REPORT

FUND. APPLIED DATA SCIENCE

Assignment 2

Mostafa Mahmoud Nofal

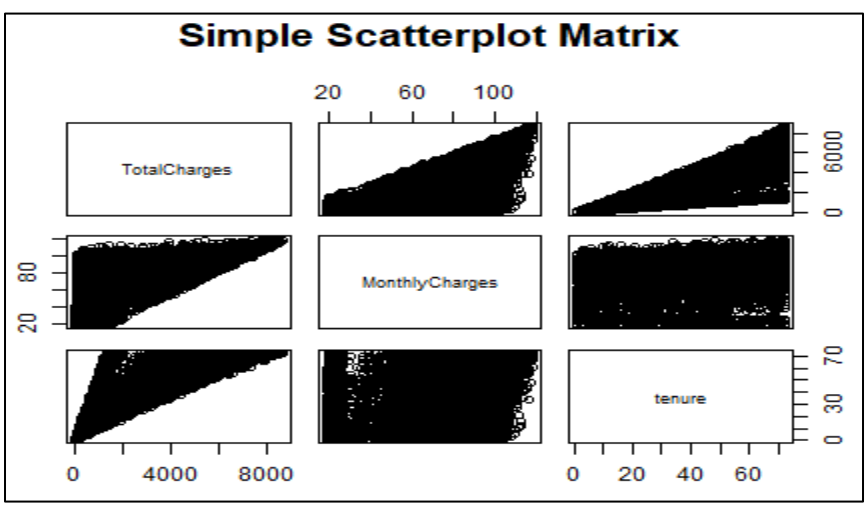
Part A: Classification

First, we read our churn dataset.

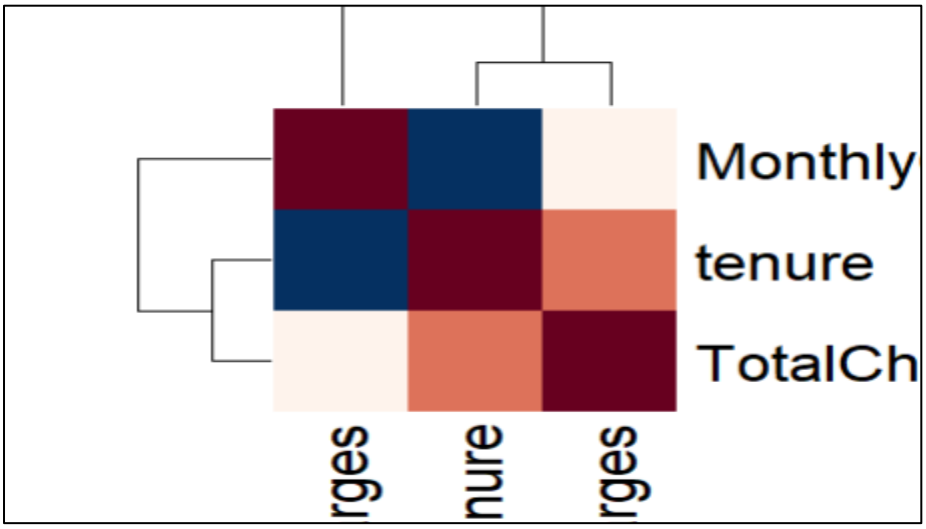
```
> str(churn)
'data.frame':   7043 obs. of  21 variables:
 $ customerID      : chr  "7590-VHVEG" "5575-GNVDE" "3668-QPYBK" "7795-CFOCW" ...
 $ gender          : chr  "Female" "Male" "Male" "Male" ...
 $ SeniorCitizen   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Partner         : chr  "Yes" "No" "No" "No" ...
 $ Dependents      : chr  "No" "No" "No" "No" ...
 $ tenure          : int  1 34 2 45 2 8 22 10 28 62 ...
 $ PhoneService    : chr  "No" "Yes" "Yes" "No" ...
 $ MultipleLines   : chr  "No phone service" "No" "No" "No phone service" ...
 $ InternetService : chr  "DSL" "DSL" "DSL" "DSL" ...
 $ OnlineSecurity  : chr  "No" "Yes" "Yes" "Yes" ...
 $ OnlineBackup    : chr  "Yes" "No" "Yes" "No" ...
 $ DeviceProtection: chr  "No" "Yes" "No" "Yes" ...
 $ TechSupport     : chr  "No" "No" "No" "Yes" ...
 $ StreamingTV     : chr  "No" "No" "No" "No" ...
 $ StreamingMovies : chr  "No" "No" "No" "No" ...
 $ Contract        : chr  "Month-to-month" "One year" "Month-to-month" "One year" ...
 $ PaperlessBilling: chr  "Yes" "No" "Yes" "No" ...
 $ PaymentMethod   : chr  "Electronic check" "Mailed check" "Mailed check" "Bank transfer (automati
c)" ...
 $ MonthlyCharges  : num  29.9 57 53.9 42.3 70.7 ...
 $ TotalCharges    : num  29.9 1889.5 108.2 1840.8 151.7 ...
 $ Churn           : chr  "No" "No" "Yes" "No" ...
```

```
> glimpse(churn)
Rows: 7,043
Columns: 21
$ customerID       <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CFOCW", "9237-HQITU"…
$ gender           <chr> "Female", "Male", "Male", "Male", "Female", "Female", "Male", "Femal…
$ SeniorCitizen    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,…
$ Partner          <chr> "Yes", "No", "No", "No", "No", "No", "No", "No", "Yes", "No", "Yes",…
$ Dependents       <chr> "No", "No", "No", "No", "No", "No", "Yes", "No", "No", "Yes", "Yes",…
$ tenure           <int> 1, 34, 2, 45, 2, 8, 22, 10, 28, 62, 13, 16, 58, 49, 25, 69, 52, 71, …
$ PhoneService     <chr> "No", "Yes", "Yes", "No", "Yes", "Yes", "Yes", "No", "Yes", "Yes", "…
$ MultipleLines    <chr> "No phone service", "No", "No", "No phone service", "No", "Yes", "Ye…
$ InternetService  <chr> "DSL", "DSL", "DSL", "DSL", "Fiber optic", "Fiber optic", "Fiber opt…
$ OnlineSecurity   <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No", "Yes", "No", "Yes", "Ye…
$ OnlineBackup     <chr> "Yes", "No", "Yes", "No", "No", "No", "Yes", "No", "No", "Yes", "No"…
$ DeviceProtection <chr> "No", "Yes", "No", "Yes", "No", "Yes", "No", "No", "Yes", "No", "No"…
$ TechSupport      <chr> "No", "No", "No", "Yes", "No", "No", "No", "No", "Yes", "No", "No", …
$ StreamingTV      <chr> "No", "No", "No", "No", "No", "Yes", "Yes", "No", "Yes", "No", "No",…
$ StreamingMovies  <chr> "No", "No", "No", "No", "No", "Yes", "No", "No", "Yes", "No", "No", …
$ Contract         <chr> "Month-to-month", "One year", "Month-to-month", "One year", "Month-t…
$ PaperlessBilling <chr> "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes", "No", "Yes", "No", "Y…
$ PaymentMethod    <chr> "Electronic check", "Mailed check", "Mailed check", "Bank transfer (…
$ MonthlyCharges   <dbl> 29.85, 56.95, 53.85, 42.30, 70.70, 99.65, 89.10, 29.75, 104.80, 56.1…
$ TotalCharges     <dbl> 29.85, 1889.50, 108.15, 1840.75, 151.65, 820.50, 1949.40, 301.90, 30…
$ Churn            <chr> "No", "No", "Yes", "No", "Yes", "Yes", "No", "No", "Yes", "No", "No"…
```

1. Then, we implemented the scatterplot matrix to show the relationships between the variables.



- Heat map to determine our correlated attributes.



