ML Final Project

Nemin Dholakia

12/8/2021

##Import any necessary packages

1)Importing all necessary packages.

2)Read in the data as an .csv file

3)Display the head of the data (first 6 rows) so we can begin to understand what the dataset looks like.

library(readr) #For reading csv files  
library(dplyr) #For data wrangling

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(lubridate) #For handling dates

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(ggplot2) #For plotting of results  
  
#Read in the data  
Online\_Retail <- read\_csv("D:/MSBA/Fundamentals of Machine Learning/ML Final Project/Online\_Retail.csv")

## Rows: 541909 Columns: 8

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (5): InvoiceNo, StockCode, Description, InvoiceDate, Country  
## dbl (3): Quantity, UnitPrice, CustomerID

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

#Convert to dataframe  
Online\_Retail = data.frame(Online\_Retail)  
#Preview the data  
head(Online\_Retail)

## InvoiceNo StockCode Description Quantity  
## 1 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6  
## 2 536365 71053 WHITE METAL LANTERN 6  
## 3 536365 84406B CREAM CUPID HEARTS COAT HANGER 8  
## 4 536365 84029G KNITTED UNION FLAG HOT WATER BOTTLE 6  
## 5 536365 84029E RED WOOLLY HOTTIE WHITE HEART. 6  
## 6 536365 22752 SET 7 BABUSHKA NESTING BOXES 2  
## InvoiceDate UnitPrice CustomerID Country  
## 1 12/1/2010 8:26 2.55 17850 United Kingdom  
## 2 12/1/2010 8:26 3.39 17850 United Kingdom  
## 3 12/1/2010 8:26 2.75 17850 United Kingdom  
## 4 12/1/2010 8:26 3.39 17850 United Kingdom  
## 5 12/1/2010 8:26 3.39 17850 United Kingdom  
## 6 12/1/2010 8:26 7.65 17850 United Kingdom

##Data Wrangling in dplyr 1)Remove any transactions that aren’t associated with a logged-in user (we can’t know how many purchases a user has made if they check out as a guest).

2)Ensure that all columns are of the appropriate data type; the invoice date column must be converted to a date type.

3)Transform the data from transactional level (the alldata dataframe) into customer-level (the customerlvl dataframe).

#How many rows are not linked with a logged-in user?  
sum(is.na(Online\_Retail$CustomerID))

## [1] 135080

#Answer: 135,080 transactions  
#We will exclude these transactions, as we are only interested in monitoring user behaviour  
Online\_Retail <- Online\_Retail %>%  
 filter(!is.na(CustomerID))  
  
#Ensure columns are of correct types  
str(Online\_Retail)

## 'data.frame': 406829 obs. of 8 variables:  
## $ InvoiceNo : chr "536365" "536365" "536365" "536365" ...  
## $ StockCode : chr "85123A" "71053" "84406B" "84029G" ...  
## $ Description: chr "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPID HEARTS COAT HANGER" "KNITTED UNION FLAG HOT WATER BOTTLE" ...  
## $ Quantity : num 6 6 8 6 6 2 6 6 6 32 ...  
## $ InvoiceDate: chr "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" "12/1/2010 8:26" ...  
## $ UnitPrice : num 2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...  
## $ CustomerID : num 17850 17850 17850 17850 17850 ...  
## $ Country : chr "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...

#In my offline version of the data (downloaded csv), I had to convert the invoice date column to a date column. Here, it looks like the correct  
#datatypes have been preserved.  
  
#Count number of unique items purchased by each customer  
uniqueitems <- Online\_Retail %>%  
 group\_by(CustomerID) %>%  
 summarise(unique\_items\_customer = n\_distinct(StockCode))  
  
#Group the data at an invoice level  
invoicelvl <- Online\_Retail %>%  
 group\_by(CustomerID, InvoiceNo) %>%  
 summarise(unique\_items\_inv = n(),  
 quantity\_purchased = sum(Quantity),  
 total\_price = sum(Quantity\*UnitPrice),  
 invoice\_date = first(InvoiceDate)) %>%  
 arrange(CustomerID,invoice\_date)

