DTI 5126: Fundamentals of Data Science

Assignment 2

Prepared by:

Mahmoud Yahia Ahmed

Part A:

1) Classification:

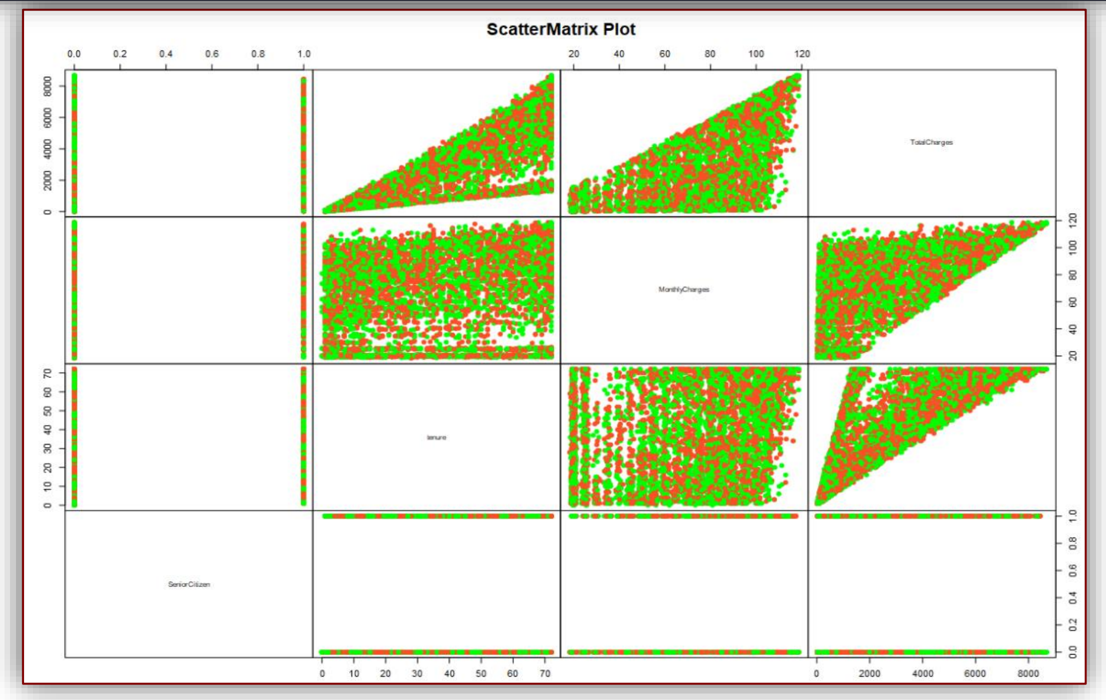
a) Generate a scatterplot matrix to show the relationships between the variables and a heatmap to determine correlated attributes.

I) Reading Data

~~~~
#reading data
DataFile = "C:/Users/mm/Documents/Churn Dataset.csv"
ChurnData = read.csv(DataFile, header=T)
```
~~~~

II) Plotting Categorical Data

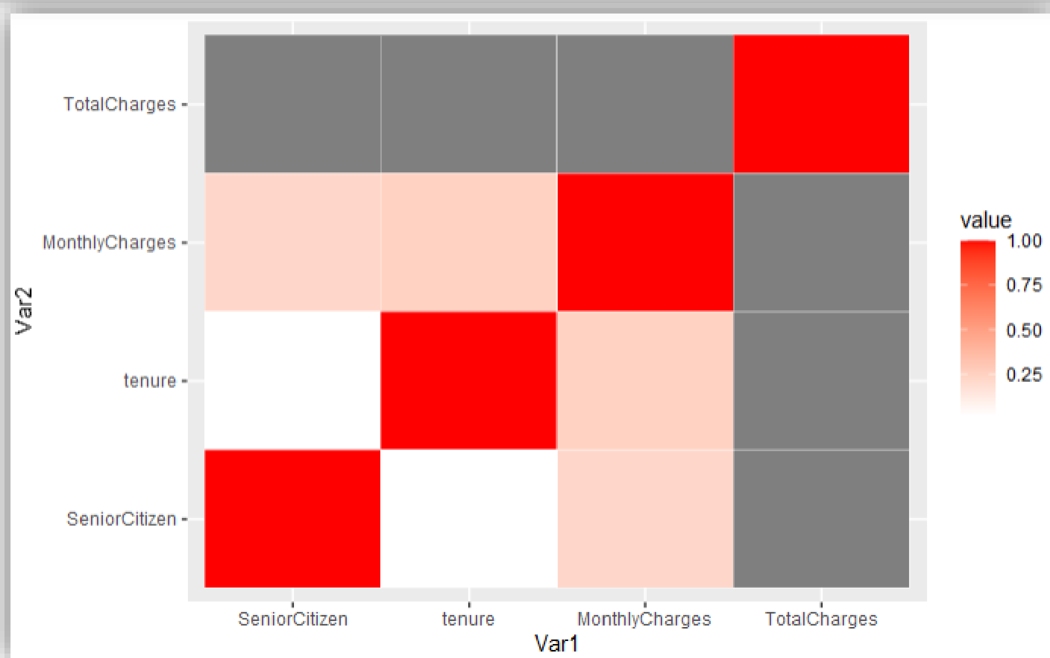
~~~~
```{r}
##ScatterMatrix Plot for Catogrical data
df <- ChurnData[c(3,6,19,20)]
pairs(df,         # Data
      pch = 19, # Pch symbol
      col = c("green", "#FC4E27"),  # Color
      main = "ScatterMatrix Plot",    # Title
      gap = 0,             # Subplots distance
      row1attop = FALSE, # Diagonal direction
      labels = colnames(df), # Labels
      cex.labels = 0.8,  # Size of diagonal texts
      font.labels = 1)
```
~~~~



III) Plotting Heat map

~~~~
```{r}
###heatmap
dta<- cor(ChurnData[sapply(ChurnData,is.numeric)])
cdf <- melt(dta)

ggplot(cdf, aes(Var1, Var2)) +
  geom_tile(aes(fill = value), colour = "white") +
  scale_fill_gradient(low = "white", high = "red")
```
~~~~



b) Ensure data is in the correct format for downstream processes (e.g., remove redundant information, convert categorical to numerical values, address missing values, etc.).

I) Removing Nulls from "TotalCharges".