2. We want to ensure that our data is in the correct format so we will perform some operations like:
   - Checking the numbers if there are missing values in each column. We removed 11 missing values.

Before removing missing values

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

After removing missing values

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

- Unifying the “No” values by changing values like “No internet service” and “No phone service” to “No”.

Yes	No	No	No internet service
Yes	Yes	DSL	No
Yes	No	DSL	Yes
Yes	No	DSL	Yes
Yes	Yes	Fiber optic	No
No	No phone service	DSL	No
Yes	Yes	DSL	Yes
Yes	No	DSL	No
Yes	Yes	Fiber optic	Yes
Yes	No	Fiber optic	No
Yes	No	DSL	Yes
Yes	No	No	No internet service
Yes	No	DSL	No
Yes	Yes	Fiber optic	Yes
Yes	No	Fiber optic	No
Yes	No	Fiber optic	No
Yes	Yes	Fiber optic	No
Yes	Yes	Fiber optic	No
Yes	No	DSL	No
Yes	Yes	DSL	Yes
Yes	No	No	No internet service
Yes	Yes	DSL	Yes
Yes	Yes	DSL	Yes
Yes	Yes	Fiber optic	No
Yes	No	DSL	No

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
1	Female	0	Yes	No	1	No
2	Male	0	No	No	34	Yes
3	Male	0	No	No	2	Yes
4	Male	0	No	No	45	No
5	Female	0	No	No	2	Yes
6	Female	0	No	No	8	Yes

	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
1	No phone service	DSL	No	Yes
2	No	DSL	Yes	No
3	No	DSL	Yes	Yes
4	No phone service	DSL	Yes	No
5	No	Fiber optic	No	No
6	Yes	Fiber optic	No	No

	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract
1	No	No	No	No	Month-to-month
2	Yes	No	No	No	One year
3	No	No	No	No	Month-to-month
4	Yes	Yes	No	No	One year
5	No	No	No	No	Month-to-month
6	Yes	No	Yes	Yes	Month-to-month

	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges
1	Yes	Electronic check	29.85	29.85
2	No	Mailed check	56.95	1889.50
3	Yes	Mailed check	53.85	108.15
4	No	Bank transfer (automatic)	42.30	1840.75
5	Yes	Electronic check	70.70	151.65
6	Yes	Electronic check	99.65	820.50

	Churn
1	No
2	No
3	Yes
4	No

- Removing redundant information by using the unique function.
- Removing the customerID column because it is independent and will not identify the sample class

- Convert categorical to numerical values by using the label encoder.

Before LabelEncoder

```
  gender SeniorCitizen Partner Dependents tenure PhoneService
1 Female             0     Yes         No      1           No
2   Male             0      No         No     34          Yes
3   Male             0      No         No      2          Yes
4   Male             0      No         No     45           No
5 Female             0      No         No      2          Yes
6 Female             0      No         No      8          Yes
     MultipleLines InternetService OnlineSecurity OnlineBackup
1 No phone service            DSL             No          Yes
2             No            DSL            Yes           No
3             No            DSL            Yes          Yes
4 No phone service            DSL            Yes           No
5             No    Fiber optic             No           No
6            Yes    Fiber optic             No           No
  DeviceProtection TechSupport StreamingTV StreamingMovies       Contract
1               No          No          No              No Month-to-month
2              Yes          No          No              No       One year
3               No          No          No              No Month-to-month
4              Yes         Yes          No              No       One year
5               No          No          No              No Month-to-month
6              Yes          No         Yes             Yes Month-to-month
  PaperlessBilling             PaymentMethod MonthlyCharges TotalCharges
1              Yes          Electronic check          29.85        29.85
2               No              Mailed check          56.95      1889.50
3              Yes              Mailed check          53.85       108.15
4               No Bank transfer (automatic)          42.30      1840.75
5              Yes          Electronic check          70.70       151.65
6              Yes          Electronic check          99.65       820.50
  Churn
1    No
2    No
3   Yes
4    No
```

After LabelEncoder

```
  customerID gender SeniorCitizen Partner Dependents tenure PhoneService
1 7590-VHVEG      0             0       0         0      1            0
2 5575-GNVDE      1             0       1         0     34            1
3 3668-QPYBK      1             0       1         0      2            1
4 7795-CFOCW      1             0       1         0     45            0
5 9237-HQITU      0             0       1         0      2            1
6 9305-CDSKC      0             0       1         0      8            1
  MultipleLines InternetService OnlineSecurity OnlineBackup DeviceProtection
1             0               0             0            1                 0
2             0               0             1            0                 1
3             0               0             1            1                 0
4             0               0             1            0                 1
5             0               1             0            0                 0
6             1               1             0            0                 1
  TechSupport StreamingTV StreamingMovies Contract PaperlessBilling
1           0           0               0        0               0
2           0           0               0        1               1
3           0           0               0        0               0
4           1           0               0        1               1
5           0           0               0        0               0
6           0           1               1        0               0
  PaymentMethod MonthlyCharges TotalCharges Churn
1             0         29.85       29.85     0
2             1         56.95     1889.50     0
3             1         53.85      108.15     1
4             2         42.30     1840.75     0
5             0         70.70      151.65     1
6             0         99.65      820.50     1
>
```

3. Splitting the dataset into 80% training and 20% testing set.

```
set.seed(147)
sample_data = sample.split(new_customer_data, SplitRatio = 0.8)
train_data = subset(new_customer_data, sample_data == TRUE)
test_data = subset(new_customer_data, sample_data == FALSE)
```

- Fitting a decision tree to the training data.

```
model <- rpart(Churn~., train_data,method = "class")
rpart.plot(model,type=5)

pred_tree <- predict(model, test_data,type = 'class')
Confusion_Matrix <- table(Predicted = pred_tree, Actual = test_data$Churn)
print("Confusion Matrix for Decision Tree"); Confusion_Matrix
```

