

```

1 rm(list=ls())
2 cat("\014") #clean console
3
4 #====Basic=====
5 library(tidyverse)
6 library(caret)
7 library(skimr)
8 library(ggplot2)
9 library(cluster)
10 library(psych)
11 library(ggcorrplot)
12 library(flexclust)
13
14 setwd('C:\\Users\\Jason Xu\\Desktop\\
    Framework_2\\Group project')
15 clean = read.csv('clean_dataset.csv',
    header = T)
16
17 #====description of the variables=====
18 # order_id - (A unique number to identity
    the order)
19 # user_id - (A unique number to identify
    the user)
20 # order_number - (The number of times a
    user purchased at hunter) [question 1]
21 # order_dow - (Day of the Week the order
    was made)
22 # order_hour_of_day - (Time of the order
    ) [question 2]
23 # days_since_prior_order - (History of the
    order)
24 # product_id - (Id of the product) [
    quesiton 2] [question 3]

```

```

25 # add_to_cart_order - (Number of items
    added to cart)
26 # reordered - (If the reorder took place,
    0000) [question 1]
27 # department_id - (Unique number allocated
    to each department)
28 # department - (Names of the departments)
29 # product_name - (Name of the products)
30 # frequency - # of time a user purchased
    at hunter
31
32 # The variables we need to use to solve
    each problem:
33 # [question 1]: order_number,
    add_to_cart_order
34 # [question 2]: order_hour_of_day,
    product_id
35 # [question 3]: order_number, product_id,
    order_hour_of_day
36
37
38 # '''
39 # purchasing_count <- ecommerce %>%
40 #   group_by(user_id, department) %>%
41 #   summarise(count = n()) %>%
42 #   pivot_wider(names_from = department,
    values_from = count, values_fill = 0)
43 # '''
44
45 #=====Dimension Reduction =====
46 #dropping id attributes
47 df <- clean %>% select(order_dow,
    order_hour_of_day, days_since_prior_order

```

```

47 , add_to_cart_order, reordered,
    first_time_purchase)#%>%left_join(
    purchase_frequency, by='user_id')%>%select
    (-user_id)
48
49 #=====Suitability for Factor Analysis=====
50
51 ggcorrplot(cor(df),colors = c('red','white
    ','green'),type = 'lower')
52 KMO(r = cor(df)) #KMO = 0.5
53 cortest.bartlett(cor(df),n = 105273) #
    significan
54
55 #=====Factor Analysis=====
56 scree_plot=scree(cor(df),factors = T, pc=T
    )
57 # 2 components suggested
58 data.frame(factor = 1:ncol(df), eigen =
    eigen(cor(df))$values)
59 # 3 components suggested
60 fa.parallel(df,fa='fa',fm = 'pa')
61 # 3 components suggested
62
63 # 3 factors selected
64 fa = fa(r = df,nfactors = 3,fm = 'pa',
    rotate = 'none')
65 fa$Vaccounted # 64% explained
66 data.frame(communality = fa$communality)
67
68 print(fa$loadings,cut=0.1)
69 fa.diagram(fa,sort = T)
70
71 #=====Principal Componenet Analysis=====

```

```

72
73 library(FactoMineR)
74 pca_facto = PCA(df,graph = F)
75 library(factoextra)
76 fviz_eig(pca_facto,ncp=11,addlabels = T)
77
78 pca = prcomp(df,scale. = F)
79 fviz_eig(pca,ncp = 11,addlabels = T)
80 pca_facto$eig[pca_facto$eig[,'eigenvalue'
    ]>1,]
81 # 3 components
82 pca_facto$eig
83
84 pca_facto = PCA(df,scale.unit = T,ncp = 3
    ,graph = F)
85 pca_facto$var$contrib %>%
86   round(2)
87 library(factoextra);library(gridExtra)
88 charts = lapply(1:3,FUN = function(x)
    fviz_contrib(pca_facto,choice = 'var',
    axes = x,title=paste('Dim',x)))
89 grid.arrange(grobs = charts)
90 fviz_pca_var(X = pca_facto,col.var = '
    contrib',gradient.cols = c('red'),col.
    circle = 'steelblue',repel = T)
91 # 59% total variance explained
92
93 trainComponents = pca_facto$ind$coord
94
95
96 #====Cluster Analysis====
97 within_ss = sapply(X = 1:9, #1 to 9 means
    to find the best number of clusters from