````
```{r}
#removing nulls from dataset
sapply(ChurnData, function(x) sum(is.na(x)))
ChurnData <- ChurnData[complete.cases(ChurnData), ]
```

      customerID           gender    SeniorCitizen          Partner       Dependents           tenure     PhoneService
               0                0                0                0                0                0                0
   MultipleLines  InternetService   OnlineSecurity     OnlineBackup DeviceProtection      TechSupport      StreamingTV
               0                0                0                0                0                0                0
 StreamingMovies         Contract PaperlessBilling    PaymentMethod   MonthlyCharges     TotalCharges            Churn
               0                0                0                0                0               11                0
````

II) Replacing Values ("No Internet/Phone service") to ("No") in occurrence Columns.

````
```{r}
for(i in 10:15){
  ChurnData[,i] <- as.factor(mapvalues(ChurnData[,i],
                                       from= c("No internet service"), to= c("No")))
}

ChurnData[,8] <- as.factor(mapvalues(ChurnData[,8],
                                     from= c("No phone service"), to= c("No")))
```
````

III) Since the minimum tenure is 1 month and maximum tenure is 72 months, we can group them into five tenure groups: and putting news groups into new column named("tenure_group")

````
```{r}
min(ChurnData$tenure); max(ChurnData$tenure)
group_tenure <- function(tenure){
    if (tenure >= 0 & tenure <= 12){
        return('0-12 Month')
    }else if(tenure > 12 & tenure <= 24){
        return('12-24 Month')
    }else if (tenure > 24 & tenure <= 48){
        return('24-48 Month')
    }else if (tenure > 48 & tenure <=60){
        return('48-60 Month')
    }else if (tenure > 60){
        return('> 60 Month')
    }
}
ChurnData$tenure_group <- sapply(ChurnData$tenure,group_tenure)
ChurnData$tenure_group <- as.factor(ChurnData$tenure_group)

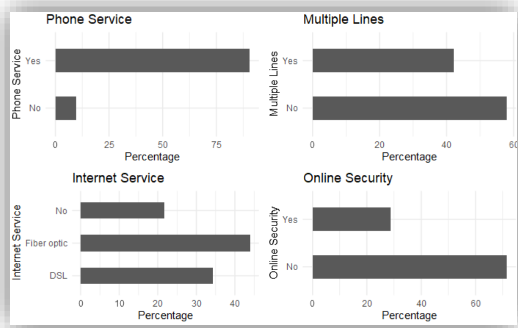
```
````

IV) Dropping "tenure, Customer ID" columns.

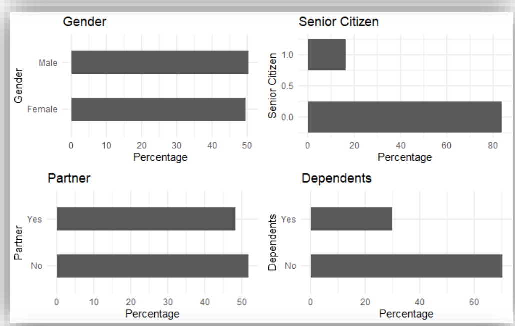
````
```{r}
ChurnData$customerID <- NULL
ChurnData$tenure <- NULL
```
````

V) Plotting all Categorical Columns to see if it's possible to drop any unnecessary.

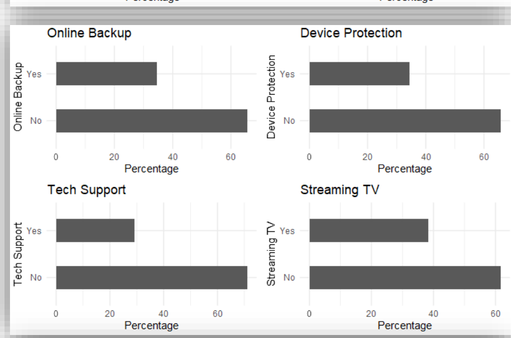
````
```{r}
p5 <- ggplot(ChurnData, aes(x=PhoneService)) + ggtitle("Phone Service") + xlab("Phone Service") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p6 <- ggplot(ChurnData, aes(x=MultipleLines)) + ggtitle("Multiple Lines") + xlab("Multiple Lines") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p7 <- ggplot(ChurnData, aes(x=InternetService)) + ggtitle("Internet Service") + xlab("Internet Service") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p8 <- ggplot(ChurnData, aes(x=OnlineSecurity)) + ggtitle("Online Security") + xlab("Online Security") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
grid.arrange(p5, p6, p7, p8, ncol=2)
````