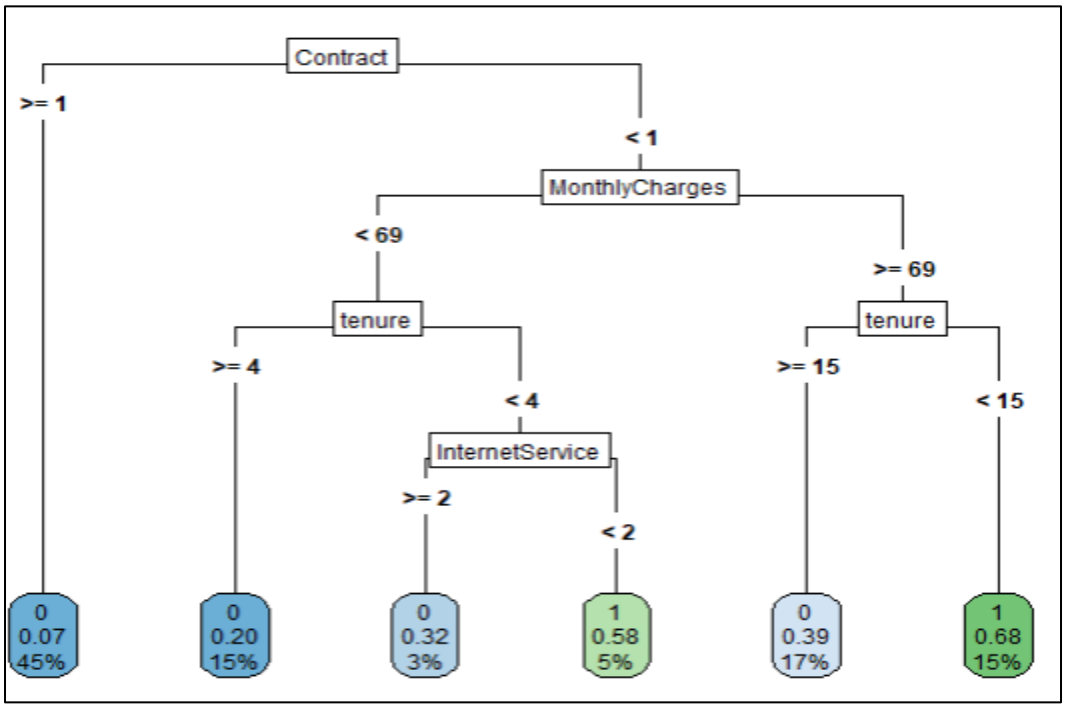
- Confusion matrix and Accuracy

```
[1] "Confusion Matrix for Decision Tree"
         Actual
Predicted  No Yes
      No  979 244
      Yes  96 162
```

```
Decision Tree Accuracy 0.77
```

```
Area under the curve: 0.6829
```

- Plot the tree



- Interpret the results:
    - Contract is our root node.
    - There is a probability of 0.07 that if the customers' contract is one year or two, 45% of customers will churn directly.
    - We need to make other branches for the rest 55% if the contract is month-to-month such as taking a look at MonthlyCharges column. If it is smaller than 69 and tenure is bigger than 4, 15% will churn by a probability of 0.2 and if the tenure is smaller than 4, we will see that if users using Internet services bigger than 2, 3% of customers will churn with a probability of 0.32. while the 5% users using internet services smaller than 2 will not be churned with a probability of 0.58.
    - On the other hand, if the MonthlyCharges is bigger than 69 and the tenure is bigger than 15, 17% of customers will churn by a probability of 0.39 and 15% of customers will not be churned by a probability of 0.68.

4. Improving the decision tree algorithm by different splitting (70% training and 30% testing)

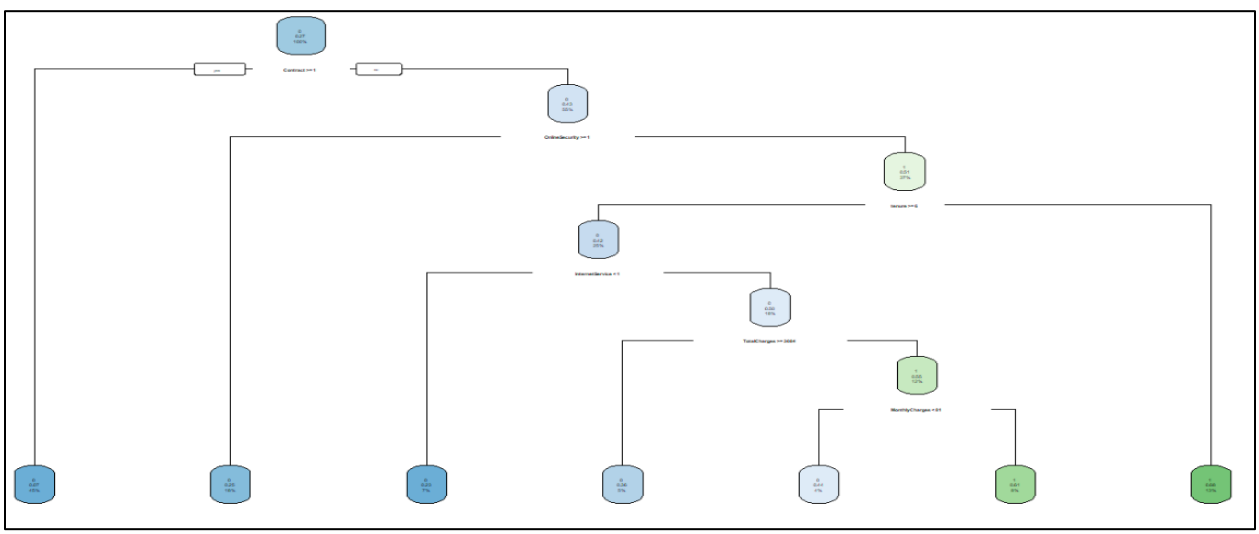
```
set.seed(50)
sample_data_2 = sample.split(new_customer_data, SplitRatio = 0.7)
train_data_2 = subset(new_customer_data, sample_data_2 == TRUE)
test_data_2 = subset(new_customer_data, sample_data_2 == FALSE)
```

```
model_2 <- rpart(Churn~., train_data_2,method = "class")
rpart.plot(model_2,type=5)

pred_tree <- predict(model_2, test_data_2,type = 'class')
Confusion_Matrix <- table(Predicted = pred_tree, Actual = test_data_2$Churn)
print("Confusion Matrix for Decision Tree"); Confusion_Matrix

accuracy_Test <- sum(diag(Confusion_Matrix)) / sum(Confusion_Matrix)
print(paste('Decision Tree Accuracy', accuracy_Test))
```

```
[1] "Confusion Matrix for Decision Tree"
         Actual
Predicted    0    1
        0 1379  284
        1  171  276
```

```
Decision Tree Accuracy 0.78
```

```
Area under the curve: 0.6913
```



- Improving the decision tree algorithm by different splitting (60% training and 40% testing)

