



# METEORITES LANDINGS

Olatounde Bachabi

## Contents

I-	Introduction .....	3
II-	Data Analytic Process .....	3
1-	Data Manipulation .....	3
2-	Data Visualisation .....	11
3-	Statistical Analysis .....	14
III-	Decision Trees .....	16
IV-	SAS Enterprise Miner: Clustering .....	21
V-	Conclusion .....	25
VI-	Reference .....	25

# **Assessment Title: ASDM Assignment: Data Mining using SAS and R**

## **Master of Science**

**Student: Olatounde Kabirou Brice Bachabi**

**Tutors: Dr Mo Saraee & Dr Yuhua Li**

## I- Introduction

This is a comparative study of the results of association rules, classification and decision tree in R language and Clustering in SAS Enterprise Miner. We have one data set which will be used for both R programming and SAS Enterprise Miner. The Data is based on Meteorite.

A meteorite is a thick or solid piece of rock or iron generally form meteoroid or asteroid which pass through earth's atmosphere and survives impact with the ground.

These meteorite are recorded in two ways. If they have been observed in their falling then are be marked as "fell" and the falling is not be observed then they are marked as "found". The other aspect of the meteorites which we are interested in in the assignment is the name type which is divided in two groups also. The "valid": when the meteorite is intact and the "relict" when the meteorite is seriously degraded by weather on earth.

According to the Meteorite Society it's not easy to classify the meteorite and there are so many classes. They have been found in different places around the world. Some of these places are not accessible. Years they have been found are a major issue too. So to make things not complicated as they already are and in terms of accuracy of the values within the variables we will concentrate on the falling and the geographical location of where they have been found.

Does a meteorite assigns "fell" to which is the one observed in its falling it's always intact ("valid")? To answer the above question we need to some adjustment call data mining on the data set we have got. Some of the denomination will be changed and some of the variables will be removed.

## II- Data Analytic Process

This is the process of transforming the data set into understanding, knowlledge and insight. There are a couple of steps to be followed to transform the data.

### 1- Data Manipulation

Data manipulation is the long and very important step in data mining. This step takes about 80% of the workload time and consist of cleaning and preparing the data set. If data manipulation is done correctly then the data will be properly structured so will be easy accessible and usable with the software or any tool we want to use in the data mining process.

The figure 1 below is the structure of the raw data we are going to be working on. They are mixte and match of variables classes. We have 45716 observations and 20 variables. Some of the rows have NA values which is a sign of missing or incorrect values.

```

R Console
[1] "C:/Worshops"
> mydata<-read.csv("meteoritel.csv", header=TRUE)
> str(mydata)
'data.frame':   45716 obs. of  20 variables:
 $ name       : Factor w/ 45716 levels "Ã-sterplana 002",...: 68 69 73 77 473 48$
 $ id         : int   1 2 6 10 370 379 390 392 398 417 ...
 $ nametype   : Factor w/ 2 levels "Relict","Valid": 2 2 2 2 2 2 2 2 2 2 ...
 $ recclass   : Factor w/ 466 levels "Acapulcoite",...: 333 197 85 1 339 85 360 $
 $ mass       : num   21 720 107000 1914 780 ...
 $ fall       : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ year       : int  1880 1951 1952 1976 1902 1919 1949 1814 1930 1920 ...
 $ reclat     : num   50.8 56.2 54.2 16.9 -33.2 ...
 $ reclong    : num    6.08 10.23 -113 -99.9 -64.95 ...
 $ GeoLocation: Factor w/ 17101 levels "", "(-1.002780, 37.150280)",...: 16779 16$
 $ X          : logi   NA NA NA NA NA NA ...
 $ X.1        : logi   NA NA NA NA NA NA ...
 $ X.2        : Factor w/ 2 levels "", "Ãspinal": 1 1 1 1 1 1 1 1 1 1 ...
 $ X.3        : Factor w/ 5 levels "", "Fell", "Found",...: 2 3 5 4 1 1 1 1 1 1 ...
 $ X.4        : logi   NA NA NA NA NA NA ...
 $ Frequency   : int  1107 44609 45641 75 NA NA NA NA NA NA ...
 $ X.5        : logi   NA NA NA NA NA NA ...
 $ reclass1    : Factor w/ 35 levels "", "Acapulcoite",...: 26 25 16 11 2 14 27 15$
 $ X.6        : logi   NA NA NA NA NA NA ...
 $ Frequency.1: int   8285 4796 4528 18 54 4211 9 7142 50 11 ...
>

```

Figure 1: Structure of Raw Data

The next figure, Figure 2 is the summary of the raw data showing some of the missing values too. We can also see that some of the means are too far higher than the median and there are a big gap between the mins and max too.

```

R Console
      reclong      GeoLocation      X
Min.   :-165.43              : 7315  Mode:logical
1st Qu.:  0.00      (0.000000, 0.000000) : 6214  NA's:45716
Median : 35.67      (-71.500000, 35.666670) : 4761
Mean   : 61.07      (-84.000000, 168.000000) : 3040
3rd Qu.: 157.17      (-72.000000, 26.000000) : 1505
Max.   : 354.47      (-79.683330, 159.750000) : 657
NA's   : 7315      (Other)                :22224

      X.1      X.2      X.3      X.4      Frequency
Mode:logical :45715      :45712  Mode:logical Min.   : 75
NA's:45716    Ãspinal: 1    Fell : 1    NA's:45716 1st Qu.: 849
                                          Median :22858
                                          Mean   :22858
                                          3rd Qu.:44867
                                          Max.   :45641
                                          NA's   :45712

      X.5      reclass1      X.6      Frequency.1
Mode:logical :45682      Mode:logical Min.   : 0.0
NA's:45716    Acapulcoite: 1    NA's:45716 1st Qu.: 18.5
      Angrite : 1
      Aubrite : 1
      C2-ung  : 1
      Cl1     : 1
      (Other) : 29
                                          Median : 74.0
                                          Mean   :1095.8
                                          3rd Qu.: 403.2
                                          Max.   :8285.0
                                          NA's   :45682

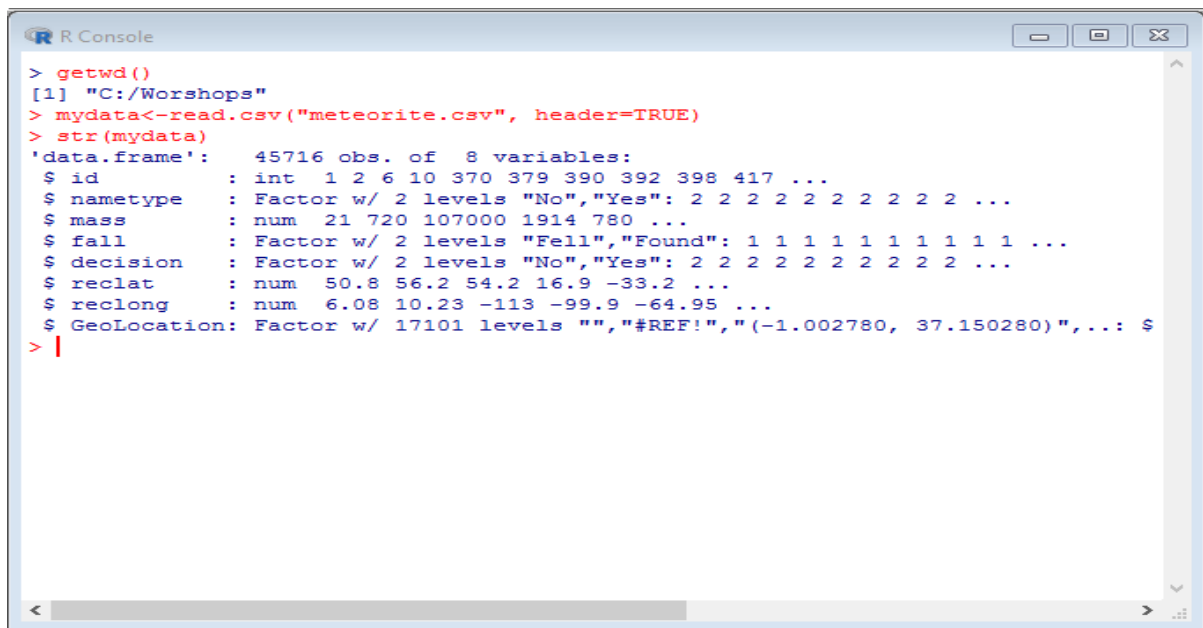
```

Figure 2: Summary of Raw Data

In our process to deal with the data correctly and to answer the question above, some of the variables are not needed so we can easily get rid of them. So we have removed some of them which

you can see in the next figure. We have changed the variable “fall” from “fell” to “yes” and from “found” to “no” . The variable nametype values also have been changed from “valid” to “yes and from “relict” to “no” for convenience of association rules. This can be done through R but we preferred to use microsoft excel.

The figure 3 below shown the structure of the raw data. The data set has not been touch yet. We can clearly see that we have 45716 observations and 8 variables. Most of the NA values have gone but we still have interger and numerical classes for the variables whose are not compatible with the association rules.



```

R Console
> getwd()
[1] "C:/Worshops"
> mydata<-read.csv("meteorite.csv", header=TRUE)
> str(mydata)
'data.frame':   45716 obs. of  8 variables:
 $ id       : int   1 2 6 10 370 379 390 392 398 417 ...
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : num   21 720 107000 1914 780 ...
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : num   50.8 56.2 54.2 16.9 -33.2 ...
 $ reclong  : num    6.08 10.23 -113 -99.9 -64.95 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!","", "(-1.002780, 37.150280)",...: $
>

```

Figure 3: New Structure

The figure 4 below is the summary of our raw data. The summary gives us a brief overview of the configuration of the variables. We may spot any irregularity.

Through the summary we can find out about outliers by observing the mean and the median. If the median is far lower than the mean then there is outlier by passing. Normally the mean is approximately equal to the median.