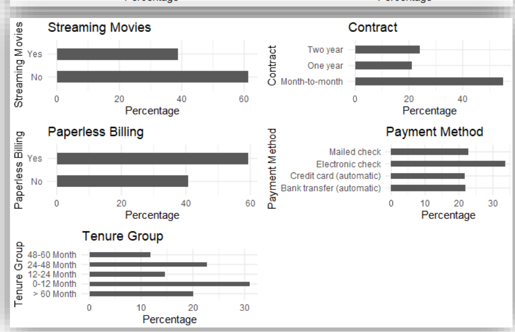

````
```{r}
p1 <- ggplot(ChurnData, aes(x=gender)) + ggtitle("Gender") + xlab("Gender") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p2 <- ggplot(ChurnData, aes(x=SeniorCitizen)) + ggtitle("Senior Citizen") + xlab("Senior Citizen") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p3 <- ggplot(ChurnData, aes(x=Partner)) + ggtitle("Partner") + xlab("Partner") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p4 <- ggplot(ChurnData, aes(x=Dependents)) + ggtitle("Dependents") + xlab("Dependents") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
grid.arrange(p1, p2, p3, p4, ncol=2)
```
````




````
```{r}
p9 <- ggplot(ChurnData, aes(x=OnlineBackup)) + ggtitle("Online Backup") + xlab("Online Backup") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p10 <- ggplot(ChurnData, aes(x=DeviceProtection)) + ggtitle("Device Protection") + xlab("Device Protection") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p11 <- ggplot(ChurnData, aes(x=TechSupport)) + ggtitle("Tech Support") + xlab("Tech Support") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p12 <- ggplot(ChurnData, aes(x=StreamingTV)) + ggtitle("Streaming TV") + xlab("Streaming TV") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
grid.arrange(p9, p10, p11, p12, ncol=2)
```
````




````
```{r}
p13 <- ggplot(ChurnData, aes(x=StreamingMovies)) + ggtitle("Streaming Movies") + xlab("Streaming Movies") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p14 <- ggplot(ChurnData, aes(x=Contract)) + ggtitle("Contract") + xlab("Contract") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p15 <- ggplot(ChurnData, aes(x=PaperlessBilling)) + ggtitle("Paperless Billing") + xlab("Paperless Billing") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p16 <- ggplot(ChurnData, aes(x=PaymentMethod)) + ggtitle("Payment Method") + xlab("Payment Method") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
p17 <- ggplot(ChurnData, aes(x=tenure_group)) + ggtitle("Tenure Group") + xlab("Tenure Group") +
  geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.5) + ylab("Percentage") + coord_flip() + theme_minimal()
grid.arrange(p13, p14, p15, p16, p17, ncol=2)
```
````



All of the categorical variables seem to have a reasonably broad distribution, therefore, all of them will be kept for the further analysis.
````

**VI) Convert categorical to numerical values.**

```{r}
columns <-c("Partner", "Dependents", "PhoneService"
           ,"MultipleLines", "OnlineSecurity","OnlineBackup",
           "DeviceProtection", "TechSupport","StreamingTV",
           "StreamingMovies","PaperlessBilling", "Churn","gender"
           ,"InternetService","Contract","PaymentMethod","tenure_group")
for (col in names(ChurnData)){
  for (j in (columns)){
  if(ChurnData[col]%in%ChurnData[j]){
   ChurnData[,j] = as.numeric(as.factor(ChurnData[,col])) - 1
  }}}
```

**VII) Checking for the highest correlated feature and removing it.**

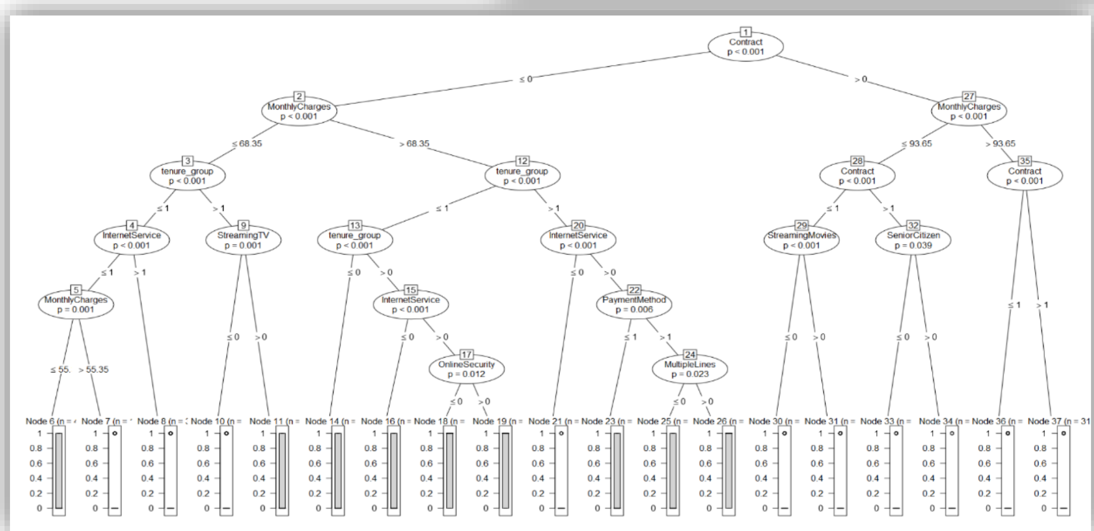
```{r}

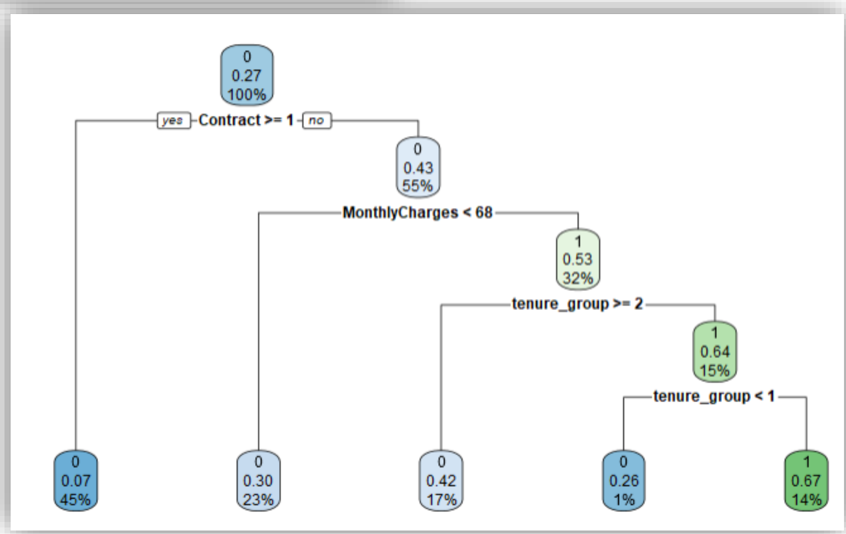
correlationMatrix <- cor(ChurnData)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.65)
# print indexes of highly correlated attributes
print(highlyCorrelated)
hc = sort(highlyCorrelated)
```

