巨量資料分析-競賽報告

競賽簡介: 透過顧客資料分析建立商品推薦系統

競賽目的: 近年隨著大數據與人工智慧議題崛起，精準行銷已成為各企業重要發展技術，本屆競賽主題將聚焦於各大產業廣泛運用的「商品推薦系統」。透過學習統計學、資料科學、機器學習技術，能直接有效地解決問題。

競賽規則: 預測客戶購買商品的準確率越高越好。

(競賽資料之內容將放在本報告附錄 A)

隊名: Caspar

組員/分工情形:

王奎貿(組長):

整合並修改組員程式 + 改善資料+word/簡報 + ensemble learning

陳立維:

跑 ridge regression & lasso regression

陳柏翰:

跑 bagging & randomforest

吳佳玲:

跑 boosting

載入需要套件

```
if(!require(dplyr)){install.packages("dplyr")}
if(!require(mice)){install.packages("mice")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(DataExplorer)){install.packages("DataExplorer")}
if(!require(VIM)){install.packages("VIM")}
```

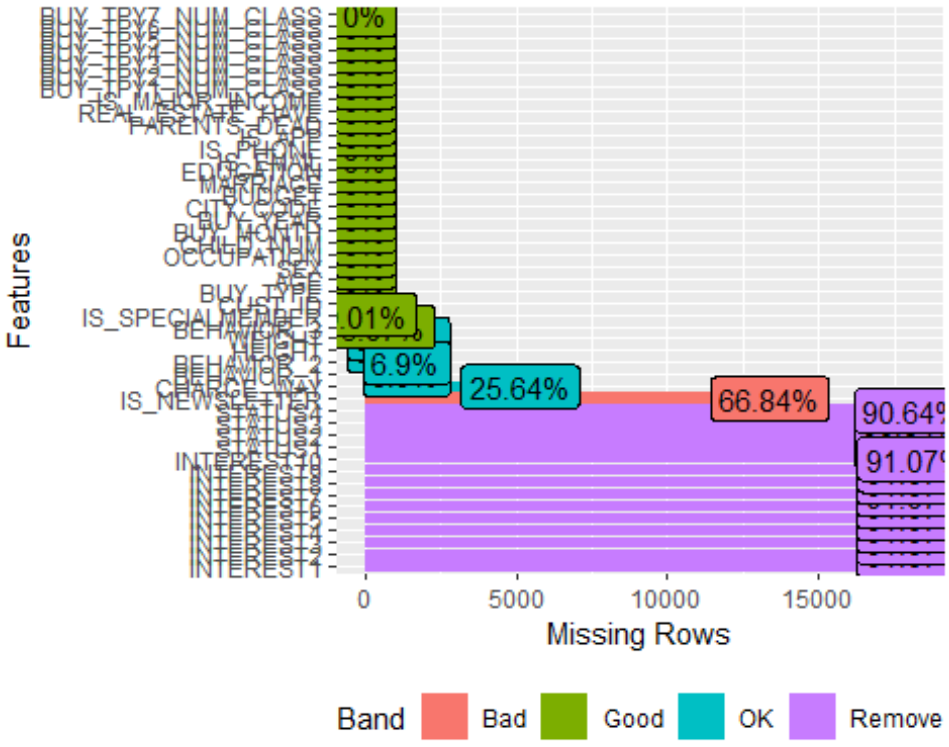
讀取需要資料集

```
train = read.csv("./train.csv", header = T, sep = ",") %>% select(-X)
test = read.csv("./test.csv", header = T, sep = ",")
```

檢視資料狀況

此資料有 48 個變數，20000 筆觀察值。

其中有 NA 值的變數有 22 個，接著列出各筆 NA 值的個數，並搭配’DataExplorer’的圖表，刪除 NA 個數佔整個觀察值 50%的變數。



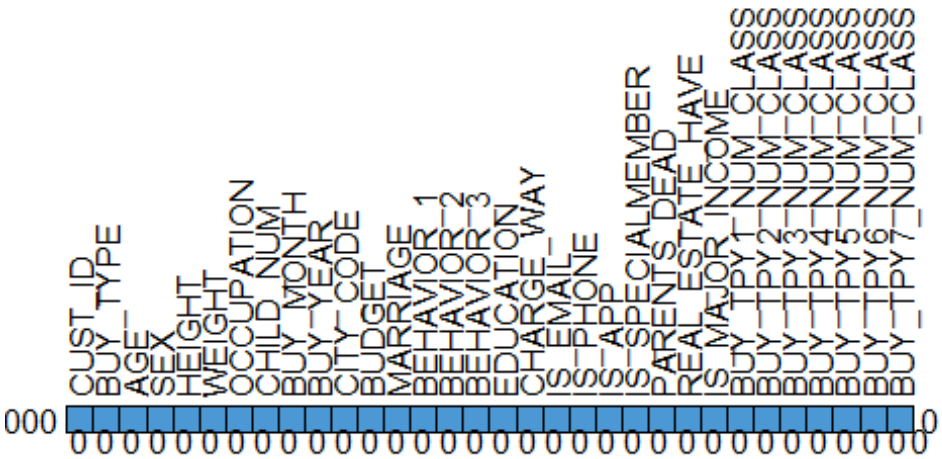
刪除 NA 大於總資料筆數 50%的變數

train:NA 的處理

欄位	變數名稱	中文說明	NA 個數	NA 處理方法
6	HEIGHT	客戶身高	1103	中位數
7	WEIGHT	客戶體重	1103	中位數
15	BEHAVIOR_1	行為 1	1380	眾數
16	BEHAVIOR_2	行為 2	1380	眾數
17	BEHAVIOR_3	行為 3	614	眾數
18	STATUS1	狀態 1	18129	刪除此變數
19	STATUS2	狀態 2	18129	刪除此變數
20	STATUS3	狀態 3	18129	刪除此變數
21	STATUS4	狀態 4	18129	刪除此變數
23	IS_NEWSLETTER	是否訂閱電子報	13368	刪除此變數
24	CHARGE_WAY	扣款方式	5129	眾數
27	INTEREST1	個人興趣 1	18214	刪除此變數
28	INTEREST2	個人興趣 2	18214	刪除此變數
29	INTEREST3	個人興趣 3	18214	刪除此變數
30	INTEREST4	個人興趣 4	18214	刪除此變數
31	INTEREST5	個人興趣 5	18214	刪除此變數
32	INTEREST6	個人興趣 6	18214	刪除此變數
33	INTEREST7	個人興趣 7	18214	刪除此變數
34	INTEREST8	個人興趣 8	18214	刪除此變數
35	INTEREST9	個人興趣 9	18214	刪除此變數
36	INTEREST10	個人興趣 10	18214	刪除此變數
38	IS_SPECIALMEMBER	是否具有特定資格	2	KNN 插補

補值完後再次確認有無 NA 值:

```
##  /\     /\
## {  `---'  }
## {  0   0  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



雙變量

我們將整理好的資料畫每個變數對 BUY_TYPE 的雙變量圖，以便我們之後做資料合併。　由於有 30 個自變數，在此只畫出最後有做處裡的雙變量圖。（詳見**附錄 C**）

變數值合併

經由上述的雙變量圖，我們有做變數處理的有: OCCUPATION, CHILD_NUM, MARRIAGE, AGE, BEHAVIOR_1, BEHAVIOR_2, BUY_TYP1_NUM_CLASS ~ BUY_TYP7_NUM_CLASS。　另外再將 HEIGHT, WEIGHT 合併為 bmi。

類別變數合併的標準是: 假如有幾個類別相較其他少，那麼就把他們的數量加到相差不多，例如: 32(A), 41(B), 174(C), 776(D), 4296(E), 2997(F), 11684(G)，其中 A、B、C 較少，那麼就把 ABC 併到 D。

- OCCUPATION: 種類眾多，將屬於同一大類的合在一起，再將過少的族群合併 (CFHJKOPUV 到 B 類)
- CHILD_NUM: 由於無小孩的人數遠大於其他有小孩的人數，所以將此變數分為兩類:有小孩&無小孩。
- MARRIAGE: 從圖顯示 B 類為最多，緊接著是 F 類，其餘都極少數，將此變數分為兩類:　B 一類&不是 B 一類。

- AGE: 由於 o~q 類的人為少數，將這幾類合併到第 n 類，使預測不會受到少數值影響。
- BEHAVIOR_1: 從圖顯示 a 與其他兩類筆數差距懸殊，但也不想失去資料的訊息，因此分成: a 一類&不是 a 一類。
- BEHAVIOR_2: 此變數在圖上僅僅顯示兩類，且兩類筆數差距懸殊，因此全部的值都設成最多的那類。
- BUY_TYP1_NUM_CLASS: A~D 的類別過少，合併成一類。
- BUY_TYP2_NUM_CLASS: A~E 的類別過少，合併成一類。
- BUY_TYP3_NUM_CLASS: C~E 的類別過少，合併成一類。
- BUY_TYP4_NUM_CLASS: A~D 的類別過少，合併到 E 成一類。
- BUY_TYP5_NUM_CLASS: B~D 的類別過少，合併到 E 成一類。
- BUY_TYP6_NUM_CLASS: A、C、D 的類別過少，合併到 F 成一類。
- BUY_TYP7_NUM_CLASS: A~C 的類別過少，合併到 D 成一類。
- BMI: WEIGHT / HEIGHT^2

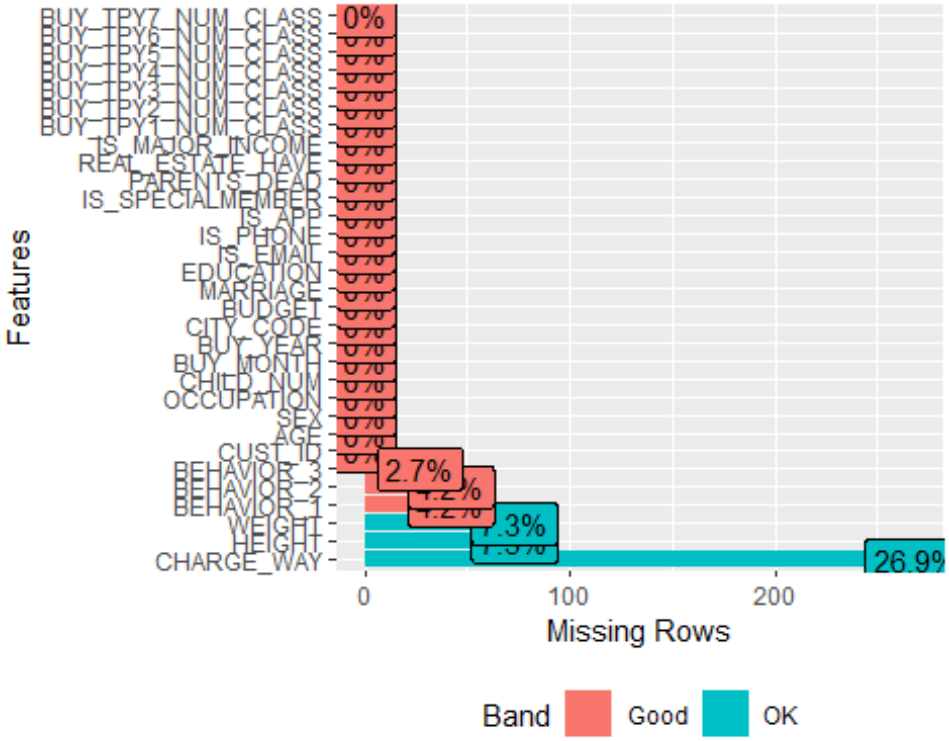
合併後再將 BUDGET, BMI 的離群值砍掉。

經過多次嘗試後發現去除 bmi 大於等於 20000 及 bmi 小於等於-20000 的資料，之後的預測分數皆會提高至少 0.3 分。

註:合併前的變數比例(百分比) 放在**附錄 B**

雙變量圖(合併之後的結果)(詳見附錄 D)

將整理完後 train 裡的變數，從 test 取出並檢視 NA 值的狀況:

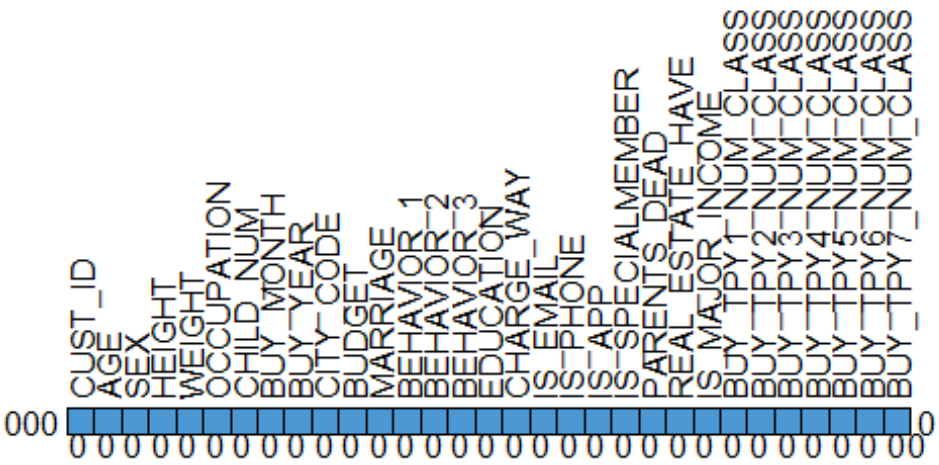


test: NA 的補值

欄位	變數名稱	中文說明	NA 個數	NA 處理方法
6	HEIGHT	客戶身高	73	中位數
7	WEIGHT	客戶體重	73	中位數
15	BEHAVIOR_1	行為 1	42	眾數
16	BEHAVIOR_2	行為 2	42	眾數
17	BEHAVIOR_3	行為 3	27	眾數
24	CHARGE_WAY	扣款方式	269	眾數

補值完後再次確認有無 NA 值:

```
##  /\     /\
## {  `---'  }
## {  0   0  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```



