ASSIGNMENT--4

Aim: Customer segmentation- k-means&PCA
Introduction:

What is Customer Segmentation:
Customer Segmentation is the process of division of customer base into several groups of individuals that share a similarity in different ways that are relevant to marketing such as gender, age, interests, and miscellaneous spending habits.

Companies that deploy customer segmentation are under the notion that every customer has different requirements and require a specific marketing effort to address them appropriately. Companies aim to gain a deeper approach of the customer they are targeting. Therefore, their aim has to be specific and should be tailored to address the requirements of each and every individual customer. Furthermore, through the data collected, companies can gain a deeper understanding of customer preferences as well as the requirements for discovering valuable segments that would reap them maximum profit. This way, they can strategize their marketing techniques more efficiently and minimize the possibility of risk to their investment.

What is K-Means Algorithm:
While using the k-means clustering algorithm, the first step is to indicate the number of clusters (k) that we wish to produce in the final output. The algorithm starts by selecting k objects from dataset randomly that will serve as the initial centers for our clusters. These selected objects are the cluster means, also known as centroids. Then, the remaining objects have an assignment of the closest centroid. This centroid is defined by the Euclidean Distance present between the object and the cluster mean. We refer to this step as "cluster assignment". When the assignment is complete, the algorithm proceeds to calculate new mean value of each cluster present in the data. After the recalculation of the centers, the observations are checked if they are closer to a different cluster. Using the updated cluster mean, the objects undergo reassignment. This goes on repeatedly through several iterations until the cluster assignments stop altering. The clusters that are present in the current iteration are the same as the ones obtained in the previous iteration.

#Customer Segmentation / Clustering with K-means,
#Principal Components Analysis and Bootstrap Evaluation

#1. Business Objective
#We want to better understand our customer, particularly
#on emphasis to the monetary value each customer contributes to the D2D business

#We look at popular "RFM segmentation"
#Recency – How recently did the customer purchase?
#Frequency – How often do they purchase?