We can also see from the min and the max if there is any that there is irregularity.

```

$ mass      : num  21 720 107000 1914 780 ...
$ fall      : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 ...
$ decision  : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 ...
$ reclat    : num  50.8 56.2 54.2 16.9 -33.2 ...
$ reclang   : num   6.08 10.23 -113 -99.9 -64.95 ...
$ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
>
> summary(mydata)
      id      nametype      mass      fall      decision
Min.   : 1      No : 75   Min.   : 0      Fell : 1107   No :44609
1st Qu.:12689   Yes:45641  1st Qu.: 7      Found:44609  Yes: 1107
Median :24262           Median : 33
Mean   :26890           Mean   : 13278
3rd Qu.:40657           3rd Qu.: 203
Max.   :57458           Max.   :60000000
NA's   :131
      reclat      reclang      GeoLocation
Min.   :-87.37   Min.   :-165.43           : 7315
1st Qu.: -76.71  1st Qu.:  0.00   (0.000000, 0.000000) : 6214
Median : -71.50  Median : 35.67   (-71.500000, 35.666670) : 4761
Mean   : -39.12  Mean   : 61.07   (-84.000000, 168.000000) : 3040
3rd Qu.:  0.00  3rd Qu.: 157.17   (-72.000000, 26.000000) : 1505
Max.   : 81.17  Max.   : 354.47   (-79.683330, 159.750000) : 657
NA's   :7315   NA's   :7315   (Other)                :22224
>

```

Figure 4: New Summary

The algorithm `sum(is.na(mydata))` will be used to find out if there is any missing values. It's clearly showed that there 14761 missing values as illustrated in figure 5 below. In those 14761 missing values we have 131, 7315 and 7315 missing values for the variables `mass`; `reclat` and `reclang` respectively.

```

1st Qu.: -76.71   1st Qu.:  0.00   (0.000000, 0.000000) : 6214
Median : -71.50   Median : 35.67   (-71.500000, 35.666670) : 4761
Mean   : -39.12   Mean   : 61.07   (-84.000000, 168.000000) : 3040
3rd Qu.:  0.00   3rd Qu.: 157.17   (-72.000000, 26.000000) : 1505
Max.   : 81.17   Max.   : 354.47   (-79.683330, 159.750000) : 657
NA's   :7315   NA's   :7315   (Other)                :22224
> sum(is.na(mydata))
[1] 14761
> sum(is.na(mydata$id))
[1] 0
> sum(is.na(mydata$nametype))
[1] 0
> sum(is.na(mydata$fall))
[1] 0
> sum(is.na(mydata$mass))
[1] 131
> sum(is.na(mydata$reclat))
[1] 7315
> sum(is.na(mydata$reclang))
[1] 7315
> sum(is.na(mydata$GeoLocation))
[1] 0
> sum(is.na(mydata$decision))
[1] 0
>

```

Figure 5: Missing Values

To treat the missing or incorrect values the packages “magrittr” and “dplyr” will be installed. These packages will help us to sift the data. In our ready we have found out that any date before 860 CE and after 2016 can be treated as missing or incorrect value. As we don’t need the years in our variables this has been purely and simply removed. There are also issues about the longitude and the latitude of certain regions or part of a continent or even country where the meteorites have been found. Any longitude and latitude equal to zero will be removed and treated as missing value.

There is algorithm we have found on the website whose can remove all those missing values using “dplyr”.

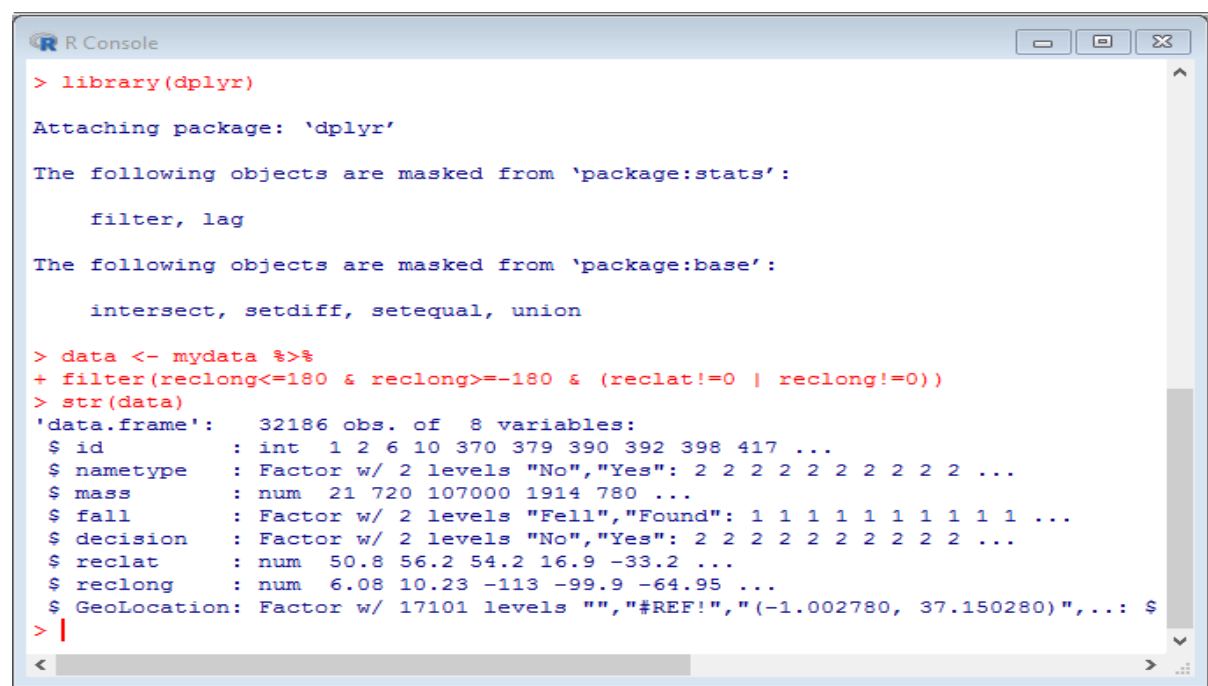
```
Meteorites.goe <- meteorites.all %>%
```

```
Filter (reclong <= 180 & reclong >= -180 & (reclat != 0 | reclong != 0)).
```

After ran this algorithm the data structure is showing 32186 observations so the missing values are gone. See figure 6.

Arithmetically there are still missing values somewhere as 32186 out of 45716 equal to 13530 while the process of finding the missing values showed us that there are 14761.

Running again the algorithm will be a long process. It will be ideal to use the omit synthase to remove any eventual left over of missing values.



```

R Console
> library(dplyr)
Attaching package: 'dplyr'
The following objects are masked from 'package:stats':
  filter, lag
The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union
> data <- mydata %>%
+ filter(reclong <= 180 & reclong >= -180 & (reclat != 0 | reclong != 0))
> str(data)
'data.frame': 32186 obs. of 8 variables:
 $ id      : int  1 2 6 10 370 379 390 392 398 417 ...
 $ nametype: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 ...
 $ mass    : num  21 720 107000 1914 780 ...
 $ fall    : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 ...
 $ decision: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 ...
 $ reclat  : num  50.8 56.2 54.2 16.9 -33.2 ...
 $ reclong : num  6.08 10.23 -113 -99.9 -64.95 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ... $
>

```

Figure 6:

After ran the omit algorithm: dataset <- na.omit(data) there is no more missing values as shown in the figure 7.



```

$ reclong      : num  6.08 10.23 -113 -99.9 -64.95 ...
$ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
> dataset <- na.omit(data)
> sum(is.na(dataset))
[1] 0
> |

```

Figure 7

The association rules is a very complex rules which needs certain conditions before it can be implemented. One of these conditions is that the classes of the variables must be factor or logical if you want to use apriori rules like in our assignment.

We need to change all the variables to factor as logical class is not compatible with our case or will make things very complicated. We have used the `colClasses = "factor"` to change all the classes which are not factor.

Figure 8 below showed that all the classes are now factors.

```

R Console
> data <- mydata %>%
+ filter(reclong<=180 & reclong>=-180 & (reclat!=0 | reclong!=0))
> str(data)
'data.frame':  32186 obs. of  8 variables:
 $ id      : int  1 2 6 10 370 379 390 392 398 417 ...
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : num  21 720 107000 1914 780 ...
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : num  50.8 56.2 54.2 16.9 -33.2 ...
 $ reclong  : num  6.08 10.23 -113 -99.9 -64.95 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
>
> dataset <- read.csv("meteorite.csv", header=T, colClasses="factor")
> str(dataset)
'data.frame':  45716 obs. of  8 variables:
 $ id      : Factor w/ 45716 levels "1","10","100",...: 1 10885 41301 2 25727$
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : Factor w/ 12577 levels "", "0", "0.01",...: 4268 10886 637 3739 11$
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : Factor w/ 12739 levels "", "-1.00278",...: 12469 12624 12579 6239$
 $ reclong  : Factor w/ 14641 levels "", "-0.0225", "-0.03333",...: 13478 2700 5$
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
> |

```

Figure 8: Factors

The Arules packages need to be installed so we can implement the apriori association rules. The rules have been created as shown in figure 9. There are 480 rules. These rules have generated as default.

```

R Console

abbreviate, write

> rules <- apriori(dataset)
Apriori

Parameter specification:
confidence minval smax arem aval originalSupport maxtime support minlen
0.8 0.1 1 none FALSE TRUE 5 0.1 1
maxlen target ext
10 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 4571

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[102780 item(s), 45716 transaction(s)] done [0.07s].
sorting and recoding items ... [12 item(s)] done [0.01s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.00s].
writing ... [480 rule(s)] done [0.00s].
creating S4 object ... done [0.01s].
> |