## `summarise()` has grouped output by 'CustomerID'. You can override using the `.groups` argument.

#Join the two arrays together  
combineddata <- left\_join(invoicelvl, uniqueitems, by='CustomerID')  
  
#Group the data at a customer level  
customerlvl <- combineddata %>%  
 group\_by(CustomerID) %>%  
 summarise(no\_orders = n(),  
 unique\_items\_purchased = min(unique\_items\_customer),  
 quantity\_items\_purchased = sum(quantity\_purchased),  
 average\_quantity\_per\_order = mean(quantity\_purchased),  
 total\_money\_spent = sum(total\_price),  
 average\_spent\_per\_order = mean(total\_price))

We now have all of the data we require to group the 4,372 users into clusters. A overview of the new dataset can be found here:

summary(customerlvl[2:7])

## no\_orders unique\_items\_purchased quantity\_items\_purchased  
## Min. : 1.000 Min. : 1.00 Min. : -303.0   
## 1st Qu.: 1.000 1st Qu.: 15.00 1st Qu.: 153.0   
## Median : 3.000 Median : 35.00 Median : 365.0   
## Mean : 5.075 Mean : 61.21 Mean : 1122.3   
## 3rd Qu.: 5.000 3rd Qu.: 77.00 3rd Qu.: 962.2   
## Max. :248.000 Max. :1794.00 Max. :196719.0   
## average\_quantity\_per\_order total\_money\_spent average\_spent\_per\_order  
## Min. : -244.0 Min. : -4287.6 Min. :-4287.6   
## 1st Qu.: 78.0 1st Qu.: 293.4 1st Qu.: 152.0   
## Median : 137.0 Median : 648.1 Median : 237.0   
## Mean : 196.8 Mean : 1898.5 Mean : 315.9   
## 3rd Qu.: 232.2 3rd Qu.: 1611.7 3rd Qu.: 370.8   
## Max. :12540.0 Max. :279489.0 Max. : 6207.7

The value of total money spent is substantially larger than the other columns. If we used k-means to the unscaled data right away, total money spent would dominate the other columns, and our customers would be categorized primarily by total money spent. The data must be scaled:

#Scale the Data  
km\_data <- scale(customerlvl[2:7])  
summary(km\_data)

## no\_orders unique\_items\_purchased quantity\_items\_purchased  
## Min. :-0.436405 Min. :-0.7048 Min. :-0.30503   
## 1st Qu.:-0.436405 1st Qu.:-0.5410 1st Qu.:-0.20744   
## Median :-0.222244 Median :-0.3068 Median :-0.16208   
## Mean : 0.000000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.:-0.008082 3rd Qu.: 0.1848 3rd Qu.:-0.03426   
## Max. :26.012519 Max. :20.2843 Max. :41.85864   
## average\_quantity\_per\_order total\_money\_spent average\_spent\_per\_order  
## Min. :-1.3726 Min. :-0.75263 Min. :-12.7437   
## 1st Qu.:-0.3700 1st Qu.:-0.19528 1st Qu.: -0.4537   
## Median :-0.1862 Median :-0.15213 Median : -0.2184   
## Mean : 0.0000 Mean : 0.00000 Mean : 0.0000   
## 3rd Qu.: 0.1102 3rd Qu.:-0.03489 3rd Qu.: 0.1521   
## Max. :38.4356 Max. :33.77283 Max. : 16.3100

##Reducing Dimensionality

In this step the following things would be done: 1)Check for multicollinearity.

2)Try to find principal components to see if the dimensionality can be reduced.

**Checking for multicollinearity**

#Print correlation matrix  
cor(km\_data)