#Monetary Value – How much do they spend?

```
library(readr)
library(skimr)
library(broom)
library(fpc)
library(scales)
library(ggrepel)
library(gridExtra)
require(tidyverse)
require(lubridate)
require(caret)
require(h2o)
require(corrplot)
require(correlationfunnel)
library(GGally)
require(Boruta)
require(skimr)
require(mice)
require(tidyquant)
require(rsample)
require(recipes)
require(iml)
require(DALEX)
require(correlationfunnel)
require(DataExplorer)
require(tictoc)
require(xgboost)
require(mlbench)
require(lime)
require(yardstick)
require(corrr)
require(knitr)
require(Matrix)

setwd("C:/Users/bokhy/Desktop/")

#Read the data
orders_tbl <- read_rds("d2d_transactions.rds")

## First, I need to create an analysis dataset
## and include average order value and number of orders
## as I want to take a look at a couple more variables
```

```
#Ideally included a 2-3 year of transaction history in your segmentation

customers_tbl <-
    orders_tbl %>%
    filter(!company_name %in% no_relationship) %>%

    #assume a cut-off date of 2019-06-01 and create a recency variable
    mutate(days_since = as.double(
        ceiling(
            difftime(
                time1 = "2019-06-01",
                time2 = ymd(as.Date(payment_date)),
                units = "days")))
) %>%

#select last 2 full years
filter(ymd(as.Date(payment_date)) <= "2019-06-01") %>%

#create analysis variables
group_by(account_id) %>%
summarise(
    recency = min(days_since),
    frequency = n(),
    #average sales
    avg_amount = mean(gross_revenue),
    #total sales
    tot_amount = sum(gross_revenue),
    #number of orders
    order_count = length(unique(ymd(as.Date(payment_date))))
) %>%
#average order value
mutate(avg_order_val = tot_amount / order_count) %>%
ungroup()


#Filter out companies that we don't work with
no_relationship <- c('4Evers Games','777 Studios','Activision','Anuman Interactive',
                     'Anvil-Soft','AtGames','Autumn Games','Badland Games',
                     'bitComposer','Born Ready Games','Bulkypix','CD Projekt RED',
                     'Deep Silver','Double Eleven','Excalibur / Merge','Feral Interactive',
                     'FireGlow','Gaijin Games','Games For All','Goblinz Studio',
                     'IQ Publishing','Juicy Beast','Limasse Five','LookAtMyGame',
                     'Mi-Clos Studio','Microids','Movie Games S.A.','Muzzy Lane',
                     'Neckbolt Games','Nexway','Nyamyam','Paradox Interactive',
```

```
                                'Ravenscourt','Rebellion','Rokaplay','Runic Games','Sanuk Games',
                                'Shadow Planet Productions','Squad','Sticky Rice Games',
                                'Viva Media')
customers_tbl <- customers_tbl %>%
  filter(!company_name %in% no_relationship)

# Outliers may represent rare occurrences of customers that have made,
# (for instance), only a few sporadic purchases over time or placed only one or two very large
orders and disappeared
# Instead, I think it' s important to include outliers so that they can be studied to understand
why they occurred and, should those customers reappear, target them with the right incentive

# (1) Recency
# Recency is right-skewed, with a mean of around 29 and 50% of observations between the
values of 13 and 135
ggplot(customers_tbl, aes(x = recency)) +
  geom_histogram()
summary(customers_tbl$recency)
skewness(customers_tbl$recency)

customers_tbl %>%
  ggplot(aes(x = recency)) +
  geom_histogram(bins = 100, fill = "#E69F00", colour = "red") +
  labs(x = "Days",
       y = "Numer of Publishers",
       title = "Days since last order") +
  coord_cartesian(xlim = c(0, 400)) +
  scale_x_continuous(breaks = seq(0, 400, 100)) +
  theme_light()

# (2) Frequency
# Frequency is right skewed and most customers have made between 17 and just under 500
purchases
ggplot(customers_tbl, aes(x = frequency)) +
  geom_histogram()
summary(customers_tbl$frequency)
skewness(customers_tbl$frequency)

# Total and Average Sales ----------------------------------------------

## I personally prefer average sales as it softens the impact of the extreme values
p1 <- customers_tbl %>%
  ggplot(aes(x = tot_amount)) +
  geom_histogram(bins = 50, fill = "light green", colour = "forest green") +
```

```
  labs(x = "",
       y = "",
       title = "Total sales per Publisher") +
  scale_x_continuous(
    labels = scales::dollar_format(scale = 1)) +
  scale_y_continuous(limits = c(0,20)) +
  theme_light()