```

Figure 9: Apriori Rules

Now we need to summarise the rules using a line code. We can that there are 3 rules in item set 1 and 18 rules in 6. Figure 10.

```

R Console

> sum(rules)
Error in sum(rules) : invalid 'type' (S4) of argument
> summary(rules)
set of 480 rules

rule length distribution (lhs + rhs):sizes
 1  2  3  4  5  6
3 51 147 171 90 18

    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000  3.000  4.000  3.725  4.000  6.000

summary of quality measures:
      support      confidence      lift
Min.   :0.1041  Min.   :0.9551  Min.   :1.000
1st Qu.:0.1041  1st Qu.:0.9998  1st Qu.:1.024
Median :0.1359  Median :1.0000  Median :1.025
Mean   :0.1546  Mean   :0.9970  Mean   :3.983
3rd Qu.:0.1597  3rd Qu.:1.0000  3rd Qu.:7.101
Max.   :0.9984  Max.   :1.0000  Max.   :9.602

mining info:
      data ntransactions support confidence
dataset          45716      0.1      0.8
> |

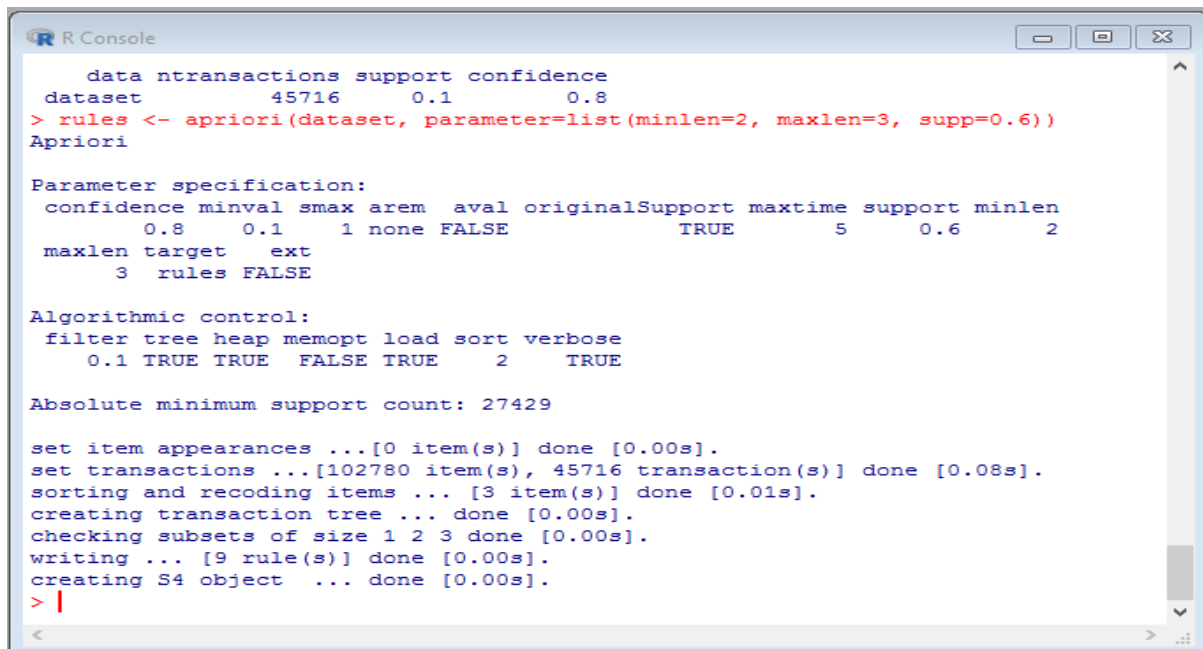
```

Figure 10: Rules Summarised

We can reduce the rules by fixing a parameter. We can define how many item we want minimum in one item set and the maximum. And also fix the support to follow those rules. The minimum length

of the item set has been set to 2 and the maximum length to 3. We have set the support to 60 percent. Figure 11: we can see that the rules have been reduced to 9 from 480 rules.

Then we will inspect the rules to see how items are put together in figure 12 using code line of the inspection.



```

R Console
data ntransactions support confidence
dataset      45716      0.1      0.8
> rules <- apriori(dataset, parameter=list(minlen=2, maxlen=3, supp=0.6))
Apriori

Parameter specification:
confidence minval smax arem aval originalSupport maxtime support minlen
0.8         0.1    1 none FALSE          TRUE         5      0.6      2
maxlen target  ext
3      rules FALSE

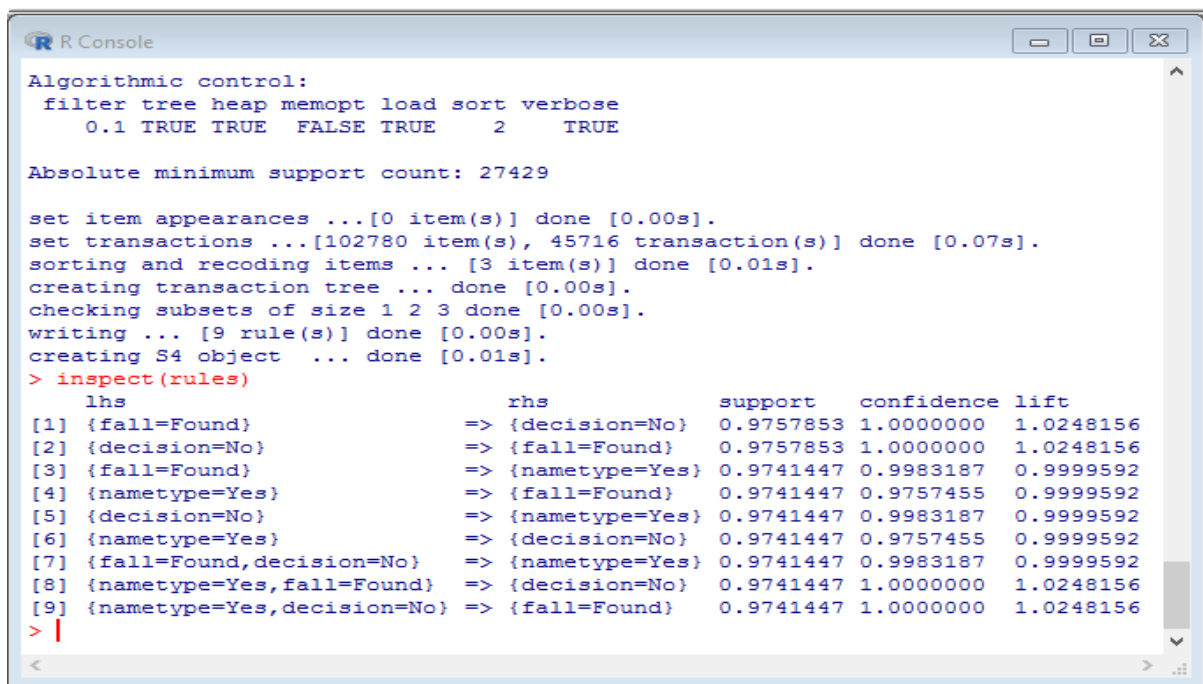
Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2      TRUE

Absolute minimum support count: 27429

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[102780 item(s), 45716 transaction(s)] done [0.08s].
sorting and recoding items ... [3 item(s)] done [0.01s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [9 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
>

```

Figure 11: Reduced Rules



```

R Console

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2      TRUE

Absolute minimum support count: 27429

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[102780 item(s), 45716 transaction(s)] done [0.07s].
sorting and recoding items ... [3 item(s)] done [0.01s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [9 rule(s)] done [0.00s].
creating S4 object ... done [0.01s].
> inspect(rules)
      lhs                                rhs      support  confidence lift
[1] {fall=Found} => {decision=No} 0.9757853 1.0000000 1.0248156
[2] {decision=No} => {fall=Found} 0.9757853 1.0000000 1.0248156
[3] {fall=Found} => {nametype=Yes} 0.9741447 0.9983187 0.9999592
[4] {nametype=Yes} => {fall=Found} 0.9741447 0.9757455 0.9999592
[5] {decision=No} => {nametype=Yes} 0.9741447 0.9983187 0.9999592
[6] {nametype=Yes} => {decision=No} 0.9741447 0.9757455 0.9999592
[7] {fall=Found,decision=No} => {nametype=Yes} 0.9741447 0.9983187 0.9999592
[8] {nametype=Yes,fall=Found} => {decision=No} 0.9741447 1.0000000 1.0248156
[9] {nametype=Yes,decision=No} => {fall=Found} 0.9741447 1.0000000 1.0248156
>

```

Figure 12: Rules Inspection

```

R Console

Apriori

Parameter specification:
confidence minval smax arem aval originalSupport maxtime support minlen
0.6 0.1 1 none FALSE TRUE 5 0.1 2
maxlen target ext
3 rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 4571

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[102780 item(s), 45716 transaction(s)] done [0.08s].
sorting and recoding items ... [12 item(s)] done [0.01s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [39 rule(s)] done [0.00s].
creating S4 object ... done [0.01s].
Warning message:
In apriori(dataset, parameter = list(minlen = 2, maxlen = 3, conf = 0.6), :
Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
>

```

Figure 13: Apriori

```

R Console

> inspect(rules)

```

	lhs	rhs	support	confiden\$
[1]	{reclat=-71.5}	=> {nametype=Yes}	0.1041430	1.000000\$
[2]	{GeoLocation=(-71.500000, 35.666670)}	=> {nametype=Yes}	0.1041430	1.000000\$
[3]	{reclong=35.66667}	=> {nametype=Yes}	0.1090428	1.000000\$
[4]	{reclong=0}	=> {nametype=Yes}	0.1359262	1.000000\$
[5]	{GeoLocation=(0.000000, 0.000000)}	=> {nametype=Yes}	0.1359262	1.000000\$
[6]	{reclat=0}	=> {nametype=Yes}	0.1408260	1.000000\$
[7]	{GeoLocation=}	=> {nametype=Yes}	0.1599003	0.99931\$
[8]	{reclong=}	=> {nametype=Yes}	0.1599003	0.99931\$
[9]	{reclat=}	=> {nametype=Yes}	0.1599003	0.99931\$
[10]	{fall=Found}	=> {nametype=Yes}	0.9741447	0.99831\$
[11]	{decision=No}	=> {nametype=Yes}	0.9741447	0.99831\$
[12]	{reclat=-71.5, GeoLocation=(-71.500000, 35.666670)}	=> {nametype=Yes}	0.1041430	1.000000\$
[13]	{reclat=-71.5, reclong=35.66667}	=> {nametype=Yes}	0.1041430	1.000000\$
[14]	{fall=Found, reclat=-71.5}	=> {nametype=Yes}	0.1041430	1.000000\$
[15]	{decision=No, reclat=-71.5}	=> {nametype=Yes}	0.1041430	1.000000\$
[16]	{reclong=35.66667, GeoLocation=(-71.500000, 35.666670)}	=> {nametype=Yes}	0.1041430	1.000000\$
[17]	{fall=Found, GeoLocation=(-71.500000, 35.666670)}	=> {nametype=Yes}	0.1041430	1.000000\$

Figure 14: Final Rules

## 2- Data Visualisation

After the rules have been set up. Now we need to visualise the rules by plotting, graphing and much more.