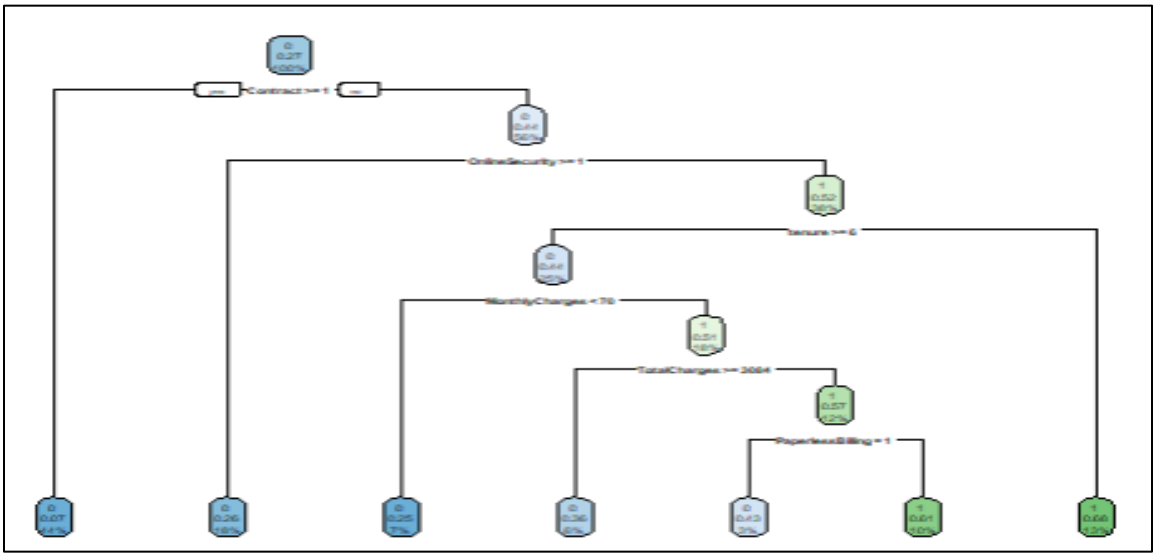
```
set.seed(50)
sample_data_3 = sample.split(new_customer_data, SplitRatio = 0.6)
train_data_3 = subset(new_customer_data, sample_data_3 == TRUE)
test_data_3 = subset(new_customer_data, sample_data_3 == FALSE)
```

```
model_3 <- rpart(Churn~., train_data_3,method = "class")
rpart.plot(model_3,type=5)

pred_tree <- predict(model_3, test_data_3,type = 'class')
Confusion_Matrix <- table(Predicted = pred_tree, Actual = test_data_3$Churn)
print("Confusion Matrix for Decision Tree"); Confusion_Matrix

accuracy_Test <- sum(diag(Confusion_Matrix)) / sum(Confusion_Matrix)
print(paste('Decision Tree Accuracy', accuracy_Test))
```

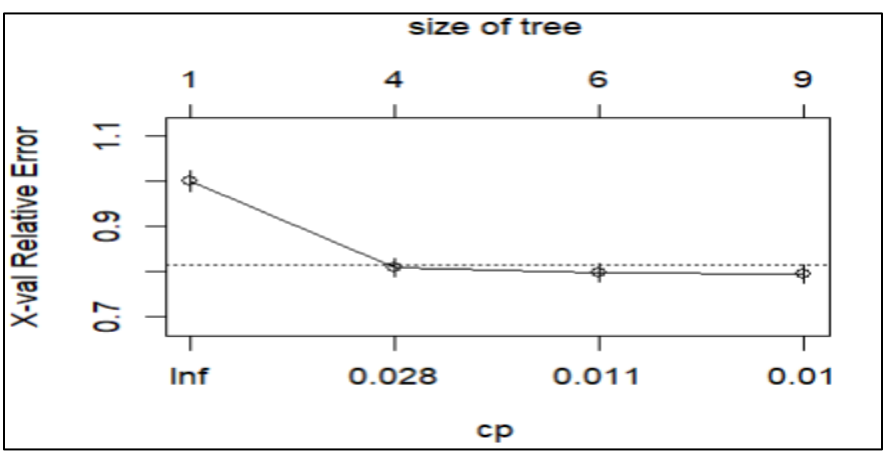
```
[1] "Confusion Matrix for Decision Tree"
         Actual
Predicted    0    1
        0 1849  332
        1  245  387
```

```
Decision Tree Accuracy 0.79
```

```
Area under the curve: 0.7106
```



- Pruning the tree




```
accuracy_pruned=sum(diag(Confusion_Matrix_pruned))/sum(Confusion_Matrix_pruned)
accuracy_pruned
0.7852063
```

- When the splitting was (60/40), the accuracy was 79%
- After pruning the tree, the accuracy became 78% so it didn't increase.

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

```
xgb_model <- xgboost(data = as.matrix(train_data[, 1:19]),label = train_data$Churn,max_depth = 3,
                     nrounds = 70, objective = "binary:logistic")


xgb_pred <- predict(xgb_model,  as.matrix(test_data[,1:19]))
xgb_pred[xgb_pred>=.5] = 1
xgb_pred[xgb_pred<.5] = 0

Confusion_Matrix_GB = table(test_data$Churn, xgb_pred)
print("Confusion_Matrix_GB for Decision Tree"); Confusion_Matrix_GB
accuracy_Gb=sum(diag(Confusion_Matrix_GB))/sum(Confusion_Matrix_GB)
accuracy_Gb

#ROC
plot(roc(test_data$Churn,as.numeric(xgb_pred)))
measurePrecisionRecall(test_data$Churn, xgb_pred)
```

- Confusion matrix and accuracy

```
[1] "Confusion_Matrix_GB for Decision Tree"
   xgb_pred
      0   1
  0 922 110
  1 188 186
```

```
Browse[1]>   accuracy_Gb
[1] 0.7880512
```

- Is there any sign of overfitting?
Mostly there is no overfitting because it's close to the accuracy we had before.

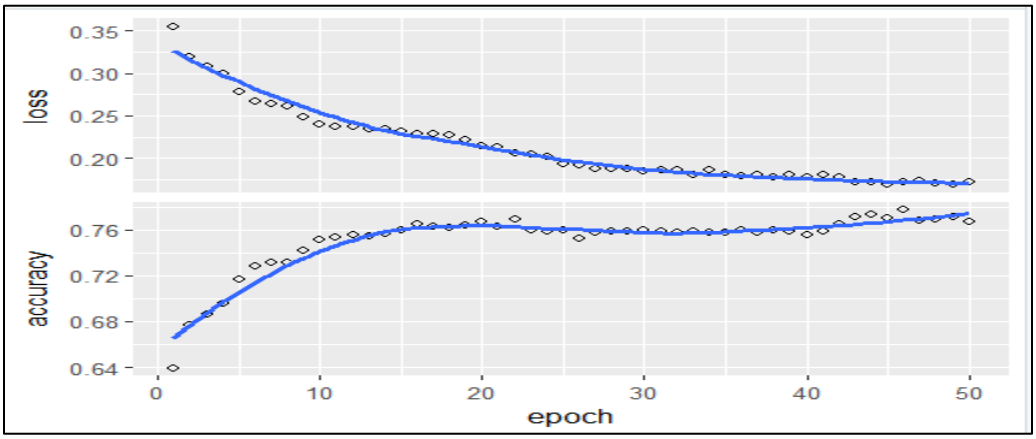
6. Train a deep neural network using Keras with 3 dense layers. Try changing the activation function or dropout rate.

```
deep_train = as.matrix(train_data)
deep_test = as.matrix(test_data)
dimnames(deep_train) = NULL

model <- keras_model_sequential()

  model %>%
    layer_dense(units = 19, activation = 'relu', input_shape = 19) %>%
    layer_dropout(rate = 0.1) %>%
    layer_dense(units = 10, activation = 'relu') %>%
    layer_dropout(rate = 0.1) %>%
    layer_dense(units = 1, activation = 'sigmoid')
```

```
model %>% compile(
  loss      = 'mean_squared_error',
  optimizer = 'adam',
  metrics = c('accuracy')
)


history = fit(model,data.matrix(train_data[,1:19]), train_data$Churn, epochs = 50,
              batch_size = 128)
plot(history)
```