補值完後，依照 train 的變數合併方式對 test 的變數做相同的動作。(不含處裡缺失值)

最後的資料維度:

- train: 19904 筆資料，32 個變數(含 BUY_TYPE)
- test: 1000 筆資料，31 個變數(不含 BUY_TYPE)

資料分析及預測

- boosting
- ridge regression / lasso regression
- bagging / randomforest
- ensemble learning

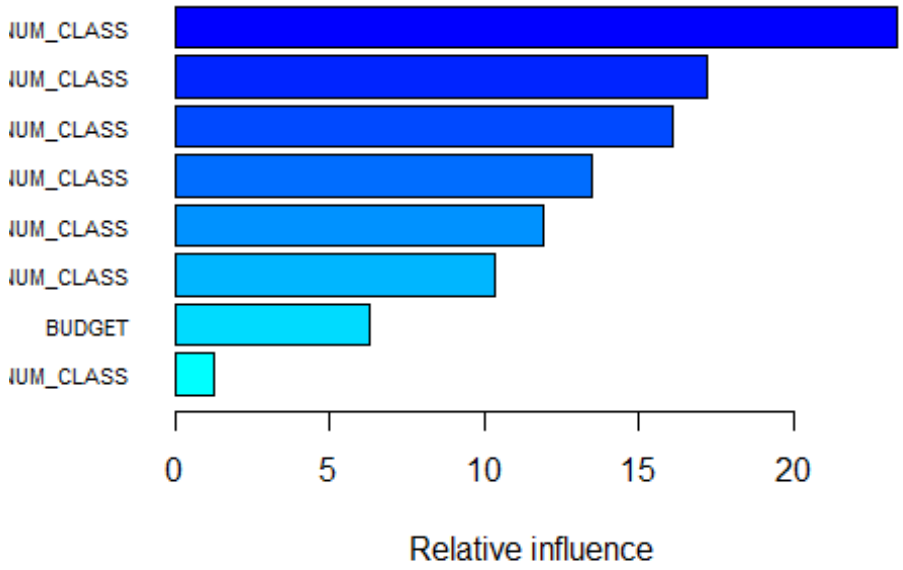
method1: boosting

由於方法眾多，為了不要把原先檔案取代掉，所以設了 tmp, tmp2 分別設為 train & test

原先將所有變數選入，後來依照重要變數圖選出重要變數再跑一次，shrinkage 的部分有試過 0.1,0.01,0.001，最後以 0.1 有最好的預測力。

最後選出來的變數有: BUDGET、BUY_TPY1_NUM_CLASS~BUY_TPY7_NUM_CLASS

```
if(!require(gbm)){install.packages("gbm")}
library(gbm)
set.seed(1)
boost.train <- gbm(BUY_TYPE~., data=tmp[,c(2,26:32,12)], distribution="multinomial", n.trees=500, interaction.depth=1, shrinkage = 0.1)  # 做 500 棵樹可能 overfitting:設壓縮率(shrinkage)
summary.gbm(boost.train, las=TRUE, cex.name=0.7)
```




```
##                                   var    rel.inf
## BUY_TPY2_NUM_CLASS BUY_TPY2_NUM_CLASS 23.364012
## BUY_TPY4_NUM_CLASS BUY_TPY4_NUM_CLASS 17.226219
## BUY_TPY6_NUM_CLASS BUY_TPY6_NUM_CLASS 16.074175
## BUY_TPY5_NUM_CLASS BUY_TPY5_NUM_CLASS 13.475110
## BUY_TPY1_NUM_CLASS BUY_TPY1_NUM_CLASS 11.943134
## BUY_TPY3_NUM_CLASS BUY_TPY3_NUM_CLASS 10.366684
## BUDGET                         BUDGET  6.303572
## BUY_TPY7_NUM_CLASS BUY_TPY7_NUM_CLASS  1.247095
```

將 test 放入 boosting model 做預測。

由於此資料為多類別，所以做出來的預測值是每個類別可能出現的機率，我們透過取最大值的方式，來把可能的類別取出。

```
## pred
##   a   b   c   d   e   f   g
## 208  60 178 212 250  60  32
```

method2: ridge regression

剛開始跟前面的方法一樣，先將所有變數放入模型中，並看估計出的參數估計如何，而 ridge regression 在參數估計上比較特別的是會針對購買類別給出各變數的估計值，我們將每個購買類別的變數估計值挑選 0.1~1 的變數，再總體來看哪些變數經常影響到購買類別。

最後選出來的變數有: BUY_TPY1_NUM_CLASS~BUY_TPY7_NUM_CLASS、BUDGET、MARRIAGE、BEHAVIOR_1、CHARGE_WAY。

用套件 glmnet，在 alpha = 0 時，建立 ridge regression。

```
library(glmnet)
# Sit a model with dependent variable of multinomial family with ridge penalty
ridge.mod=glmnet(x,y,alpha=0, family = "multinomial")  #配適 Ridge
# Summary of fit model
```

透過交叉驗證法將模型的參數最佳解求得，且因為要跑多類別，所以 family = multinomial.

```
cvfit <- cv.glmnet(data.matrix(x), y,alpha = 0, family = "multinomial")
```

透過最小的錯誤分類率來觀察一下參數的估計結果，還有最佳λ值

```
# Pring the minimum lambda - regularization factor
print(cvfit$lambda.min)

## [1] 0.0284669

print(cvfit$lambda.1se)

## [1] 0.0284669

## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)         -2.1396719
## BUY_TPY1_NUM_CLASS -0.8604801
## BUY_TPY2_NUM_CLASS  0.3650673
## BUY_TPY3_NUM_CLASS  0.5457219
## BUY_TPY4_NUM_CLASS  0.2664762
## BUY_TPY5_NUM_CLASS  0.2169577
## BUY_TPY6_NUM_CLASS  0.2641484
```

```
## BUY_TPY7_NUM_CLASS  0.2479258
## CHARGE_WAY          0.3809085
## BEHAVIOR_1          .
## MARRIAGE           -0.1150624
## BUDGET              1.2391991
```

ridge regression 在預測的時候，test 只能放預測時所用的變數。

由於此資料為多類別，所以做出來的預測值是每個類別可能出現的機率，我們透過取最大值的方式，來把可能的類別取出，跟 boosting 一樣。

```
## pred
##   a   b   c   d   e   f   g
## 216  75 156 251 230  63   9
```

method3: lasso regression

做 lasso regression 時，我們放的變數是根據 ridge regression 所產生的，之所以不再看每個類別的各變數估計值是因為在 lasso regression 中，大部分的估計值都過小，無法有效選出影響變數，因此我們決定延用 ridge regression 所選出來的變數。

最後選出來的變數有: BUY_TPY1_NUM_CLASS~BUY_TPY7_NUM_CLASS、BUDGET、MARRIAGE、BEHAVIOR_1、CHARGE_WAY。

```
lasso <- cv.glmnet(data.matrix(x), y,alpha = 1, family = "multinomial")
```

lasso 的最佳λ值

```
print(lasso$lambda.min)
```

```
## [1] 2.84669e-05
```

```
##                                                      V1 feature
## a <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       a
## b <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       b
## c <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       c
## d <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       d
## e <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       e
## f <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       f
## g <S4 class 'dgCMatrix' [package "Matrix"] with 6 slots>       g
```

跟上述的 ridge regression, boosting 一樣，預測值都是類別的機率，我們一樣透過最大值的方式將最有可能的類別取出。

```
## pred
##   a   b   c   d   e   f   g
## 221  74 149 241 225  64  26
```

method4: Bagging 演算法

```
#install.packages("randomForest")
library(randomForest)
```

為了之後跑模型，所以將 train 的 BUY_TYPE 做變數轉換。(變成 factor)

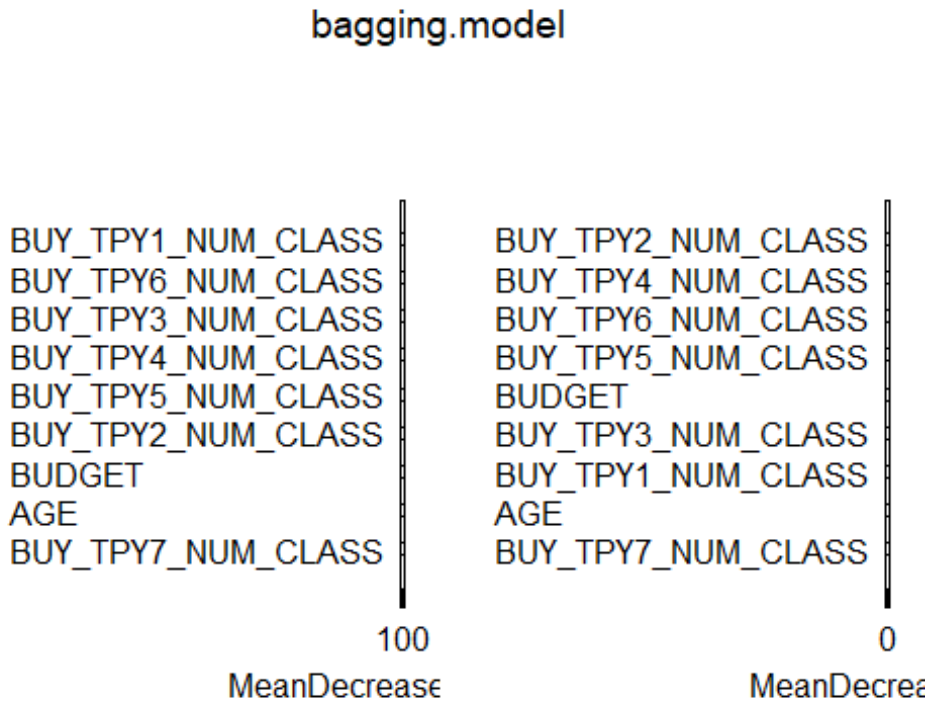
建立 training data 和 validation data，一開始一樣是先把全部變數丟入，最後看重要變數圖，把一些不重要的變數砍掉，從頭跑模型預測，再慢慢的把變數加回來讓預測值達到最高後停止。

```
train$AGE <- as.factor(train$AGE)
train$BUY_MONTH <- as.factor(train$BUY_MONTH)
train$BUY_TPY1_NUM_CLASS <- as.factor(train$BUY_TPY1_NUM_CLASS )
train$BUY_TPY2_NUM_CLASS <- as.factor(train$BUY_TPY2_NUM_CLASS )
train$BUY_TPY3_NUM_CLASS <- as.factor(train$BUY_TPY3_NUM_CLASS )
train$BUY_TPY4_NUM_CLASS <- as.factor(train$BUY_TPY4_NUM_CLASS )
train$BUY_TPY5_NUM_CLASS <- as.factor(train$BUY_TPY5_NUM_CLASS )
train$BUY_TPY6_NUM_CLASS <- as.factor(train$BUY_TPY6_NUM_CLASS )
train$BUY_TPY7_NUM_CLASS <- as.factor(train$BUY_TPY7_NUM_CLASS )

test$AGE <- as.factor(test$AGE)
test$BUY_MONTH <- as.factor(test$BUY_MONTH)
test$BUY_TPY1_NUM_CLASS <- as.factor(test$BUY_TPY1_NUM_CLASS )
test$BUY_TPY2_NUM_CLASS <- as.factor(test$BUY_TPY2_NUM_CLASS )
test$BUY_TPY3_NUM_CLASS <- as.factor(test$BUY_TPY3_NUM_CLASS )
test$BUY_TPY4_NUM_CLASS <- as.factor(test$BUY_TPY4_NUM_CLASS )
test$BUY_TPY5_NUM_CLASS <- as.factor(test$BUY_TPY5_NUM_CLASS )
test$BUY_TPY6_NUM_CLASS <- as.factor(test$BUY_TPY6_NUM_CLASS )
test$BUY_TPY7_NUM_CLASS <- as.factor(test$BUY_TPY7_NUM_CLASS )
set.seed(150)
#從原本的訓練集裡面，依照 0.8:0.2 的比例抽出訓練集及測試集

#選擇"BUY_TYPE"為 y.train 及 y.test 的變數，
#除了變數 CUST_ID、BUY_TYPE，其餘均列為參考變數"為 x.train 及 x.test
train$BUY_TYPE=as.factor(train$BUY_TYPE)
y.train = train[,"BUY_TYPE"]
x.train = train[,-c(1,2,5,6,11,17,25,15,10,24,23,14,4,8,7,22,19,13,20,21,33,18,
16,9)]
```