```{r}
ChurnData$TotalCharges<- NULL
```

**c) Split the dataset into 80 training/20 test set and fit a decision tree to the training data. Plot the tree, and interpret the results.**

```{r}
set.seed(1112)
split_train_test <- createDataPartition(ChurnData$Churn,p=0.8,list=FALSE)
dtrain<- ChurnData[split_train_test,]
dtest<-  ChurnData[-split_train_test,]
#Plot decision tree
rtree <- rpart(Churn ~ ., dtrain ,method ="class")
#plot conditional parting plot
ctree_ <- ctree(Churn ~ ., dtrain)
plot(ctree_)
rpart.plot(rtree)
pred  <- predict(rtree , newdata = dtest,type="class")
confusionMatrix(pred, as.factor( dtest$Churn) ) #check accuracy
```






```
          Reference
Prediction    0    1
         0 1004  218
         1   49  135

               Accuracy : 0.8101
                 95% CI : (0.7886, 0.8303)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 3.069e-08

                  Kappa : 0.3995

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9535
            Specificity : 0.3824
         Pos Pred Value : 0.8216
         Neg Pred Value : 0.7337
              Precision : 0.8216
                 Recall : 0.9535
                     F1 : 0.8826
             Prevalence : 0.7489
         Detection Rate : 0.7141
   Detection Prevalence : 0.8691
      Balanced Accuracy : 0.6680

       'Positive' Class : 0
```

**<u>From the shown confusion matrix</u>**

**The model predicts most of the test points right but there is something confusing with the model so it misses some of the points**

**d)** **Try different ways to improve the decision tree algorithm (e.g., use different splitting strategies, prune the tree after splitting).**

```
```{r}

etnhancetree2 <- rpart(Churn ~ ., dtrain ,method ="class",parms=list(split ="information"))
epred2  <- predict(etnhancetree2 , newdata = dtest,type="class",mode="everything")
confusionMatrix(epred2, as.factor( dtest$Churn) ) #check accuracy
```

```
```{r}
DT_poprune <- rpart(Churn ~ ., dtrain ,
                    method = "class",
                    control = rpart.control(cp = 0.0005)  )
plotcp(DT_poprune)
DT_model_pruned <- prune(DT_poprune, cp = 0.004)
Pred_poprune <- predict(DT_model_pruned, newdata = subset(dtest, select = -c(Churn)),type = "class" )
table_mat_prune <- table(dtest$Churn,Pred_poprune)
acc_Test_prune <- sum(diag(table_mat_prune)) / sum(table_mat_prune)
print(paste("Accuracy:", acc_Test_prune*100))

confusionMatrix(Pred_poprune , as.factor( dtest$Churn), mode = "everything" ) #check accuracy
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1004  218
         1   49  135

               Accuracy : 0.8101
                 95% CI : (0.7886, 0.8303
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 3.069e-08

                  Kappa : 0.3995

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9535
            Specificity : 0.3824
         Pos Pred Value : 0.8216
         Neg Pred Value : 0.7337
              Precision : 0.8216
                 Recall : 0.9535
                     F1 : 0.8826
             Prevalence : 0.7489
         Detection Rate : 0.7141
   Detection Prevalence : 0.8691
      Balanced Accuracy : 0.6680
```

```
          Reference
Prediction   0   1
         0 976 187
         1  77 166

               Accuracy : 0.8122
                 95% CI : (0.7908, 0.8323)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 1.002e-08

                  Kappa : 0.443

 Mcnemar's Test P-Value : 1.967e-11

            Sensitivity : 0.9269
            Specificity : 0.4703
         Pos Pred Value : 0.8392
         Neg Pred Value : 0.6831
              Precision : 0.8392
                 Recall : 0.9269
                     F1 : 0.8809
             Prevalence : 0.7489
         Detection Rate : 0.6942
   Detection Prevalence : 0.8272
      Balanced Accuracy : 0.6986

       'Positive' Class : 0
```

**Does pruning the tree improve the accuracy?** Yes, but not that much

**e)** **Classify the data using the XGBoost model with nrounds = 70 and max depth = 3. Evaluate the performance.**

```
```{r}
#splitting for XGBoost classifier

xtrain = data.matrix(dtrain[1:5626,-18])
ytrain = data.matrix(dtrain[1:5626,18])
xgtest = data.matrix(dtest[1:5626,-18])
ygtest = data.matrix(dtest[1:5626,18])

# train an xgboost model
bstDense <- xgboost(data =xtrain ,label = ytrain
                    ,max.depth = 3,eta = 0.2,
                    nrounds = 70, gamma=0.8)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 978 175
         1  75 178

               Accuracy : 0.8222
                 95% CI : (0.8012, 0.8418)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 3.116e-11

                  Kappa : 0.478

 Mcnemar's Test P-Value : 3.818e-10

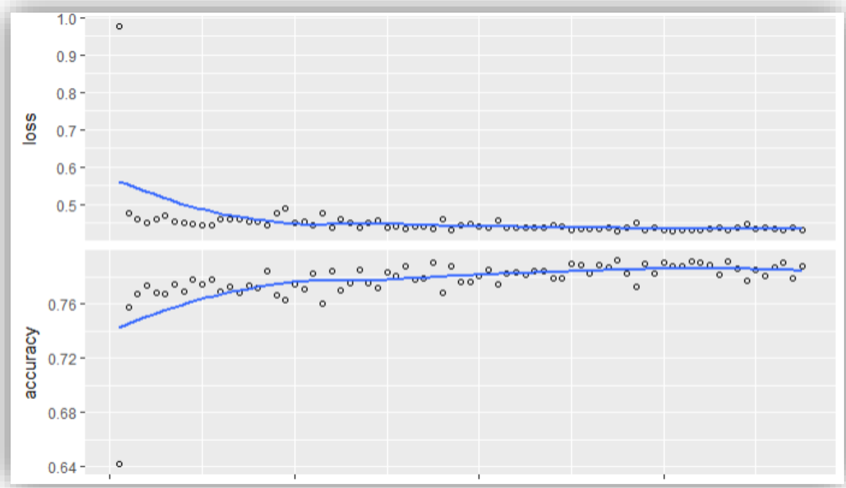
            Sensitivity : 0.9288
            Specificity : 0.5042
         Pos Pred Value : 0.8482
         Neg Pred Value : 0.7036
              Precision : 0.8482
                 Recall : 0.9288
                     F1 : 0.8867
             Prevalence : 0.7489
         Detection Rate : 0.6956
   Detection Prevalence : 0.8201
      Balanced Accuracy : 0.7165

       'Positive' Class : 0
```