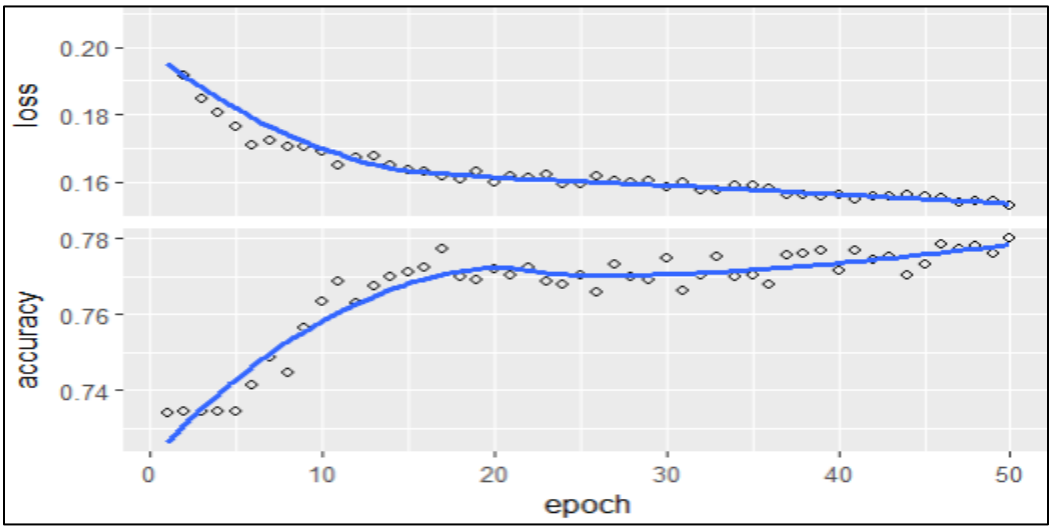
- Relu activation function accuracy.

```
Epoch 47/50
44/44 [==============================] - 0s 5ms/step - loss: 0.1734 - accuracy: 0.7680
Epoch 48/50
44/44 [==============================] - 0s 7ms/step - loss: 0.1711 - accuracy: 0.7693
Epoch 49/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1688 - accuracy: 0.7716
Epoch 50/50
44/44 [==============================] - 0s 7ms/step - loss: 0.1714 - accuracy: 0.7672
```



- Changing the activation function to Tanh and print its accuracy.

```
Epoch 46/50
44/44 [==============================] - 0s 8ms/step - loss: 0.1556 - accuracy: 0.7785
Epoch 47/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1542 - accuracy: 0.7773
Epoch 48/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1544 - accuracy: 0.7782
Epoch 49/50
44/44 [==============================] - 0s 7ms/step - loss: 0.1543 - accuracy: 0.7762
Epoch 50/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1529 - accuracy: 0.7800
```



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

The accuracy of Tanh activation function is higher than the accuracy of the relu activation function.

7. Compare the performance of the models in terms of the following criteria: precision, recall, accuracy, F-measure.
   - Function to calculate the precision, recall and f-measure.

```
measurePrecisionRecall <- function(actual_labels, predict){
  conMatrix = table(actual_labels, predict)
  precision <- conMatrix['0','0'] / ifelse(sum(conMatrix[,'0'])== 0, 1, sum(conMatrix[,'0']))
  recall <- conMatrix['0','0'] / ifelse(sum(conMatrix['0',])== 0, 1, sum(conMatrix['0',]))
  fmeasure <- 2 * precision * recall / ifelse(precision + recall == 0, 1, precision + recall)

  cat('precision:  ')
  cat(precision * 100)
  cat('%')
  cat('\n')

  cat('recall:     ')
  cat(recall * 100)
  cat('%')
  cat('\n')

  cat('f-measure:  ')
  cat(fmeasure * 100)
  cat('%')
  cat('\n')
}
```

   - The decision tree model

```
>   accuracy_Test <- sum(diag(Confusion_Matrix)) / sum(Confusion_Matrix)
>   print(paste('Decision Tree Accuracy', accuracy_Test))
[1] "Decision Tree Accuracy 0.778805120910384"
```

```
>   measurePrecisionRecall(test_data[,15],pred_tree)
precision:  41.88034%
recall:     55.54156%
f-measure:  47.75311%
```

   - Xgboost

```
>   accuracy_Gb
[1] 0.7965861
```

```
Browse[4]>   measurePrecisionRecall(test_data$Churn, xgb_pred)
precision:  83.06306%
recall:     89.34109%
f-measure:  86.08777%
```

   - Deep neural network

```
>   accuracy_Dnn
[1] 0.7596017
```

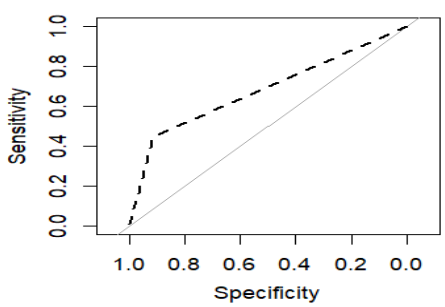
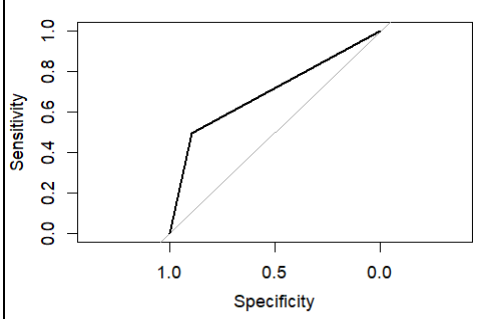
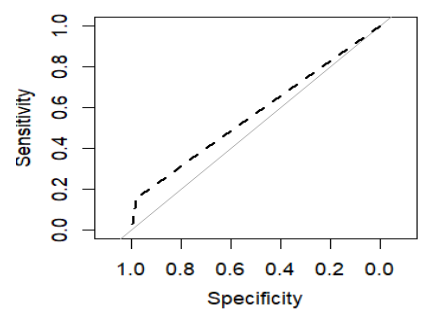
```
Browse[4]>   measurePrecisionRecall(test_data$Churn,pred_data)
precision:  73.39972%
recall:     100%
f-measure:  84.65956%
```

- Identify the model that performed best and worst according to each criterion.

