**1.**

# Loading necessary library library(pROC)

# Defining the data data <- data.frame(

Tuple = 1:10,  
Class = c("p", "n", "p", "p", "n", "p", "n", "n", "n", "p"),  
Prob = c(0.95, 0.85, 0.78, 0.66, 0.6, 0.55, 0.53, 0.52, 0.51, 0.4)

)

# Sorting data by probability data <- data[order(-data$Prob), ]

# Initializing empty columns data$TP <- 0  
data$FN <- 0  
data$FP <- 0

data$TN <- 0 data$TPR <- 0 data$FPR <- 0

# Calculating TP, FN, FP, TN, TPR, FPR for each threshold for (i in 1:nrow(data)) {

threshold <- data$Prob[i]  
data$TP[i] <- sum(data$Class == "p" & data$Prob >= threshold) data$FN[i] <- sum(data$Class == "p" & data$Prob < threshold) data$FP[i] <- sum(data$Class == "n" & data$Prob >= threshold) data$TN[i] <- sum(data$Class == "n" & data$Prob < threshold) data$TPR[i] <- data$TP[i] / (data$TP[i] + data$FN[i]) data$FPR[i] <- data$FP[i] / (data$FP[i] + data$TN[i])

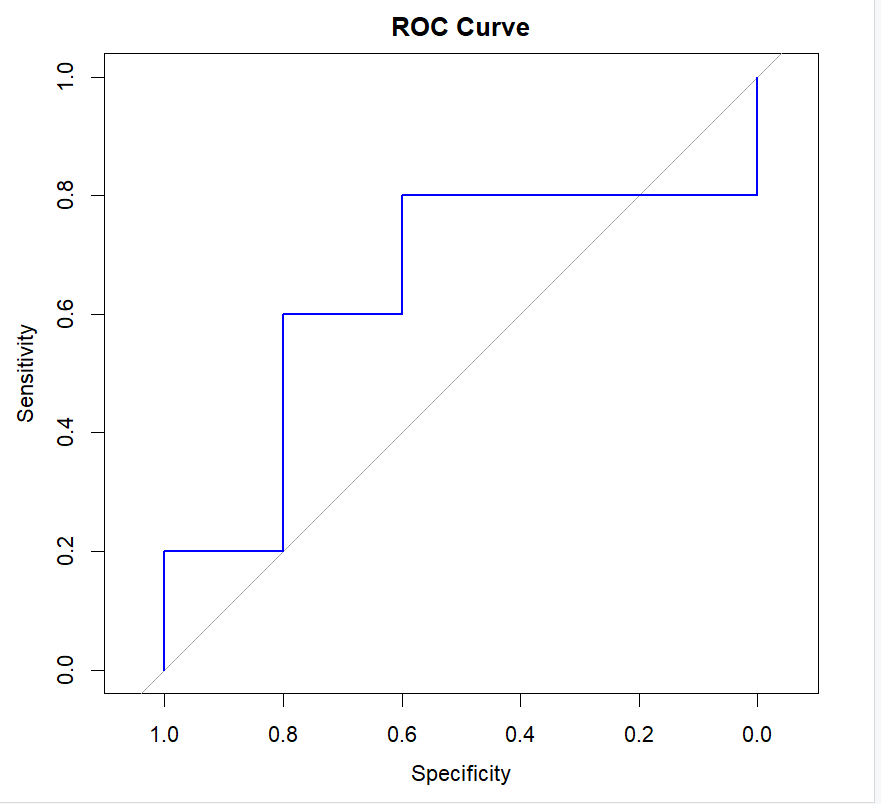
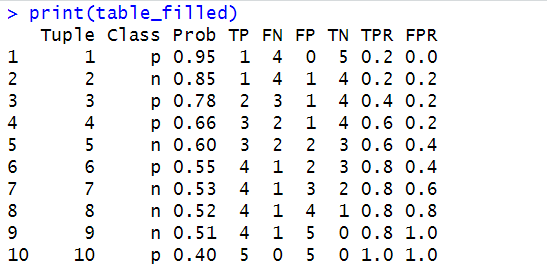
}