**Is there any sign of overfitting?** **No**
```

**f)** **Train a deep neural network using Keras with 3 dense layers. Try changing the activation function or dropout rate.**

**I) Using Relu activation function and softmax in the end.**




```
          Reference
Prediction   0   1
         0 930 123
         1 138 215

               Accuracy : 0.8144
                 95% CI : (0.793, 0.8344)
    No Information Rate : 0.7596
    P-Value [Acc > NIR] : 4.602e-07

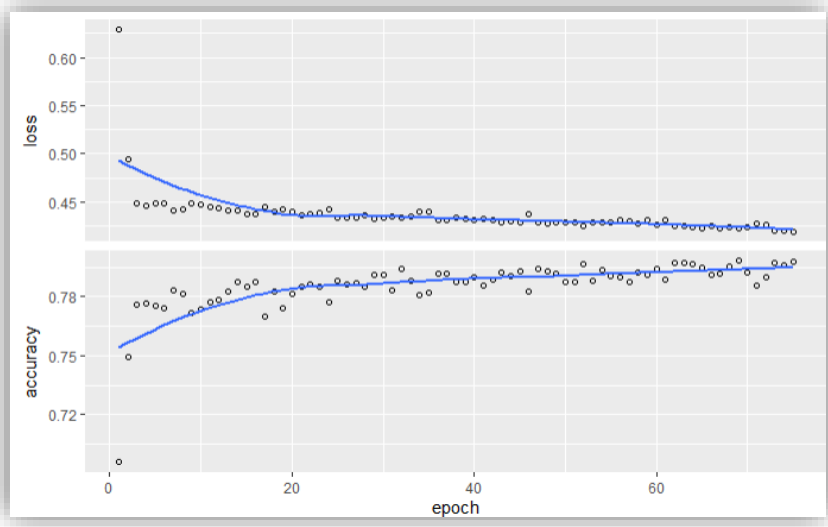
                  Kappa : 0.4993

 Mcnemar's Test P-Value : 0.3862

            Sensitivity : 0.8708
            Specificity : 0.6361
         Pos Pred Value : 0.8832
         Neg Pred Value : 0.6091
              Precision : 0.8832
                 Recall : 0.8708
                     F1 : 0.8769
             Prevalence : 0.7596
         Detection Rate : 0.6615
   Detection Prevalence : 0.7489
      Balanced Accuracy : 0.7534

       'Positive' Class : 0
```

**II) Using (Tanh) activation function and softmax in the end.**




```
          Reference
Prediction   0   1
         0 920 133
         1 132 221

               Accuracy : 0.8115
                 95% CI : (0.7901, 0.8317)
    No Information Rate : 0.7482
    P-Value [Acc > NIR] : 1.038e-08

                  Kappa : 0.4993

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8745
            Specificity : 0.6243
         Pos Pred Value : 0.8737
         Neg Pred Value : 0.6261
              Precision : 0.8737
                 Recall : 0.8745
                     F1 : 0.8741
             Prevalence : 0.7482
         Detection Rate : 0.6543
   Detection Prevalence : 0.7489
      Balanced Accuracy : 0.7494

       'Positive' Class : 0
```

**What effects does any of these have on the result?**

**No effects when I replaced the**

**Relu function obtained higher accuracy**

**g)** **Compare the performance of the models in terms of the following criteria: precision, recall, accuracy, F-measure. Identify the model that performed best and worst according to each criterion.**

**DNN model(RELU) achieved the Highest Precision among all models** → the classifier is very conservative - does not risk too much in saying that a sample is Positive

**Base Decisiontree&enhanced achieved the Highest recall among all models** → The model **maximizes the number of True Positives But it could be wrong sometimes!**

**Pruned Decision tree achieved the Highest F1 among all models** → high F-score can be the result of an imbalance between Precision and Recall

DT&enhanced DT worst model

```
          Reference
Prediction    0    1
         0 1004  218
         1   49  135

               Accuracy : 0.8101
                 95% CI : (0.7886, 0.8303)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 3.069e-08

                  Kappa : 0.3995

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9535
            Specificity : 0.3824
         Pos Pred Value : 0.8216
         Neg Pred Value : 0.7337
              Precision : 0.8216
                 Recall : 0.9535
                     F1 : 0.8826
             Prevalence : 0.7489
         Detection Rate : 0.7141
   Detection Prevalence : 0.8691
      Balanced Accuracy : 0.6680

       'Positive' Class : 0
```

Pruned DT

```
          Reference
Prediction   0   1
         0 976 187
         1  77 166

               Accuracy : 0.8122
                 95% CI : (0.7908, 0.8323)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 1.002e-08

                  Kappa : 0.443

 Mcnemar's Test P-Value : 1.967e-11

            Sensitivity : 0.9269
            Specificity : 0.4703
         Pos Pred Value : 0.8392
         Neg Pred Value : 0.6831
              Precision : 0.8392
                 Recall : 0.9269
                     F1 : 0.8809
             Prevalence : 0.7489
         Detection Rate : 0.6942
   Detection Prevalence : 0.8272
      Balanced Accuracy : 0.6986