## no\_orders unique\_items\_purchased  
## no\_orders 1.00000000 0.6956634  
## unique\_items\_purchased 0.69566343 1.0000000  
## quantity\_items\_purchased 0.57416943 0.4409264  
## average\_quantity\_per\_order 0.04117428 0.1149862  
## total\_money\_spent 0.56612197 0.4120146  
## average\_spent\_per\_order 0.08752532 0.1888894  
## quantity\_items\_purchased average\_quantity\_per\_order  
## no\_orders 0.5741694 0.04117428  
## unique\_items\_purchased 0.4409264 0.11498617  
## quantity\_items\_purchased 1.0000000 0.31661947  
## average\_quantity\_per\_order 0.3166195 1.00000000  
## total\_money\_spent 0.9216490 0.23682905  
## average\_spent\_per\_order 0.3816547 0.61226503  
## total\_money\_spent average\_spent\_per\_order  
## no\_orders 0.5661220 0.08752532  
## unique\_items\_purchased 0.4120146 0.18888939  
## quantity\_items\_purchased 0.9216490 0.38165474  
## average\_quantity\_per\_order 0.2368291 0.61226503  
## total\_money\_spent 1.0000000 0.42007813  
## average\_spent\_per\_order 0.4200781 1.00000000

Total\_money\_spent and quantity\_items\_purchased are highly co-related and we do not need both these variables in cluster analysis so I choose to remove total\_money\_spent.

#Remove the 5th column & scale  
km\_data\_reduced <- scale(customerlvl[c(2:5,7)])  
summary(km\_data\_reduced)

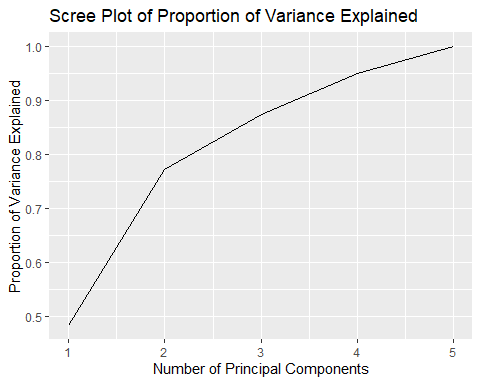
## no\_orders unique\_items\_purchased quantity\_items\_purchased  
## Min. :-0.436405 Min. :-0.7048 Min. :-0.30503   
## 1st Qu.:-0.436405 1st Qu.:-0.5410 1st Qu.:-0.20744   
## Median :-0.222244 Median :-0.3068 Median :-0.16208   
## Mean : 0.000000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.:-0.008082 3rd Qu.: 0.1848 3rd Qu.:-0.03426   
## Max. :26.012519 Max. :20.2843 Max. :41.85864   
## average\_quantity\_per\_order average\_spent\_per\_order  
## Min. :-1.3726 Min. :-12.7437   
## 1st Qu.:-0.3700 1st Qu.: -0.4537   
## Median :-0.1862 Median : -0.2184   
## Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.1102 3rd Qu.: 0.1521   
## Max. :38.4356 Max. : 16.3100

**Checking for Principal Components**

km\_data.pca <- prcomp(km\_data\_reduced, center = T,scale = T)  
#Summary of Principal Components  
summary(km\_data.pca)

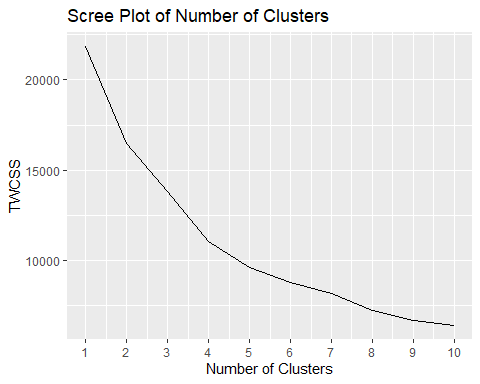
## Importance of components:  
## PC1 PC2 PC3 PC4 PC5  
## Standard deviation 1.5533 1.2011 0.7158 0.62225 0.49526  
## Proportion of Variance 0.4825 0.2885 0.1025 0.07744 0.04906  
## Cumulative Proportion 0.4825 0.7710 0.8735 0.95094 1.00000

#Proportion of variance explained:  
var <- km\_data.pca$sdev^2  
pve <- var/sum(var)  
#Scree Plot of Cumulative Variance Explained:  
g <- qplot(x = 1:5, y = cumsum(pve), geom = 'line', xlab="Number of Principal Components", ylab='Proportion of Variance Explained', main="Scree Plot of Proportion of Variance Explained")  
g + scale\_x\_continuous(breaks = seq(0, 5, by = 1))

