS)), ]#creating a copy

k <- 10
nmethod <- 1
folds <- cut(seq(1,nrow(DOS_Cross)),breaks=k,labels=FALSE)
model.err <- matrix(-1,k,nmethod,dimnames=list(paste0("Fold", 1:k),
                                               c("LogitReg")))
```

In order to get the mean error rate, we put our models through a loop as follows:

```
for(i in 1:k)
{
    testindexes <- which(folds==i, arr.ind=TRUE)
    test_Cross <- DOS_Cross[testindexes, ]
    train_Cross <- DOS_Cross[-testindexes, ]


    LogitModel<- glm(target~., data = train_Cross[, !colnames(train_Cross) %in%
                 c("CustomerCode", "CountryName")], family = "binomial")

    predicted <- predict(LogitModel, newdata = test_Cross, type = "response")
    pred_class <- as.factor(ifelse(predicted >= 0.5,"YES", "NO"))
    model.err[i] <- mean(test_Cross$target != pred_class)
}

mean(model.err)
```

```
## [1] 0.1391753
```

```
model.err
```

```
##         LogitReg
## Fold1  0.1357388
## Fold2  0.1323024
## Fold3  0.1649485
## Fold4  0.1357388
## Fold5  0.1529210
## Fold6  0.1254296
## Fold7  0.1305842
## Fold8  0.1580756
## Fold9  0.1237113
## Fold10 0.1323024
```

Here we can say safely that according to cross-validation, the mean error rate for all our models is **0.1396907**, which translates to **13.96%**.

Evaluation of Best Model

First we simply created four models to check their accuracy

```
set.seed(123)
trainIndex_best <- createDataPartition(DOS$target, p = 0.8, list = FALSE)
training_best <- DOS[trainIndex_best, ]
testing_best <- DOS[-trainIndex_best, ]

#DecisionTree
fit.tree <- rpart(target ~ ., data=training_best, method="class")
pred.tree <- predict(fit.tree, testing_best, type="class")

#NaiveBayes
fit.nb <- naiveBayes(target ~ ., data = training_best)
pred.nb <- predict(fit.nb, newdata = testing_best, type = 'class')

#RandomForest
fit.rf <- randomForest(target ~. -`Hand Tufted` -`Double Back` -`Hand Woven`,
                       data=training_best, ntree=100)

pred.rf <- predict(fit.rf, testing_best)

#Logistic Regression
fit.logit <- glm(target ~ ., data = training_best, family = binomial)
pred.logit<- predict(fit.logit, newdata = testing_best, type = "response")

#NeuralNetwork
fit.nn <- nnet(target ~ ., data=training_best, size=5)
```

```
## # weights:  401
## initial  value 2800.946811
## iter  10 value 1923.909904
## iter  20 value 1576.881866
## iter  30 value 1315.171910
## iter  40 value 1137.556220
## iter  50 value 1094.168433
## iter  60 value 1075.895070
## iter  70 value 1048.251568
## iter  80 value 1026.579438
## iter  90 value 1005.514757
## iter 100 value 991.934092
## final  value 991.934092
## stopped after 100 iterations
```

```
pred.nn <- predict(fit.nn, testing_best, type="class")
```

Summary

Calculating the accuracy values for each model and their percentages:

```
acc.tree <- sum(pred.tree == testing_best$target)/nrow(testing_best)
acc.nb <- sum(pred.nb == testing_best$target)/nrow(testing_best)
acc.rf <- sum(pred.rf == testing_best$target)/nrow(testing_best)
acc.logit <- sum(pred.logit == testing_best$target)/nrow(testing_best)
acc.nn <- sum(pred.nn == testing_best$target)/nrow(testing_best)

acc.tree_pct <-
paste(round(acc.tree*100, 2), "%", sep = "")

acc.nb_pct <-
paste(round(acc.nb*100, 2), "%", sep = "")

acc.rf_pct <-
paste(round(acc.rf*100, 2), "%", sep = "")

acc.logit_pct <-
paste(round(acc.logit*100, 2), "%", sep = "")

acc.nn_pct <-
paste(round(acc.nn*100, 2), "%", sep = "")
```

Displaying the above calculated values using "kable" function in R:

```
Accu_Table <- data.frame(ModelName = c("Decision Tree Accuracy",
"Naive Bayes Accuracy", "Random Forest Accuracy",
"Logistic Regression Accuracy", "Neural Network Accuracy"),
ModelValue = c(acc.tree_pct, acc.nb_pct, acc.rf_pct, acc.logit_pct, acc.nn_pct))

kable(Accu_Table)
```

ModelName	ModelValue
Decision Tree Accuracy	90.97%
Naive Bayes Accuracy	81.6%
Random Forest Accuracy	90.89%
Logistic Regression Accuracy	0%
Neural Network Accuracy	91.14%

- The accuracy of the **Neural Network model is 91.14%**, which means that the model correctly predicted 91.14% of the target's results on the test set, making it the best model for us.
- Second and third best models are **Decision Tree** and **Random Forest**.
- The worst performing models for our data set are **Naive Bayes** and **Logistic Regression**.

Part D

Q. Discuss the data strategy for building customer segmentation using clustering. What are the benefits Champo Carpets can expect from clustering? Hint: Data strategy should clearly identify the data that should be used and how it should be used, including any feature engineering that may be performed before the model building.

A. The data strategy for establishing customer segmentation using clustering is to categorize consumers according to their behavior using a number of clustering algorithms. As a result, Champo Carpet will be able to recognize different customer segments and focus its marketing efforts on them. The benefits of this method include the ability to monitor the success of marketing initiatives and more accurately target clients with special offers. This way, Champo Carpets would also be able to increase their marketing efforts and ROI, not to mention greater lifetime value and client retention. For this purpose, they can divide their customer base into groups utilizing clustering techniques like K-means and hierarchical clustering.

Based on the description, the main issue for Champo carpet is the low sample conversion rate. In 2019, Champo carpet's sample conversion rate was **20%**, which is 15 percentage points below the conversion rate of the industry as a whole. For carpet the carpet manufacturing cost can be divided into three parts:

- raw material and its dyeing cost **(30%)**
- weaving cost **(60%)**
- finishing cost **(10%)**

Therefore, reducing the weaving cost can quickly reduce the manufacturing cost of carpet. The clustering idea for Champo carpet can focus on the clustering of weaving methods. By understanding customer needs and preferences, Champo Carpets enhances the customer experience, and by providing a personalized experience for customers, Champo Carpets can increase overall customer satisfaction and build brand loyalty. Finally customer segmentation helps Champo Carpets to identify high value customers and create personalized up-sell and cross-sell opportunities, thereby increasing revenue.

Additionally, Champo Carpet can create tailored marketing efforts with a higher chance of success by comprehending the behavior of various client segments. In order to better enhance their marketing efforts, Champo Carpet may also track the outcomes of these campaigns.

Part E

Q. Discuss clustering algorithms that can be used for segmenting Champo Carpets's customers. Please justify your choices. Discuss what distance and similarity measures are suitable in this case (Again, this is a conceptual question where you need to discuss which clustering method seems proper for this application and why). First, k-means and hierarchical clustering algorithms are both common clustering algorithms, but the difference between them is the clustering method. **k-means algorithm** requires a pre-specified number of clusters and assigns the data points to the nearest cluster at the center of each cluster. Hierarchical clustering, on the other hand, gradually merges data points into larger and larger clusters based on the similarity or distance between them until all data points are merged into one cluster or a preset stopping condition is reached.

A. Here are our thoughts to the question:

1. For Champo carpet company's data, we think hierarchical clustering is more appropriate. First by looking at the table, each customer group may not have placed orders for certain weaving methods, so there are many cells in this data set with 0. The data is sparse because there are many 0 values in the data set. In k-means clustering, it is assumed that each data point belongs to a certain cluster, which may lead to inaccurate clustering results. On the other hand, **hierarchical clustering** can adapt to sparse data and gradually merge similar points into a cluster, thus producing more accurate clustering results.

2. Second, the number of clusters for Champo carpet is unknown. k-means requires specifying the number of clusters, but in this case, we do not know how many customer groups are appropriate. Hierarchical clustering can help us to determine the most appropriate number of clusters by generating different numbers of clusters with different linking methods and clustering distances.

3. When dealing with data with many zero values, the use of traditional distance measures such as **Euclidean distance** or **Manhattan distance** may lead to biased clustering results. Therefore, some similarity measures specifically for dealing with data with a large number of zero values have been proposed.

4. One of the commonly used methods is the **cosine similarity measure**, which measures the cosine of the angle between vectors rather than the length of the vectors. In the cosine similarity measure, the similarity is 1 if the angle between two vectors is 0 degrees, and 0 if the angle is 90 degrees. cosine similarity measure is usually used to deal with textual data, but it is also applicable to sparse data, such as the carpet order data in this example. Since the data has many cells with zero data, the cosine similarity measure seems to be a reasonable choice because it can ignore the effect of zero values. In addition, if a hierarchical clustering method is used, one can choose to use the single-join clustering method, which will pick the distance between the two closest samples in two clusters as the distance between them, which is also applicable to the case of sparse data.

Part F

Q. Develop customer segmentation using k-means clustering. Discuss the optimal number of clusters., significant variables, and cluster characteristics. Notice that when the scree plot does not provide a clear choice of k for the number of clusters, you can look at other measures that we have discussed, such as the Silhouette measure. In many clustering applications, you need to consider more than one measure to obtain the number of desirable clusters.

A. For this question, we will need the DataForClustering data. First, we copy the data set into something more easy to work with:

```
cd <- read_excel("C:/Champo Carpets.xlsx",sheet = "Data for Clustering",
                 col_names = TRUE)

#We need to set the first column to the row name

cd1 <- cd[, -1]
row.names(cd1) <- cd$`Row Labels`

good <- cd1

#We also check their structure and their data types

str(good)
```

```
## tibble [45 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Sum of QtyRequired: num [1:45] 2466 131 18923 624 464 ...
##  $ Sum of TotalArea  : num [1:45] 140 2086 53626 203 8452 ...
##  $ Sum of Amount     : num [1:45] 185404 6247 1592080 14811 58627 ...
##  $ DURRY             : num [1:45] 1021 0 3585 581 0 ...
##  $ HANDLOOM          : num [1:45] 1445 0 0 0 0 ...
##  $ DOUBLE BACK       : num [1:45] 0 25 175 0 459 0 0 0 0 3 ...
##  $ JACQUARD          : num [1:45] 0 106 714 2 5 0 0 0 0 0 ...
##  $ HAND TUFTED       : num [1:45] 0 0 11716 0 0 ...
##  $ HAND WOVEN        : num [1:45] 0 0 2116 41 0 ...
##  $ KNOTTED           : num [1:45] 0 0 617 0 0 0 453 0 0 0 ...
##  $ GUN TUFTED        : num [1:45] 0 0 0 0 0 0 0 0 0 19 ...
##  $ Powerloom Jacquard: num [1:45] 0 0 0 0 0 0 0 0 0 0 ...
##  $ INDO TEBETAN      : num [1:45] 0 0 0 0 0 0 0 0 0 0 ...
```

For k-means there cannot be any missing values or NA, so we need to check NA and missing value first. Upon checking, we can confirm that this data set does not have any missing / NA values:

```
sum(is.na(good))
```

```
## [1] 0
```

```
anyNA(good)
```

```
## [1] FALSE
```

```
any(is.null(good))
```

```
## [1] FALSE
```

Now we need to normalize the data, which we do using the min-max transformation:

```
library(dplyr)

myscale <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

c_data <- good %>% mutate_if(is.numeric, myscale)
```

Next up, we can begin using k-means to create our model. To create graphs of the clusters generated with the kmeans() function, will use the "factoextra" package. We run the kmeans() function on Champo data for clustering dataset to group instances into two clusters (centers = 2)

```
library(ggplot2)
library(factoextra)

km1 <- kmeans(c_data, centers = 2, nstart = 200)
str(km1)
```

```
## List of 9
##  $ cluster     : int [1:45] 2 2 2 2 2 2 2 2 2 2 ...
##  $ centers     : num [1:2, 1:13] 0.4516 0.0436 0.491 0.0316 0.2595 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:13] "Sum of QtyRequired" "Sum of TotalArea" "Sum of Amount" "DURRY" ...
##  $ totss       : num 17.7
##  $ withinss    : num [1:2] 4.76 8.24
##  $ tot.withinss: num 13
##  $ betweenss   : num 4.7
##  $ size        : int [1:2] 3 42
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
km1
```

```
## K-means clustering with 2 clusters of sizes 3, 42
##
## Cluster means:
##   Sum of QtyRequired Sum of TotalArea Sum of Amount      DURRY   HANDLOOM
## 1         0.45163133       0.49095247    0.25954037 0.39638394 0.44432344
## 2         0.04362532       0.03162803    0.04739456 0.02619529 0.02238342
##   DOUBLE BACK  JACQUARD HAND TUFTED HAND WOVEN     KNOTTED GUN TUFTED
## 1  0.62082491 0.4495798  0.17390898 0.26605654 0.460534624 0.33333333
## 2  0.03600977 0.1020742  0.05203569 0.04594403 0.008346614 0.02087912
##   Powerloom Jacquard INDO TEBETAN
```

```
## 1          0.3333333   0.00000000
## 2          0.0000000   0.03809524
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 4.760036 8.237222
##  (between_SS / total_SS =  26.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Based on the above data, we can deduce the following:

1. **K-means clustering with 2 clusters of sizes 3, 42** indicates that the data set is divided into 2 categories using K-means algorithm, one category has 3 data points and the other category has 42 data points.
2. **Cluster means** shows the coordinates of the centroids of each category, for each column of data there is a coordinate, there are 13 columns of data.
3. **Clustering vector** shows which category each data point belongs to; there are 45 data points in total.
4. **Within cluster sum of squares by cluster** indicates the sum of squared errors of data points within each category.
5. **Available components** lists the available information contained in this result.

```r
km1$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2
```

```r
km1$centers
```

```
##   Sum of QtyRequired Sum of TotalArea Sum of Amount      DURRY   HANDLOOM
## 1         0.45163133       0.49095247    0.25954037 0.39638394 0.44432344
## 2         0.04362532       0.03162803    0.04739456 0.02619529 0.02238342
##   DOUBLE BACK  JACQUARD HAND TUFTED HAND WOVEN     KNOTTED GUN TUFTED
## 1  0.62082491 0.4495798  0.17390898 0.26605654 0.460534624 0.33333333
## 2  0.03600977 0.1020742  0.05203569 0.04594403 0.008346614 0.02087912
##   Powerloom Jacquard INDO TEBETAN
## 1          0.3333333   0.00000000
## 2          0.0000000   0.03809524
```

```r
km1$withinss
```

```
## [1] 4.760036 8.237222
```

```r
km1$betweenss
```

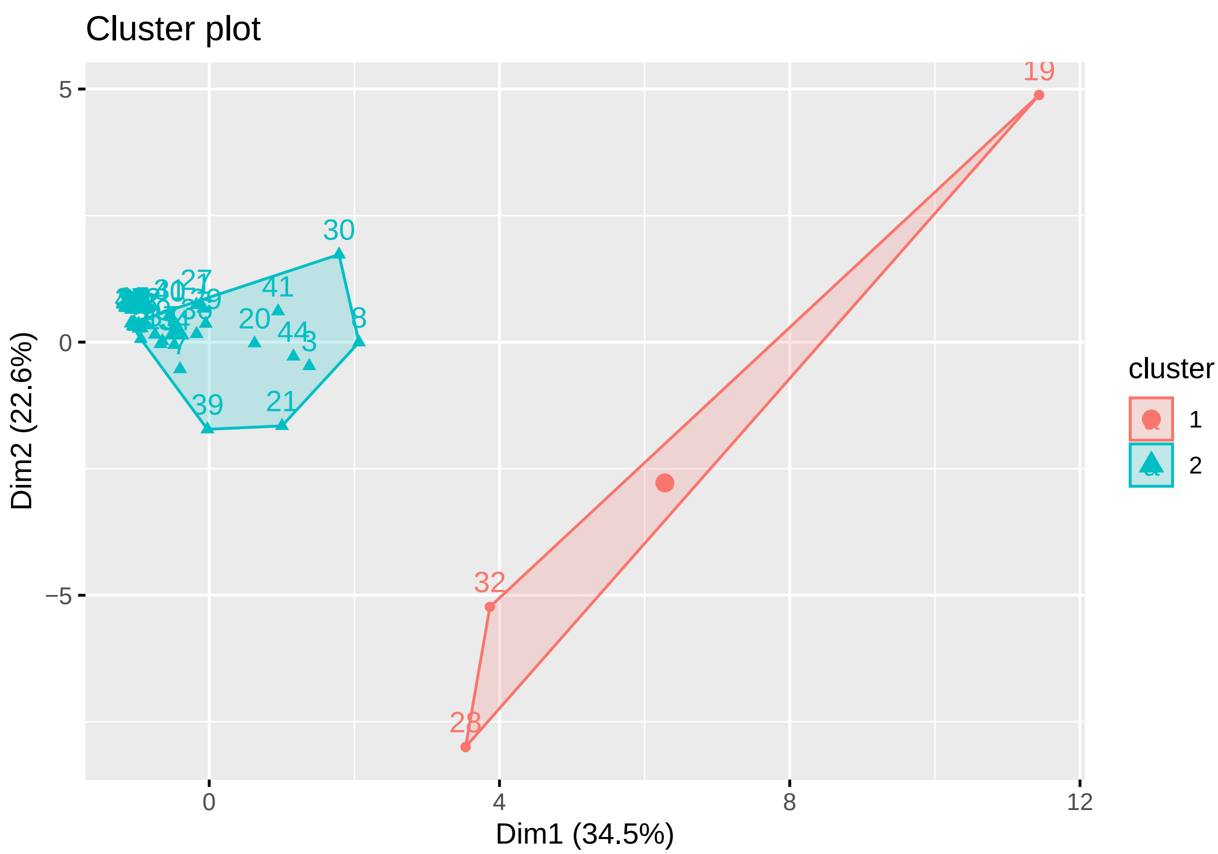
```
## [1] 4.699134
```

```r
km1$size
```

```
## [1]  3 42
```