       'Positive' Class : 0
```

XGBOOST classifier

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 978 175
         1  75 178

               Accuracy : 0.8222
                 95% CI : (0.8012, 0.8418)
    No Information Rate : 0.7489
    P-Value [Acc > NIR] : 3.116e-11

                  Kappa : 0.478

 Mcnemar's Test P-Value : 3.818e-10

            Sensitivity : 0.9288
            Specificity : 0.5042
         Pos Pred Value : 0.8482
         Neg Pred Value : 0.7036
              Precision : 0.8482
                 Recall : 0.9288
                     F1 : 0.8867
             Prevalence : 0.7489
         Detection Rate : 0.6956
   Detection Prevalence : 0.8201
      Balanced Accuracy : 0.7165

       'Positive' Class : 0
```

Dnn relu AF best model

```
          Reference
Prediction   0   1
         0 930 123
         1 138 215

               Accuracy : 0.8144
                 95% CI : (0.793, 0.8344)
    No Information Rate : 0.7596
    P-Value [Acc > NIR] : 4.602e-07

                  Kappa : 0.4993

 Mcnemar's Test P-Value : 0.3862

            Sensitivity : 0.8708
            Specificity : 0.6361
         Pos Pred Value : 0.8832
         Neg Pred Value : 0.6091
              Precision : 0.8832
                 Recall : 0.8708
                     F1 : 0.8769
             Prevalence : 0.7596
         Detection Rate : 0.6615
   Detection Prevalence : 0.7489
      Balanced Accuracy : 0.7534

       'Positive' Class : 0
```

Dnn tanh AF

```
          Reference
Prediction   0   1
         0 920 133
         1 132 221

               Accuracy : 0.8115
                 95% CI : (0.7901, 0.8317)
    No Information Rate : 0.7482
    P-Value [Acc > NIR] : 1.038e-08

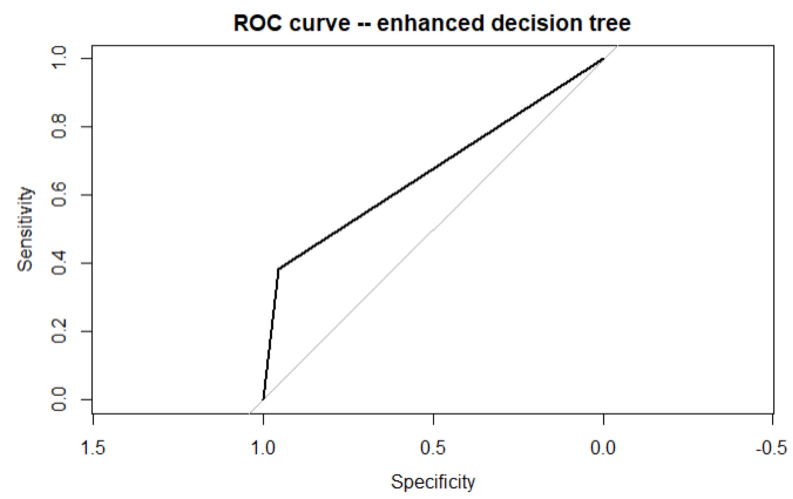
                  Kappa : 0.4993

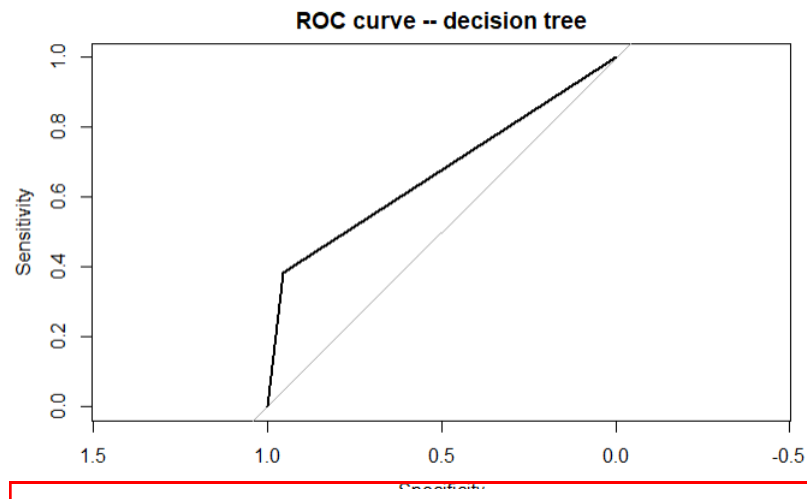
 Mcnemar's Test P-Value : 1

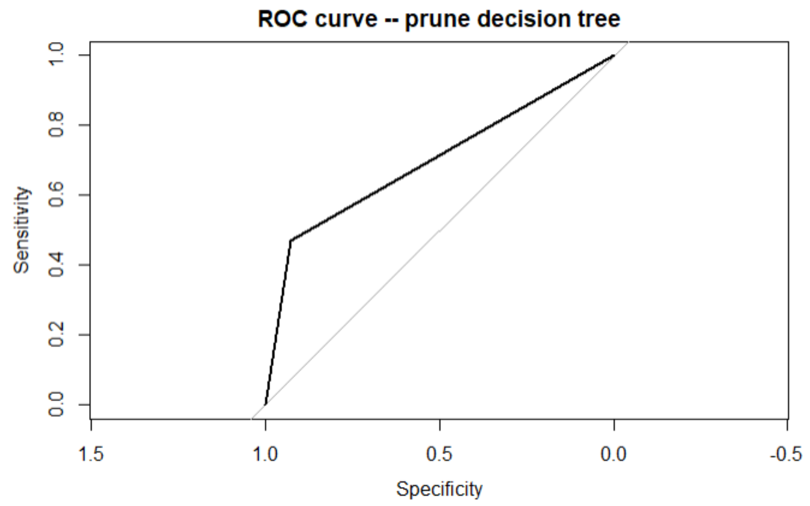
            Sensitivity : 0.8745
            Specificity : 0.6243
         Pos Pred Value : 0.8737
         Neg Pred Value : 0.6261
              Precision : 0.8737
                 Recall : 0.8745
                     F1 : 0.8741
             Prevalence : 0.7482
         Detection Rate : 0.6543
   Detection Prevalence : 0.7489
      Balanced Accuracy : 0.7494

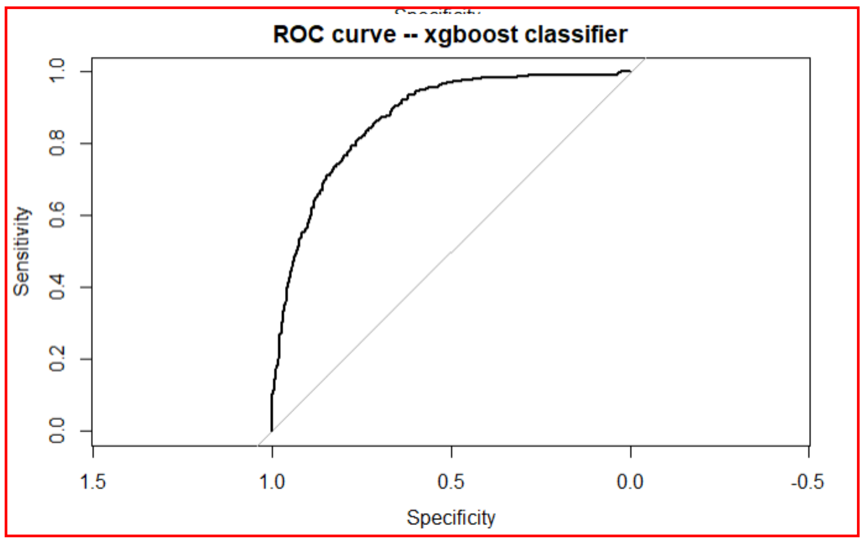
       'Positive' Class : 0
```