p2 <- customers_tbl %>%
  ggplot(aes(x = avg_amount)) +
  geom_histogram(bins = 50, fill = "forest green", colour = "dark green") +
  labs(x = "",
       y = "",
       title = "Average sales per Publisher") +
  scale_x_continuous(
    labels = scales::dollar_format(scale = 1)) +
  scale_y_continuous(limits = c(0, 10)) +
  theme_light()

grid.arrange(p1, p2, nrow = 2)


# Number of Orders ----------------------------------------------------

summary(customers_tbl$order_count)

# peak at around 10 and another one at about 50.
# This suggests the potential for distinct subgroups in the data.
customers_tbl %>%
  ggplot(aes(x = order_count)) +
  geom_histogram(bins = 60, fill = "firebrick3", colour = "sandybrown") +
  labs(x = "",
       y = "",
       title = "Number of Orders per Publisher") +
  scale_x_continuous(breaks = seq(0, 700, 50)) +
  theme_light()


# Average Order Value -------------------------------------------------

summary(customers_tbl$avg_order_val)

# small amount of outliers around $200 value
customers_tbl %>%
```

```
ggplot(aes(x = avg_order_val)) +
geom_histogram(
    bins = 50,
    fill = "purple", colour = "black") +
labs(x = "",
         y = "",
         title = "Average Order Value") +
scale_x_continuous(
    labels = scales::dollar_format(scale = 1)) +
    theme_light()



#Segmentation Analysis ----------------------------------------


## 1. Scale the variables so that the relative difference in their magnitudes does not affect the
calculations
clust_data <-
    customers_tbl %>%
    select(-account_id) %>%
    scale() %>%
    as_tibble()

## 2. Calculate kmeans for any number of centres
kmeans_map <- function(centers = centers) {
    set.seed(0623)                          #for reproducibility
    clust_data[,1:3] %>%
        kmeans(centers = centers,
                    nstart = 100,
                    iter.max = 50)
}

#Create a nested tibble
kmeans_map_tbl <-
    tibble(centers = 1:10) %>%          #create column with centres
    mutate(k_means = centers %>%
                map(kmeans_map)) %>%      #iterate `kmeans_map` row-wise to gather
    #kmeans models for each centre in column 2

    mutate(glance = k_means %>%          #apply `glance()` row-wise to gather each
                map(glance))

## 3. build a scree plot
```

```
kmeans_map_tbl %>%
    unnest(glance) %>%                          # unnest the glance column
    select(centers, tot.withinss) %>%           # select centers and tot.withinss

    ggplot(aes(x = centers, y = tot.withinss)) +
    geom_line(colour = 'grey30', size = .8) +
    geom_point(colour = 'green4', size = 3) +
    geom_label_repel(aes(label = centers),
                                colour = 'grey30') +
    labs(title = 'Scree Plot') +
    scale_x_continuous(breaks = seq(0, 10, 2)) +
    theme_light()

## We look for the “elbow”, the point where the gain of adding an extra clusters in explained
"tot.withinss" starts to level off.
## It seems that the optimal number of clusters is 4.


# Evaluating the Clusters ---------------------------------------------

kmeans_4_tbl <-
    kmeans_map_tbl %>%
    pull(k_means) %>%
    pluck(3) %>%                    # pluck element 4
    augment(customers_tbl)          # attach .cluster to the tibble

kmeans_4_tbl %>%
    ggplot(aes(x = log(recency), y = log(avg_amount))) +
    geom_point(aes(colour = .cluster)) +
    labs(x = "Log Recency",
             y = "Log Average Sales",
             title = "") +
    theme_light()

## Looking at below, the clusters are not particularly well balanced,
## with the 1st group containing nearly %74 of all retailers, and there's only one in 4th group
options(digits = 2)

kmeans_4_tbl %>%
    group_by(.cluster) %>%
    summarise(
        Retailers = n(),
        Recency = median(recency),
        Frequency = median(frequency),
```

```
    Avg.Sales = median(avg_amount)
) %>%
ungroup() %>%
mutate(`Retailers(%)` = 100*Retailers / sum(Retailers)) %>%
arrange((.cluster)) %>%
select(c(1,2,6,3:5)) %>%
kable()
```

```
#Principal Components Analysis ----------------------------------------


pca_obj <-
    customers_tbl[,2:4] %>%
    # ALWAYS scale and centre your data! For some reason, this is not the default!
    prcomp(center = TRUE,
                scale. = TRUE)

summary(pca_obj)

data.frame(summary(pca_obj)$importance) %>%    # extract importance as a dataframe
    rownames_to_column() %>%                    # get metrics names in a column
    pivot_longer(c(2:4),