Next we use the fviz_cluster function to illusterate the clusters. Since our data set has more than two variables, the fviz_cluster function performs principal component analysis (PCA) first and plot the data points according to the first two principal components that explain the majority of the variance:

```
fviz_cluster(km1, data = c_data)
```

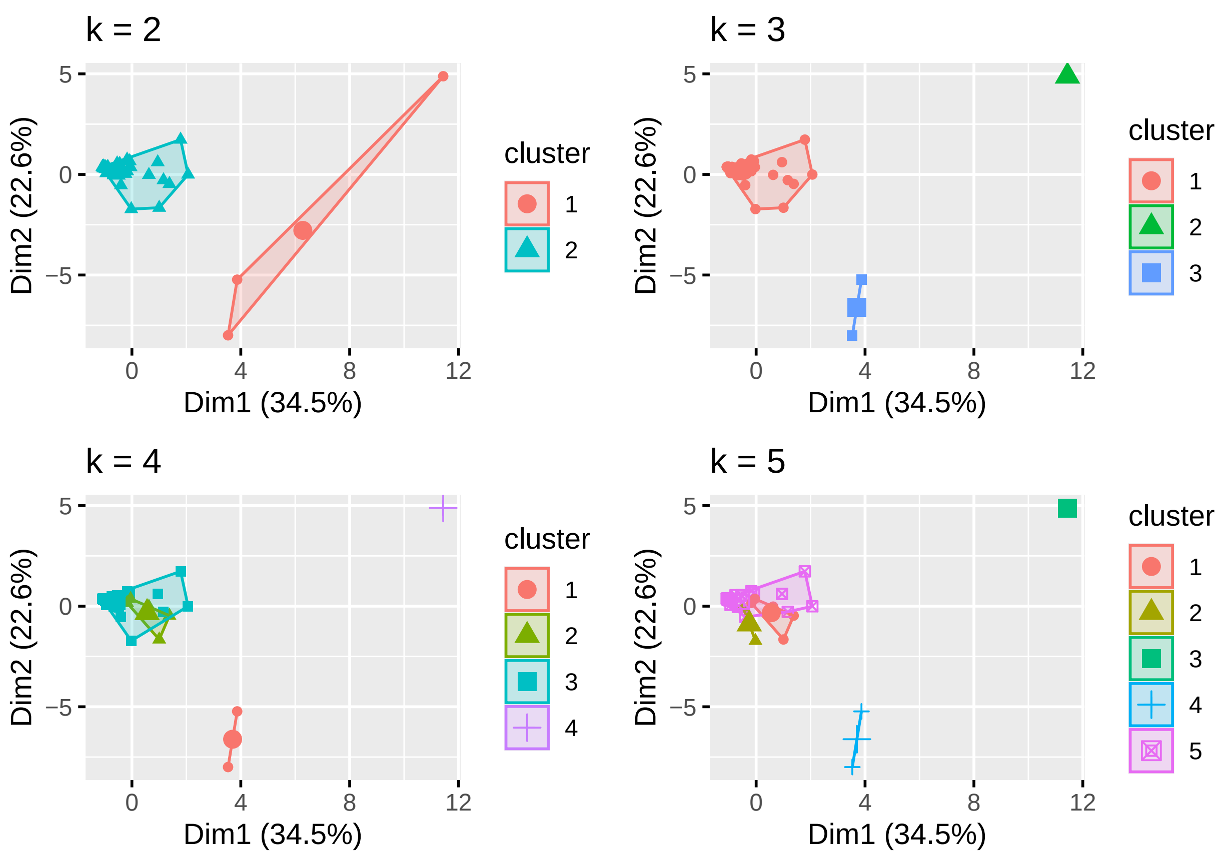


Now, we will execute the k-means algorithm for **3, 4, and 5 clusters** and see how clusters change:

```
km2 <- kmeans(c_data, centers = 3, nstart = 200)
km3 <- kmeans(c_data, centers = 4, nstart = 200)
km4 <- kmeans(c_data, centers = 5, nstart = 200)
```

Plots to compare:

```
library(gridExtra)

p1 <- fviz_cluster(km1, geom = "point", data = c_data) + ggtitle("k = 2")
p2 <- fviz_cluster(km2, geom = "point", data = c_data) + ggtitle("k = 3")
p3 <- fviz_cluster(km3, geom = "point", data = c_data) + ggtitle("k = 4")
p4 <- fviz_cluster(km4, geom = "point", data = c_data) + ggtitle("k = 5")

grid.arrange(p1, p2, p3, p4, nrow = 2)
```

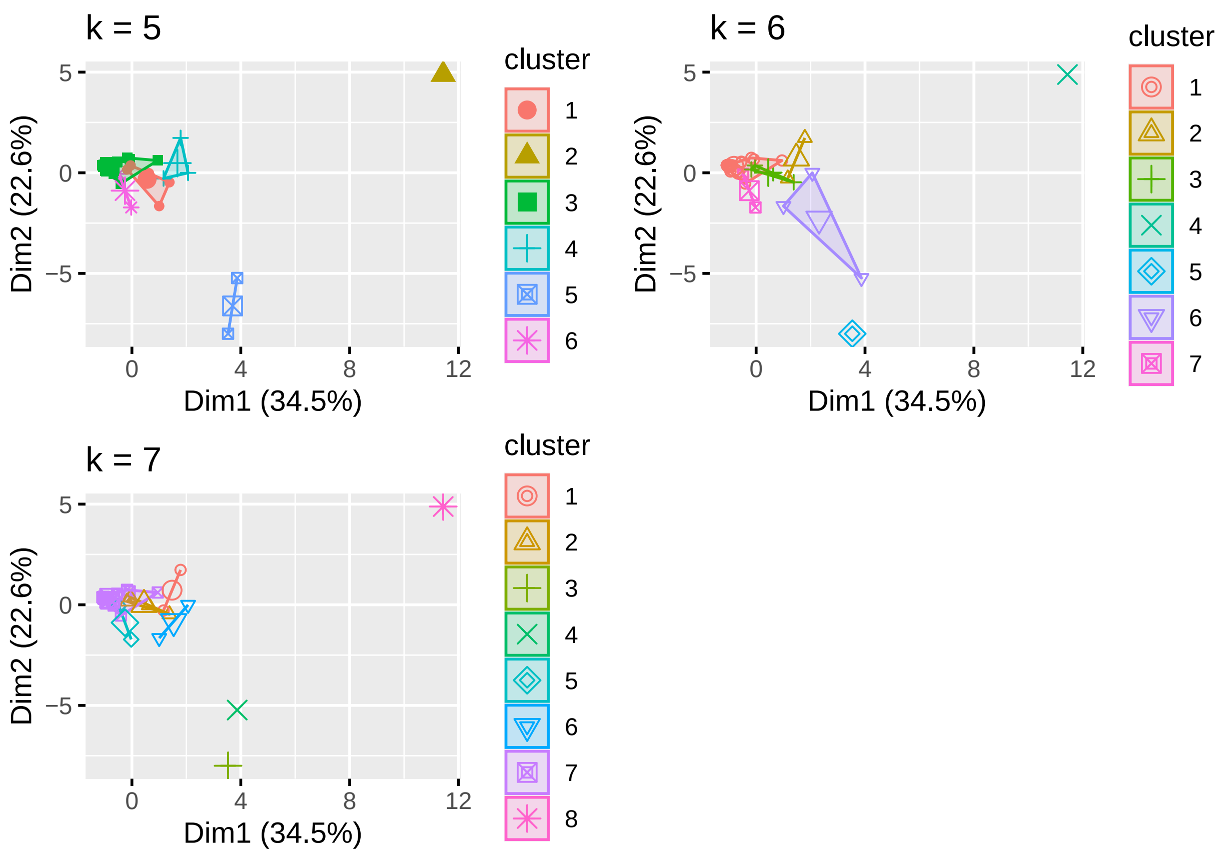


Post that, we will execute the k-means algorithm for **6, 7, and 8 clusters** and see how clusters change:

```
km5 <- kmeans(c_data, centers = 6, nstart = 200)
km6 <- kmeans(c_data, centers = 7, nstart = 200)
km7 <- kmeans(c_data, centers = 8, nstart = 200)
```

