502 M5 UE Wang

2022-11-27

Introduction to Data Mining:

Chapter 5:

Q6:

a. R = 3(d square) - 2(d+1 square) +1 = 602.
b. ID 6 and 9. Thus: 4.
c. (6 * 5 * 4) / (3 * 2 * 1) = 20
d. {Bread, Butter}.
e. (Beer, Cookies) or (Bread, Butter).

Q8:

a. {1, 2, 3, 4},{1, 2, 3, 5},{1, 2, 3, 6}. {1, 2, 4, 5},{1, 2, 4, 6},{1, 2, 5, 6}. {1, 3, 4, 5},{1, 3, 4, 6},{2, 3, 4, 5}. {2, 3, 4, 6},{2, 3, 5, 6}.
b. {1,2,3,4}, {1,2,3,5}, {1,2,4,5}, {2,3,4,5}, {2,3,4,6}.
c. {1,2,3,4}

Q15:

a. Data set (e)
b. Data set (d)
c. Data set (e).
d. Data set (b).
e. Data set (e).

DS Python and R:

Q11:

```
churn <- read.csv(file = "/Users/UE/Desktop/M5 Dataset/Churn_Training_File")
head(churn,5)
```

```
##   State Account.Length Area.Code    Phone Intl.Plan VMail.Plan VMail.Message
## 1    KS            128       415 382-4657        no        yes            25
## 2    OH            107       415 371-7191        no        yes            26
## 3    NJ            137       415 358-1921        no         no             0
## 4    OH             84       408 375-9999       yes         no             0
## 5    OK             75       415 330-6626       yes         no             0
##   Day.Mins Day.Calls Day.Charge Eve.Mins Eve.Calls Eve.Charge Night.Mins
## 1    265.1       110      45.07    197.4        99      16.78      244.7
## 2    161.6       123      27.47    195.5       103      16.62      254.4
## 3    243.4       114      41.38    121.2       110      10.30      162.6
## 4    299.4        71      50.90     61.9        88       5.26      196.9
## 5    166.7       113      28.34    148.3       122      12.61      186.9
##   Night.Calls Night.Charge Intl.Mins Intl.Calls Intl.Charge CustServ.Calls
## 1          91        11.01      10.0          3        2.70              1
## 2         103        11.45      13.7          3        3.70              1
## 3         104         7.32      12.2          5        3.29              0
## 4          89         8.86       6.6          7        1.78              2
## 5         121         8.41      10.1          3        2.73              3
##   Churn
## 1 False
## 2 False
## 3 False
## 4 False
## 5 False
```

```
min.churn <- subset(churn,
                    select = c("Churn",
                               "Intl.Plan",
                               "VMail.Plan",
                               "CustServ.Calls"))
```

```
min.churn$CustServ.Calls <- as.factor(min.churn$CustServ.Calls)
```

```
t1 <- table(min.churn$Intl.Plan)
t1
```

```
##
##   no  yes
## 2705  295
```

```
t11 <- rbind(t1, round(prop.table(t1), 4))
t11
```

```
##           no      yes
## t1 2705.0000 295.0000
##       0.9017   0.0983
```

```
colnames(t11) <- c("Intl.Plan = no", "Intl.Plan = yes")
rownames(t11) <- c("Count", "Proportion")
t11
```

```
##            Intl.Plan = no Intl.Plan = yes
## Count           2705.0000        295.0000
## Proportion         0.9017          0.0983
```

Intl.Plan’s baseline distribution is mainly on Intl.Plan = No, about 90%

```
t2 <- table(min.churn$VMail.Plan)
t2
```

```
##
##   no  yes
## 2170  830
```

```
t22 <- rbind(t2, round(prop.table(t2), 4))
t22
```

```
##           no      yes
## t2 2170.0000 830.0000
##       0.7233   0.2767
```

```
colnames(t22) <- c("VMail.Plan = no", "VMail.Plan = yes")
rownames(t22) <- c("Count", "Proportion")
t22
```

```
##            VMail.Plan = no VMail.Plan = yes
## Count            2170.0000         830.0000
## Proportion          0.7233           0.2767
```

VMail.Plan’s baseline distribution is mainly on VMail.Plan = No, about 72%

```
t3 <- table(min.churn$CustServ.Calls)
t3
```

```
##
##    0    1    2    3    4    5    6    7    8    9
##  626 1068  679  383  149   61   22    8    2    2
```

```
t33 <- rbind(t3, round(prop.table(t3), 4))
t33
```

```
##           0        1        2        3        4       5       6      7      8
## t3 626.0000 1068.000 679.0000 383.0000 149.0000 61.0000 22.0000 8.0000 2e+00
##      0.2087    0.356   0.2263   0.1277   0.0497  0.0203  0.0073 0.0027 7e-04
##        9
## t3 2e+00
##    7e-04
```

```
rownames(t33) <- c("Count", "Proportion")
t33
```

```
##                   0        1        2        3        4       5       6      7
## Count      626.0000 1068.000 679.0000 383.0000 149.0000 61.0000 22.0000 8.0000
## Proportion   0.2087    0.356   0.2263   0.1277   0.0497  0.0203  0.0073 0.0027
##                8     9
## Count      2e+00 2e+00
## Proportion 7e-04 7e-04
```

CustServ.Calls’s baseline distribution is a bellshape that skew to the left. baseline is mainly on point 4, about 50%

```
t4 <- table(min.churn$Churn)
t4
```

```
##
## False  True
##  2564   436
```

```
t44 <- rbind(t4, round(prop.table(t4), 4))
t44
```

```
##        False     True
## t4 2564.0000 436.0000
##       0.8547   0.1453
```

```
colnames(t44) <- c("Churn = no", "Churn = yes")
rownames(t44) <- c("Count", "Proportion")
t44
```

```
##            Churn = no Churn = yes
## Count       2564.0000    436.0000
## Proportion     0.8547      0.1453
```