 ##Applying K-means

**Identify the optimal number of clusters**

#Remove randomness from iterations by setting a seed  
set.seed(123)  
  
#Determine the maximum number of clusters  
#k\_max = 10 means we will test all possible numbers of clusters from 1 to 10 to see which performs the best.  
k\_max <- 10  
  
#Calculate the total within sum of squares for each of 1:k\_max  
twcss <- sapply(1:k\_max, function(k){kmeans(km\_data\_reduced, k)$tot.withinss})  
#Vizualize the results  
g <- qplot(x = 1:k\_max, y = twcss, geom = 'line', xlab="Number of Clusters", ylab='TWCSS', main="Scree Plot of Number of Clusters")  
g + scale\_x\_continuous(breaks = seq(0, 10, by = 1))

 We’re looking for a ‘elbow’ in the plot where the model’s quality no longer improves considerably as the model’s complexity grows. There is some ambiguity in this plan because there isn’t a defined elbow. This is due to a lack of separation between clusters (see the pairs plot below).

I’ll use k=5 in this case. When we test our model assumptions later, we’ll be able to see if this was a good decision.

**Applying K-means clustering with 5 cluster**

# k-means clustering  
km <-kmeans(km\_data\_reduced, centers = 5, nstart=20)  
#Check total within sum of squares with a variety of nstart and iter.max values  
#to ensure algorithm convergence  
km$tot.withinss

## [1] 9572.027

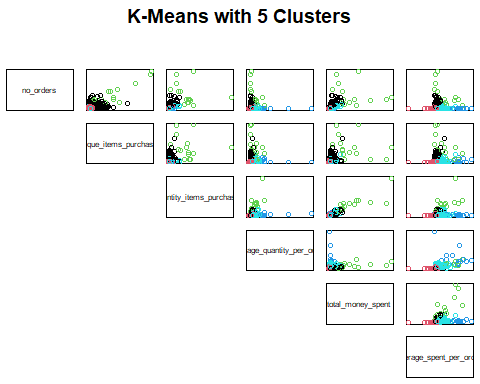
#What are the characteristics of the 5 groups?  
km$centers

## no\_orders unique\_items\_purchased quantity\_items\_purchased  
## 1 1.2511948 1.5535037 0.4581963  
## 2 -0.2030739 -0.2716794 -0.1471688  
## 3 9.9718354 7.8008540 12.5302400  
## 4 -0.3436019 -0.1374824 0.9231006  
## 5 -0.1593412 0.1638137 0.2269996  
## average\_quantity\_per\_order average\_spent\_per\_order  
## 1 0.007378346 0.05365236  
## 2 -0.182832544 -0.23331610  
## 3 2.530813432 3.89646370  
## 4 10.408260674 8.83434582  
## 5 1.198118802 1.62395165

#How many users belong to each group?  
table(km$cluster)

##   
## 1 2 3 4 5   
## 496 3480 15 15 366

# plot the dataset with clusters  
par(pty="m")  
pairs(km\_data, col = km$cluster, main="K-Means with 5 Clusters", lower.panel=NULL,  
 xaxt="n",yaxt="n", oma=c(0,0,5,0))

 This scale is really hard to interpret - for example, category 2, where the ‘number of orders’ variable is centred around a negative number. Transforming centers back to original scale for added interpretability.

**Backtransform onto original scale of variables**

data.orig = t(apply(km$centers, 1, function(r)r\*attr(km\_data\_reduced,'scaled:scale') + attr(km\_data\_reduced, 'scaled:center')))  
print(data.orig)

## no\_orders unique\_items\_purchased quantity\_items\_purchased  
## 1 16.760081 193.91935 3263.3992  
## 2 3.179023 38.00287 434.6552  
## 3 98.200000 727.60000 59673.5333  
## 4 1.866667 49.46667 5435.8000  
## 5 3.587432 75.20492 2183.0656  
## average\_quantity\_per\_order average\_spent\_per\_order  
## 1 199.1762 335.2646  
## 2 138.0920 231.6009  
## 3 1009.5511 1723.4314  
## 4 3539.3111 3507.1787  
## 5 581.5701 902.5153