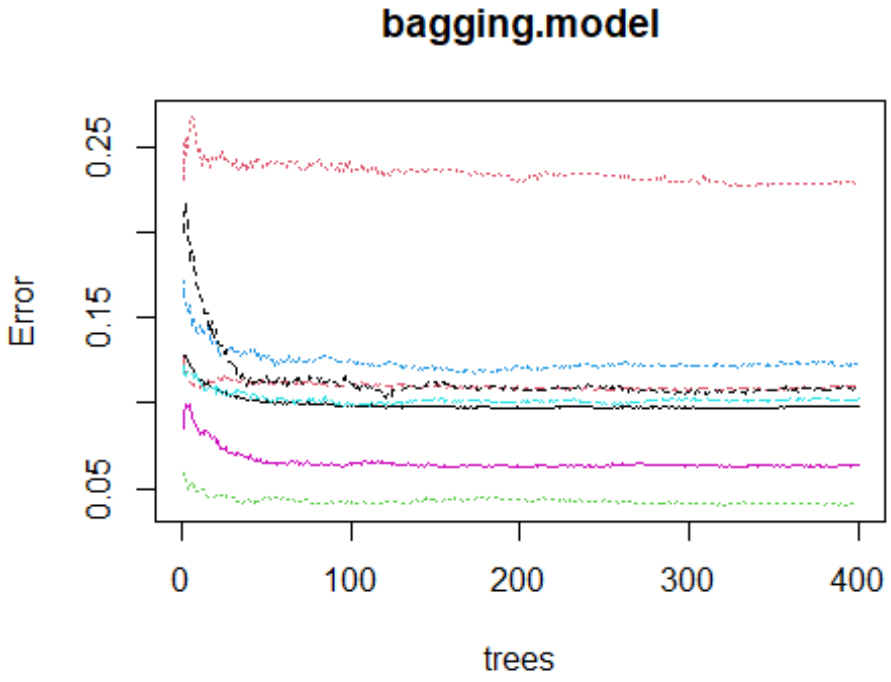
利用建立好的訓練集建構 bagging 模型，mtry = 變數總個數。

ntree 的值，一開始設為 1000，從 error 圖看差不多在 150 時趨近穩定。

```
#Bagging 模型
library(randomForest)
bagging.model = randomForest(x = x.train, y = y.train, ntree = 400, mtry = 9,i
mportance=TRUE)
bagging.model

##
## Call:
##  randomForest(x = x.train, y = y.train, ntree = 400, mtry = 9,       importa
nce = TRUE)
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 9.81%
## Confusion matrix:
##      a    b    c    d    e    f   g class.error
## a 3725   19  119  115  113   71  23  0.10991637
## b   18 1844   10   18   19   13   2  0.04158004
## c   83    6 2594   88   93   78  15  0.12275955
## d   72   17  112 3671  109   95  14  0.10244499
## e   56    9   61   70 3895   64   5  0.06370192
## f   33    9   65   57   37 1680   4  0.10875332
## g   20    9   40   42   21   28 543  0.22759602

#importance(bagging.model)
varImpPlot(bagging.model, sort = TRUE)
```




```
plot(bagging.model)
```




```
round(importance(bagging.model))
```

```
##                          a     b   c   d   e   f   g MeanDecreaseAccuracy
## AGE                      0    40  46  44  66  43  10                  102
## BUDGET                 118    53 119 110  86  61 142                  219
## BUY_TPY1_NUM_CLASS     724    45  34  99  43  30  69                  546
## BUY_TPY2_NUM_CLASS     157    63  80 178 167  51  88                  240
## BUY_TPY3_NUM_CLASS      71  1040  46  59  45  36  53                  326
## BUY_TPY4_NUM_CLASS     127    50 185 156  63  65  67                  257
## BUY_TPY5_NUM_CLASS     100    42  82 123  83 189  68                  257
## BUY_TPY6_NUM_CLASS     140    61  50 894  66  14  69                  508
## BUY_TPY7_NUM_CLASS      59    17  12  47  31  13  27                   75
##                      MeanDecreaseGini
## AGE                               747
## BUDGET                           2067
## BUY_TPY1_NUM_CLASS               1109
## BUY_TPY2_NUM_CLASS               3174
## BUY_TPY3_NUM_CLASS               2053
## BUY_TPY4_NUM_CLASS               2355
## BUY_TPY5_NUM_CLASS               2275
## BUY_TPY6_NUM_CLASS               2340
## BUY_TPY7_NUM_CLASS                363
```

從圖可以看到，BUY_TYPE1_NUM_CLASS~BUY_TYPE7_NUM_CLASS、BUDGET、AGE 都相對重要。之後對此模型做預測

Note: 即使 bagging 不做變數選擇，把所有變數丟進去，預測結果也不差，將資料丟到競賽網站中分數為 89.9。

method5: Random Forest(隨機森林)

建立 training data 和 validation data，變數選擇的部分一樣是透過先把全部變述丟進去後，再剔除不重要的變數。

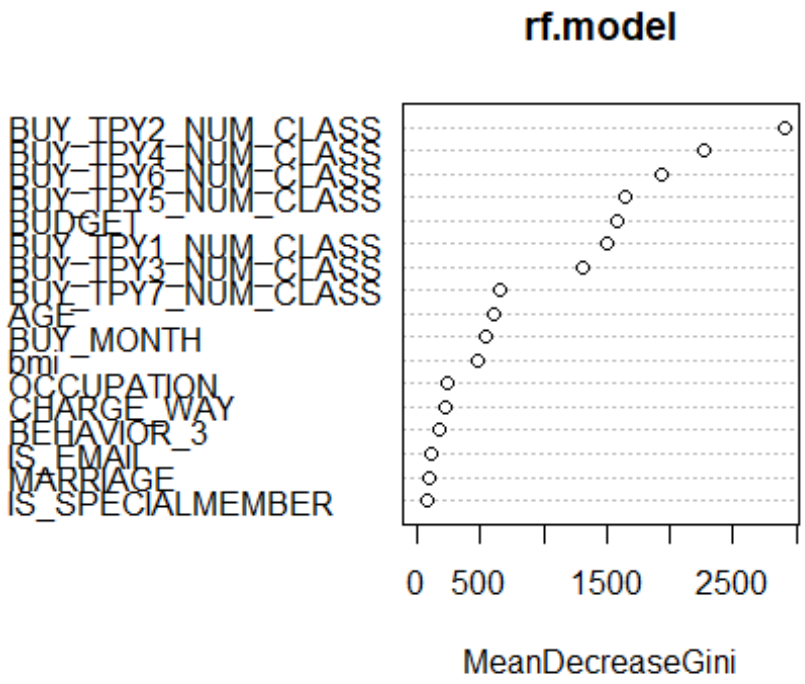
最後選出來的變數有: AGE、OCCUPATION、BUY_MONTH、BUDGET、MARRIAGE、BEHAVIOR_3、CHARGE_WAY、IS_EMAIL、IS_SPECIALMEMBER、BUY_TPY1_NUM_CLASS~BUY_TPY7_NUM_CLASS、bmi。

```
train$AGE <- as.factor(train$AGE)
train$BUY_MONTH <- as.factor(train$BUY_MONTH)
train$BUY_TPY1_NUM_CLASS <- as.factor(train$BUY_TPY1_NUM_CLASS )
train$BUY_TPY2_NUM_CLASS <- as.factor(train$BUY_TPY2_NUM_CLASS )
train$BUY_TPY3_NUM_CLASS <- as.factor(train$BUY_TPY3_NUM_CLASS )
train$BUY_TPY4_NUM_CLASS <- as.factor(train$BUY_TPY4_NUM_CLASS )
train$BUY_TPY5_NUM_CLASS <- as.factor(train$BUY_TPY5_NUM_CLASS )
train$BUY_TPY6_NUM_CLASS <- as.factor(train$BUY_TPY6_NUM_CLASS )
```

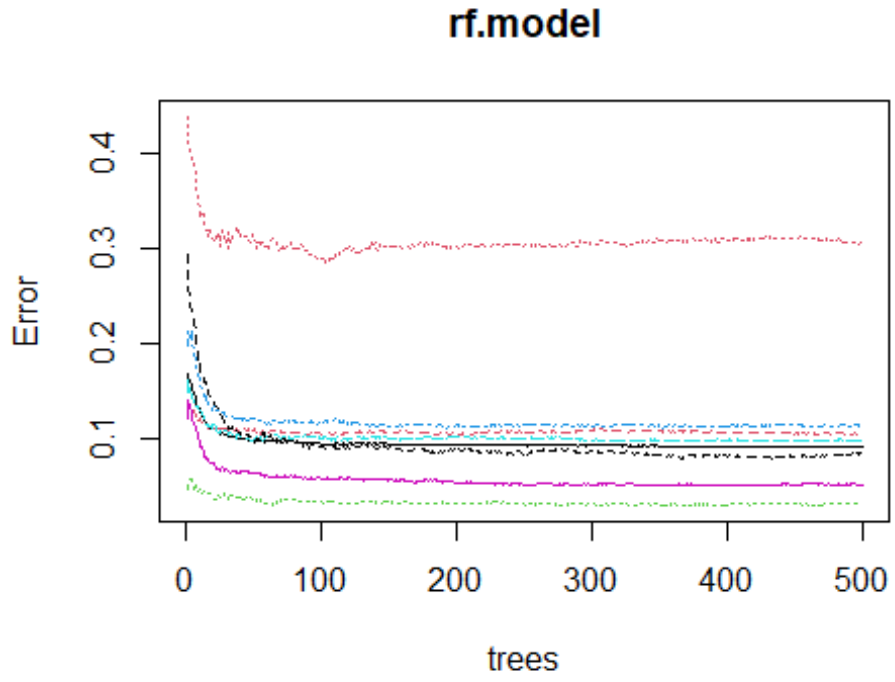
```
train$BUY_TPY7_NUM_CLASS <- as.factor(train$BUY_TPY7_NUM_CLASS )

test$AGE <- as.factor(test$AGE)
test$BUY_MONTH <- as.factor(test$BUY_MONTH)
test$BUY_TPY1_NUM_CLASS <- as.factor(test$BUY_TPY1_NUM_CLASS )
test$BUY_TPY2_NUM_CLASS <- as.factor(test$BUY_TPY2_NUM_CLASS )
test$BUY_TPY3_NUM_CLASS <- as.factor(test$BUY_TPY3_NUM_CLASS )
test$BUY_TPY4_NUM_CLASS <- as.factor(test$BUY_TPY4_NUM_CLASS )
test$BUY_TPY5_NUM_CLASS <- as.factor(test$BUY_TPY5_NUM_CLASS )
test$BUY_TPY6_NUM_CLASS <- as.factor(test$BUY_TPY6_NUM_CLASS )
test$BUY_TPY7_NUM_CLASS <- as.factor(test$BUY_TPY7_NUM_CLASS )
set.seed(150)



#選擇"Survived"為 y.train 及 y.test 的變數，
#除了變數 CUST_ID、BUY_TYPE，其餘均列為參考變數"為 x.train 及 x.test
train$BUY_TYPE=as.factor(train$BUY_TYPE)
y.train = train[,"BUY_TYPE"]
x.train = train[,-c(1,2,5,6,11,10,25,15,23,17,24,14,21,20,4,8)]
```

利用建立好的訓練集建構 randomforest 模型，mtry < p。我們從 mtry = 1 ~ $\sqrt{p}$ 慢慢試，最後在 mtry = 8 時有最好的準確力。之後對此模型做預測。

```
#隨機森林模型
library(randomForest)
rf.model = randomForest(x = x.train, y = y.train)
#importance(rf.model)
varImpPlot(rf.model, sort = TRUE)
```




```
plot(rf.model)
```




```
round(importance(rf.model))
```

```
##                       MeanDecreaseGini
## AGE                                612
## OCCUPATION                         245
## BUY_MONTH                          543
## BUDGET                            1579
## MARRIAGE                            90
## BEHAVIOR_3                         180
## CHARGE_WAY                         217
## IS_EMAIL                           104
## IS_SPECIALMEMBER                    84
## BUY_TPY1_NUM_CLASS                1498
## BUY_TPY2_NUM_CLASS                2909
## BUY_TPY3_NUM_CLASS                1315
## BUY_TPY4_NUM_CLASS                2268
## BUY_TPY5_NUM_CLASS                1653
## BUY_TPY6_NUM_CLASS                1935
## BUY_TPY7_NUM_CLASS                 647
## bmi                                480
```

```
#tuneRF(x=x.train, y=y.train,ntreeTry= 400)

#進行模型預測
# Predicting on testing data set
rfpred <- predict(rf.model,newdata=test,type='class')
# Checking classification accuracy
test$BUY_TYPE <- rfpred
#boost.confus.matrix <-table(real=y.test, predict=rfpred)
#sum(diag(boost.confus.matrix))/sum(boost.confus.matrix) # 對角線的數量/總數量

Predictions = data.frame(test[c("CUST_ID","BUY_TYPE")])
readr::write_csv(file = "BUY_TYPE_random forest.csv", x = Predictions)
```

Method6: 集成學習法(ensemble learning)

將上述所有方法(除了 ridge regression & lasso regression，因為預測不佳)的預測結果以多數決的方式整合。

```
boost <-  read.csv("./boosting.csv", header = T, sep = ",")
#ridge <- read.csv("./Q1.csv", header = T, sep = ",")
#lasso <- read.csv("./Q2.csv", header = T, sep = ",")
bagging <- read.csv("./BUY_TYPE_bagging.csv", header = T, sep = ",")
rf <- read.csv("./BUY_TYPE_random forest.csv", header = T, sep = ",")

overall = cbind(boost$pred, bagging$BUY_TYPE, rf$BUY_TYPE)
```

```
buy = rep(0,nrow(overall))
for(i in 1:nrow(overall)){
  buy[i] = names(table(overall[i,]))[table(overall[i,])==max(table(overall
[i,]))]
}
buy = as.factor(buy)
overall$pred <- buy

test$BUY_TYPE <- overall$pred
Predictions = data.frame(test[c("CUST_ID","BUY_TYPE")])
readr::write_csv(file = "BUY_TYPE_ensemble.csv", x = Predictions)
```

結論

由下列圖顯示: 最好的模型為 ensemble learning 的 91.8

分析方法	預測分數	均方根誤差
Boosting	91.1	0.298
ridge regression	80	0.447
lasso regression	80.3	0.444
bagging	91.5	0.292
random forest	91.6	0.29
ensemble learning	91.8	0.286