# Drawing ROC curve  
roc\_curve <- roc(data$Class, data$Prob) plot(roc\_curve, main = "ROC Curve", col = "blue")

# Filling in the table with calculated values  
table\_filled <- cbind(data[, c("Tuple", "Class", "Prob")], data[, c("TP", "FN", "FP", "TN", "TPR", "FPR")])  
rownames(table\_filled) <- NULL

# Printing the filled table print(table\_filled)

*OUTPUT :*



**2.**

# Calculating minimum support  
min\_support <- ceiling(0.2 \* length(transactions)) / length(transactions)

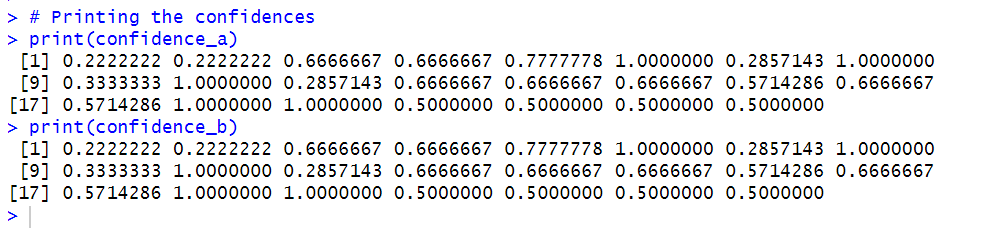
# Finding frequent itemsets with minimum support  
frequent\_itemsets <- eclat(trans, parameter = list(support = min\_support))

# Generating association rules  
rules <- apriori(trans, parameter = list(supp = min\_support, conf = 0, target = "rules"))

# Calculating confidences for the rules  
confidence\_a <- interestMeasure(rules, measure = "confidence", subset = "lhs %in% c('A', 'B') & rhs == 'E'")  
confidence\_b <- interestMeasure(rules, measure = "confidence", subset = "lhs == 'A' & rhs %in% c('B', 'E')")

# Printing the confidences print(confidence\_a) print(confidence\_b)

*OUTPUT :*



**3.**

Step 1: Calculating the support of single items

●  {A,} (support: 6)

●  {B,} (support: 7)

●  {E} (support: 2)

●  {D} (support: 2)

●  {C} (support: 5)

●  {C,} (support: 1)

●  {} (support: 1)

Step 2: Keeping only the frequent items

{C,} is pruned because its support count (1) is less than the minimum support count (2) and {} was pruned because its support count (1) is less than the minimum support count (2)

●  {A,} (support: 6)

●  {B,} (support: 7)

●  {E} (support: 2)

●  {D} (support: 2)

●  {C} (support: 5)

Step 3: Joining frequent itemsets to create candidates of size 2

●  {A,} and {B,} are joined to obtain {A, B}

●  {A,} and {E} are joined to obtain {A, E}

●  {A,} and {D} are joined to obtain {A,D}

●  {A,} and {C} are joined to obtain {A, C}

●  {B,} and {E} are joined to obtain {B, E}

●  {B,} and {D} are joined to obtain {B, D}

●  {B,} and {C} are joined to obtain {B,C}

●  {D} and {D} are joined to obtain {D, E}

●  {C} and {E} are joined to obtain {C, E}

●  {C} and {D} are joined to obtain {C, D}

Step 4: Calculating the support of candidate itemsets

●  {A, B,} (support: 4)

●  {A, E} (support: 2)

●  {A, D} (support: 1)

●  {A, C} (support: 3)

●  {B, E} (support: 2)

●  {B, D} (support: 2)

●  {B, C} (support: 3)

●  {D, E} (support: 0)

●  {C, E} (support: 0)

●  {C, D} (support: 0)

Step 5: Keeping only the candidate itemsets that are frequent

{A,, D} will be pruned because its support count (1) is less than the minimum support count (2) ; {D, E} will be pruned because its support count (0) is less than the minimum support count (2) ; {C, E} will be pruned because its support count (0) is less than the minimum support count (2) and {C, D} will be pruned because its support count (0) is less than the minimum support count (2)

●  {A, B,} (support: 4)

●  {A, E} (support: 2)