We need install the ArulesViz packages so we can project the visually the rules. We are going to plot the rules (figure 15) first of all by using the code line: plot(rules).

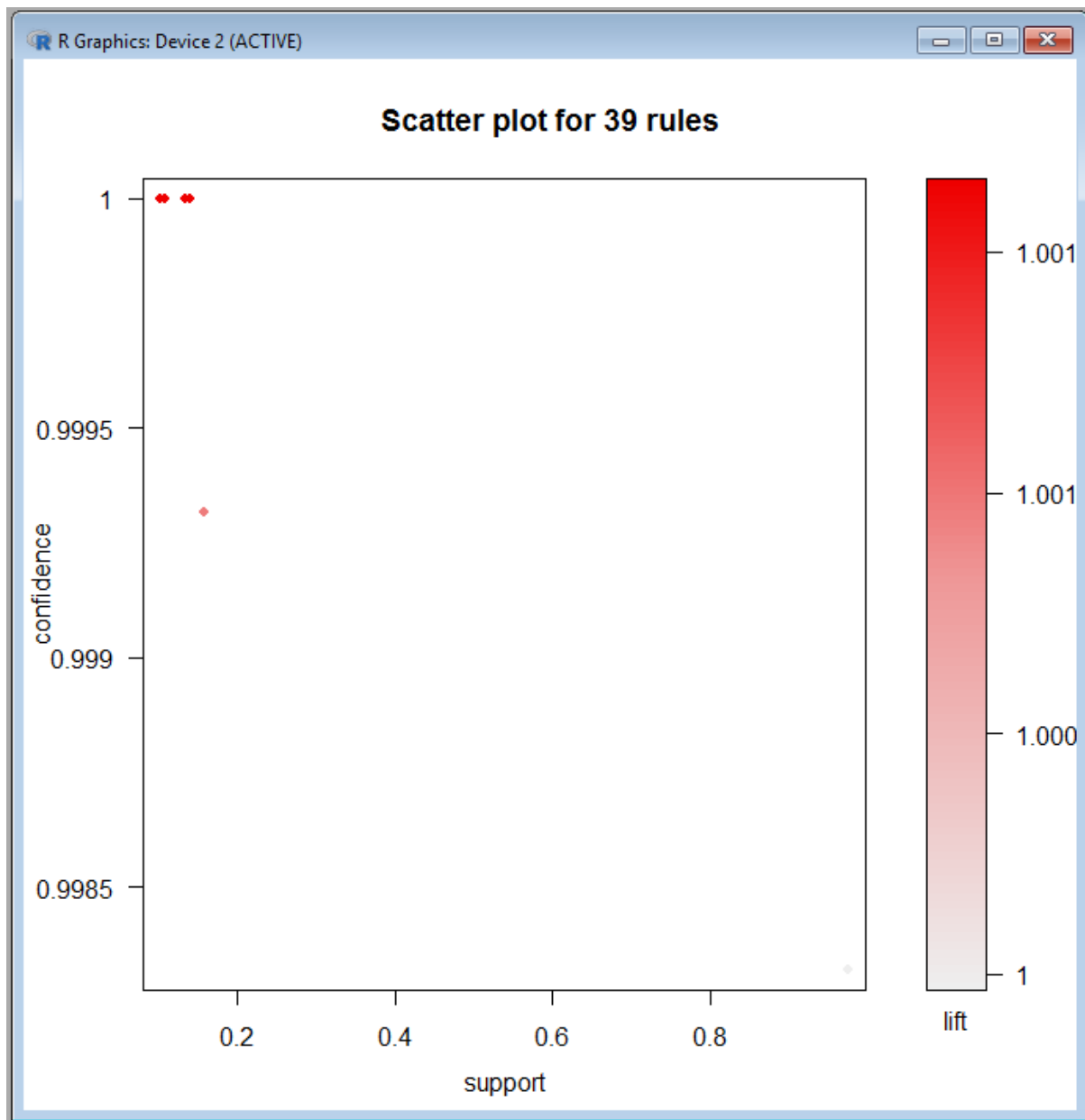


Figure 15: Scatter plot for 39 rules.

We can also plot in groups by using the plot (rules, method = "grouped") as shown in figure 16

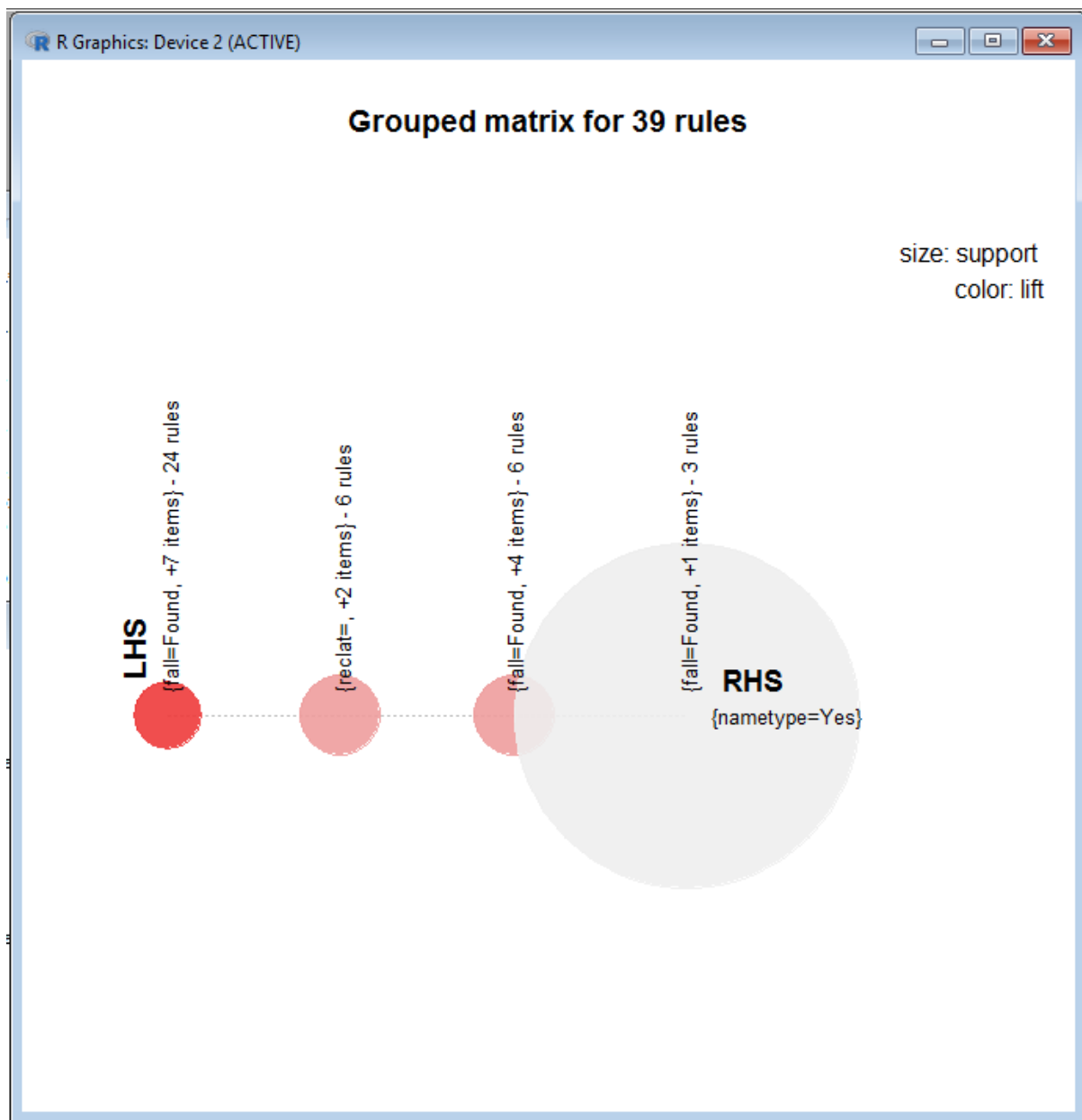


Figure 16: Grouped matrix for 39 rules

We can also create a graph to see these rules by using the code line of method.