                        names_to = "PC",
                        values_to = "value") %>%          # using tidyr::pivot_longer, the new gather

    filter(rowname == 'Proportion of Variance') %>%
    ggplot(aes(x = PC, y = value)) +
    geom_col(aes(fill =     value)) +
    scale_fill_gradient(high = "grey5", low = "grey50") +
    scale_y_continuous(labels = scales::percent) +
    labs(x = "Principal Component",
            y = "Percentage of Variance Explained",
            title = "Variance Explained by Principal Component")
## The first 2 components are explaining 72.% of the variation in the data,
## which means that using the first 2 PCs will give us a very good understanding of the data and
every subsequent PC will add very little information



#PCA visualisation - 3 clusters ----------------------------------------
pca_tbl <-
    pca_obj$x %>%                       # extract "x", which contains the PCs co-ordinates
    as_tibble() %>%                     # change to a tibble
```

```
    bind_cols(customers_tbl %>%      # append retailer_code, my primary key
                  select(account_id))

pca_tbl

km_pca_4_tbl <-
    kmeans_map_tbl_alt %>%
    pull(k_means) %>%
    pluck(3) %>%                          # pluck element 3
    augment(customers_tbl) %>%        # attach .cluster to the tibble
    left_join(pca_tbl,                    # left_join by my primary key, retailer_code
              by = 'account_id')
km_pca_4_tbl



km_pca_4_tbl %>%
    ggplot(aes(x = PC1, y = PC2, colour = .cluster)) +
    geom_point(aes(shape = .cluster)) +
    labs(title      = 'K-Means 4 Clusters Against First Two Principal Components',
         caption    = "") +
    theme(legend.position = 'none') +
    theme_light() +
    xlim(-5,10) +
    ylim(-5,20)

# The chart confirms that the segments are well separated in the 4-cluster configuration.
# Segments 1 and 3 show significant variability in different directions and there is a degree of
overlapping between segments 2 and 4.
options(digits = 2)

kmeans_map_tbl %>%
    pull(k_means) %>%
    pluck(3) %>%
    augment(customers_tbl) %>%
    group_by(.cluster) %>%
    summarise(
    Customers = n(),
    Avg.Sales = median(avg_amount),
    Orders = mean(order_count),
    Avg.Order.Val = median(avg_order_val),
    `Tot.Sales($)` = sum(tot_amount)
) %>%
ungroup() %>%
mutate(`Tot.Sales(%)` = 100 * `Tot.Sales($)` / sum(`Tot.Sales($)`),
```

```
            `Customers(%)` = 100*Customers / sum(Customers))    %>%
  select(c(1, 2, 8, 3:7)) %>%
  kable()

## Understading chart in PROGRESS


## Clusterboot Evaluation
# One last step worth taking is verifying how 'genuine' your clusters are
# by validating whether they capture non-random structure in the dat

# The clusterboot algorithm uses bootstrap resampling to evaluate how stable a given cluster is
to perturbations in the data.

kmeans_boot100 <-
  clusterboot(
    clust_data[,1:3],
    B = 50,                          # number of resampling runs
    bootmethod = "boot",             # resampling method: nonparametric bootstrap
    clustermethod = kmeansCBI, # clustering method: k-means
    k = 7,                           # number of clusters
    seed = 0623)                      # for reproducibility

bootMean_df <-
  data.frame(cluster = 1:7,
             bootMeans = kmeans_boot100$bootmean)
bootMean_df
# To interpret the results, I visualise the tests output with a simple chart.

#Remember that values:

# above 0.8 (segment 2 and 4) indicates highly stable clusters
# between 0.6 and 0.75 (segments 1 and 6) signal a acceptable degree of stability
# below 0.6 (segment 7) should be considered unstable

# So, 4 cluster is highly stable

bootMean_df %>%
  ggplot(aes(x = cluster, y = bootMeans)) +
  geom_point(colour = 'green4', size = 4) +
  geom_hline(yintercept = c(0.6, 0.8)) +
  theme_light() +
```

```
  labs(x = "Cluster",
       title = "Clusterboot Stability Evaluation") +
  theme(legend.position = "none")
```

Conclusion ----------

This analysis should provide a solid base for discussion with relevant business stakeholders.
Normally I would present my client with a variety of customer profiles based on different combinations of customer features and formulate my own data-driven recommendations.
However, it is ultimately down to them to decide how many groups they want settle for and what characteristics each segment should have.

#Reference
#
https://diegousai.io/2019/09/steps-and-considerations-to-run-a-successful-segmentation