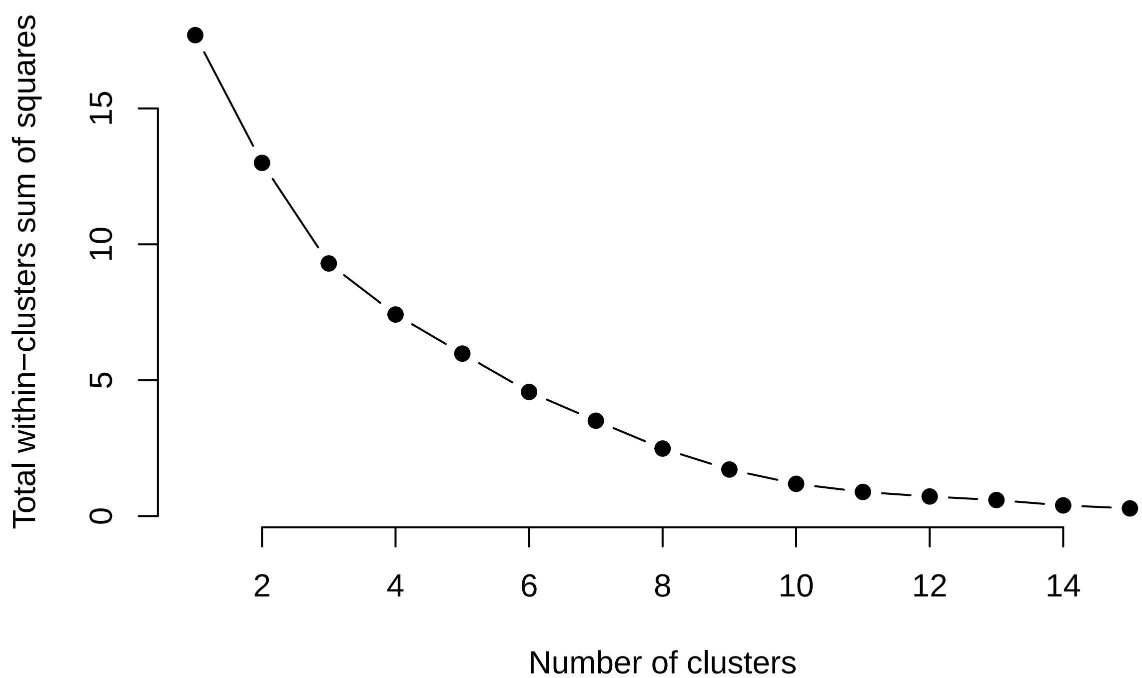
Plots to compare:

```
library(gridExtra)
p5 <- fviz_cluster(km5, geom = "point", data = c_data) + ggtitle("k = 5")
p6 <- fviz_cluster(km6, geom = "point", data = c_data) + ggtitle("k = 6")
p7 <- fviz_cluster(km7, geom = "point", data = c_data) + ggtitle("k = 7")

grid.arrange(p5, p6, p7, nrow = 2)
```



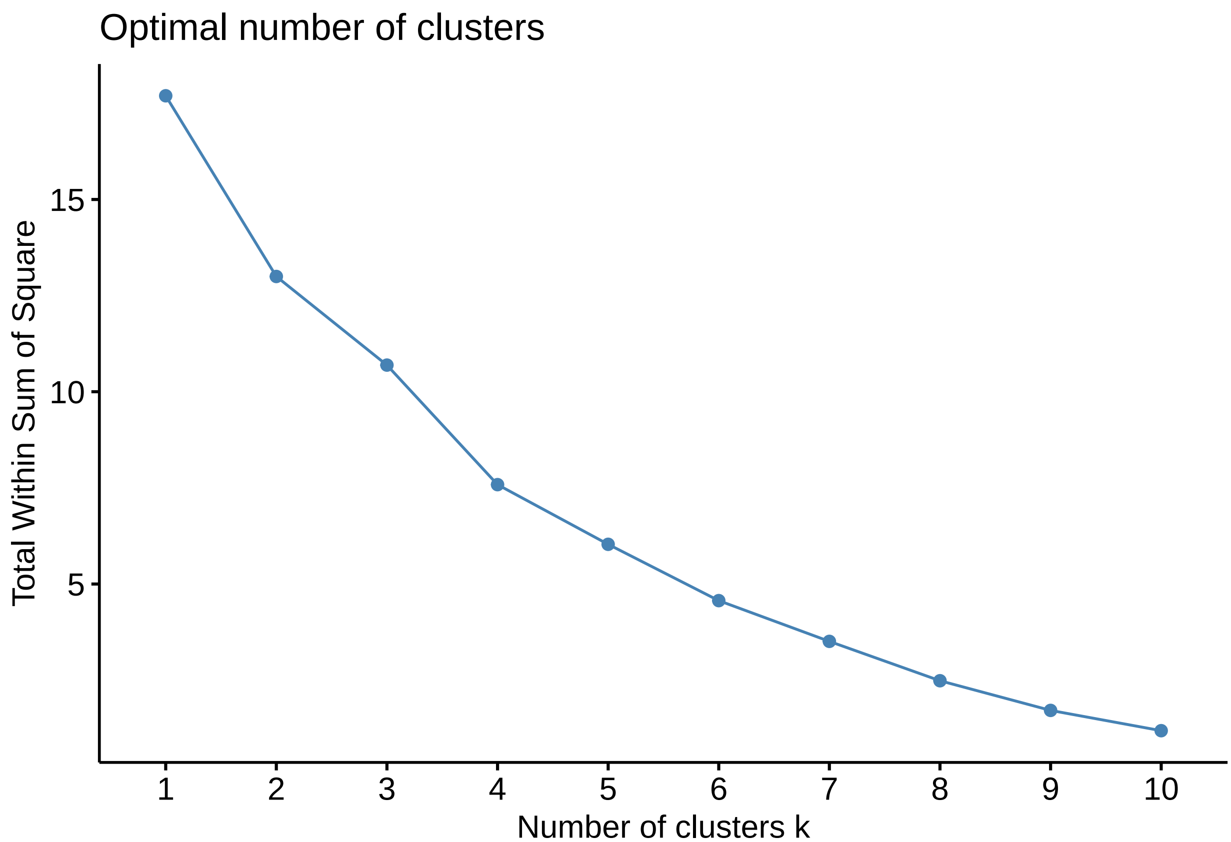
Solution 1: To determine the numbers of clusters we can use the elbow method. The following code draws the scree plot:

```
library(tidyverse)

set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(c_data, centers = k, nstart = 100)$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters

wss_values <- map_dbl(k.values, wss)
plot(k.values, wss_values,
type="b", pch = 19, frame = FALSE,
xlab="Number of clusters",
ylab="Total within-clusters sum of squares")
```