●  {A, C} (support: 3)

●  {B,E} (support: 2)

●  {B, D} (support: 2)

●  {B,C} (support: 3)

Step 6: Joining frequent itemsets to create candidates of size 3

●  {A, B} and {A,E} are joined to obtain {A, B, E}

●  {A,B} and {A,C} are joined to obtain {A,B,C}

●  {A, C} and {A,C} are joined to obtain {A,C, E}

{A,C, E} is discarded because it contains a subset that is not frequent: {C, E}

●  {B,D} and {B,E} are joined to obtain {B,D, E}

{B,D, E} is discarded because it contains a subset that is not frequent: {D, E}

●  {B,C} and {B,E} are joined to obtain {B,C, E}

{B, C, E} is discarded because it contains a subset that is not frequent: {C, E}

●  {B,C} and {B,D} are joined to obtain {B,, C, D}

{B,C, D} is discarded because it contains a subset that is not frequent: {C, D}

Step 7: Calculating the support of candidate itemsets

{A,B, E} (support: 2) {A,B, C} (support: 1)

Step 8: Keeping only the candidate itemsets that are frequent

{A,B, C} will be pruned because its support count (1) is less than the minimum support count (2)

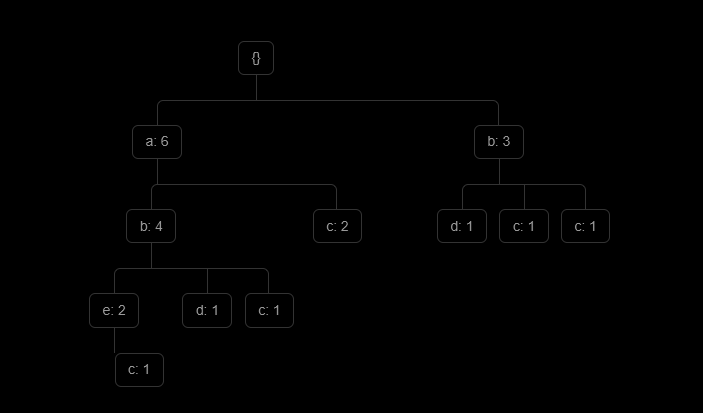
{A,B,E} (support: 2)  
Step 9: Joining frequent itemsets to create candidates of size 4 No more candidates can be generated.  
**Total number of frequent itemsets found: 12**

|  |  |
| --- | --- |
| **Itemset** | **Support Count** |
| {A} | 6 |
| {B} | 7 |
| {E} | 2 |
| {D} | 2 |
| {C} | 5 |
| {A,B} | 4 |
| {A,E} | 2 |
| {A,C} | 3 |
| {B,E} | 2 |
| {B,D} | 2 |
| {B,C} | 3 |
| {A,B,E} | 2 |