#Churn’s baseline distribution is mainly on Chun = No, about 85%

Q13:

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

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
all.rules <- apriori(data = min.churn,
                     parameter = list(target = "rules",
                                      supp = 0.01,
                                      minlen = 2,
                                      maxlen = 2,
                                      conf = 0.4))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen target  ext
##       2  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16 item(s), 3000 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2
```

```
## Warning in apriori(data = min.churn, parameter = list(target = "rules", : Mining
## stopped (maxlen reached). Only patterns up to a length of 2 returned!
```

```
##  done [0.00s].
## writing ... [32 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(all.rules, by = "lift", n = 5))
```

```
##     lhs                   rhs              support    confidence coverage
## [1] {CustServ.Calls=5} => {Churn=True}     0.01200000 0.5901639  0.02033333
## [2] {CustServ.Calls=4} => {Churn=True}     0.02266667 0.4563758  0.04966667
## [3] {Intl.Plan=yes}    => {Churn=True}     0.04233333 0.4305085  0.09833333
## [4] {Churn=True}       => {VMail.Plan=no} 0.12066667 0.8302752  0.14533333
## [5] {CustServ.Calls=3} => {VMail.Plan=no} 0.09933333 0.7780679  0.12766667
##     lift     count
## [1] 4.060761  36
## [2] 3.140201  68
## [3] 2.962214 127
## [4] 1.147846 362
## [5] 1.075670 298
```

Q14:

```
all.rules.ant.df <- as(as(attr(all.rules, "lhs"), "transactions"), "data.frame")
head(all.rules.ant.df, 15)
```

```
##                 items
## 1  {CustServ.Calls=5}
## 2  {CustServ.Calls=5}
## 3  {CustServ.Calls=5}
## 4  {CustServ.Calls=4}
## 5  {CustServ.Calls=4}
## 6  {CustServ.Calls=4}
## 7  {CustServ.Calls=4}
## 8     {Intl.Plan=yes}
## 9     {Intl.Plan=yes}
## 10    {Intl.Plan=yes}
## 11 {CustServ.Calls=3}
## 12 {CustServ.Calls=3}
## 13 {CustServ.Calls=3}
## 14       {Churn=True}
## 15       {Churn=True}
```

```
t1 <- all.rules.ant.df$items == "{Churn=True}"
t2 <- all.rules.ant.df$items == "{Churn=False}"
non.churn.ant <- abs(t1+t2-1)
non.churn.ant
```

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

```
good.rules <- all.rules[non.churn.ant == 1]
```

```
inspect(head(good.rules, by = "lift", n = 5))
```

```
##     lhs                   rhs              support    confidence coverage
## [1] {CustServ.Calls=5} => {Churn=True}     0.01200000 0.5901639  0.02033333
## [2] {CustServ.Calls=4} => {Churn=True}     0.02266667 0.4563758  0.04966667
## [3] {Intl.Plan=yes}    => {Churn=True}     0.04233333 0.4305085  0.09833333
## [4] {CustServ.Calls=3} => {VMail.Plan=no} 0.09933333 0.7780679  0.12766667
## [5] {VMail.Plan=yes}   => {Churn=False}    0.25200000 0.9108434  0.27666667
##     lift     count
## [1] 4.060761  36
## [2] 3.140201  68
## [3] 2.962214 127
## [4] 1.075670 298
## [5] 1.065729 756
```

```
t.csc.churn <- table(min.churn$Churn, min.churn$CustServ.Calls)
t.csc.churn
```

```
##
##           0   1   2   3   4   5   6   7   8   9
##   False 540 960 602 343  81  25   8   4   1   0
##   True   86 108  77  40  68  36  14   4   1   2
```

```
colnames(t.csc.churn) <- c("CSC = 0", "CSC = 1", "CSC = 2", "CSC = 3", "CSC = 4",
                           "CSC = 5", "CSC = 6", "CSC = 7", "CSC = 8", "CSC = 9")
rownames(t.csc.churn) <- c("Churn = False", "Churn = True")
addmargins(A = t.csc.churn, FUN = list(Total = sum), quiet = TRUE)
```

```
##
##                CSC = 0 CSC = 1 CSC = 2 CSC = 3 CSC = 4 CSC = 5 CSC = 6 CSC = 7
##   Churn = False     540     960     602     343      81      25       8       4
##   Churn = True       86     108      77      40      68      36      14       4
##   Total             626    1068     679     383     149      61      22       8
##
##                CSC = 8 CSC = 9 Total
##   Churn = False       1       0  2564
##   Churn = True        1       2   436
##   Total               2       2  3000
```

Q15:

```
rules.confdiff <- apriori(data = min.churn, parameter =
    list(arem = "diff", aval = TRUE,
           minval = 0.4, supp = 0.01, target = "rules", conf
           = 0.05, minlen = 2, maxlen = 2))
```

```
## Warning: Column(s) 1, 2, 3 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem aval originalSupport maxtime support minlen maxlen
##        0.05    0.4    1 diff TRUE            TRUE       5    0.01      2      2
##  target  ext
##   rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 30
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[16 item(s), 3000 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2
```

```
## Warning in apriori(data = min.churn, parameter = list(arem = "diff", aval =
## TRUE, : Mining stopped (maxlen reached). Only patterns up to a length of 2
## returned!
```

```
##  done [0.00s].
## writing ... [1 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(head(rules.confdiff, by = "lift", n = 10))
```

```
##     lhs                   rhs          support confidence diff      coverage
## [1] {CustServ.Calls=5} => {Churn=True} 0.012   0.5901639  0.4448306 0.02033333
##     lift     count
## [1] 4.060761 36
```