	Accuracy	Precision	Recall	f-measure
Decision Tree	**77%**	**41%**	**55%**	**47%**
Xgboost	**79%**	**83%**	**89%**	**86%**
DNN	**75%**	**73%**	**100%**	**84%**
Best	**Xgboost**	**Xgboost**	**DNN**	**Xgboost**
Worst	**DNN**	**Decision Tree**	**Decision Tree**	**Decision Tree**

- Our best model is Xgboost and the worst is Decision Tree.

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

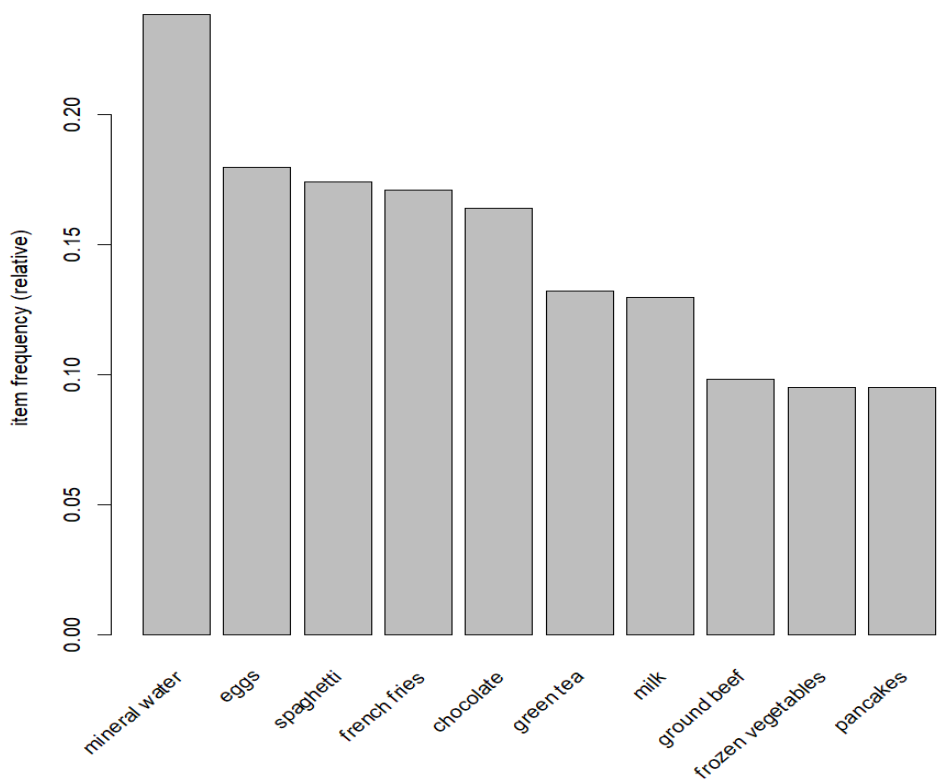
Decision Tree ROC	**Xgboost ROC**	**DNN ROC**
		
`Area under the curve: 0.7078`	`Area under the curve: 0.7012`	`Area under the curve: 0.5686`

- Part B: Association Rule

- Read the transactions dataset

```
library(arules)
library(arulesViz)

t <- read.transactions("transactions.csv", header=FALSE,sep = ',', rm.duplicates = TRUE)

itemFrequencyPlot(t, topN = 20)
```

A - Generate a plot of the top 10 transactions.



B - Generating association rules with minimum support = 0.002, minimum confidence = 0.20, and maximum length = 3.

```
r1 <- apriori(t, parameter = list(support = 0.002, confidence =0.20, maxlen = 3))
```