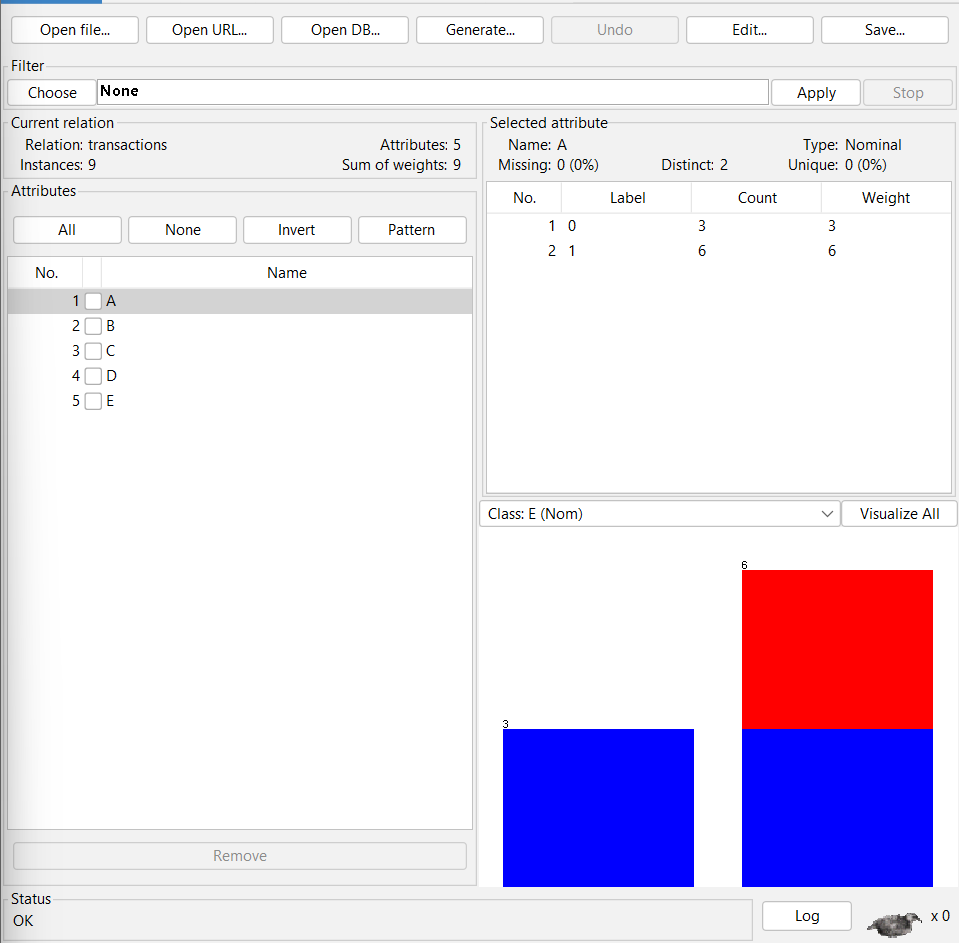
**4.**

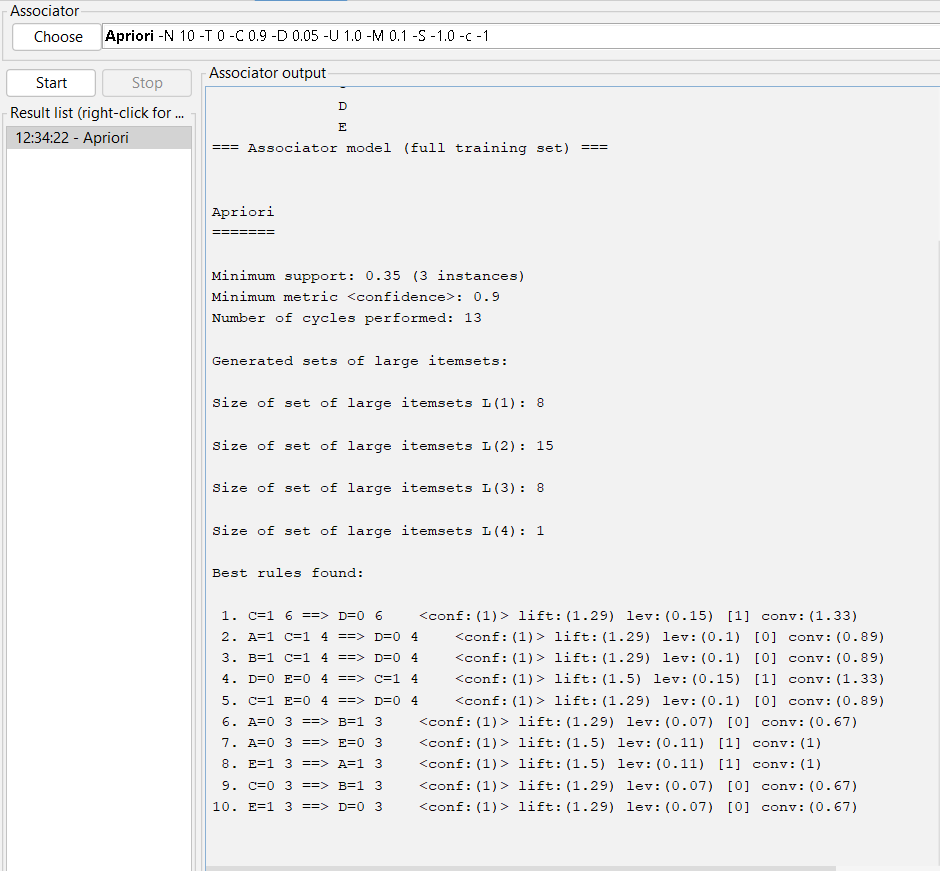
|  |  |  |
| --- | --- | --- |
| ROW | ITEMS | FREQUENT ITEMS |
| 1 | {a, b, e} | {a b e} |
| 2 | {b, d} | {b d} |
| 3 | {b,c} | {b c} |
| 4 | {a,b,d} | {a b d} |
| 5 | {a, c} | {a c} |
| 6 | {b,c} | {b c} |
| 7 | {a,c} | {a c} |
| 8 | {a,b,c,e} | {a b c e} |
| 9 | {a,b,c} | {a b c} |