附錄 A（資料之變數內容）

欄位	中文名稱	類型	備註
CUST_ID	客戶編號	字元	
BUY_TYPE（預測變數）	當次購買商品類別(預測變數)	字元	共七種商品類別，並已轉換為代碼
AGE	客戶年齡	字元	以 5 歲為一檻，並已轉換為代碼
SEX	客戶性別	字元	已轉換為代碼
HEIGHT	客戶身高	數字	標準化處理後數值
WEIGHT	客戶體重	數字	標準化處理後數值
OCCUPATION	客戶職業類別	字元	已轉換為代碼(前 1 碼為大分類)
CHILD_NUM	客戶的小孩數量	數字	
BUY_MONTH	客戶購買月	數字	
BUY_YEAR	客戶購買年	數字	
CITY_CODE	通訊城市	字元	已轉換為代碼
BUDGET	預算	數字	標準化處理後數值
MARRIAGE	婚姻狀況	字元	已轉換為代碼
BEHAVIOR_1	行為 1	字元	已轉換為代碼
BEHAVIOR_2	行為 2	字元	已轉換為代碼
BEHAVIOR_3	行為 3	字元	已轉換為代碼
STATUS1	狀態 1	字元	已轉換為代碼
STATUS2	狀態 2	字元	已轉換為代碼
STATUS3	狀態 3	字元	已轉換為代碼
STATUS4	狀態 4	字元	已轉換為代碼
EDUCATION	教育程度/學歷	字元	已轉換為代碼
IS_NEWSLETTER	是否訂閱電子報	字元	已轉換為代碼
CHARGE_WAY	扣款方式	字元	已轉換為代碼
IS_EMAIL	是否有 E-MAIL	字元	已轉換為代碼
IS_PHONE	是否有手機	字元	已轉換為代碼
INTEREST1	個人興趣 1	字元	已轉換為代碼
INTEREST2	個人興趣 2	字元	已轉換為代碼
INTEREST3	個人興趣 3	字元	已轉換為代碼
INTEREST4	個人興趣 4	字元	已轉換為代碼
INTEREST5	個人興趣 5	字元	已轉換為代碼
INTEREST6	個人興趣 6	字元	已轉換為代碼
INTEREST7	個人興趣 7	字元	已轉換為代碼
INTEREST8	個人興趣 8	字元	已轉換為代碼
INTEREST9	個人興趣 9	字元	已轉換為代碼
INTEREST10	個人興趣 10	字元	已轉換為代碼
IS_APP	是否使用行動 APP	字元	已轉換為代碼

IS_SPECIALMEMBER	是否具有特定資格	字元	已轉換為代碼
PARENTS_DEAD	父母是否存在	字元	已轉換為代碼
REAL_ESTATE_HAVE	是否有房地產	字元	已轉換為代碼
IS_MAJOR_INCOME	是否為家庭主要經濟來源	字元	已轉換為代碼
BUY_TPY1_NUM_CLASS	過去 TYPE1 已購買件數區間	字元	已轉換為代碼
BUY_TPY2_NUM_CLASS	過去 TYPE2 已購買件數區間	字元	已轉換為代碼(同以上編碼)
BUY_TPY3_NUM_CLASS	過去 TYPE3 已購買件數區間	字元	已轉換為代碼(同以上編碼)
BUY_TPY4_NUM_CLASS	過去 TYPE4 已購買件數區間	字元	已轉換為代碼(同以上編碼)
BUY_TPY5_NUM_CLASS	過去 TYPE5 已購買件數區間	字元	已轉換為代碼(同以上編碼)
BUY_TPY6_NUM_CLASS	過去 TYPE6 已購買件數區間	字元	已轉換為代碼(同以上編碼)
BUY_TPY7_NUM_CLASS	過去 TYPE7 已購買件數區間	字元	已轉換為代碼(同以上編碼)

附錄 B（train 合併前的變數比例(百分比)）

OCCUPATION					
A: 2.870	B: 0.750	C: 0.360	D: 13.020	E: 1.950	F: 0.190
G: 1.015	H: 0.035	I: 32.900	J: 0.175	K: 0.225	L: 1.745
M: 3.520	N: 1.915	O: 0.065	P: 0.035	Q: 14.625	R: 5.265
S: 6.365	T: 12.805	U: 0.135	V: 0.035		

CHILD_NUM					
0: 79.320	1: 5.635	2: 9.350	3: 4.515	4: 0.930	5: 0.200
6: 0.025	7: 0.015	8: 0.010			

AGE					
a: 8.165	b: 3.025	c: 3.715	d: 6.500	e: 9.265	f: 9.000
g: 9.665	h: 9.725	i: 8.825	j: 8.035	k: 8.030	l: 6.645
m: 4.585	n: 2.835	o: 1.185	p: 0.560,	q: 0.240	

MARRIAGE:					
a: 1.875	b: 51.545	c: 0.115	d: 0.650	e: 0.095	f: 45.720

BEHAVIOR_1		
a: 97.855	b: 0.195	c: 1.950

BEHAVIOR_2:	
a: 0.44	b: 99.56

BUY_TYP1_NUM_CLASS					
A: 0.160	B: 0.205	C: 0.870	D: 3.880	E: 21.480	F: 14.985
G: 58.420					

BUY_TPY2_NUM_CLASS:					
A, 0.020	B: 0.010,	C: 0.080,	D: 0.540	E: 5.920	F: 22.725
G: 70.705					

BUY_TPY3_NUM_CLASS:				
C: 0.005	D: 0.010	E: 3.845	F: 11.060	G: 85.080

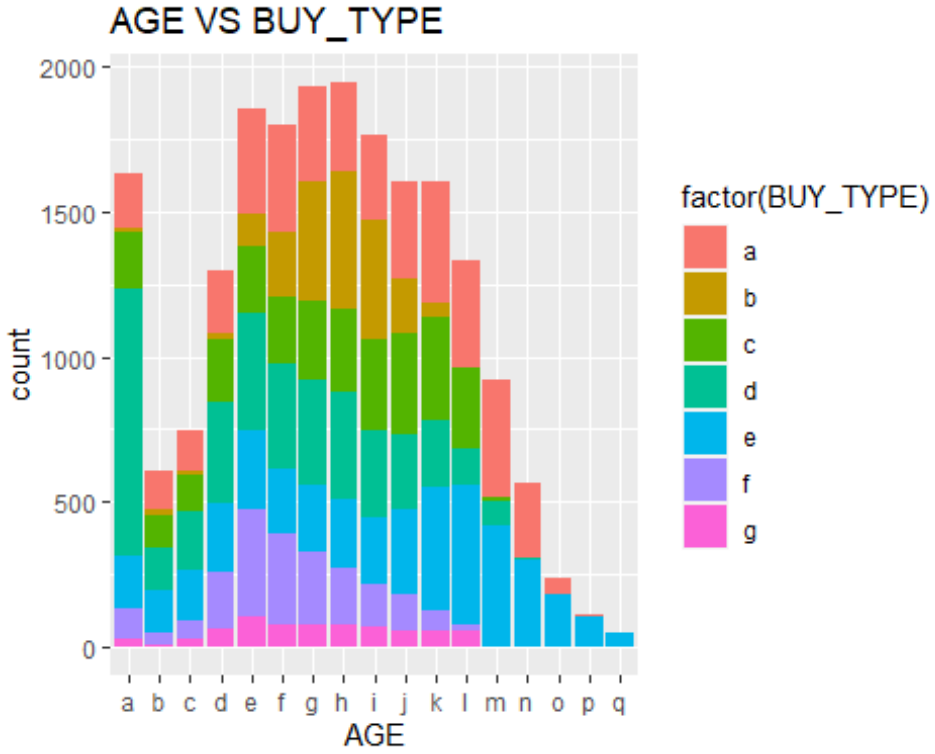
BUY_TPY4_NUM_CLASS:					
A: 0.010	B: 0.015	C: 0.035	D: 0.485	E: 9.050	F: 16.025
G: 74.380					

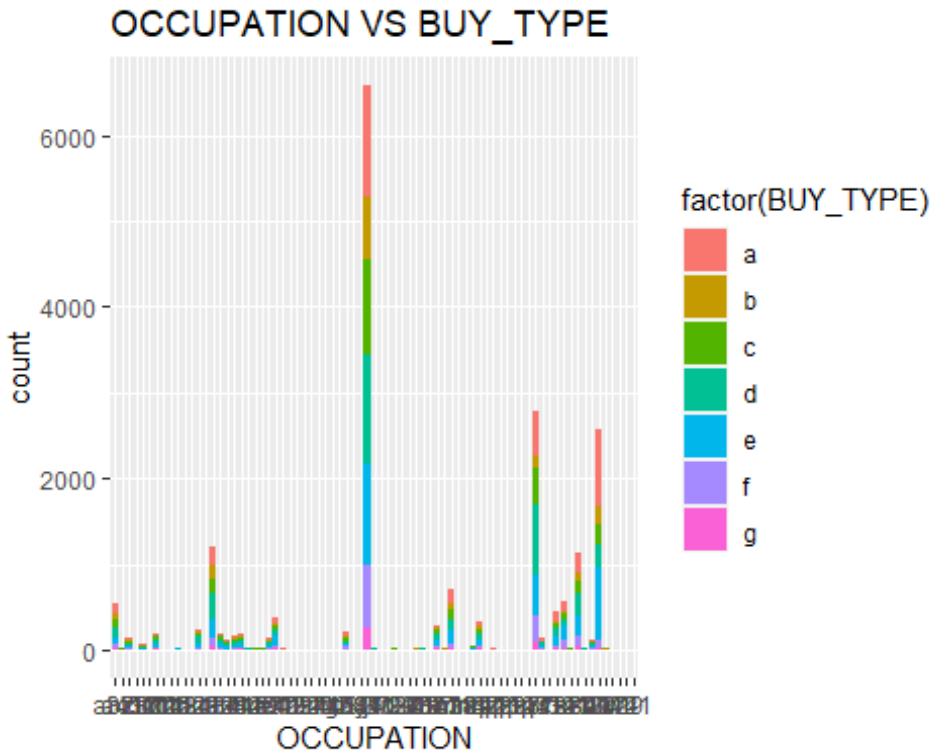
BUY_TPY5_NUM_CLASS:					
B: 0.005	C: 0.005	D: 0.230	E: 4.975	F: 9.670	G: 85.115

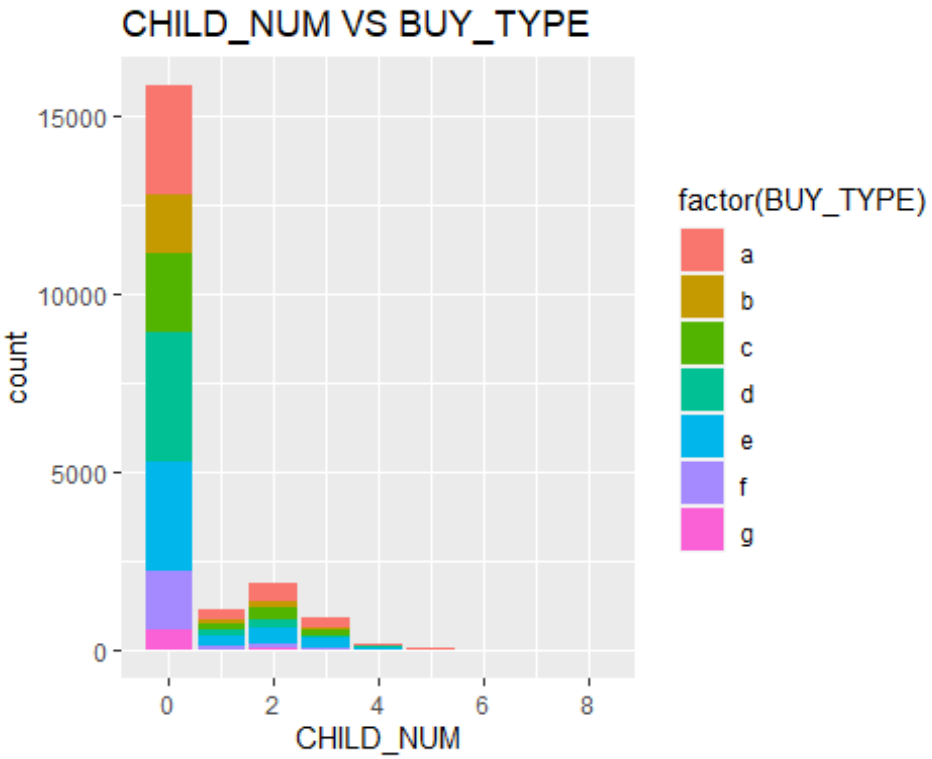
BUY_TPY6_NUM_CLASS:					
A: 0.005	C: 0.125	D: 4.270	E: 26.095	F: 7.815	G: 61.690

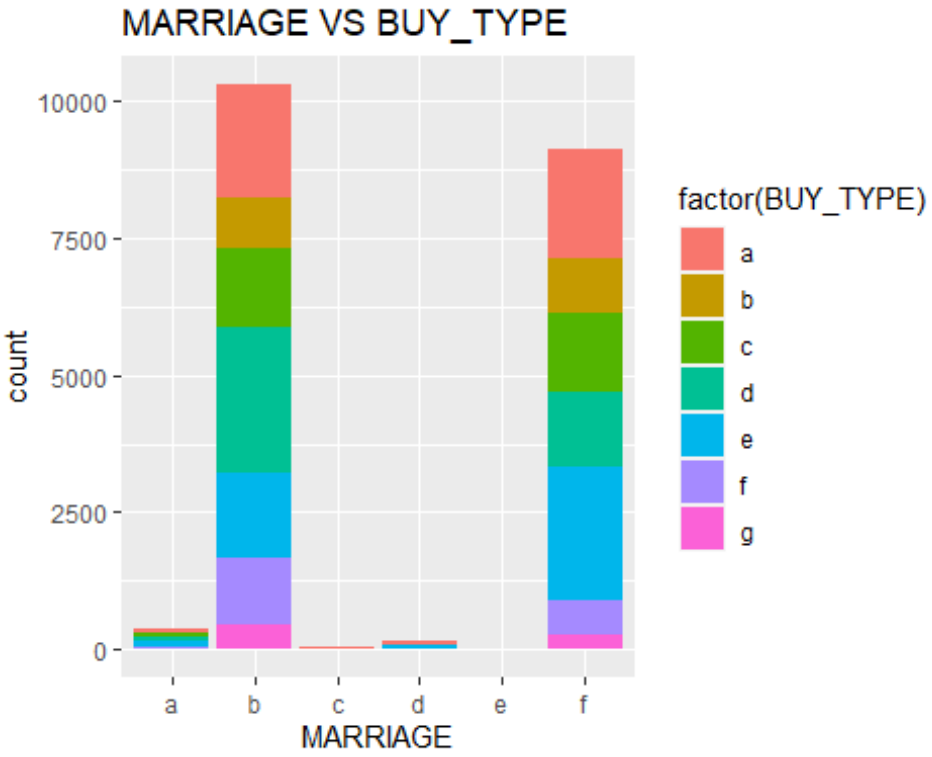
BUY_TPY7_NUM_CLASS:					
A: 0.005	B: 0.020	C: 0.130	D: 2.960	E: 24.305	F: 12.405
G: 60.175					