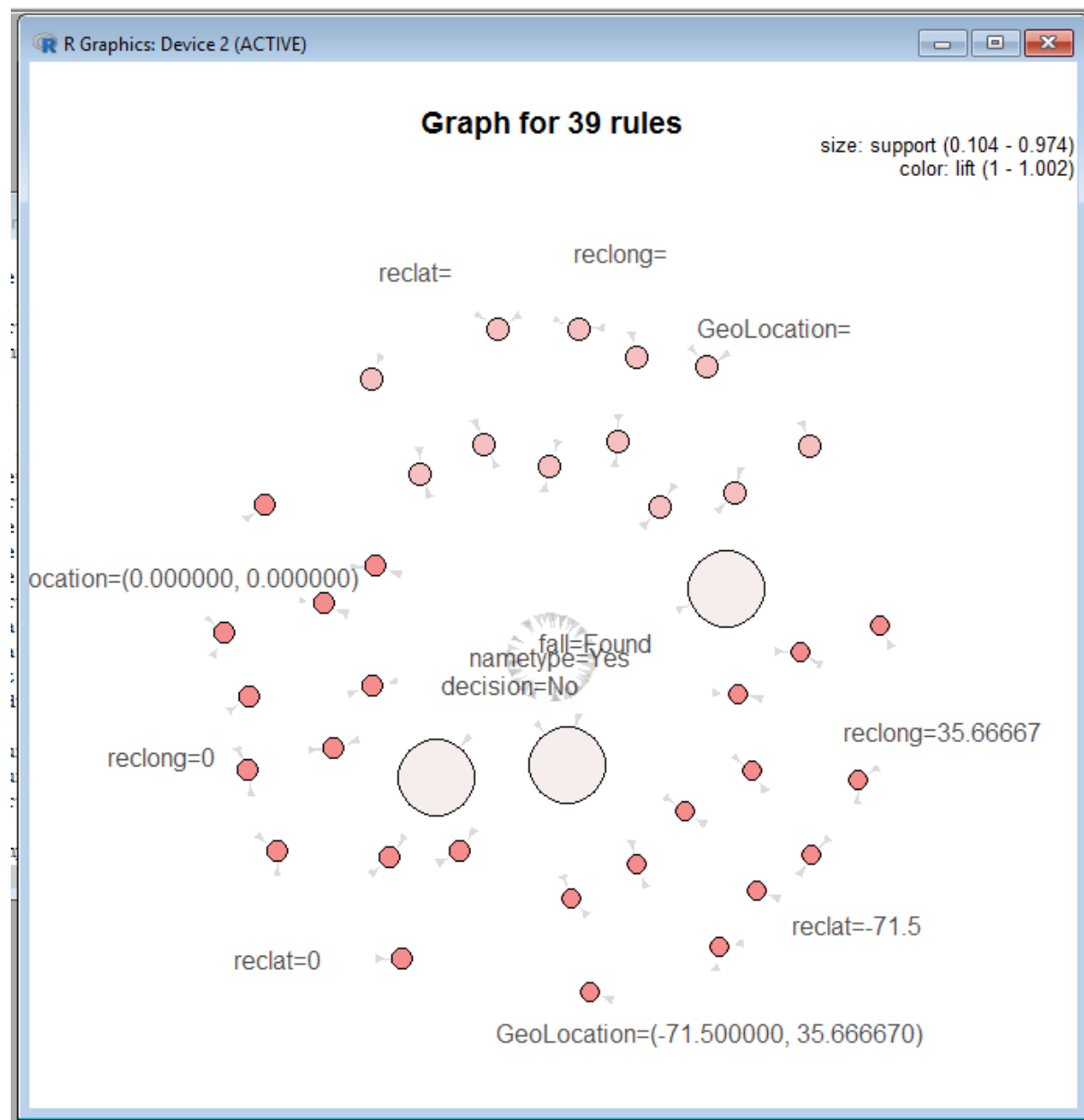


Figure 17: Graph for 39 rules

### 3- Statistical Analysis

In figure 14 we have noticed that the rule 1 is an independent rule with no extra help from the rule 2 as the rule 1 tell us that all the meteorite in the range of the reclat are valid. Such rule like rule 2 is a super rule to rule 1 and has the same or lower lift is considered to be redundant. So we will need to prune redundant rules.

After ran the line code we have noticed that 28 rules are redundant (figure 18)

```

R Console
> subset.matrix <- is.subset(rules, rules)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
> which(redundant)
      {nametype=Yes, reclat=-71.5, GeoLocation=(-71.500000, 35.666670)}
      12
      {nametype=Yes, reclat=-71.5, reclong=35.66667}
      13
      {nametype=Yes, fall=Found, reclat=-71.5}
      14
      {nametype=Yes, decision=No, reclat=-71.5}
      15
{nametype=Yes, reclong=35.66667, GeoLocation=(-71.500000, 35.666670)}
      16
      {nametype=Yes, fall=Found, GeoLocation=(-71.500000, 35.666670)}
      17
      {nametype=Yes, decision=No, GeoLocation=(-71.500000, 35.666670)}
      18
      {nametype=Yes, fall=Found, reclong=35.66667}
      19
      {nametype=Yes, decision=No, reclong=35.66667}
      20
      {nametype=Yes, reclong=0, GeoLocation=(0.000000, 0.000000)}
      21
      {nametype=Yes, reclat=0, reclong=0}

```

Figure 18: redundant

To remove the redundant rules the line code:

```
Rules.pruned <- rules[!redundant]
```

```
Rules.pruned <- sorte(rules.pruned, by="lift")Inspect(rules.pruned)
```

```

R Console

      lhs                                rhs                                support
[1] {reclat=-71.5}                      => {nametype=Yes} 0.1041430
[2] {GeoLocation=(-71.500000, 35.666670)} => {nametype=Yes} 0.1041430
[3] {reclong=35.66667}                  => {nametype=Yes} 0.1090428
[4] {reclong=0}                         => {nametype=Yes} 0.1359262
[5] {GeoLocation=(0.000000, 0.000000)} => {nametype=Yes} 0.1359262
[6] {reclat=0}                          => {nametype=Yes} 0.1408260
[7] {GeoLocation=}                      => {nametype=Yes} 0.1599003
[8] {reclong=}                          => {nametype=Yes} 0.1599003
[9] {reclat=}                           => {nametype=Yes} 0.1599003
[10] {fall=Found}                       => {nametype=Yes} 0.9741447
[11] {decision=No}                      => {nametype=Yes} 0.9741447

confidence lift
[1] 1.0000000 1.0016433
[2] 1.0000000 1.0016433
[3] 1.0000000 1.0016433
[4] 1.0000000 1.0016433
[5] 1.0000000 1.0016433
[6] 1.0000000 1.0016433
[7] 0.9993165 1.0009586
[8] 0.9993165 1.0009586
[9] 0.9993165 1.0009586
[10] 0.9983187 0.9999592
[11] 0.9983187 0.9999592
> |

```

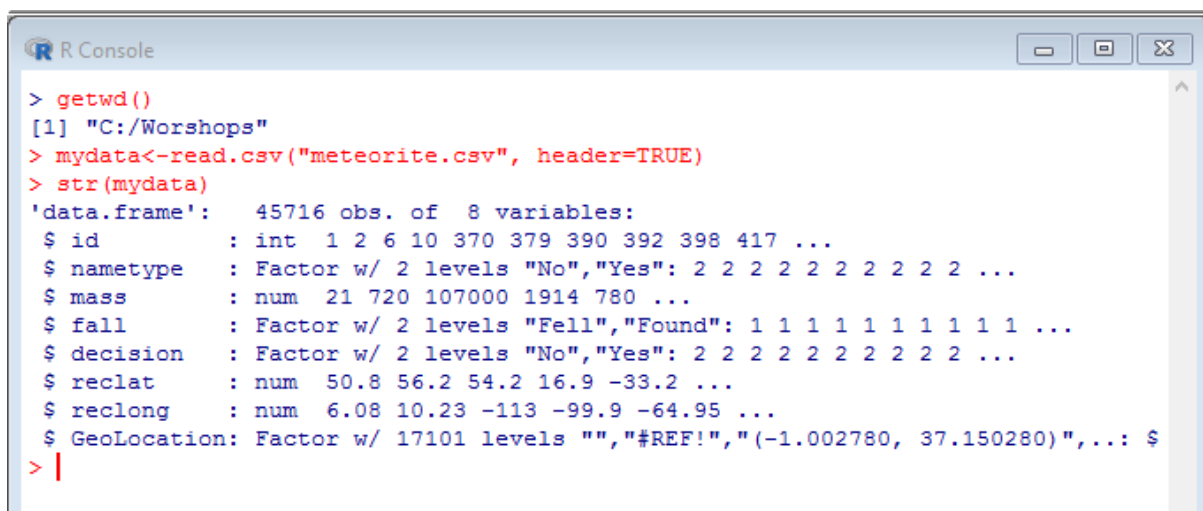
Figure 19: New rules



From the above figure we can conclude that all the meteorite assign “fell” to as the have been observed in their falling are high likely to be intact “valid”.

### III- Decision Trees

We are using the same data set. We already know that there are 45716 observations and 8 variables. We also have numerical, integer and logical classes.



```
> getwd()
[1] "C:/Worshops"
> mydata<-read.csv("meteorite.csv", header=TRUE)
> str(mydata)
'data.frame':  45716 obs. of  8 variables:
 $ id      : int  1 2 6 10 370 379 390 392 398 417 ...
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : num  21 720 107000 1914 780 ...
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : num  50.8 56.2 54.2 16.9 -33.2 ...
 $ reclong  : num  6.08 10.23 -113 -99.9 -64.95 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
> |
```

Figure 1: Structure

A formula need to be created which will be our first parameter for the decision tree. This will help to define the variable we are interested on and also to create the independent variables list.

Decision trees need categorical data from the class of factor for the prediction. Our target variable is the “nametype” which is already factor so we don’t need to convert it.

To create a decision tree we need to specify our train and validation data. To test our result, small sample of observations will be taking from the data set. We will have 80% of training and 20% of validation for our experiment.

We need to set seed of random number generator using the code line: “seed”.