```

```

97 1 to 9
98          FUN = function(x)
      kmeans(clean,centers = x,iter.max = 100)$
      tot.withinss)
99
100 ratio_ss = sapply(X = 1:9,
101                   FUN = function(x) {
102                       km = kmeans(clean,
      centers = x,iter.max = 100)
103                       ratio = km$betweenss/
      km$totss
104                       return(ratio)
105                   })
106
107 dat = data.frame(clusters=1:9,within_ss,
      ratio_ss)
108 dat #check the result
109
110 #=====Plot=====
111 par(mar = c(4, 4, 2, 2)) # set the figure
      margin
112
113 # Elbow in Ratio Plot
114 ggplot(dat,aes(x=clusters,y=ratio_ss))+
115   geom_line(color='steelblue',size=1.4)+
116   scale_x_continuous(breaks=1:9,
      minor_breaks = 1:9)+
117   geom_vline(xintercept=4) # 4 clusters
      would be the best number of clusters
118
119 # Elbow in Within_ss (within sum of
      squares Plot)
120 ggplot(dat,aes(x=clusters,y=within_ss))+

```

```

121   geom_line(color='steelblue',size=1.4)+
122   scale_x_continuous(breaks=1:9,
    minor_breaks = 1:9)+
123   geom_vline(xintercept=4) # 4 clusters
    would be the best number of clusters
124
125 # Conclusion: We decide to choose 4
    clusters for further analysis
126
127 #====Analysis part====
128 set.seed(100)
129 km = kmeans(clean, centers = 4,iter.max =
    100, nstart =1) # k-means analysis
130 table(km$cluster)
131 k_cluster = km$cluster #□□□□□□
132
133 data2 = cbind(clean,k_cluster) #
    □□□□□□□□□□cluster
134 data2
135
136 data2 %>%
137   select(order_id:first_time_purchase,
    k_cluster) %>% # select the range of
    columns
138   group_by(k_cluster) %>%
139   summarize_all(function(x) round(mean(x,
    na.rm=T),4)) %>%
140   data.frame() #□□□□cluster□□□□□□□□,
    □□□□insights
141
142 #====Cluster then Predict Using
    Regression====
143 #-----research question 1-----

```

```

144 #split data
145 set.seed(100)
146 split = createDataPartition(y=clean$
      add_to_cart_order,p = 0.7,list = F,groups
      = 100)
147 train = clean[split,]
148 test = clean[-split,]
149
150 #regression predict
151 linear = lm(add_to_cart_order~.,train)
152 summary(linear)
153 sseLinear = sum(linear$residuals^2);
      sseLinear
154
155 #test dataset
156 predLinear = predict(linear,newdata=test)
157 sseLinear = sum((predLinear-test$
      add_to_cart_order)^2); sseLinear
158
159 #remove outcome
160 trainNorm = subset(train,select=-c(
      add_to_cart_order)) # remove outcome
161 testNorm = subset(test,select=-c(
      add_to_cart_order)) # remove outcome
162
163 # Since our data is clean already, so we
      do not need to normalize the data again
164
165 set.seed(100)
166 km = kmeans(x = trainNorm,centers = 2,
      iter.max=100,nstart=1)
167 #km$center
168

```

```

169 # Total within sum of squares Plot
170 within_ss = sapply(1:10,FUN = function(x
    ) kmeans(x = trainNorm,centers = x,iter.
      max = 100,nstart = 1)$tot.withinss)
171 ggplot(data=data.frame(cluster = 1:10,
    within_ss),aes(x=cluster,y=within_ss))+
172   geom_line(col='steelblue',size=1.2)+
173   geom_point()+
174   scale_x_continuous(breaks=seq(1,10,1))+
175   geom_vline(xintercept=2)# result: 2
    clsuters
176
177 # Ratio Plot
178 ratio_ss = sapply(1:10,FUN = function(x
    ) {km = kmeans(x = trainNorm,centers = x,
      iter.max = 100,nstart = 1)
179 km$betweenss/km$totss} )
180 ggplot(data=data.frame(cluster = 1:10,
    ratio_ss),aes(x=cluster,y=ratio_ss))+
181   geom_line(col='steelblue',size=1.2)+
182   geom_point()+
183   scale_x_continuous(breaks=seq(1,10,1))+
184   geom_vline(xintercept=2)# result: 2
    clsuters
185
186 # Apply Clustering Solution from Train to
    Test
187 km_kcca = as.kcca(km,trainNorm) #
    flexclust uses objects of the classes
    kcca
188 clusterTrain = predict(km_kcca)
189 clusterTest = predict(km_kcca,newdata=
    testNorm)