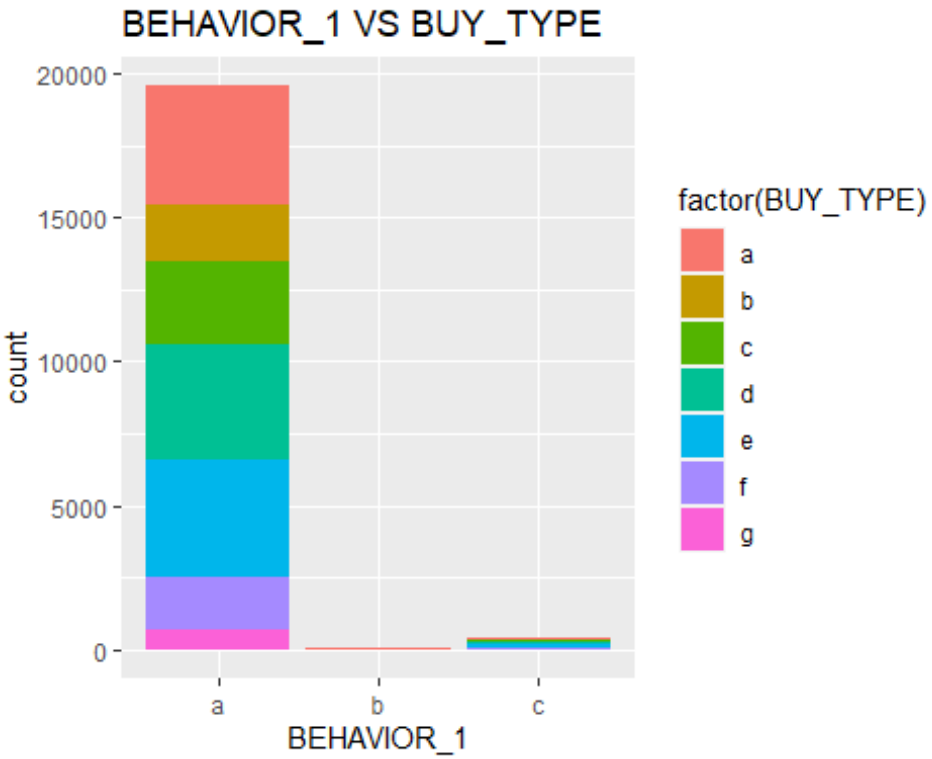
附錄 C（雙變量圖表）

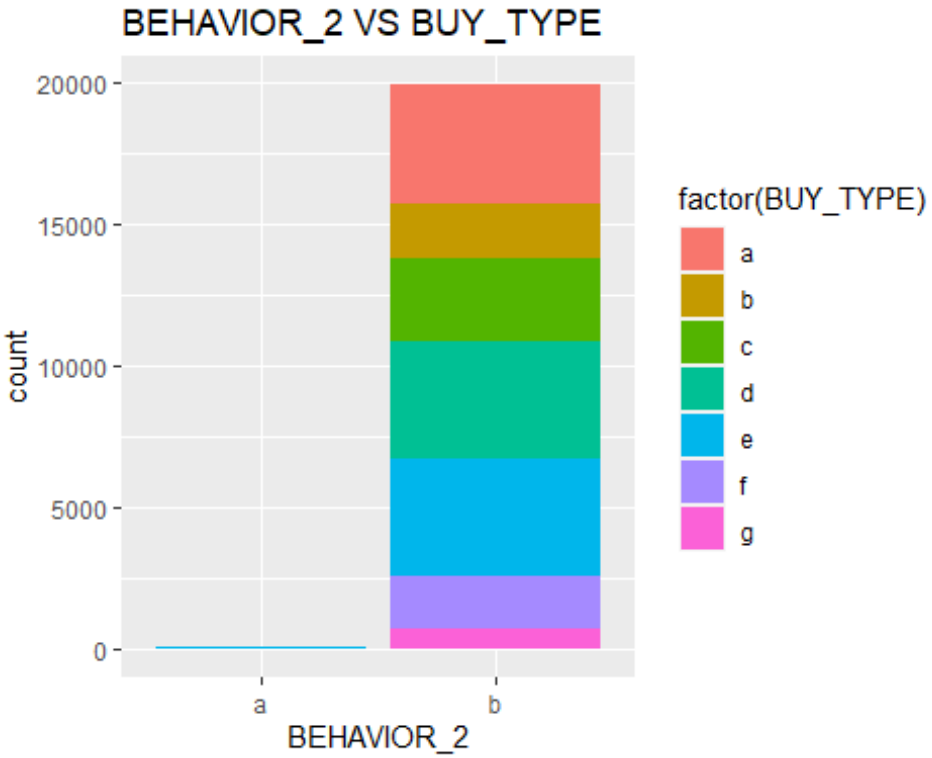

AGE VS BUY_TYPE
count
2000
1500
1000
500
0
a b c d e f g h i j k l m n o p q
AGE
factor(BUY_TYPE)
a
b
c
d
e
f
g

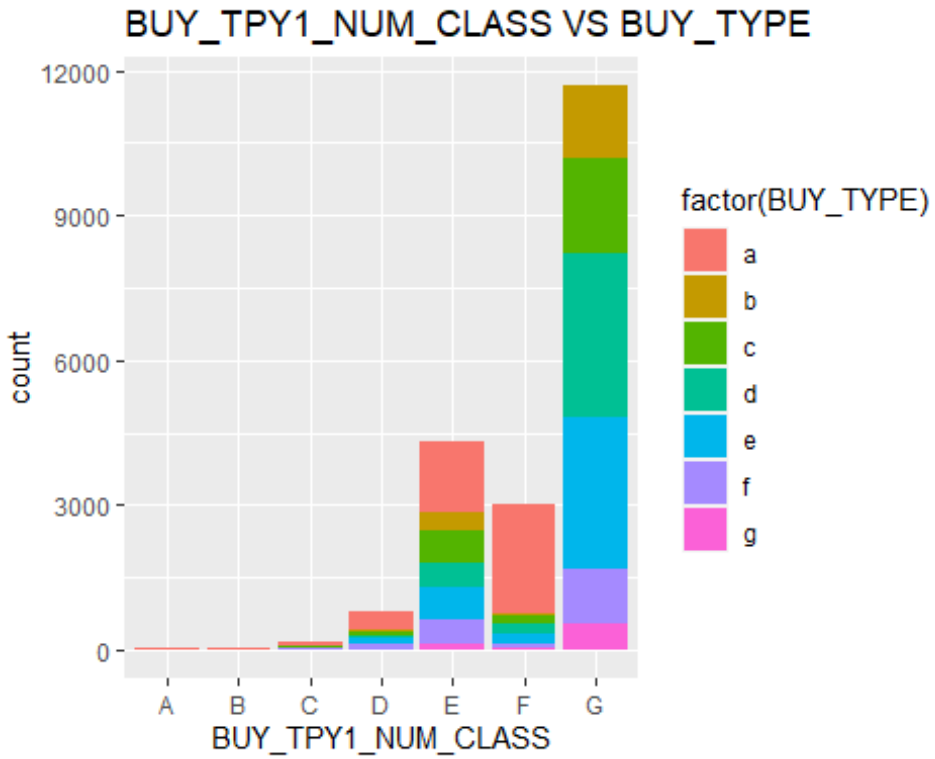

OCCUPATION VS BUY_TYPE
count
6000
4000
2000
0
OCCUPATION
factor(BUY_TYPE)
a
b
c
d
e
f
g

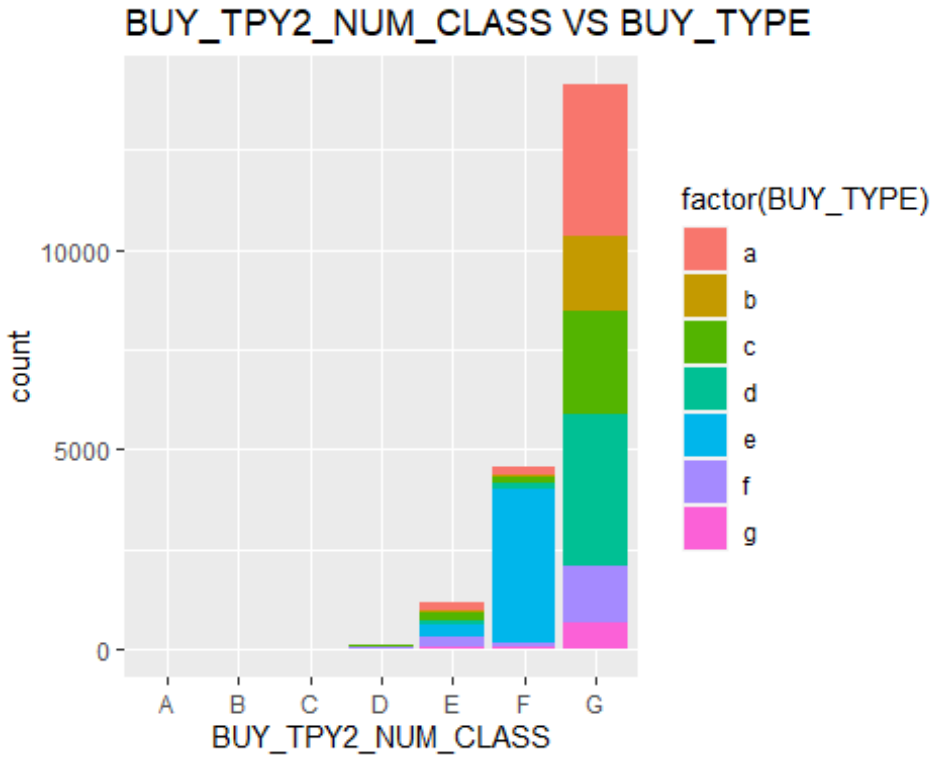

CHILD_NUM VS BUY_TYPE
count
15000
10000
5000
0
0
2
4
6
8
CHILD_NUM
factor(BUY_TYPE)
a
b
c
d
e
f
g

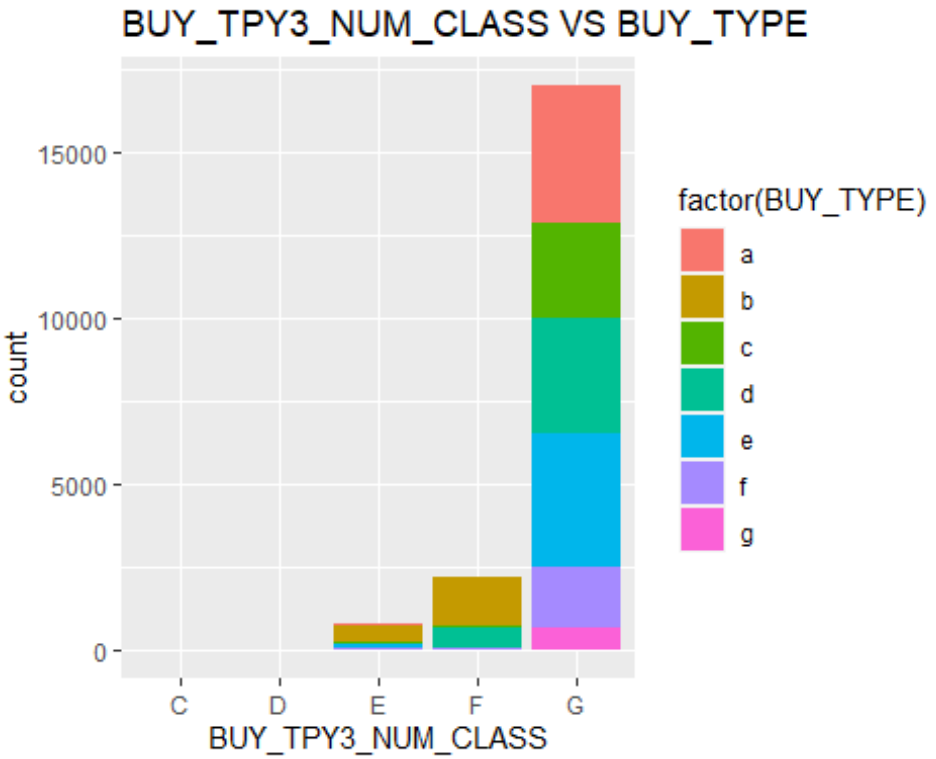

MARRIAGE VS BUY_TYPE
count
10000
7500
5000
2500
0
a
b
c
d
e
f
MARRIAGE
factor(BUY_TYPE)
a
b
c
d
e
f
g