The see of the training has 36601 observations within the 8 variables (figure2).

```
> str(train)
'data.frame': 36601 obs. of 8 variables:
 $ id      : int  1 2 6 10 379 390 392 398 417 423 ...
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : num  21 720 107000 1914 4239 ...
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : num  50.8 56.2 54.2 16.9 32.1 ...
 $ reclong  : num  6.08 10.23 -113 -99.9 71.8 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
> |
```

Figure 2: Training Structure

The size of the validation has 9115 observations within the 8 variables (figure 3).

```
> str(validate)
'data.frame': 9115 obs. of 8 variables:
 $ id      : int  370 426 432 463 466 2276 2301 2302 4903 4905 ...
 $ nametype : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ mass     : num  780 779 3000 908 228000 32000 NA 1500 50000 9330 ...
 $ fall     : Factor w/ 2 levels "Fell","Found": 1 1 1 1 1 1 1 1 1 1 ...
 $ decision : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
 $ reclat   : num  -33.17 29.52 8.92 44.88 45.27 ...
 $ reclong  : num  -64.95 35.05 8.43 8.75 10.15 ...
 $ GeoLocation: Factor w/ 17101 levels "", "#REF!", "(-1.002780, 37.150280)", ...: $
> |
```

Figure 3: Validation Structure

The decision trees packages "party" needs to be installed so we can implement it. Only four variables will be used to make the decision tree so we can have a clear view of the diagram. The four variables will be: fall, decision, reclat and reclong.

The code line will be adopted to generate the tree and also draw the tree.

```
Loading required package: sandwich
> mytree <- ctree(nametype~fall+decision+reclat+reclong, mydata,
+ controls=ctree_control(mincriterion=0.9, minsplit=50))
> |
```

Figure 4: Code line.

The tree configuration is shown in figure 5 below.

```
R Console
Loading required package: sandwich
> mytree <- ctree(nametype~fall+decision+reclat+reclong, mydata,
+ controls=ctree_control(mincriterion=0.9, minsplit=50))
> print(mytree)

      Conditional inference tree with 5 terminal nodes

Response:  nametype
Inputs:    fall, decision, reclat, reclong
Number of observations: 45716

1) reclat <= 50.53333; criterion = 1, statistic = 268.952
  2) reclong <= 136.7667; criterion = 0.92, statistic = 5.365
    3)* weights = 33235
  2) reclong > 136.7667
    4) reclat <= -33.9; criterion = 0.999, statistic = 14.303
      5)* weights = 11973
    4) reclat > -33.9
      6)* weights = 122
1) reclat > 50.53333
  7) decision == {No}; criterion = 1, statistic = 60.552
    8)* weights = 219
  7) decision == {Yes}
    9)* weights = 167
> |
```

Figure 5: Configuration and structure

The decision tree below in figure 6 showed that the most important variable in the class variable detection is reclat as it is the root.

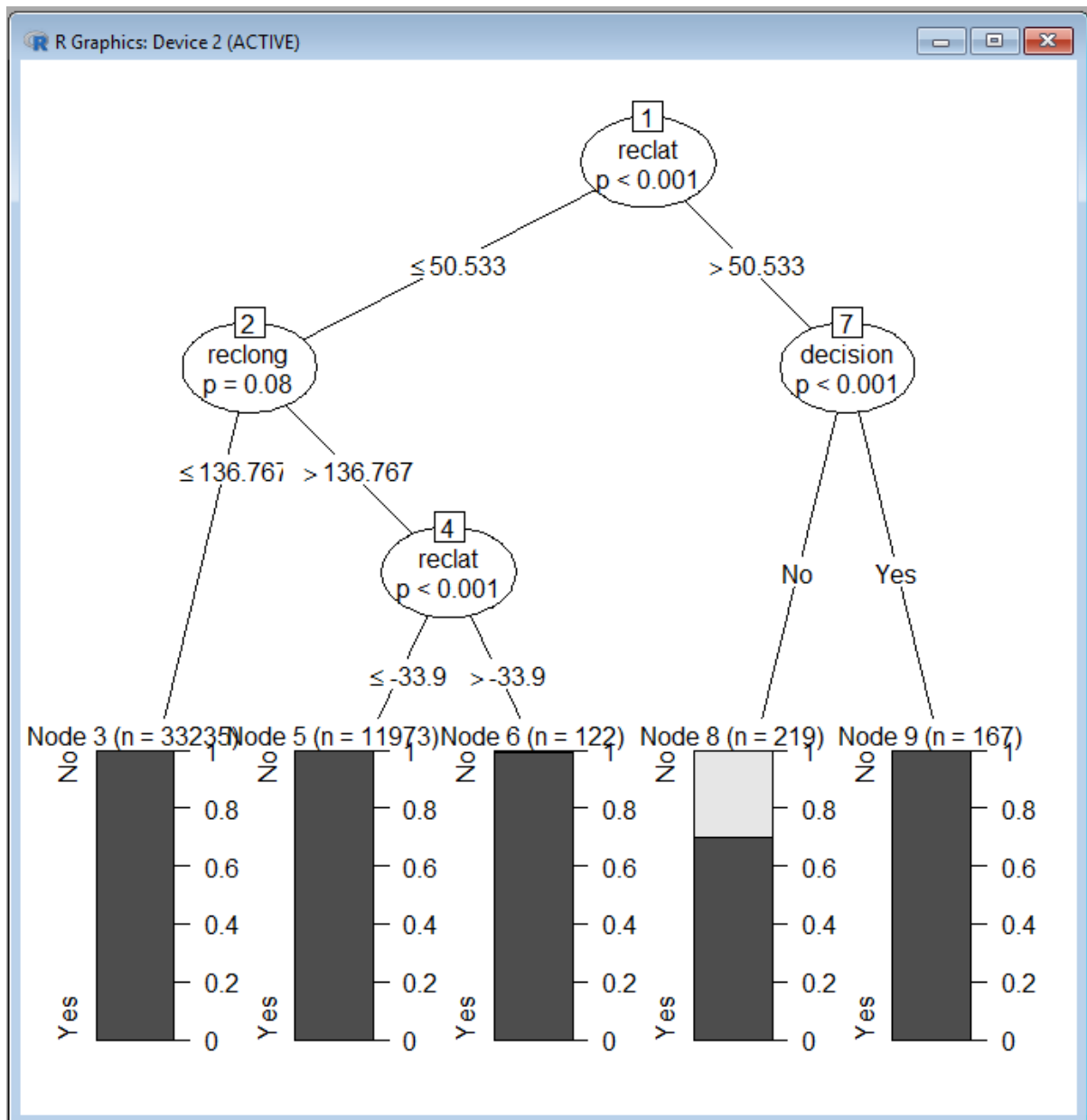


Figure 6: Decision Tree 1

Another form of decision tree using code line type = "simple" code line. Figure 7. And we can also check the classification error with a code shown in figure 8.

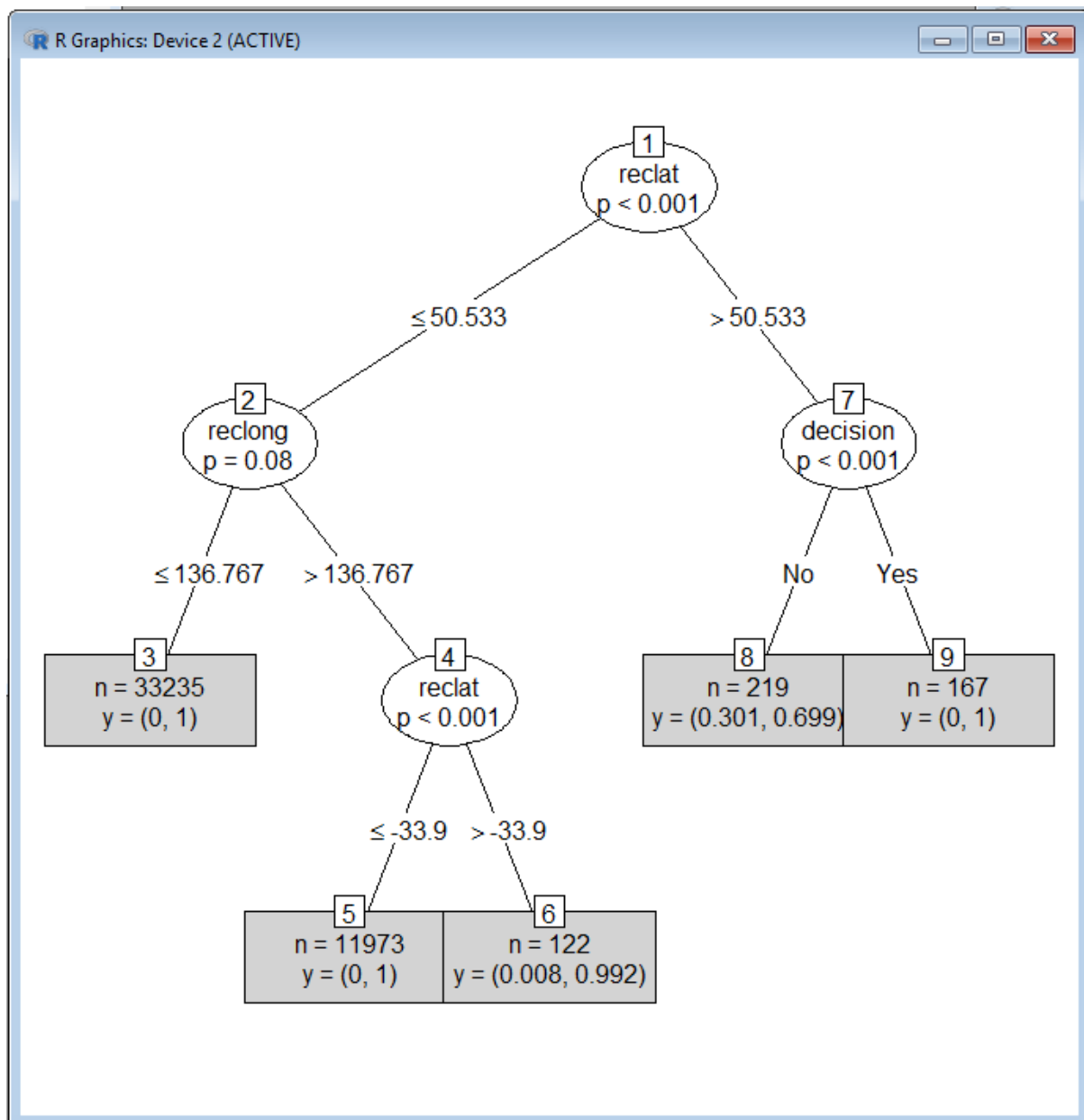


Figure 7: Decision tree 2

```

      No   Yes
No      0    0
Yes    75 45641
> |
  
```

Figure 8 shows the R console output of a confusion matrix. The matrix is displayed as a table with 'No' and 'Yes' as both row and column headers. The values are 0 for (No, No), 0 for (Yes, No), 75 for (No, Yes), and 45641 for (Yes, Yes). The prompt '>' is followed by a vertical bar '|'. The console window has a scrollbar on the right.