- Display the rules, sorted by descending lift value

```
inspect(sort(r1, by = "lift",decreasing = TRUE))
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{escalope, mushroom cream sauce}	=>	{pasta}	0.002532996	0.4418605	0.005732569	28.088096	19
[2]	{escalope, pasta}	=>	{mushroom cream sauce}	0.002532996	0.4318182	0.005865885	22.650826	19
[3]	{mushroom cream sauce, pasta}	=>	{escalope}	0.002532996	0.9500000	0.002666311	11.976387	19
[4]	{parmesan cheese, tomatoes}	=>	{frozen vegetables}	0.002133049	0.6666667	0.003199573	6.993939	16
[5]	{mineral water, whole wheat pasta}	=>	{olive oil}	0.003866151	0.4027778	0.009598720	6.115863	29
[6]	{frozen vegetables, parmesan cheese}	=>	{tomatoes}	0.002133049	0.3902439	0.005465938	5.706081	16
[7]	{burgers, herb & pepper}	=>	{ground beef}	0.002266364	0.5483871	0.004132782	5.581345	17
[8]	{light cream, mineral water}	=>	{chicken}	0.002399680	0.3272727	0.007332356	5.455273	18
[9]	{ground beef, shrimp}	=>	{herb & pepper}	0.002932942	0.2558140	0.011465138	5.172131	22
[10]	{fromage blanc}	=>	{honey}	0.003332889	0.2450980	0.013598187	5.164271	25
[11]	{ground beef, low fat yogurt}	=>	{herb & pepper}	0.002399680	0.2500000	0.009598720	5.054582	18
[12]	{spaghetti, tomato sauce}	=>	{ground beef}	0.003066258	0.4893617	0.006265831	4.980600	23
[13]	{meatballs, spaghetti}	=>	{tomatoes}	0.002133049	0.3333333	0.006399147	4.873944	16
[14]	{light cream}	=>	{chicken}	0.004532729	0.2905983	0.015597920	4.843951	34
[15]	{frozen vegetables, herb & pepper}	=>	{ground beef}	0.002799627	0.4666667	0.005999200	4.749616	21
[16]	{mineral water, tomato sauce}	=>	{ground beef}	0.002666311	0.4651163	0.005732569	4.733836	20
[17]	{pasta}	=>	{escalope}	0.005865885	0.3728814	0.015731236	4.700812	44
[18]	{french fries, herb & pepper}	=>	{ground beef}	0.003199573	0.4615385	0.006932409	4.697422	24
[19]	{cereals, spaghetti}	=>	{ground beef}	0.003066258	0.4600000	0.006665778	4.681764	23
[20]	{french fries, ground beef}	=>	{herb & pepper}	0.003199573	0.2307692	0.013864818	4.665768	24
[21]	{chicken, ground beef}	=>	{herb & pepper}	0.002133049	0.2253521	0.009465405	4.556243	16
[22]	{grated cheese, ground beef}	=>	{herb & pepper}	0.002532996	0.2235294	0.011331822	4.519391	19
[23]	{pasta}	=>	{shrimp}	0.005065991	0.3220339	0.015731236	4.506672	38
[24]	{chocolate, herb & pepper}	=>	{ground beef}	0.003999467	0.4411765	0.009065458	4.490183	30
[25]	{chicken, herb & pepper}	=>	{ground beef}	0.002133049	0.4324324	0.004932676	4.401188	16
[26]	{cake, frozen vegetables}	=>	{tomatoes}	0.003066258	0.2987013	0.010265298	4.367560	23
[27]	{milk, tomatoes}	=>	{soup}	0.003066258	0.2190476	0.013998134	4.335293	23
[28]	{herb & pepper, shrimp}	=>	{ground beef}	0.002932942	0.4150943	0.007065725	4.224725	22
[29]	{eggs, ground beef}	=>	{herb & pepper}	0.004132782	0.2066667	0.019997334	4.178455	31
[30]	{milk, olive oil}	=>	{soup}	0.003599520	0.2109375	0.017064391	4.174781	27
[31]	{herb & pepper, low fat yogurt}	=>	{ground beef}	0.002399680	0.4090909	0.005865885	4.163624	18
[32]	{whole wheat pasta}	=>	{olive oil}	0.007998933	0.2714932	0.029462738	4.122410	60
[33]	{frozen vegetables, soup}	=>	{olive oil}	0.002133049	0.2666667	0.007998933	4.049123	16
[34]	{frozen smoothie, shrimp}	=>	{frozen vegetables}	0.002799627	0.3818182	0.007332356	4.005620	21
[35]	{herb & pepper, spaghetti}	=>	{ground beef}	0.006399147	0.3934426	0.016264498	4.004360	48
[36]	{herb & pepper, milk}	=>	{ground beef}	0.003599520	0.3913043	0.009198773	3.982597	27
[37]	{almonds, eggs}	=>	{burgers}	0.002266364	0.3469388	0.006532462	3.979186	17
[38]	{herb & pepper, mineral water}	=>	{ground beef}	0.006665778	0.3906250	0.017064391	3.975683	50
[39]	{grated cheese, herb & pepper}	=>	{ground beef}	0.002532996	0.3877551	0.006532462	3.946474	19
[40]	{frozen smoothie, frozen vegetables}	=>	{shrimp}	0.002799627	0.2800000	0.009998667	3.918433	21
[41]	{parmesan cheese, spaghetti}	=>	{frozen vegetables}	0.002532996	0.3725490	0.006799093	3.908378	19
[42]	{chocolate, ham}	=>	{burgers}	0.002133049	0.3404255	0.006265831	3.904483	16
[43]	{extra dark chocolate}	=>	{chicken}	0.002799627	0.2333333	0.011998400	3.889407	21
[44]	{cereals, ground beef}	=>	{spaghetti}	0.003066258	0.6764706	0.004532729	3.885303	23
[45]	{frozen vegetables, soup}	=>	{milk}	0.003999467	0.5000000	0.007998933	3.858539	30
[46]	{chicken, olive oil}	=>	{milk}	0.003599520	0.5000000	0.007199040	3.858539	27
[47]	{tomato sauce}	=>	{ground beef}	0.005332622	0.3773585	0.014131449	3.840659	40
[48]	{cake, tomatoes}	=>	{frozen vegetables}	0.003066258	0.3650794	0.008398880	3.830014	23
[49]	{milk, turkey}	=>	{whole wheat rice}	0.002532996	0.2235294	0.011331822	3.819349	19
[50]	{mushroom cream sauce}	=>	{escalope}	0.005732569	0.3006993	0.019064125	3.790833	43
[51]	{french fries, honey}	=>	{frozen smoothie}	0.002133049	0.2388060	0.008932142	3.771123	16
[52]	{barbecue sauce}	=>	{turkey}	0.002532996	0.2345679	0.010798560	3.751586	19
[53]	{cake, turkey}	=>	{burgers}	0.002266364	0.3269231	0.006932409	3.749618	17
[54]	{shrimp, tomatoes}	=>	{frozen vegetables}	0.003999467	0.3571429	0.011198507	3.746753	30
[55]	{chocolate, soup}	=>	{chicken}	0.002266364	0.2236842	0.010131982	3.728567	17
[56]	{grated cheese, pancakes}	=>	{ground beef}	0.002532996	0.3653846	0.006932409	3.718792	19
[57]	{herb & pepper, pancakes}	=>	{ground beef}	0.002666311	0.3636364	0.007332356	3.700999	20
[58]	{milk, whole wheat pasta}	=>	{olive oil}	0.002399680	0.2432432	0.009865351	3.693457	18
[59]	{chicken, milk}	=>	{olive oil}	0.003599520	0.2432432	0.014798027	3.693457	27
[60]	{olive oil, tomatoes}	=>	{frozen vegetables}	0.002532996	0.3518519	0.007199040	3.691246	19
[61]	{fresh tuna, mineral water}	=>	{olive oil}	0.002133049	0.2424242	0.008798827	3.681021	16
[62]	{avocado, spaghetti}	=>	{burgers}	0.002532996	0.3166667	0.007998933	3.631983	19
[63]	{almonds, spaghetti}	=>	{ground beef}	0.002133049	0.3555556	0.005999200	3.618755	16
[64]	{burgers, cookies}	=>	{green tea}	0.002666311	0.4761905	0.005599253	3.604344	20
[65]	{milk, soup}	=>	{olive oil}	0.003599520	0.2368421	0.015197974	3.596260	27
[66]	{french wine, ground beef}	=>	{spaghetti}	0.002399680	0.6206897	0.003866151	3.564926	18
[67]	{french wine, spaghetti}	=>	{burgers}	0.002399680	0.3103448	0.007732302	3.559475	18
[68]	{milk, olive oil}	=>	{chicken}	0.003599520	0.2109375	0.017064391	3.516094	27
[69]	{olive oil, tomatoes}	=>	{spaghetti}	0.004399413	0.6111111	0.007199040	3.509912	33
[70]	{frozen vegetables, shrimp}	=>	{tomatoes}	0.003999467	0.2400000	0.016664445	3.509240	30
[71]	{cottage cheese, mineral water}	=>	{frozen smoothie}	0.002133049	0.2222222	0.009598720	3.509240	16
[72]	{spaghetti, whole wheat pasta}	=>	{milk}	0.003999467	0.4545455	0.008798827	3.507763	30
[73]	{fresh tuna, spaghetti}	=>	{pancakes}	0.002266364	0.3333333	0.006799093	3.506779	17
[74]	{frozen vegetables, spaghetti}	=>	{tomatoes}	0.006665778	0.2392344	0.027862952	3.498046	50
[75]	{frozen vegetables, tomatoes}	=>	{shrimp}	0.003999467	0.2479339	0.016131183	3.469687	30
[76]	{burgers, pancakes}	=>	{turkey}	0.002266364	0.2151899	0.010531929	3.441661	17
[77]	{pepper, spaghetti}	=>	{ground beef}	0.003332889	0.3378378	0.009865351	3.438428	25
[78]	{chicken, soup}	=>	{milk}	0.002666311	0.4444444	0.005999200	3.429813	20
[79]	{mineral water, soup}	=>	{olive oil}	0.005199307	0.2254335	0.023063592	3.423030	39
[80]	{chicken, ground beef}	=>	{olive oil}	0.002133049	0.2253521	0.009465405	3.421794	16
[81]	{milk, whole wheat rice}	=>	{turkey}	0.002532996	0.2134831	0.011865085	3.414365	19
[82]	{soup, tomatoes}	=>	{milk}	0.003066258	0.4423077	0.006932409	3.413323	23
[83]	{ground beef, milk}	=>	{olive oil}	0.004932676	0.2242424	0.021997067	3.404944	37
[84]	{pancakes, soup}	=>	{ground beef}	0.002266364	0.3333333	0.006799093	3.392583	17
[85]	{frozen vegetables, green tea}	=>	{tomatoes}	0.003332889	0.2314815	0.014398080	3.384683	25