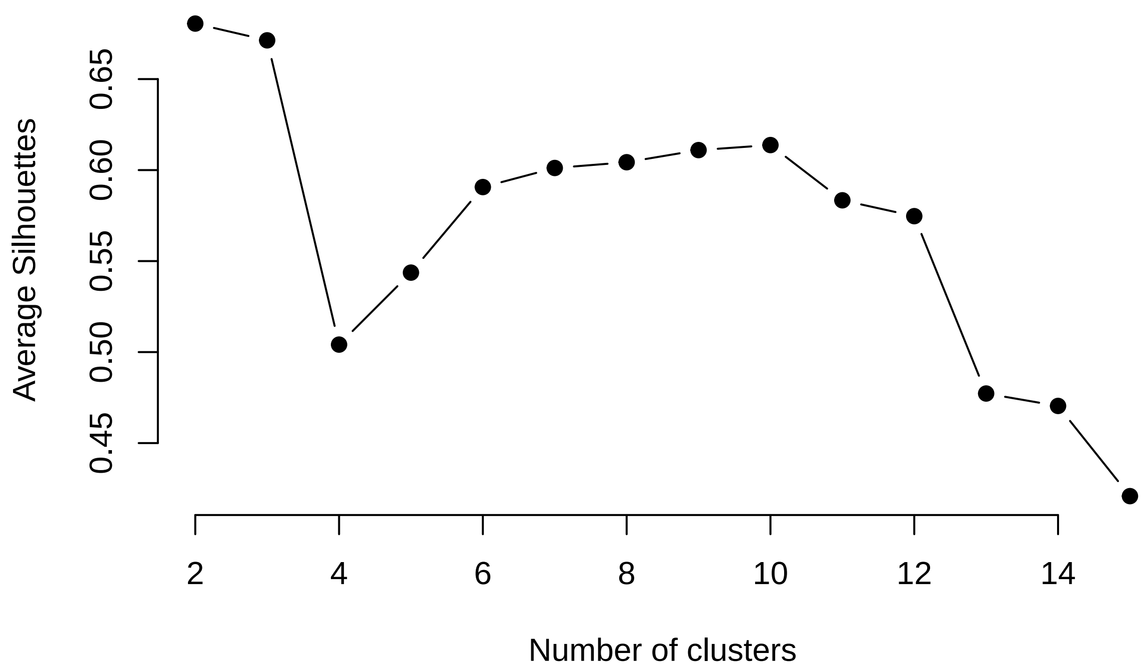
To get the scree plot, we can also use the “fviz_nbclust” function:

```
set.seed(123)
fviz_nbclust(c_data, kmeans, method = "wss")
```



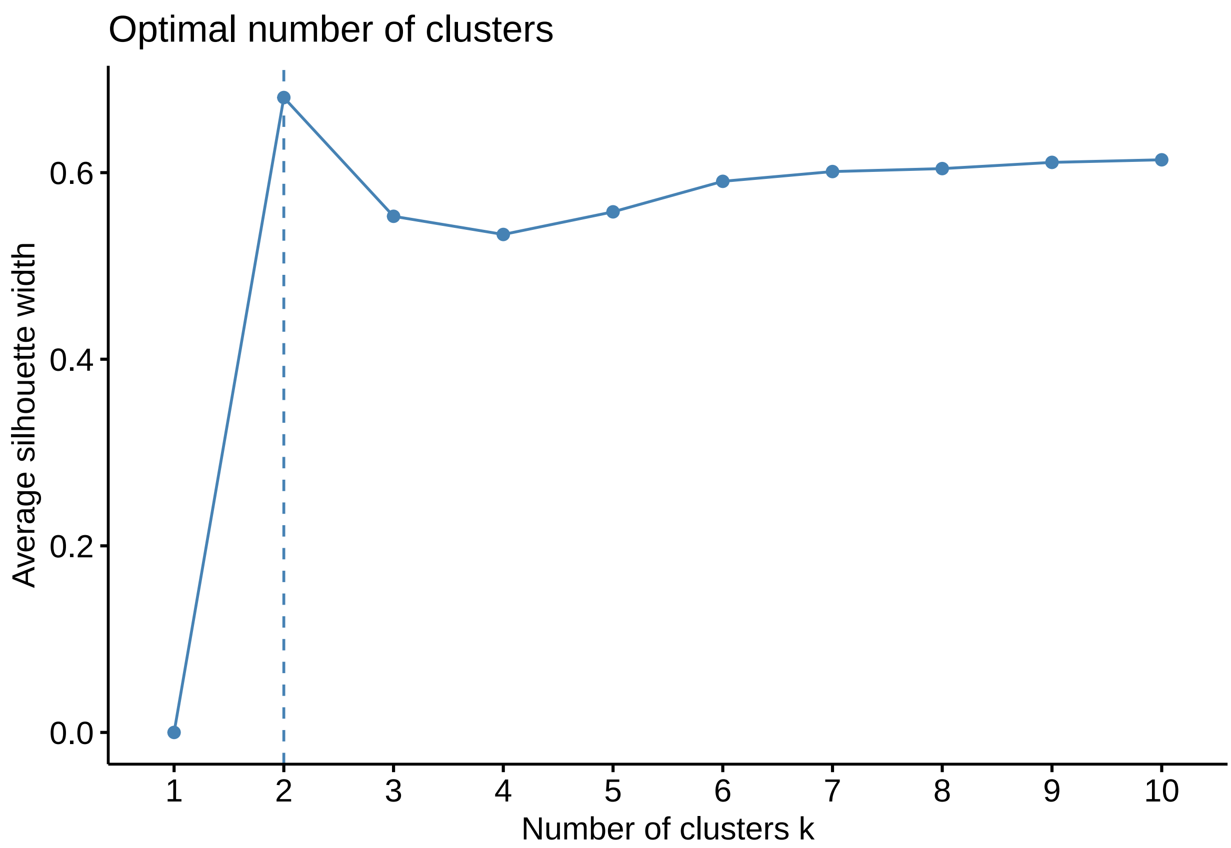
Solution 2: Another measure we learned that can be used to check the performance of clusters is the Silhouette measure. We can use the "silhouette" function from the "cluster" package to compute the average silhouette:

```
# function to compute average silhouette for k clusters
library(cluster)

avgsil <- function(k) {
  kmModel <- kmeans(c_data, centers = k, nstart = 100)
  ss <- silhouette(kmModel$cluster, dist(c_data))
  mean(ss[, 3])
}

# Compute and plot wss for k = 2 to k = 15
k.values <- 2:15

# extract avg silhouette for 2-15 clusters
avgsil_values <- map_dbl(k.values, avgsil)
plot(k.values, avgsil_values,
type = "b", pch = 19, frame = FALSE,
xlab = "Number of clusters",
ylab = "Average Silhouettes")
```



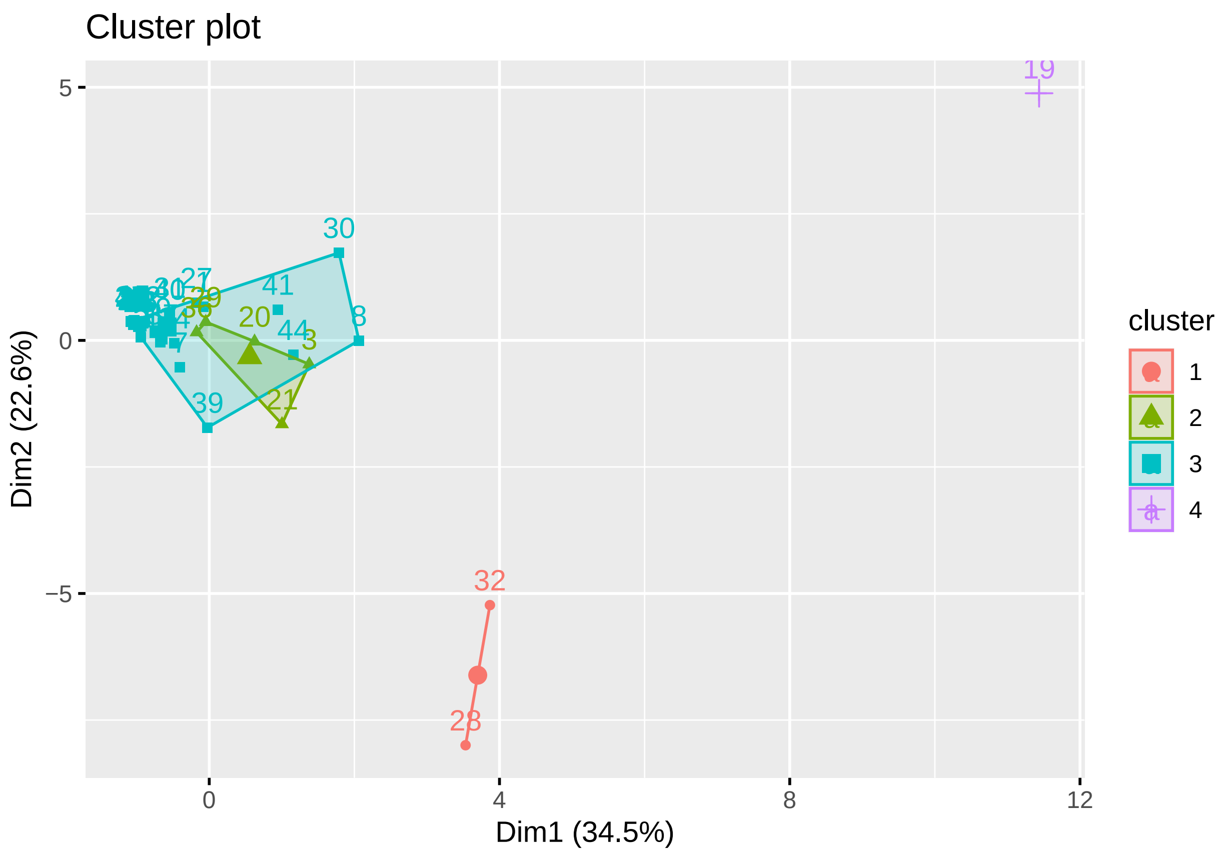
Similar to the elbow method, the "average silhoutte method" can be found in fviz_nbclust function:

```
fviz_nbclust(c_data, kmeans, method = "silhouette")
```



Suppose km3 is our final model:

```
fviz_cluster(km3, data = c_data)
```



During data processing we found that when we converted the first column to a row name and named it good, the first column automatically became a number when we performed the next operation. So we need to check the good and c_data against each other. For example, 1 in the figure is A-11 and 2 is A-6 etc.