Figure 8:

We can calculate the accuracy of the classification. The figure 9 is telling us that our classification is 99% accurate. The equivalent classification error is 0.01% (figure: 10)

```
> sum(diag(tab))/sum(tab)
[1] 0.9983594
> |
```

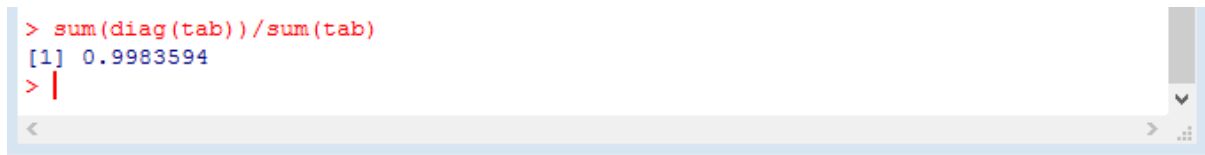
A screenshot of a SAS command window. The prompt is '>'. The command entered is 'sum(diag(tab))/sum(tab)'. The output is '[1] 0.9983594'. The cursor is on the next line, showing '> |'.

Figure 9: Accuracy

```
> 1-sum(diag(tab))/sum(tab)
[1] 0.001640563
> |
```

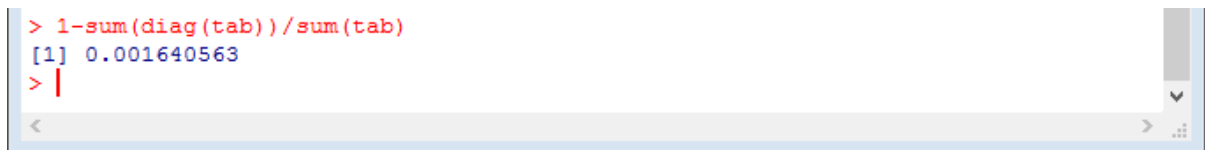
A screenshot of a SAS command window. The prompt is '>'. The command entered is '1-sum(diag(tab))/sum(tab)'. The output is '[1] 0.001640563'. The cursor is on the next line, showing '> |'.

Figure 10: Equivalently classification error.

Also trying to resolve the same question with a decision tree we can see that our accuracy is 99% and the error is under 1% which mean it's high likely that all the observed meteorites in their falling are intact "valid" as already demonstrated with association rules.

## IV- SAS Enterprise Miner: Clustering

In this part of the assignment we are going to use SAS enterprise Miner to try to answer the question in the introduction. In the train section of the SAS Miner we have set the specification method to User Specify. This will allow us to choose the maximum number of clusters with want to use for our analysis in another word we are doing the K mean method of clustering as SAS Enterprise Miner gives you 2 options: K mean and the Hierarchical method of clustering analysis. We are going to use the K mean clustering as that the method which give more information about the analysis as the user has control of the clusters to be generated. So we will have 3 clusters.

We have chosen to use a maximum of three clusters in the vision line to our question so we won't be carry away from our initial idea to answer the question asked in the introduction.

We can see from figure 6 below the first cluster has a frequency of 93%.