```
[86]  {extra dark chocolate}              => {olive oil}           0.002666311 0.2222222  0.011998400  3.374269  20
[87]  {eggs, herb & pepper}               => {ground beef}         0.004132782 0.3297872  0.012531662  3.356491  31
[88]  {spaghetti, tomatoes}               => {frozen vegetables}   0.006665778 0.3184713  0.020930543  3.341054  50
[89]  {eggs, whole wheat pasta}           => {milk}                0.002133049 0.4324324  0.004932676  3.337115  16
[90]  {burgers, french wine}              => {spaghetti}           0.002399680 0.5806452  0.004132782  3.334931  18
[91]  {fresh bread, mineral water}        => {chicken}             0.002666311 0.2000000  0.013331556  3.333778  20
[92]  {ground beef, tomato sauce}         => {spaghetti}           0.003066258 0.5750000  0.005332622  3.302508  23
[93]  {herb & pepper}                     => {ground beef}         0.015997867 0.3234501  0.049460072  3.291994 120
[94]  {grated cheese, spaghetti}          => {ground beef}         0.005332622 0.3225806  0.016531129  3.283144  40
[95]  {cooking oil, ground beef}          => {spaghetti}           0.004799360 0.5714286  0.008398880  3.281995  36
[96]  {black tea, mineral water}          => {milk}                0.002266364 0.4250000  0.005332622  3.279758  17
[97]  {light cream, spaghetti}            => {milk}                0.002266364 0.4250000  0.005332622  3.279758  17
[98]  {grated cheese, milk}               => {ground beef}         0.002532996 0.3220339  0.007865618  3.277580  19
[99]  {frozen vegetables, olive oil}      => {tomatoes}            0.002532996 0.2235294  0.011331822  3.268410  19
[100] {frozen vegetables, olive oil}      => {milk}                0.004799360 0.4235294  0.011331822  3.268410  36
[101] {chocolate, red wine}               => {spaghetti}           0.002799627 0.5675676  0.004932676  3.259820  21
[102] {light cream, milk}                 => {spaghetti}           0.002266364 0.5666667  0.003999467  3.254645  17
[103] {chocolate, frozen vegetables}      => {shrimp}              0.005332622 0.2325581  0.022930276  3.254512  40
[104] {fresh bread, milk}                 => {shrimp}              0.002133049 0.2318841  0.009198773  3.245079  16
[105] {milk, turkey}                      => {burgers}             0.003199573 0.2823529  0.011331822  3.238424  24
[106] {gluten free bar}                   => {pancakes}            0.002133049 0.3076923  0.006932409  3.237027  16
[107] {frozen vegetables, soup}           => {ground beef}         0.002532996 0.3166667  0.007998933  3.222953  19
[108] {avocado, spaghetti}                => {milk}                0.003332889 0.4166667  0.007998933  3.215449  25
[109] {cake, milk}                        => {burgers}             0.003732836 0.2800000  0.013331556  3.211437  28
[110] {avocado, burgers}                  => {spaghetti}           0.002532996 0.5588235  0.004532729  3.209598  19
[111] {mineral water, vegetables mix}     => {frozen vegetables}   0.002399680 0.3050847  0.007865618  3.200616  18
[112] {mineral water, shrimp}             => {frozen vegetables}   0.007199040 0.3050847  0.023596854  3.200616  54
[113] {spaghetti, tomatoes}               => {olive oil}           0.004399413 0.2101911  0.020930543  3.191586  33
[114] {cooking oil, frozen vegetables}    => {milk}                0.002799627 0.4117647  0.006799093  3.177620  21
[115] {chocolate, tomato sauce}           => {spaghetti}           0.002799627 0.5526316  0.005065991  3.174035  21
[116] {spaghetti, whole wheat rice}       => {tomatoes}            0.003066258 0.2169811  0.014131449  3.172662  23
[117] {ground beef, soup}                 => {milk}                0.003999467 0.4109589  0.009732036  3.171402  30
[118] {meatballs, tomatoes}               => {spaghetti}           0.002133049 0.5517241  0.003866151  3.168823  16
[119] {milk, parmesan cheese}             => {spaghetti}           0.002133049 0.5517241  0.003866151  3.168823  16
[120] {almonds, ground beef}              => {spaghetti}           0.002133049 0.5517241  0.003866151  3.168823  16
[121] {fresh bread, shrimp}               => {milk}                0.002133049 0.4102564  0.005199307  3.165981  16
[122] {frozen vegetables, spaghetti}      => {ground beef}         0.008665511 0.3110048  0.027862952  3.165328  65
[123] {honey, milk}                       => {burgers}             0.002133049 0.2758621  0.007732302  3.163978  16
[124] {mineral water, rice}               => {ground beef}         0.002399680 0.3103448  0.007732302  3.158611  18
[125] {french wine, spaghetti}            => {ground beef}         0.002399680 0.3103448  0.007732302  3.158611  18
```

C - Select the rule from (B) with the greatest lift. Compare this rule with the highest lift rule for maximum length of 2.

- Highest lift rule for maximum length of 2:

```
rule1 <- sort(r1, by = "lift",decreasing = TRUE)[0:1]

r2 <- apriori(t, parameter = list(support = 0.002, confidence =0.20, maxlen = 2))
rule2 <- sort(r2, by = "lift",decreasing = TRUE)[0:1]
```

```
if(rule1@quality$lift > rule2@quality$lift){
  print("Rule1 with maximum length of 3 has a better Lift")
}else{
  print("Rule2 with maximum length of 2 has a better Lift")
}


if(rule1@quality$support > rule2@quality$support){
  print("Rule1 with maximum length of 3 has a greater Support")
}else{
  print("Rule2 with maximum length of 2 has a greater Support")
}
```

i) Comparing the two rules will tell us that the best lift was Rule1 (Maximum length is 3) and the best support was Rule2 (Maximum length is 2).

ii) I would choose Rule1 because its lift is much bigger than the lift of Rule 2 and the support is very close to it.

```
> print(rule1@quality$lift)
[1] 28.0881
> print(rule2@quality$lift)
[1] 5.164271
>
> print(rule1@quality$support)
[1] 0.002532996
> print(rule2@quality$support)
[1] 0.003332889
```