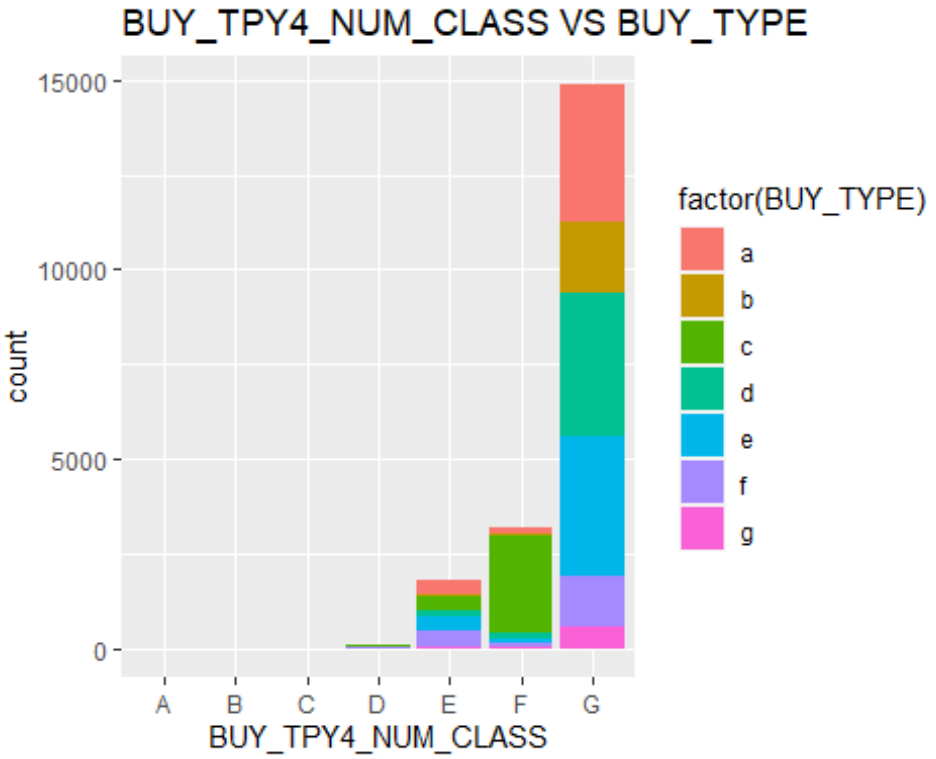

BEHAVIOR_1 VS BUY_TYPE
20000
15000
10000
5000
0
count
a
b
c
BEHAVIOR_1
factor(BUY_TYPE)
a
b
c
d
e
f
g

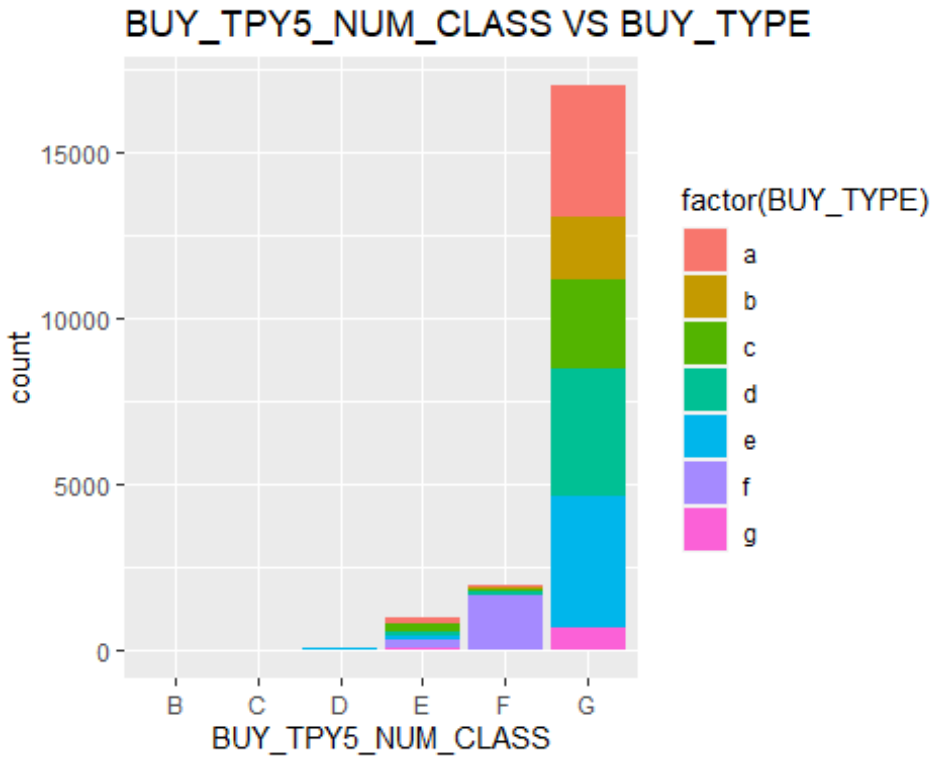

BEHAVIOR_2 VS BUY_TYPE
20000
15000
10000
5000
0
count
a
b
BEHAVIOR_2
factor(BUY_TYPE)
a
b
c
d
e
f
g

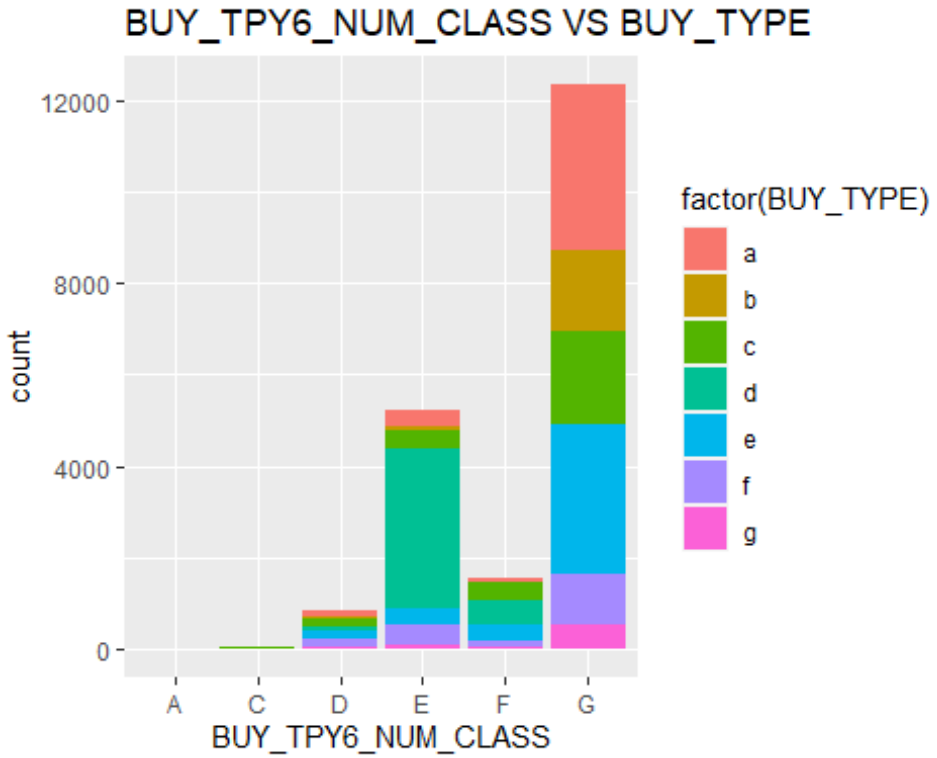

BUY_TPY1_NUM_CLASS VS BUY_TYPE
12000
9000
6000
3000
0
count
A
B
C
D
E
F
G
BUY_TPY1_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

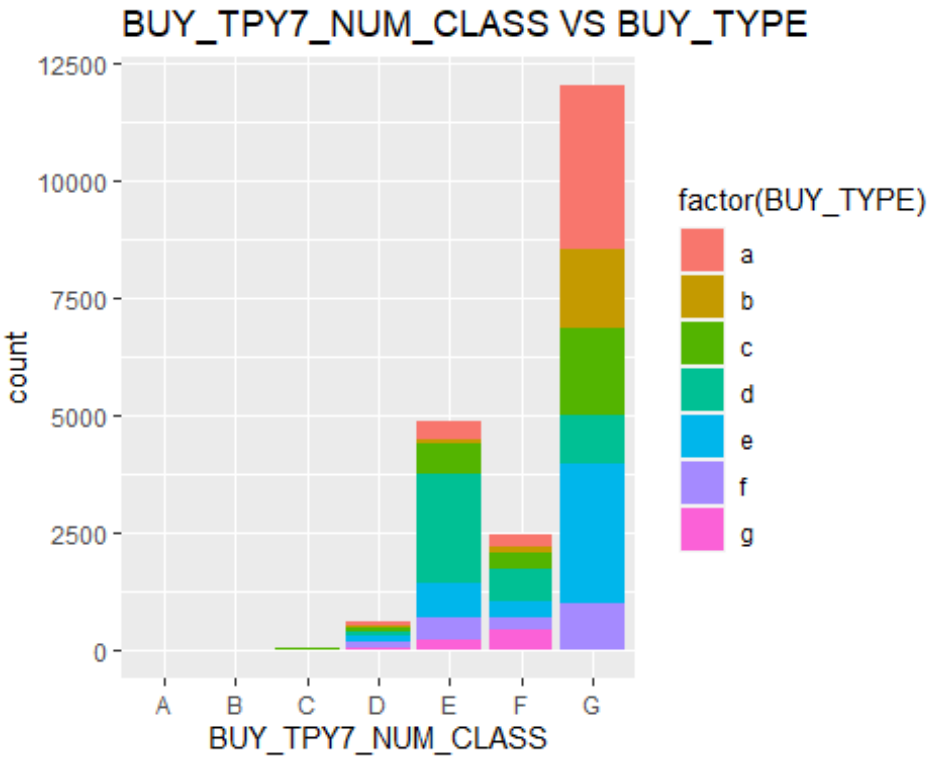

BUY_TPY2_NUM_CLASS VS BUY_TYPE
10000
5000
0
count
A
B
C
D
E
F
G
BUY_TPY2_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY3_NUM_CLASS VS BUY_TYPE
15000
10000
5000
0
count
C
D
E
F
G
BUY_TPY3_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

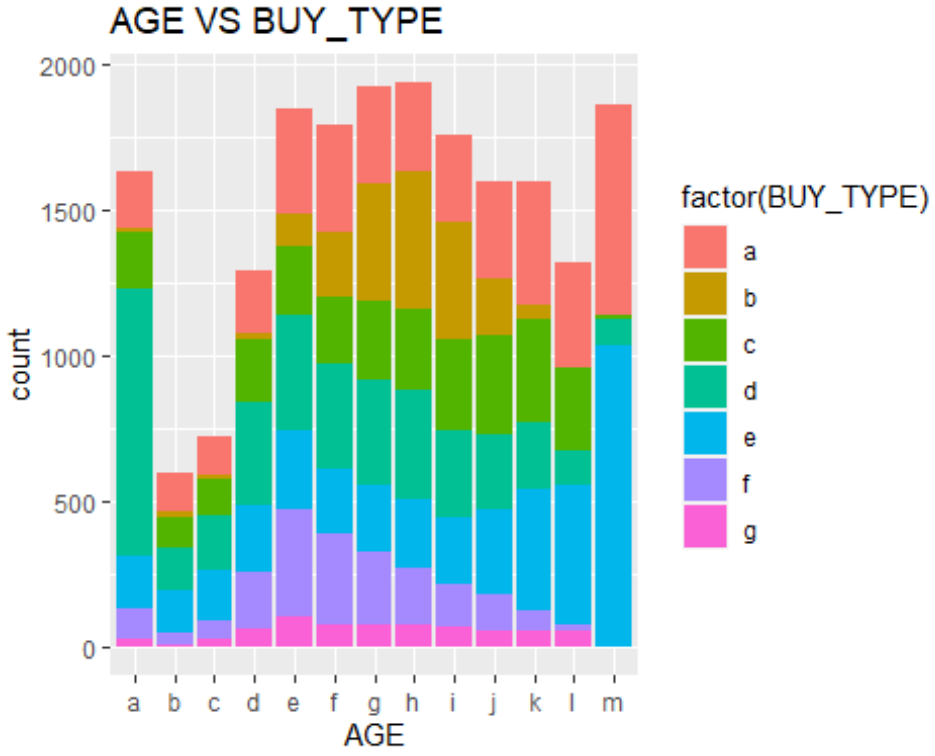

BUY_TPY4_NUM_CLASS VS BUY_TYPE
15000
10000
5000
0
count
A
B
C
D
E
F
G
BUY_TPY4_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

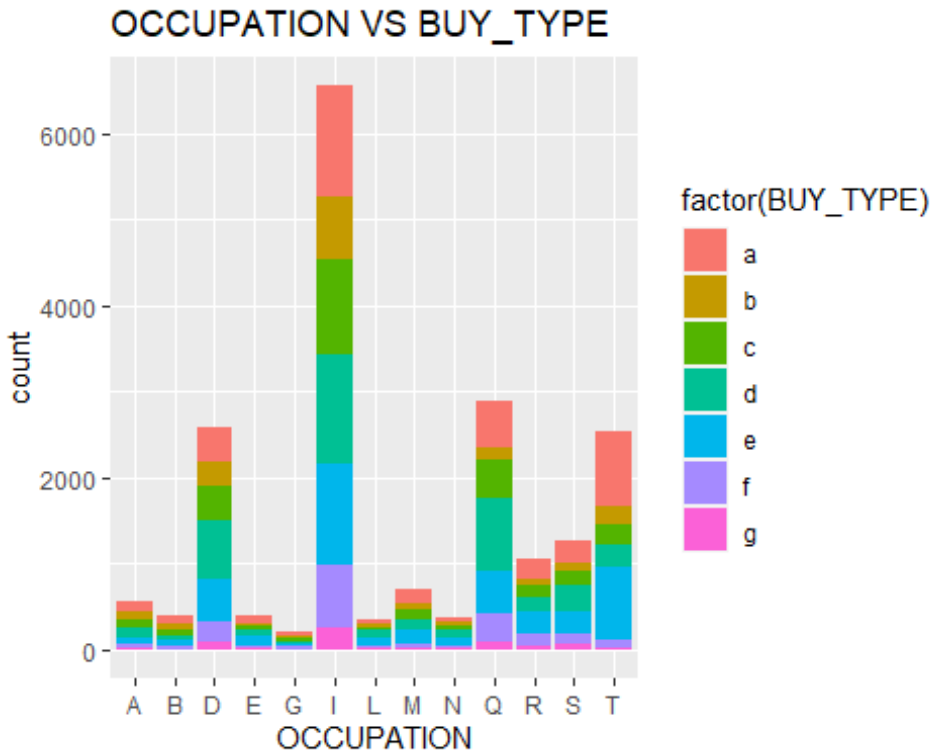

BUY_TPY5_NUM_CLASS VS BUY_TYPE
count
15000
10000
5000
0
B
C
D
E
F
G
BUY_TPY5_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