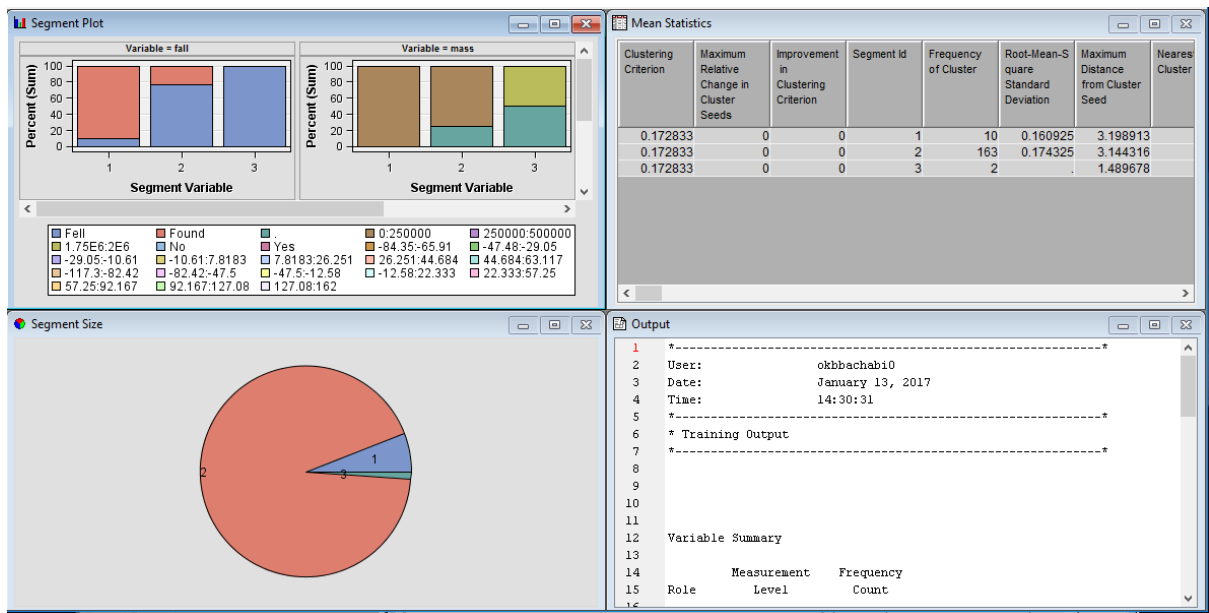


Figure 1

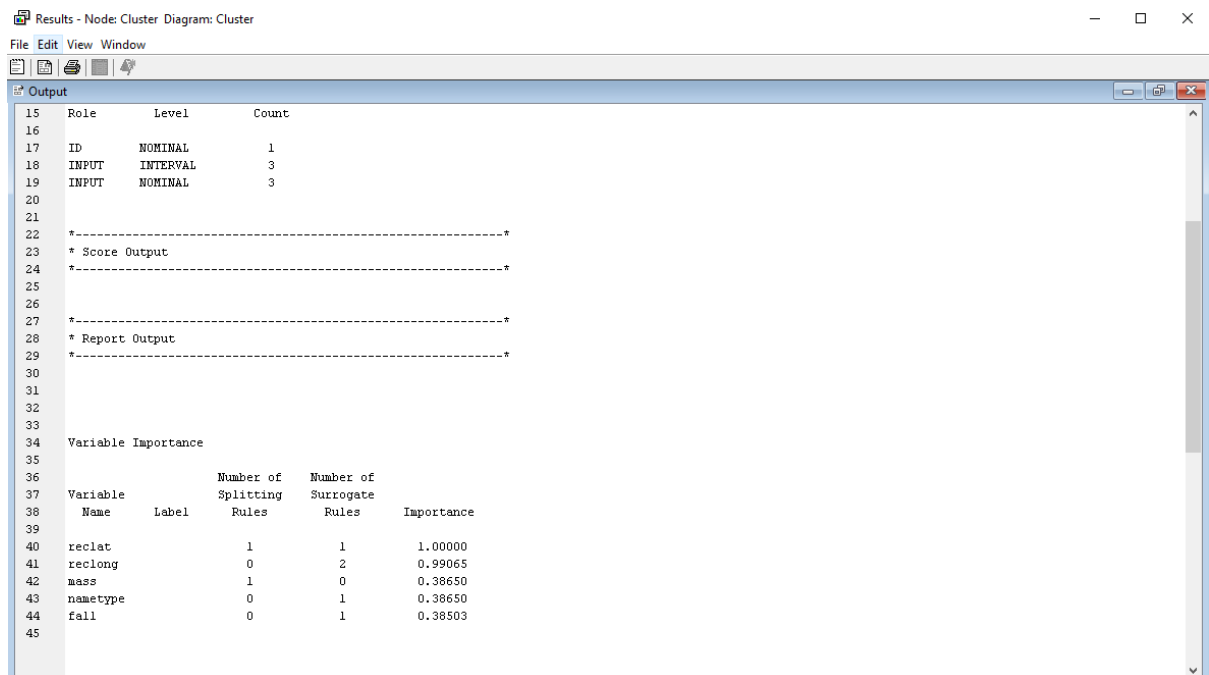


Figure 2

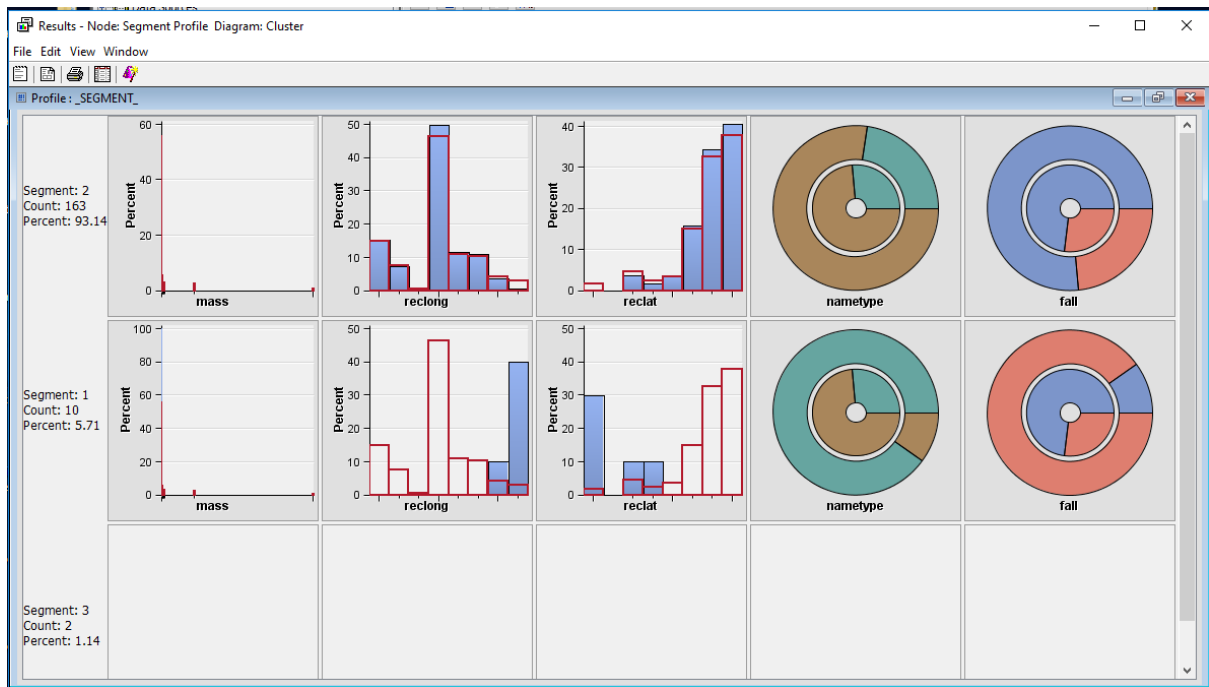


Figure 3

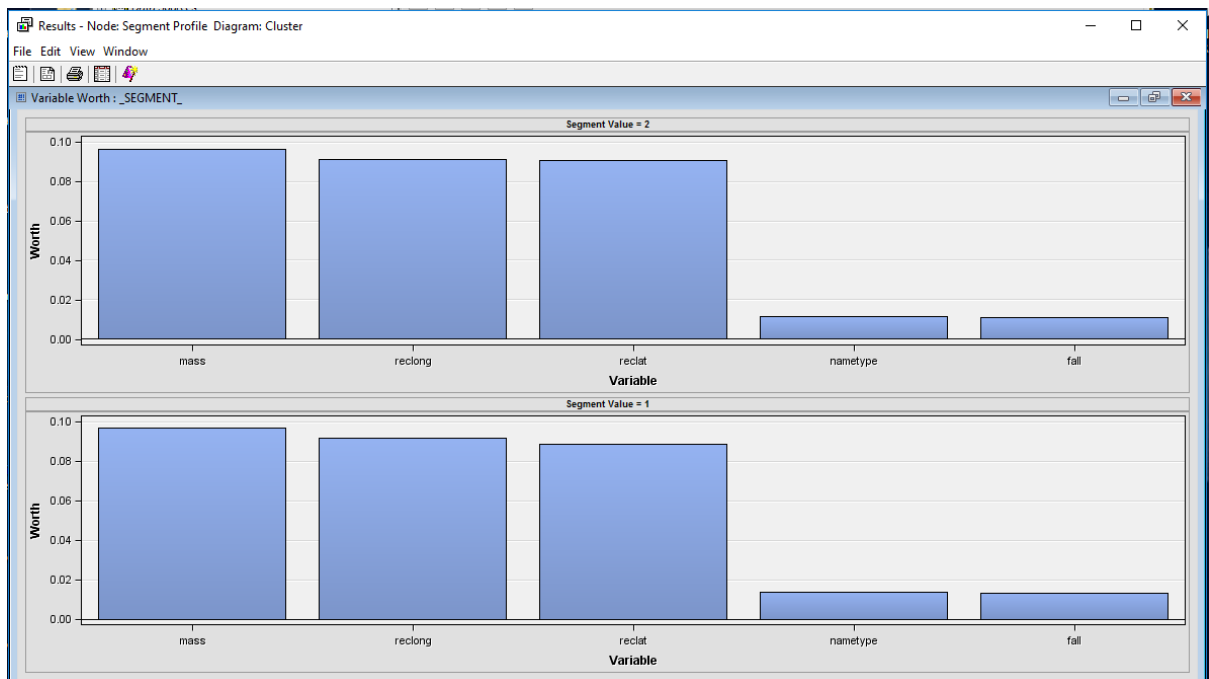


Figure 4



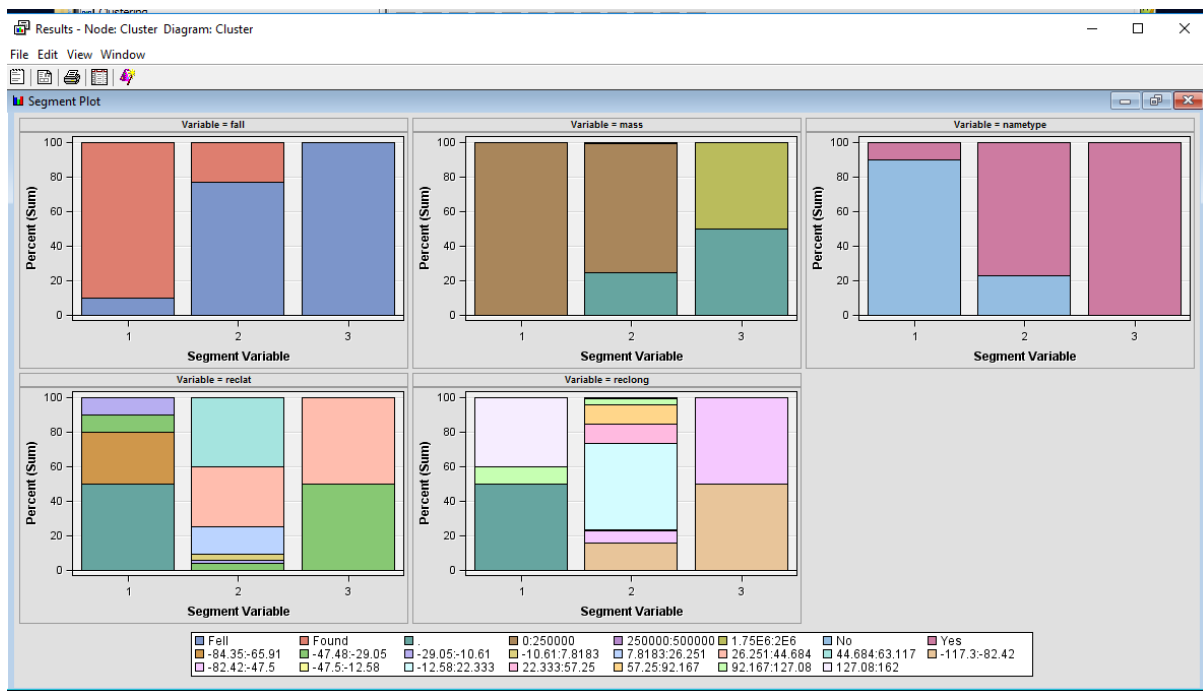


Figure 5

Figure 6 displays the results of a segment profile analysis, showing the frequency and percentage of observations for each segment (1, 2, 3) across various variables.

**Variable Summary**

Role	Measurement Level	Frequency Count
DECISION	NOMINAL	1
ID	NOMINAL	1
INPUT	INTERVAL	3
INPUT	NOMINAL	3
REJECTED	INTERVAL	1
REJECTED	NOMINAL	1
SEGMENT	NOMINAL	1

**Frequencies: \_SEGMENT\_**

Segment Variable	Segment Value	Frequency Count	Percent of Total Frequency
_SEGMENT_	2	163	93.1429
_SEGMENT_	1	10	5.7143
_SEGMENT_	3	2	1.1429

Variable: \_SEGMENT\_ Segment: 2 Count: 163  
Decision Tree Importance Profiles

Figure 6

## V- Conclusion

We have through our works in this assignment to answer this question: Does a meteorite assigns “fell” to which is the one observed in its falling it’s always intact (“valid”)? We have used three different data analysis techniques: association rules and classifications, decision tree and SAs Enterprise Miner. We have seen that the percentage of the results in every single analysis method we have used is very high to conclude that the prediction is accurate and the marge of the error is very narrow.

Three different techniques and very similar results can only be a confirmation of our these.

## VI- Reference

Bater Makhabel (2015). Learning Data Mining with R. Birmingham: Packt Publishing Ltd. P8-11, p24-35, p44-50, p74-81, p148-151.

Edwin de Jonge & Mark van der Loo (2013). An Introduction to Data cleaning with R. Netherlands: Statistics Netherlands. P7-10, p31-45.

Jiawey Han & Micheline Kamber & Jian Pei (2012). Data Mining Concepts and Techniques. 3<sup>rd</sup> ed. USA: Morgan Kaufmann. P1-2, p23-26, p40-44, p88-91.

Wiley (2015). Data Science & Big Data Analytics. InIndianapolis: John Wiley & Sons. P2-15, p54-59, p67-99, p118-130, p138-206

SAS. (2016). Data Visualization. Available: [www.sas.com](http://www.sas.com). Last accessed 13<sup>th</sup> Jan 2017

NASA. (2016). Meteorite Landings. Available: [www.kaggle.com/nasa/meteorite-landings](http://www.kaggle.com/nasa/meteorite-landings). Last accessed 13<sup>th</sup> Jan 2017

Meteoritical Society. (2016). Meteoritics & Planetary Science. Available: [www.meteoriticalsociety.org](http://www.meteoriticalsociety.org). Last accessed 13<sup>th</sup> Jan 2017