```



```

190
191 table(clusterTrain) # check train cluster
192 table(clusterTest) # check test cluster
193
194 # Split train and test based on cluster
    membership
195 train1 = subset(train,clusterTrain==1)
196 train2 = subset(train,clusterTrain==2)
197
198 test1 = subset(test,clusterTest==1)
199 test2 = subset(test,clusterTest==2)
200
201 # Predict for each Cluster then Combine
202 lm1 = lm(add_to_cart_order~.,train1)
203 lm2 = lm(add_to_cart_order~.,train2)
204
205 pred1 = predict(lm1,newdata=test1)
206 pred2 = predict(lm2,newdata=test2)
207
208 sse1 = sum((test1$add_to_cart_order-pred1
    )^2); sse1
209 sse2 = sum((test2$add_to_cart_order-pred2
    )^2); sse2
210
211 predOverall = c(pred1,pred2)
212 qualityOverall = c(test1$
    add_to_cart_order,test2$add_to_cart_order
    )
213
214 sseOverall = sum((predOverall -
    qualityOverall)^2); sseOverall # get the
    result of sse, can be used to prove our
    conclusion is more persuasive

```

```

215
216 # Compare Results
217 paste('SSE for model on entire data',
      sseLinear)
218 paste('SSE for model on clusters',
      sseOverall)
219
220
221 #-----research question 2-----
222 #split data
223 set.seed(100)
224 split = createDataPartition(y=clean$
      product_id,p = 0.7,list = F,groups = 100)
225 train = clean[split,]
226 test = clean[-split,]
227
228 #regression predict
229 linear = lm(product_id~.,train)
230 summary(linear)
231 sseLinear = sum(linear$residuals^2);
      sseLinear
232
233 #test dataset
234 predLinear = predict(linear,newdata=test)
235 sseLinear = sum((predLinear-test$
      product_id)^2); sseLinear
236
237 #remove outcome
238 trainNorm = subset(train,select=-c(
      product_id)) # remove outcome
239 testNorm = subset(test,select=-c(
      product_id)) # remove outcome
240

```

```

241 # Since our data is clean already, so we
    do not need to normalize the data again
242
243 set.seed(100)
244 km = kmeans(x = trainNorm,centers = 2,
    iter.max=100,nstart=1)
245 #km$center
246
247 # Total within sum of squares Plot
248 within_ss = sapply(1:10,FUN = function(x
    ) kmeans(x = trainNorm,centers = x,iter.
    max = 100,nstart = 1)$tot.withinss)
249 ggplot(data=data.frame(cluster = 1:10,
    within_ss),aes(x=cluster,y=within_ss))+
250     geom_line(col='steelblue',size=1.2)+
251     geom_point()+
252     scale_x_continuous(breaks=seq(1,10,1))+
253     geom_vline(xintercept=2) # result: 2
    clsuters
254
255 # Ratio Plot
256 ratio_ss = sapply(1:10,FUN = function(x
    ) {km = kmeans(x = trainNorm,centers = x,
    iter.max = 100,nstart = 1)
257 km$betweenss/km$totss} )
258 ggplot(data=data.frame(cluster = 1:10,
    ratio_ss),aes(x=cluster,y=ratio_ss))+
259     geom_line(col='steelblue',size=1.2)+
260     geom_point()+
261     scale_x_continuous(breaks=seq(1,10,1))+
262     geom_vline(xintercept=2) # result: 2
    clsuters
263

```

```

264 # Apply Clustering Solution from Train to
    Test
265 km_kcca = as.kcca(km,trainNorm) #
    flexclust uses objects of the classes
    kcca
266 clusterTrain = predict(km_kcca)
267 clusterTest = predict(km_kcca,newdata=
    testNorm)
268
269 table(clusterTrain) # check train cluster
270 table(clusterTest) # check test cluster
271
272 # Split train and test based on cluster
    membership
273 train1 = subset(train,clusterTrain==1)
274 train2 = subset(train,clusterTrain==2)
275
276 test1 = subset(test,clusterTest==1)
277 test2 = subset(test,clusterTest==2)
278
279 # Predict for each Cluster then Combine
280 lm1 = lm(product_id~.,train1)
281 lm2 = lm(product_id~.,train2)
282
283 pred1 = predict(lm1,newdata=test1)
284 pred2 = predict(lm2,newdata=test2)
285
286 sse1 = sum((test1$product_id-pred1)^2);
    sse1
287 sse2 = sum((test2$product_id-pred2)^2);
    sse2
288
289 predOverall = c(pred1,pred2)

```

```

290 qualityOverall = c(test1$product_id,test2
    $product_id)
291
292 sseOverall = sum((predOverall -
    qualityOverall)^2); sseOverall # get the
    result of sse, can be used to prove our
    conclusion is more persuasive
293
294 # Compare Results
295 paste('SSE for model on entire data',
    sseLinear)
296 paste('SSE for model on clusters',
    sseOverall)
297

```