By looking at the above diagram we find that **28 and 32** are divided into one category, that is, M-1 and P-5 have high similarity. m-1 has purchased **Gun Tufted** and P-5 has not, so we can recommend **Gun Tufted** carpet to customer with code **P-5**.

Part G

Q. Write your own collaborative filtering function as a recommender system. Hint: Collaborative filtering technique is based on an aggregation of customer purchase history. For each customer, you can use various measures such as Pearson correlation, Euclidean distance, or cosine similarity to find the nearest neighbors. You can then use the nearest neighbors to recommend products. For example, suppose using cosine similarity, you find out that the closest customer to customer H-2 is customer T-5. Customer T-5 has purchased carpet type double black and gray color, which are not purchased by H-2. Hence these products can be recommended.

A. To calculate the nearest neighbors, we used the cosine similarity as follows:

```
# define a function to calculate cosine similarity between two rows
cosine_sim <- function(x, y) {
     (sum(x * y)) / (sqrt(sum(x^2)) * sqrt(sum(y^2)))
 }
```

```
# create an empty matrix to store cosine similarities
similarity_matrix <- matrix(NA, nrow = nrow(good), ncol = nrow(good))
```

```
# calculate cosine similarity between all pairs of rows
for (i in 1:nrow(good)) {
     for (j in 1:nrow(good)) {
         similarity_matrix[i, j] <-
        cosine_sim(good[i,  c(4,5,6,7,8,9,10,11,12,13)],
                   good[j, c(4,5,6,7,8,9,10,11,12,13)])
     }
 }
```

```
# find the row j with the largest similarity with row i
max_similarities <- apply(similarity_matrix, 1, max)
most_similar_row_indices <- apply(similarity_matrix, 1, which.max)
```

```
# Create an empty vector to store the largest value in each row except for 1
max_values <- vector(mode = "numeric", length = nrow(similarity_matrix))
```

```
# Create an empty vector to store the column number corresponding to
#the maximum value of each row
max_cols <- vector(mode = "numeric", length = nrow(similarity_matrix))
```

```
for (i in 1:nrow(similarity_matrix)) {

  max_values[i] <- max(similarity_matrix[i, similarity_matrix[i,] != 1])

  max_cols[i] <- which(similarity_matrix[i,] == max_values[i])
}
```

```r
# Output the largest value in each row except 1 and the
#corresponding row and column

result <- data.frame(Maximum_Value = max_values,
              Row_Number = 1:nrow(similarity_matrix), Column_Number = max_cols)
print(result)
```

```
##    Maximum_Value Row_Number Column_Number
## 1      1.0000000          1             1
## 2      0.8731991          2            16
## 3      0.9834904          3            29
## 4      0.9977566          4            31
## 5      0.9999407          5            23
## 6      0.9801335          6            30
## 7      1.0000000          7             7
## 8      0.9557607          8            41
## 9      1.0000000          9             9
## 10     1.0000000         10            10
## 11     0.9999745         11            10
## 12     0.7087369         12            45
## 13     0.9118937         13             5
## 14     0.9902291         14            29
## 15     0.9999374         15            31
## 16     0.8731991         16             2
## 17     0.9999374         17            31
## 18     1.0000000         18            18
## 19     0.9953973         19            41
## 20     1.0000000         20            20
## 21     0.7903961         21             8
## 22     1.0000000         22            22
## 23     0.9999407         23             5
## 24     0.8902003         24            39
## 25     0.9999745         25            10
## 26     0.9999374         26            31
## 27     0.9999374         27            31
## 28     0.8040762         28            37
## 29     1.0000000         29            29
## 30     1.0000000         30            30
## 31     1.0000000         31            31
## 32     0.9570103         32            22
## 33     0.9999745         33            10
## 34     0.9981121         34            31
## 35     0.9999374         35            31
## 36     0.9196745         36             3
## 37     0.9030477         37            13
## 38     0.8623167         38            20
## 39     1.0000000         39            39
## 40     1.0000000         40            40
## 41     0.9953973         41            19
## 42     0.9990589         42            44
## 43     1.0000000         43            43
## 44     0.9999745         44            10
## 45     1.0000000         45            45
```

Based on this result. row 2 and row 16 are similar. There is a high degree of similarity between customers with code N-6 and G-1. G-1 purchased Handwoven carpets, so they can recommend Handwoven to N-6 as well.

For this question, we load the Raw Data-Order and Sample tab again:

```
library(arules)
raw <- read_excel("C:/Champo Carpets.xlsx",sheet = "Raw Data-Order and Sample",
                  col_names = TRUE)

as(raw, "transactions")
```

```
## transactions in sparse format with
##  18955 transactions (rows) and
##  9186 items (columns)
```

As we noticed above, columns that are not logical or factor are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16, so we remove those columns from our data set.

```
raw2<-raw[,-c(1,2,4,5,6,7,8,9,10,12,13,16)]
```

To see the most frequent items we can use the eclat() function that takes in a transactions object and gives the most frequent items in the data based the support you provide as min_support in "supp" argument. The "maxlen" defines the maximum number of items in each itemset of frequent items:

```
frequency <- eclat(raw2, parameter = list(supp=0.07, maxlen=15))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.07      1     15 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 1326
##
## create itemset ...
## set transactions ...[878 item(s), 18955 transaction(s)] done [0.01s].
## sorting and recoding items ... [11 item(s)] done [0.00s].
## creating bit matrix ... [11 row(s), 18955 column(s)] done [0.00s].
## writing  ... [22 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
inspect(frequency)
```

```
##      items                       support count
## [1]  {CustomerCode=A-9,
##       ShapeName=REC}          0.07338433  1391
## [2]  {ITEM_NAME=KNOTTED,
##       ShapeName=REC}          0.08251121  1564
## [3]  {CustomerCode=P-5,
##       ShapeName=REC}          0.10023740  1900
## [4]  {ITEM_NAME=HANDWOVEN,
##       ShapeName=REC}          0.12065418  2287
## [5]  {ITEM_NAME=DOUBLE BACK,
##       ShapeName=REC}          0.12993933  2463
## [6]  {CustomerCode=M-1,
##       ShapeName=REC}          0.13131100  2489
## [7]  {CustomerCode=CC,
##       ITEM_NAME=HAND TUFTED,
##       ShapeName=REC}          0.08208916  1556
## [8]  {CustomerCode=CC,
##       ShapeName=REC}          0.21593247  4093
## [9]  {CustomerCode=CC,
##       ITEM_NAME=HAND TUFTED}  0.08272224  1568
## [10] {ITEM_NAME=DURRY,
##       ShapeName=REC}          0.22621999  4288
## [11] {ITEM_NAME=HAND TUFTED,
##       ShapeName=REC}          0.35900818  6805
## [12] {ShapeName=REC}          0.97694540 18518
## [13] {ITEM_NAME=HAND TUFTED}  0.37430757  7095
```

```
## [14] {ITEM_NAME=DURRY}         0.22975468  4355
## [15] {CustomerCode=CC}         0.21814825  4135
## [16] {CustomerCode=M-1}        0.13183857  2499
## [17] {ITEM_NAME=DOUBLE BACK}  0.13051965  2474
## [18] {ITEM_NAME=HANDWOVEN}    0.12292271  2330
## [19] {CustomerCode=P-5}        0.10182010  1930
## [20] {ITEM_NAME=KNOTTED}      0.08309153  1575
## [21] {CustomerCode=A-9}        0.07359536  1395
## [22] {ColorName=GREY}          0.07037721  1334
```