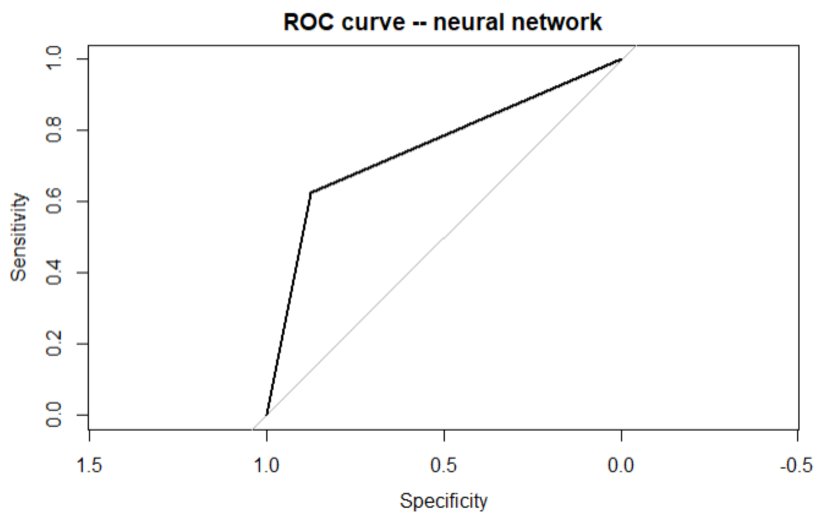
**h) Use a ROC graph to compare the performance of the DT, XGboost & DNN techniques.**