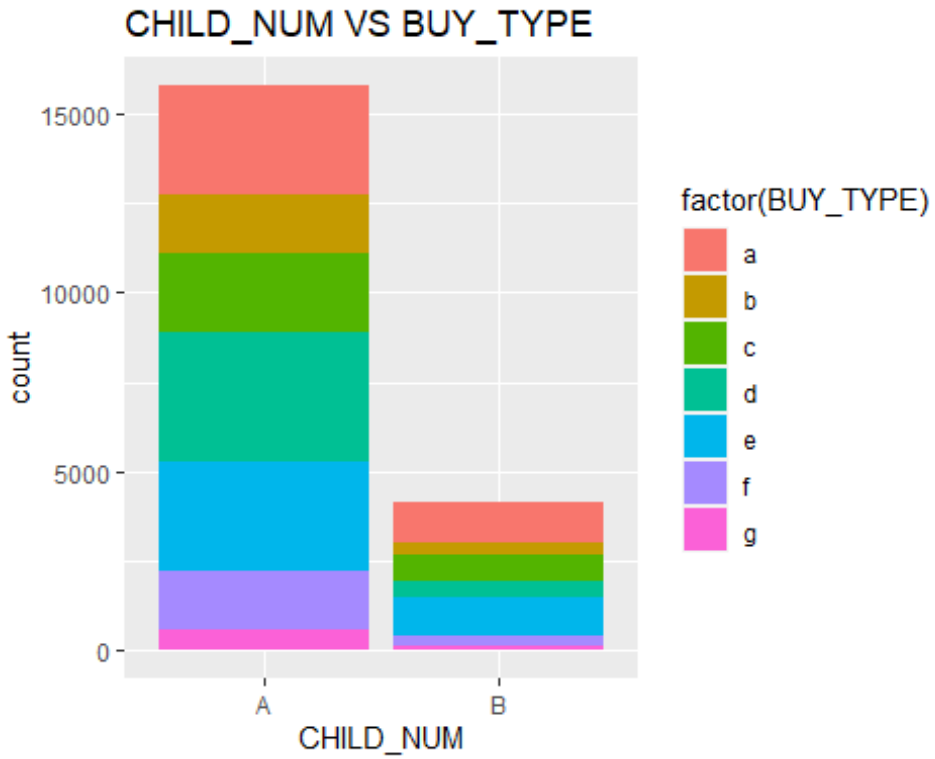

BUY_TPY6_NUM_CLASS VS BUY_TYPE
count
12000
8000
4000
0
A
C
D
E
F
G
BUY_TPY6_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

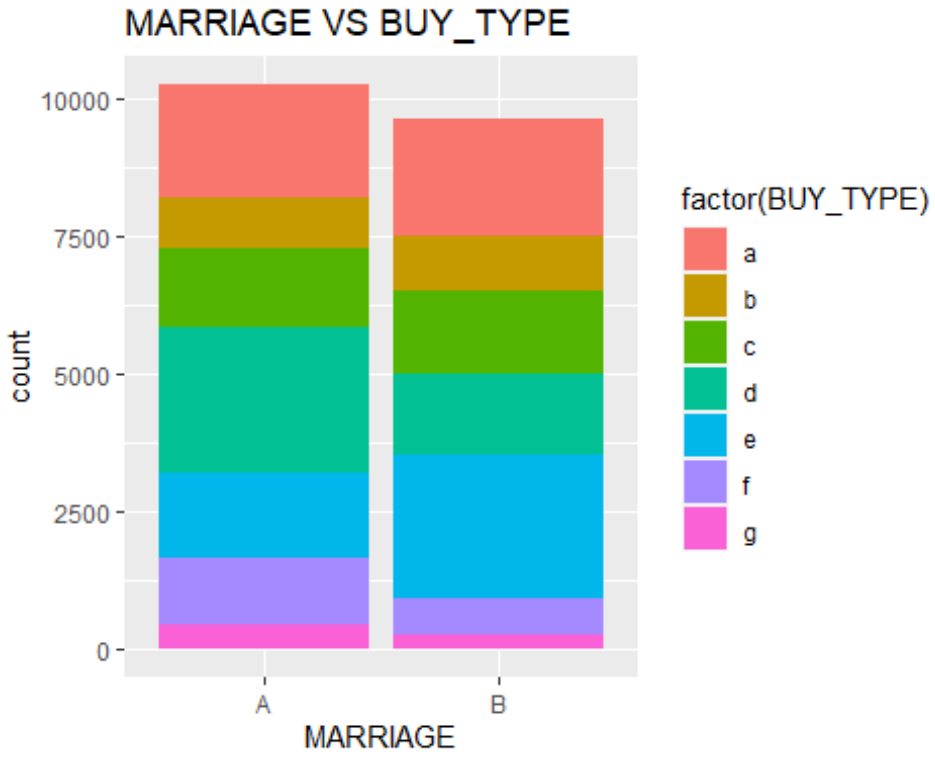

BUY_TPY7_NUM_CLASS VS BUY_TYPE
12500
10000
7500
5000
2500
0
count
A
B
C
D
E
F
G
BUY_TPY7_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

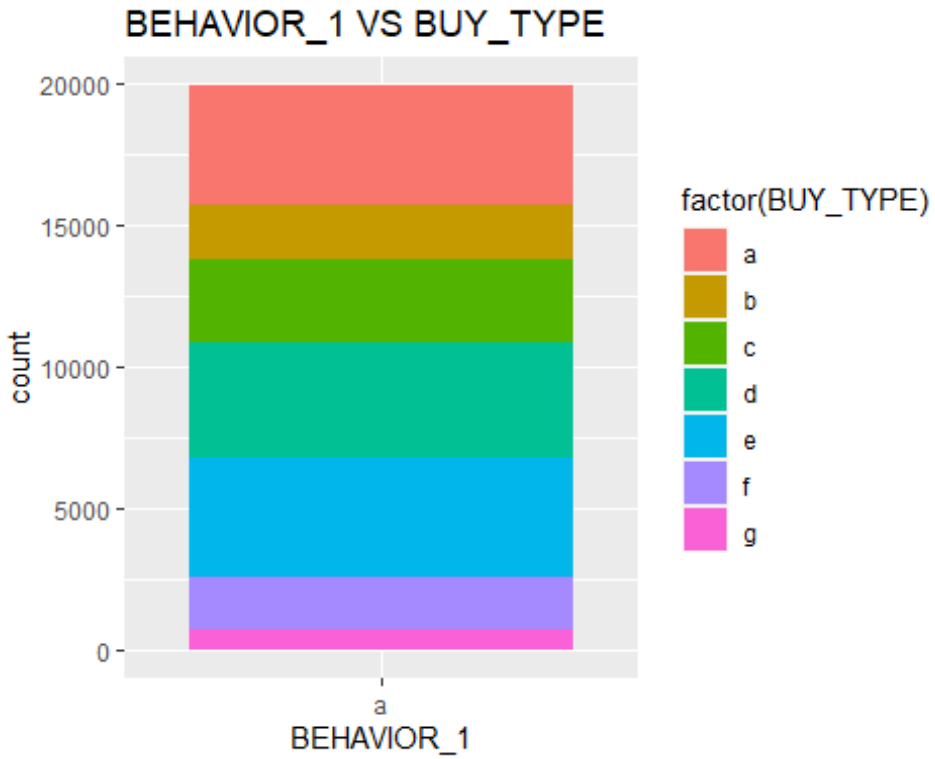
附錄 D（合併後雙變量圖表）

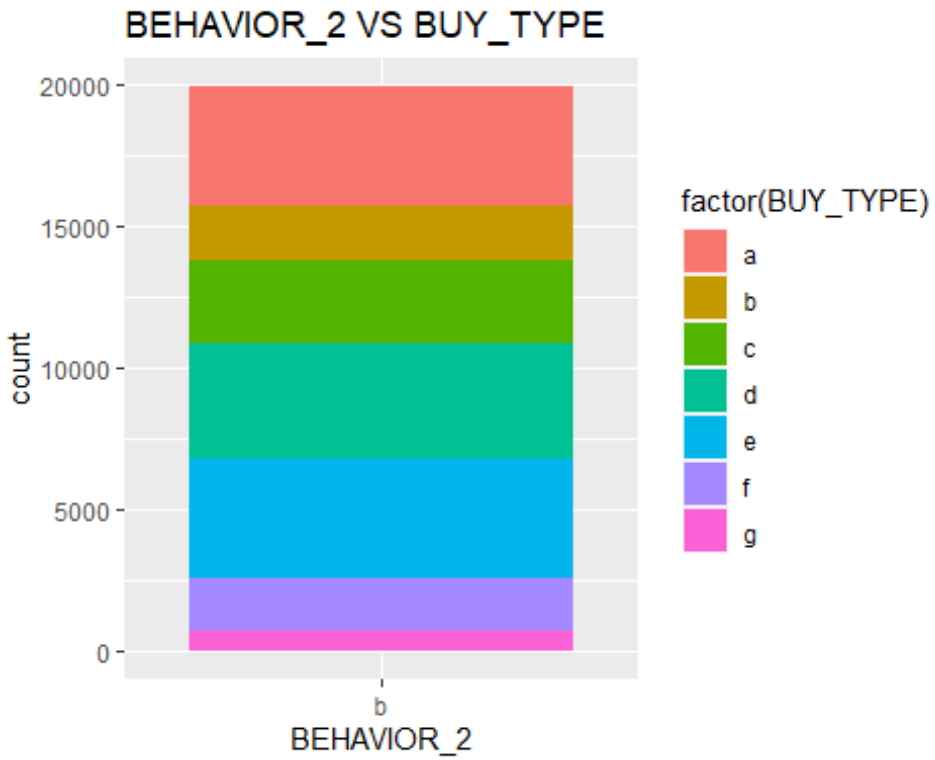


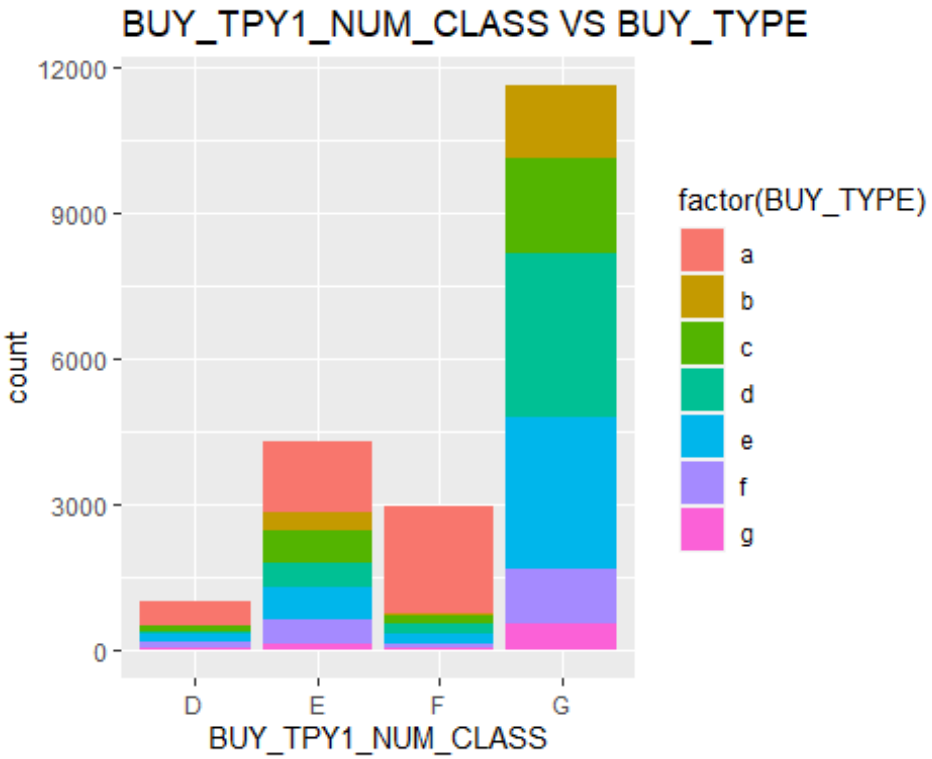


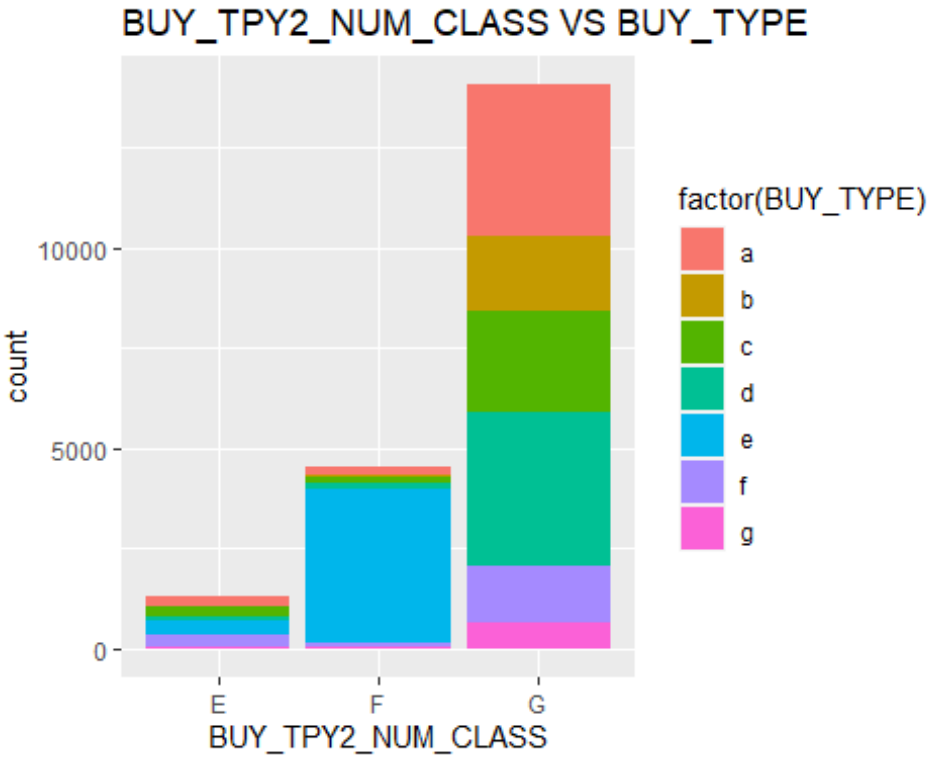

CHILD_NUM VS BUY_TYPE
15000
10000
5000
0
count
A
B
CHILD_NUM
factor(BUY_TYPE)
a
b
c
d
e
f
g