Next up, we use apriori() function to generate the rules. We can adjust the maxlen, supp, and conf arguments in the apriori function to control the number of rules generated. The "control" argument can be adjusted to control the algorithmic performance (for example, verbose which is a logical argument indicating whether to report progress of algorithm):

```
library(arulesViz)
# Produce recommendation rules
# Min Support as 0.001, confidence as 0.5
rules <- apriori (raw2, parameter = list(supp=0.001, conf= 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 18
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[878 item(s), 18955 transaction(s)] done [0.01s].
## sorting and recoding items ... [200 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [2029 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

The function inspect() prints the internal representation of an R object. Here, it displaying the first 10 strong association rules based on confidence:

```
# Sort rules by Confidence
rules_conf <- sort (rules, by="confidence", decreasing=TRUE)
# show the support, lift and confidence for all rules
inspect(rules_conf[1:10])
```

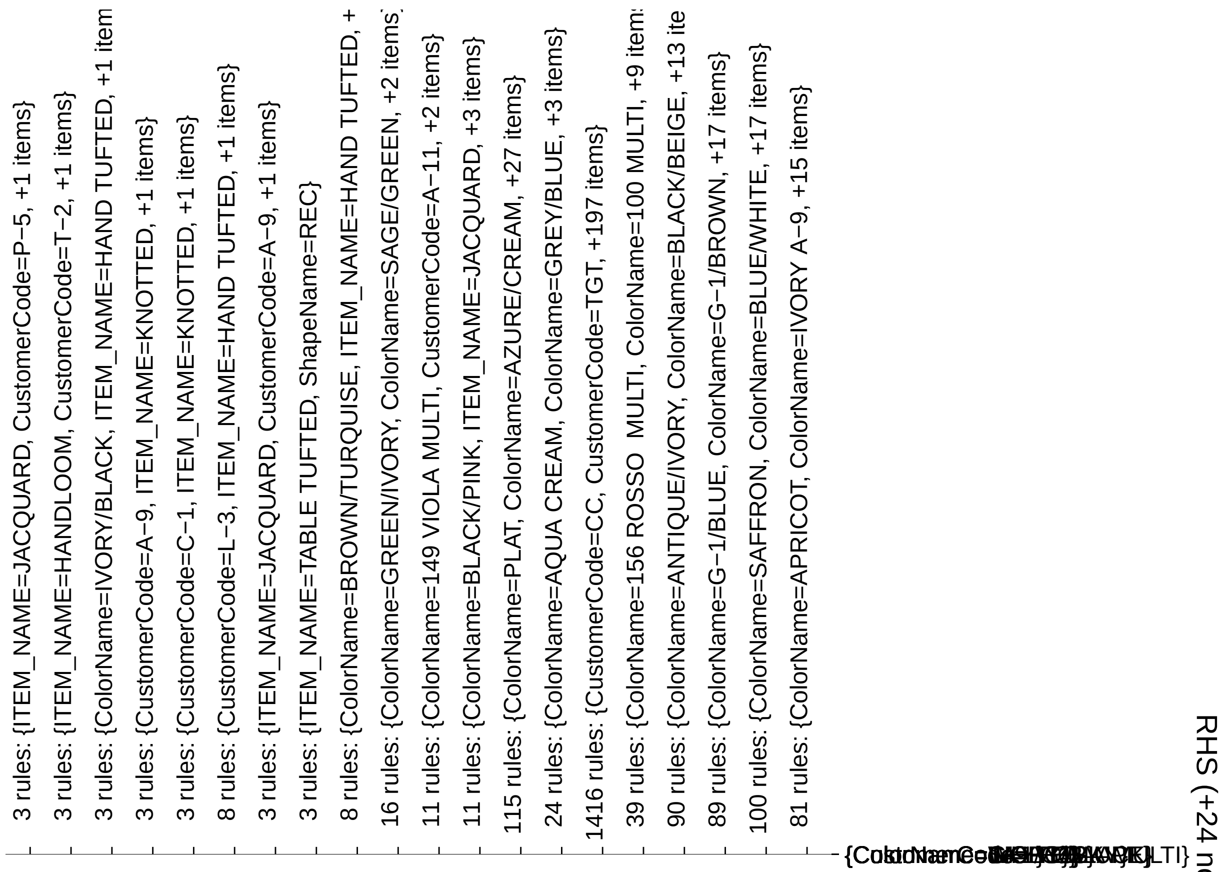
```
##      lhs                           rhs                     support
## [1]  {ColorName=PLUM}           => {ShapeName=REC}         0.001002374
## [2]  {ColorName=CHARCOAL/GREY} => {ShapeName=REC}         0.001002374
## [3]  {ColorName=RED/BROWN}     => {CustomerCode=C-1}      0.001002374
## [4]  {ColorName=RED/BROWN}     => {ITEM_NAME=HAND TUFTED} 0.001002374
## [5]  {ColorName=RED/BROWN}     => {ShapeName=REC}         0.001002374
## [6]  {ColorName=BLUE/18-19}    => {CustomerCode=M-1}      0.001002374
## [7]  {ColorName=BLUE/18-19}    => {ITEM_NAME=DURRY}       0.001002374
## [8]  {ColorName=BLUE/18-19}    => {ShapeName=REC}         0.001002374
## [9]  {ColorName=NAVY/BLUE}     => {ShapeName=REC}         0.001002374
## [10] {ColorName=IVORY-JL}      => {ITEM_NAME=HANDWOVEN}   0.001055131
##      confidence coverage    lift      count
## [1]  1          0.001002374  1.023599 19
## [2]  1          0.001002374  1.023599 19
## [3]  1          0.001002374 17.278943 19
## [4]  1          0.001002374  2.671600 19
## [5]  1          0.001002374  1.023599 19
## [6]  1          0.001002374  7.585034 19
## [7]  1          0.001002374  4.352468 19
## [8]  1          0.001002374  1.023599 19
## [9]  1          0.001002374  1.023599 19
## [10] 1          0.001055131  8.135193 20
```

We can now sort this based on the **lift** values as follows:

```r
rules_lift <- sort (rules, by="lift", decreasing=TRUE)
inspect(rules_lift[1:10])
```

```
##      lhs                            rhs                             support confidence    coverage
## [1]  {CustomerCode=P-5,
##       ITEM_NAME=JACQUARD}      => {ColorName=WHITE/SKY}       0.001318913  1.0000000 0.001318913 758.
## [2]  {CustomerCode=P-5,
##       ITEM_NAME=JACQUARD,
##       ShapeName=REC}           => {ColorName=WHITE/SKY}       0.001318913  1.0000000 0.001318913 758.
## [3]  {CustomerCode=T-2,
##       ITEM_NAME=HANDLOOM}      => {ColorName=149 VIOLA MULTI} 0.001266157  0.5333333 0.002374044 421.
## [4]  {CustomerCode=T-2,
##       ITEM_NAME=HANDLOOM,
##       ShapeName=REC}           => {ColorName=149 VIOLA MULTI} 0.001266157  0.5333333 0.002374044 421.
## [5]  {ITEM_NAME=HAND TUFTED,
##       ColorName=IVORY/BLACK}   => {CustomerCode=L-3}          0.001635452  0.5849057 0.002796096 291.
## [6]  {ITEM_NAME=HAND TUFTED,
##       ColorName=IVORY/BLACK,
##       ShapeName=REC}           => {CustomerCode=L-3}          0.001635452  0.5849057 0.002796096 291.
## [7]  {CustomerCode=A-9,
##       ITEM_NAME=KNOTTED}       => {ColorName=IVORY/TAUPE}     0.002426800  0.6571429 0.003692957 270.
## [8]  {CustomerCode=A-9,
##       ITEM_NAME=KNOTTED,
##       ShapeName=REC}           => {ColorName=IVORY/TAUPE}     0.002426800  0.6571429 0.003692957 270.
## [9]  {CustomerCode=C-1,
##       ITEM_NAME=KNOTTED}       => {ColorName=G-1/G-2}         0.003323661  0.7682927 0.004326035 220.
## [10] {CustomerCode=C-1,
##       ITEM_NAME=KNOTTED,
##       ShapeName=REC}           => {ColorName=G-1/G-2}         0.003270905  0.7654321 0.004273279 219.
```