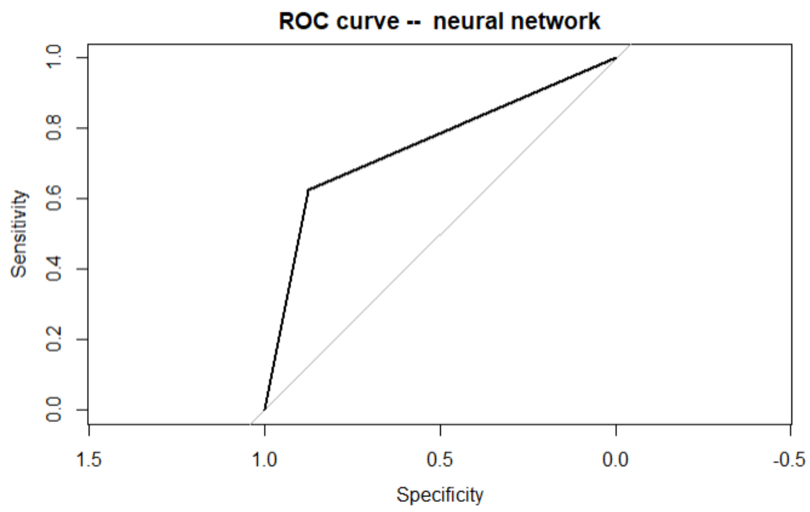


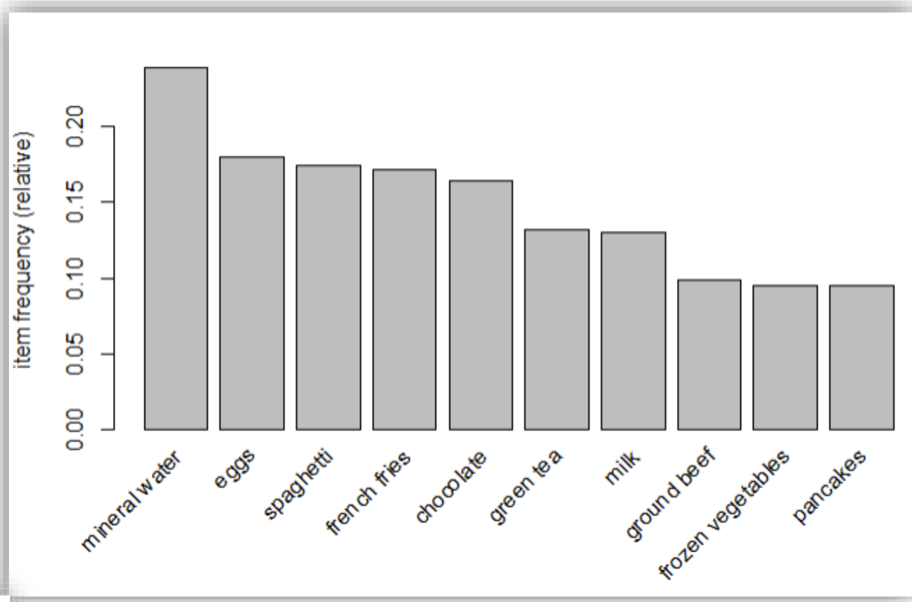



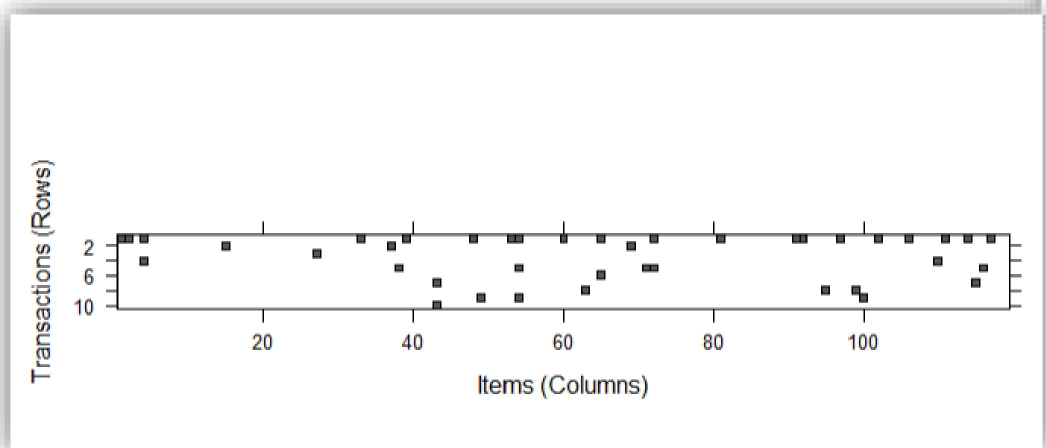

<u>**The highlighted curve has the high Area under the curve so It has better performance**</u>

## Part B:

1) **Generate a plot of the top 10 transactions.**

```{r}
#reading data
df = "C:/Users/Queen/Documents/transactions.csv"
trans = read.transactions(df, sep = ",")
#plotting most 10 frq transcations
itemFrequencyPlot(trans, topN = 10)
image(trans[1:10])
```



2) **Generate association rules using minimum support of 0.002, minimum confidence of 0.20, and maximum length of 3. Display the rules, sorted by descending lift value.**

```{r}
# default settings result in zero rules learned
apriori(trans)

transrules <- apriori(trans, parameter = list(support = 0.002, confidence =0.20, maxlen = 3))
lift <-inspect(sort(transrules,by="lift"))

|

```

| | lhs <chr> | <chr> | rhs <chr> | support <dbl> | confidence <dbl> |
|---|---|---|---|---|---|
| [991] | {milk, mineral water} | => | {spaghetti} | 0.015731236 | 0.3277778 |
| [992] | {milk, spaghetti} | => | {chocolate} | 0.010931876 | 0.3082707 |
| [993] | {burgers, french fries} | => | {green tea} | 0.005465938 | 0.2484848 |
| [994] | {olive oil, spaghetti} | => | {chocolate} | 0.007065725 | 0.3081395 |
| [995] | {almonds} | => | {green tea} | 0.005065991 | 0.2483660 |
| [996] | {mineral water, oil} | => | {spaghetti} | 0.002399680 | 0.3272727 |
| [997] | {olive oil, spaghetti} | => | {mineral water} | 0.010265298 | 0.4476744 |
| [998] | {french fries, grated cheese} | => | {chocolate} | 0.003199573 | 0.3076923 |
| [999] | {soup, tomatoes} | => | {spaghetti} | 0.002266364 | 0.3269231 |
| [1000] | {protein bar, spaghetti} | => | {mineral water} | 0.002266364 | 0.4473684 |

991-1000 of 2,023 rows | 1-6 of 8 columns Previous 1 ... 95 96 97 98 99 100 Next

3) **Select the rule from QII-b with the greatest lift. Compare this rule with the highest lift rule for maximum length of 2.**

```{r}
# set better support and confidence levels to learn more rules
transrules1 <- apriori(trans, parameter = list(support = 0.002, confidence =0.20, maxlen = 2))
lift <-inspect(sort(transrules1,by="lift"))
transrules1
```

| | lhs <chr> | <chr> | rhs <chr> | support <dbl> | confidence <dbl> | coverage <dbl> | lift <dbl> | count <int> |
|---|---|---|---|---|---|---|---|---|
| [351] | {green tea} | => | {mineral water} | 0.031062525 | 0.2351160 | 0.132115718 | 0.9863565 | 233 |
| [352] | {parmesan cheese} | => | {mineral water} | 0.004666045 | 0.2348993 | 0.019864018 | 0.9854474 | 35 |
| [353] | {melons} | => | {mineral water} | 0.002799627 | 0.2333333 | 0.011998400 | 0.9788777 | 21 |
| [354] | {candy bars} | => | {mineral water} | 0.002266364 | 0.2328767 | 0.009732036 | 0.9769621 | 17 |
| [355] | {light mayo} | => | {mineral water} | 0.006265831 | 0.2303922 | 0.027196374 | 0.9665389 | 47 |
| [356] | {escalope} | => | {mineral water} | 0.017064391 | 0.2151261 | 0.079322757 | 0.9024947 | 128 |
| [357] | {energy drink} | => | {mineral water} | 0.005599253 | 0.2100000 | 0.026663112 | 0.8809899 | 42 |
| [358] | {yogurt cake} | => | {mineral water} | 0.005599253 | 0.2048780 | 0.027329689 | 0.8595024 | 42 |

351-358 of 358 rows Previous 1 ... 31 32 33 34 35 36 Next

**I) Which rule has the better lift? Which rule has the greater support?**

**The second rule has the greater support**

**II) If you were a marketing manager, and could fund only one of these rules, which would it be, and why?**

**The second rule, because it has the highest support and trust, it will reflect positively on my sales!**