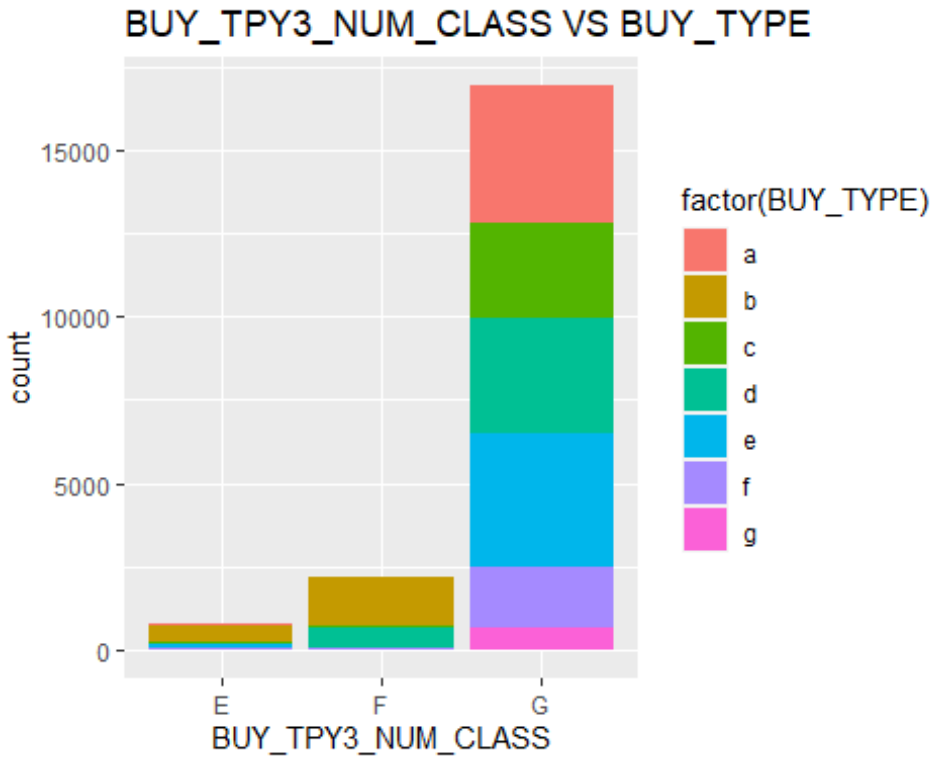

MARRIAGE VS BUY_TYPE
10000
7500
5000
2500
0
count
A
B
MARRIAGE
factor(BUY_TYPE)
a
b
c
d
e
f
g

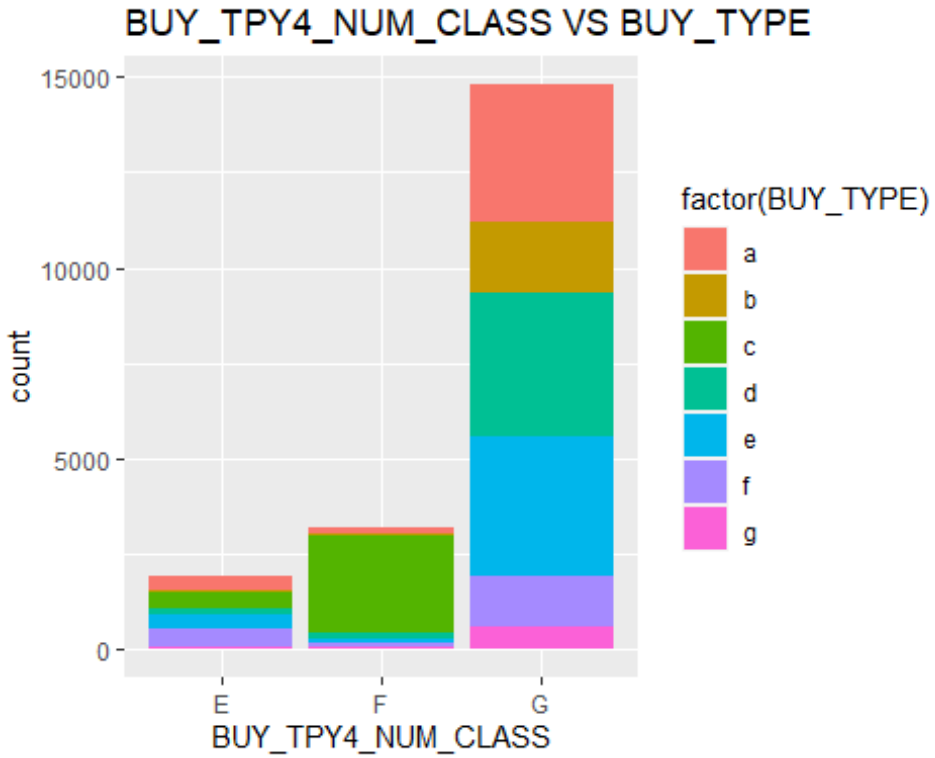

BEHAVIOR_1 VS BUY_TYPE
20000
15000
10000
5000
0
count
a
BEHAVIOR_1
factor(BUY_TYPE)
a
b
c
d
e
f
g

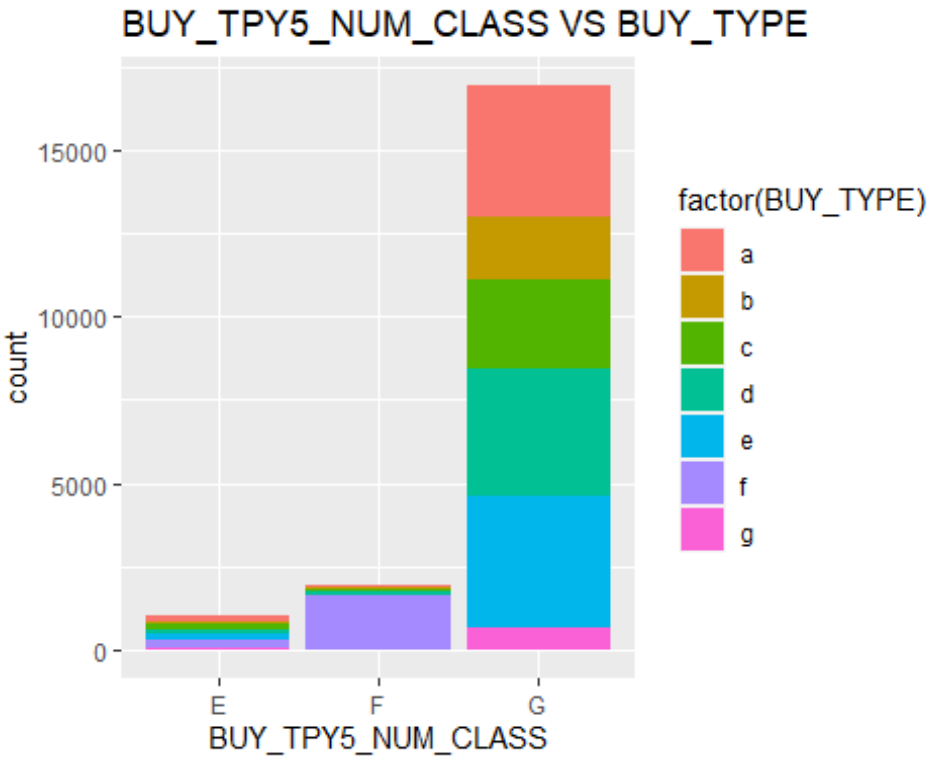

BEHAVIOR_2 VS BUY_TYPE
20000
15000
10000
5000
0
count
b
BEHAVIOR_2
factor(BUY_TYPE)
a
b
c
d
e
f
g

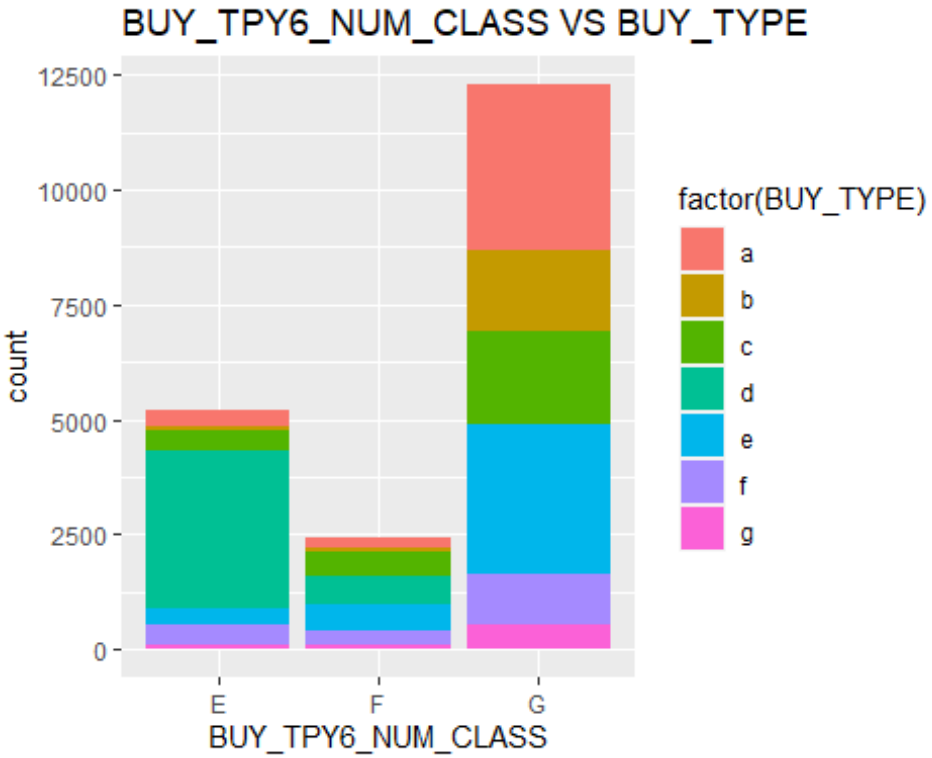

BUY_TPY1_NUM_CLASS VS BUY_TYPE
12000
9000
6000
3000
0
count
D
E
F
G
BUY_TPY1_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g

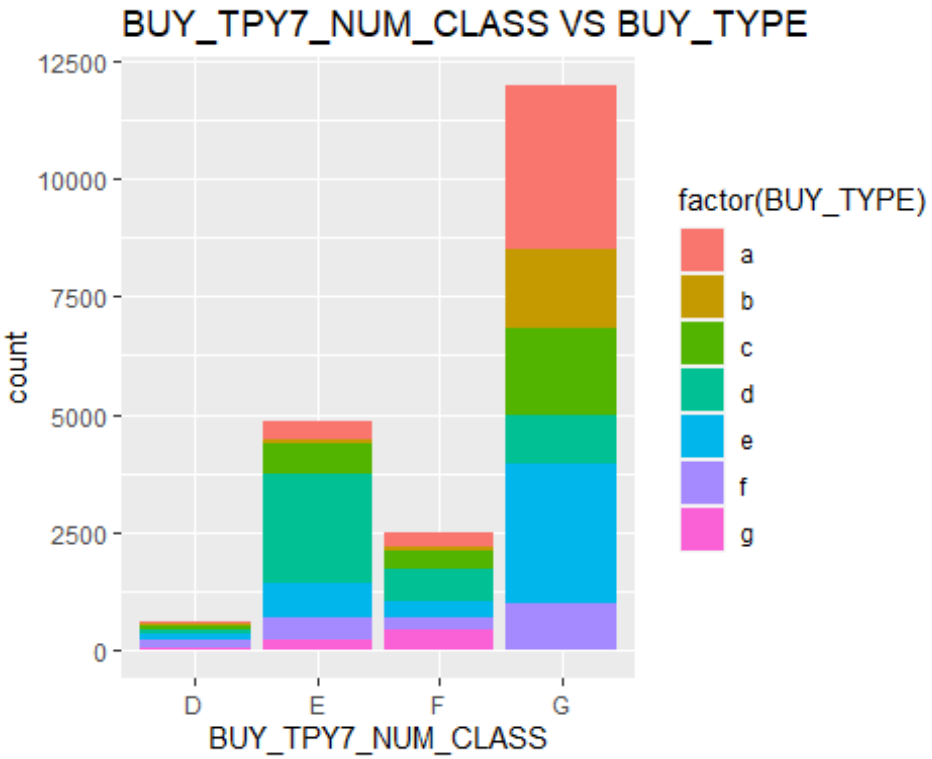

BUY_TPY2_NUM_CLASS VS BUY_TYPE
10000
5000
0
count
E
F
G
BUY_TPY2_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY3_NUM_CLASS VS BUY_TYPE
15000
10000
5000
0
count
E
F
G
BUY_TPY3_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY4_NUM_CLASS VS BUY_TYPE
15000
10000
5000
0
count
E
F
G
BUY_TPY4_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY5_NUM_CLASS VS BUY_TYPE
15000
10000
5000
0
count
E
F
G
BUY_TPY5_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY6_NUM_CLASS VS BUY_TYPE
12500
10000
7500
5000
2500
0
count
E
F
G
BUY_TPY6_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g


BUY_TPY7_NUM_CLASS VS BUY_TYPE
12500
10000
7500
5000
2500
0
count
D
E
F
G
BUY_TPY7_NUM_CLASS
factor(BUY_TYPE)
a
b
c
d
e
f
g