```r
write(rules_lift, file = "clusterRules", sep = ",")
plot(rules_lift, method = "grouped")
```



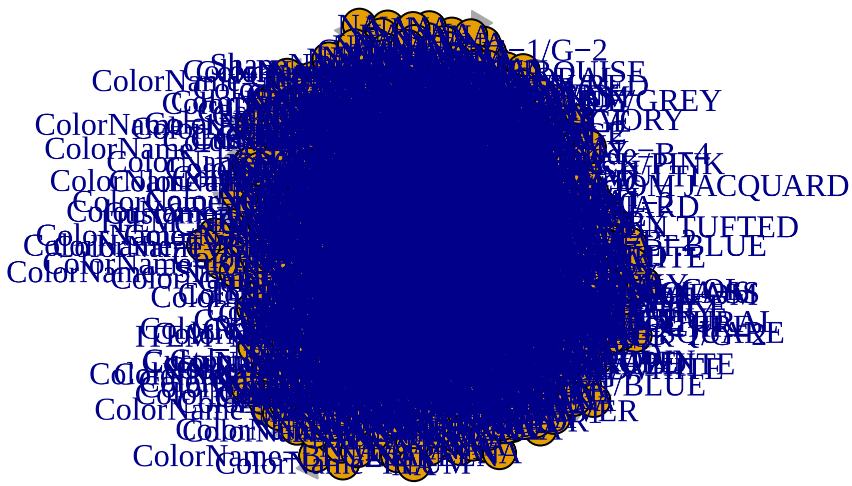

```r
plot(rules_lift, method = "graph", control = list(type = 'item'),
     interactive = F)
```

```
## Available control parameters (with default values):
## layout     =  stress
## circular   =  FALSE
## ggraphdots     =  NULL
## edges      =  <environment>
## nodes      =  <environment>
## nodetext   =  <environment>
## colors     =  c("#EE0000FF", "#EEEEEEFF")
## engine     =  ggplot2
## max    =  100
## verbose    =  FALSE
```


CustomerCode=A−11
ColorName=156 ROSSO MULTI
ColorName=COL−150 MULTI
CustomerCode=C−1
CustomerCode=T−2
ColorName=IVORY/TAUPE
ShapeName=ROUND
ITEM_NAME=KNOTTED
ColorName=159 ARANCIO MULTI
ColorName=100 MULTI
CustomerCode=A−9
ColorName=SKY
ShapeName=REC
ColorName=160 NERO MULTI
CustomerCode=P−5
ITEM_NAME=HAND TUFTED
ColorName=COL−156 MULTI
ColorName=WHITE/SKY
CustomerCode=TGT
CustomerCode=A−6
ITEM_NAME=JACQUARD
ColorName=STONE GREY
ColorName=ICE
ColorName=IVORY/BLACK
ColorName=BEIGE
CustomerCode=RC
ColorName=BLUSH PINK
lift
600
400
200
support
0.002
0.003
0.004
0.005
0.006

To visualize this, we utilize the arulesViz package and after that the iGraph package:

```
library(arulesViz)
library(igraph)
saveAsGraph(rules_lift, file = "rules.graphml")
g<-read_graph("rules.graphml",format ="graphml")

require(igraph)
plot(g,width=10,arrow.size=0.5)
```



We can also control the number of rules in the output:

```r
rules3 <- apriori(raw2, parameter = list(supp = 0.001, conf = 0.5, maxlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##       3  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 18
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[878 item(s), 18955 transaction(s)] done [0.01s].
## sorting and recoding items ... [200 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [1512 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(head(rules3))
```

```
##     lhs                             rhs                      support     confidence
## [1] {}                           => {ShapeName=REC}          0.976945397 0.9769454
## [2] {ColorName=PLUM}             => {ShapeName=REC}          0.001002374 1.0000000
## [3] {ColorName=CHARCOAL/GREY} => {ShapeName=REC}          0.001002374 1.0000000
## [4] {ColorName=RED/BROWN}        => {CustomerCode=C-1}       0.001002374 1.0000000
## [5] {ColorName=RED/BROWN}        => {ITEM_NAME=HAND TUFTED} 0.001002374 1.0000000
## [6] {ColorName=RED/BROWN}        => {ShapeName=REC}          0.001002374 1.0000000
##     coverage    lift      count
## [1] 1.000000000  1.000000 18518
## [2] 0.001002374  1.023599    19
## [3] 0.001002374  1.023599    19
## [4] 0.001002374 17.278943    19
## [5] 0.001002374  2.671600    19
## [6] 0.001002374  1.023599    19
```

We can use "appearance" argument in apriori() function to control the itemsets in antecedent and consequent parts of the decision rules:

```
# Find rules related to given item(s)
# Get rules that lead to buying 'ShapeName=REC'
rules <- apriori (data=raw2, parameter=list(supp=0.001, conf=0.08),
appearance = list(default="lhs",rhs="ShapeName=REC"), control = list(verbose=F))

# 'high-confidence' rules.
rules_conf <- sort (rules, by="confidence", decreasing=TRUE)
inspect(head(rules_conf))
```

```
##     lhs                            rhs             support     confidence
## [1] {ColorName=RED/BROWN}       => {ShapeName=REC} 0.001002374 1
## [2] {ColorName=BLUE/18-19}      => {ShapeName=REC} 0.001002374 1
## [3] {ColorName=PLUM}            => {ShapeName=REC} 0.001002374 1
## [4] {ColorName=NAVY/BLUE}       => {ShapeName=REC} 0.001002374 1
## [5] {ColorName=CHARCOAL/GREY}   => {ShapeName=REC} 0.001002374 1
## [6] {ColorName=CHARCOAL/IVORY} => {ShapeName=REC} 0.001055131 1
##     coverage    lift     count
## [1] 0.001002374 1.023599 19
## [2] 0.001002374 1.023599 19
## [3] 0.001002374 1.023599 19
## [4] 0.001002374 1.023599 19
## [5] 0.001002374 1.023599 19
## [6] 0.001055131 1.023599 20
```

Summary

The rules with confidence of 1 imply that, whenever the LHS item was purchased, the RHS item was also purchased 100% of the time. This means that according to our data, whenever **Plum** colored carpet was bought, it was bought in **Rec** shape. Same with the other items listed above.

Similarly, a rule with a lift of **17.27** means that items in LHS and RHS are 18 times more likely to be purchased together compared to the purchases when they are assumed to be unrelated. In our case, **RED/BROWN** carpet is **17.27** times more likely to be purchased by customer with customer code **C-1** when they are assumed to be unrelated.

Part H

Q. What will be your final recommendation to Champo Carpets?

A. Through the models we've discussed above, ML models or clustering models, we can come up with a strategy for Champo Carpets. Some highlights are as follows:

1. Our Decision Tree model suggests that the important attributes contributing to an order conversion are: **AreaFt**, **CustomerCode**, and **ITEM_NAME**.
2. Our Random Forest model suggests that the important attributes are: **QtyRequired**, **CustomerCode**, **ITEM_NAME**, and **AreaFt** according to MeanDecreaseAccuracy and MeanDecreaseGini.
3. Our Logistic Regression model suggests that the important attributes are: **ITEM_NAME**, especially **Gun Tufted**, **Knotted**, **Table Tufted**, and **Power Loom Jacquard**, followed by **Shape Name (Round)**, and **AreaFt**
4. According to all models combined, we can recommend that **QtyRequired**, **AreaFt**, **CustomerCode**, and **ITEM_NAME** are playing the most important role in conversion of orders.
5. Using K-Means clustering model, key knowledge about the segmentation of their customers can be obtained by Champo Carpets. This will help them form better strategies targeting different segments.
6. **Reducing the weaving cost** can quickly reduce the manufacturing cost of carpet. The clustering idea for Champo carpet can focus on the clustering of weaving methods
7. Champo Carpet can create tailored **marketing efforts** with a higher chance of success by comprehending the behavior of various client segments.
8. An effective recommendation system needs to be developed using high-quality user feedback data, as a recommendation model is only as good as the data it is trained on.
9. As per the Cosine Similarity calculation we did above:

- The **second row** has a high correlation with row **16**.
- The **third row** has high correlation with row **29**.
- The **fourth row** has a high correlation with row **31**.
- The **fifth row** has a high correlation with row **23**.
- The **Sixth row** has high correlation with row **30**.
- The **Eigth row** has high correlation with the row **41**.
- The similar relationship pairs that can be seen are:
  - **N-6 & G-1**
  - **A-9 & M-2**
  - **B-2 & P-4**
  - **B-3 & K-3**
  - **B-4 & N-1**
  - **C-2 & T-5**

We can advise that customer with code **G-1** purchased hand-woven, so they can recommend hand woven to customer with code **N-6** as well. Customer with code **A-9** bought **Double Back** and **Knotted**, so they can recommend Double Back and Knotted to customer with code **M-2** as well.

References

1. Association Rules
2. Association Rules SaveAsGraph Documentation
3. Association Rules iGraph
4. Agglomerative Methods in Machine Learning
5. Decision Tree / Naive Bayes / Random Forest / Logistic Regression / Neural Network / Cluster Lecture Notes