page7image9802240



**5.**

page9image9593472



Therefore , the association rules found are :  
● C=1 6 ==> D=0 6 <conf:(1)> lift:(1.29) lev:(0.15) [1] conv:(1.33)

●  A=1 C=1 4 ==> D=0 4

●  B=1 C=1 4 ==> D=0 4

●  D=0 E=0 4 ==> C=1 4

●  C=1 E=0 4 ==> D=0 4

<conf:(1)> lift:(1.29) lev:(0.1) [0] conv:(0.89) <conf:(1)> lift:(1.29) lev:(0.1) [0] conv:(0.89) <conf:(1)> lift:(1.5) lev:(0.15) [1] conv:(1.33) <conf:(1)> lift:(1.29) lev:(0.1) [0] conv:(0.89)

* ●  A=0 3 ==> B=1 3

●  A=0 3 ==> E=0 3

●  E=1 3 ==> A=1 3

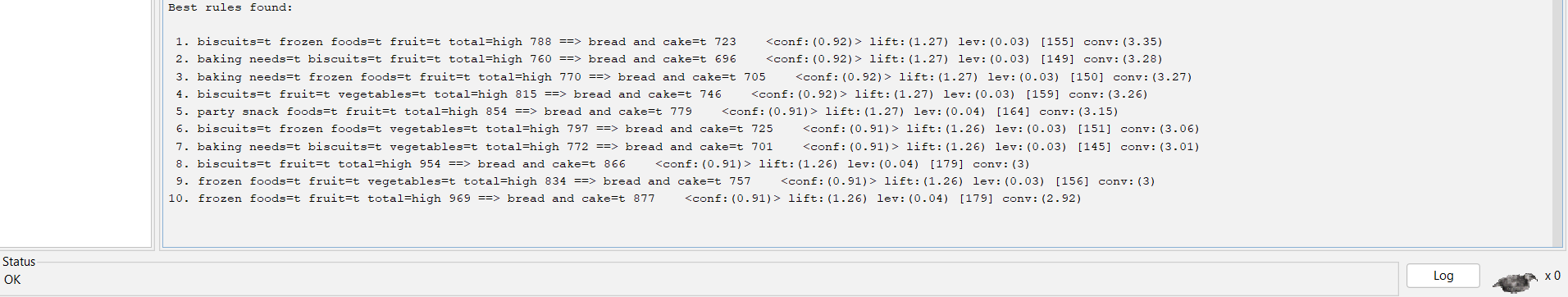
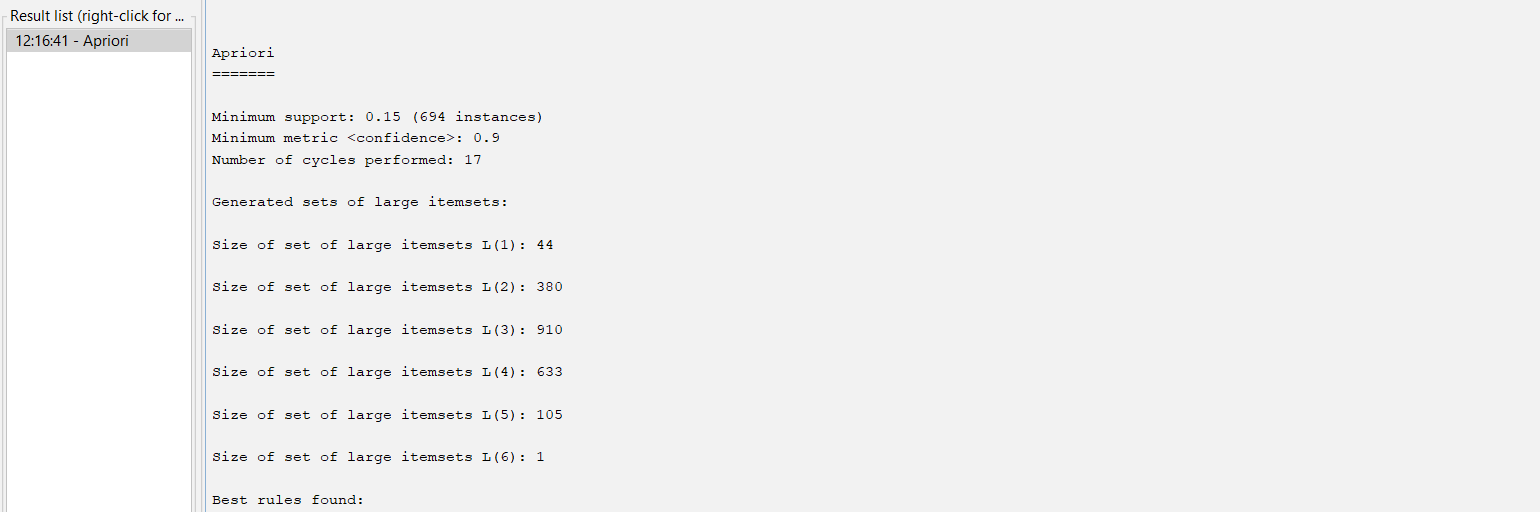
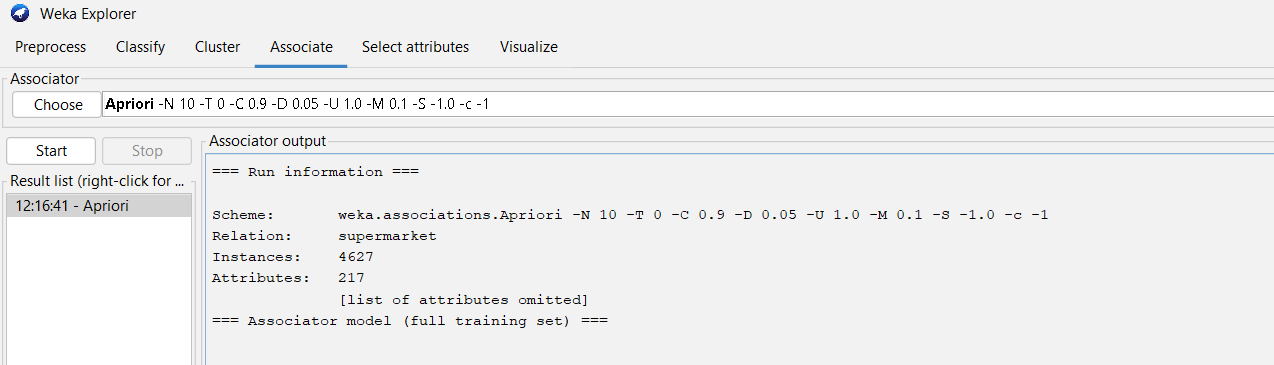
<conf:(1)> lift:(1.29) lev:(0.07) [0] conv:(0.67) <conf:(1)> lift:(1.5) lev:(0.11) [1] conv:(1) <conf:(1)> lift:(1.5) lev:(0.11) [1] conv:(1)

●  C=0 3 ==> B=1 3 <conf:(1)> lift:(1.29) lev:(0.07) [0] conv:(0.67)

●  E=1 3 ==> D=0 3 <conf:(1)> lift:(1.29) lev:(0.07) [0] conv:(0.67)

**6.**

*APRIORI ALGORITHM*

page11image9600384

*FP GROWTH ALGORITHM*

