R Notebook

Part 1

1

abc)

```
#install.packages('arules')
#isntall.packages('arulesViz')
library(readr)

## Warning: package 'readr' was built under R version 4.1.2

library(arules)

## Warning: package 'arules' was built under R version 4.1.2

## Loading required package: Matrix

##
## Attaching package: 'arules'

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

library(arulesViz)

## Warning: package 'arulesViz' was built under R version 4.1.2

df <- read_csv("air_quality.csv")

## Rows: 153 Columns: 6

## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## dbl (6): Ozone, Solar.R, Wind, Temp, Month, Day

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

str(df)
```

```
## spec_tbl_df [153 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Ozone  : num [1:153] 41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: num [1:153] 190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num [1:153] 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : num [1:153] 67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : num [1:153] 5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : num [1:153] 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Ozone = col_double(),
##   ..   Solar.R = col_double(),
##   ..   Wind = col_double(),
##   ..   Temp = col_double(),
##   ..   Month = col_double(),
##   ..   Day = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
summary(df)
```

```
##      Ozone           Solar.R           Wind             Temp      
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.75  
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.983   Mean   :77.95  
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.750   3rd Qu.:85.00  
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
##  NA's   :37       NA's   :7       NA's   :2        NA's   :1      
##      Month            Day      
##  Min.   :5.000   Min.   : 1.00  
##  1st Qu.:6.000   1st Qu.: 8.50  
##  Median :7.000   Median :16.00  
##  Mean   :6.993   Mean   :15.99  
##  3rd Qu.:8.000   3rd Qu.:23.50  
##  Max.   :9.000   Max.   :31.00  
##                  NA's   :2      
```

```
head(df)
```

```
## # A tibble: 6 x 6
##   Ozone Solar.R  Wind  Temp Month   Day
##   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    41     190   7.4    67     5     1
## 2    36     118   8      72     5     2
## 3    12     149  12.6    74     5     3
## 4    18     313  11.5    62     5     4
## 5    NA      NA  14.3    56     5     5
## 6    28      NA  14.9    66     5     6
```

```
tail(df)
```

```
## # A tibble: 6 x 6
##   Ozone Solar.R  Wind  Temp Month   Day
##   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    14      20  16.6    63     9    25
## 2    30     193   6.9    70     9    26
## 3    NA     145  13.2    77     9    27
## 4    14     191  14.3    75     9    28
## 5    18     131   8      76     9    29
## 6    20     223  11.5    NA     9    30
```

```
colSums(is.na(df))
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       2       1       0       2
```

```
df= na.omit(df)
```

```
df = data.frame(scale(df))
summary(df)
```

```
##      Ozone            Solar.R             Wind               Temp
##  Min.   :-1.2478   Min.   :-1.9777   Min.   :-2.1301   Min.   :-2.1555
##  1st Qu.:-0.7414   1st Qu.:-0.7729   1st Qu.:-0.7074   1st Qu.:-0.7036
##  Median :-0.3243   Median : 0.2778   Median :-0.0657   Median : 0.1260
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6140   3rd Qu.: 0.7785   3rd Qu.: 0.4365   3rd Qu.: 0.7482
##  Max.   : 3.7268   Max.   : 1.6202   Max.   : 3.0031   Max.   : 1.9927
##      Month              Day
##  Min.   :-1.4689   Min.   :-1.76465
##  1st Qu.:-0.7914   1st Qu.:-0.82643
##  Median :-0.1140   Median :-0.00548
##  Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.9022   3rd Qu.: 0.75683
##  Max.   : 1.2409   Max.   : 1.75369
```

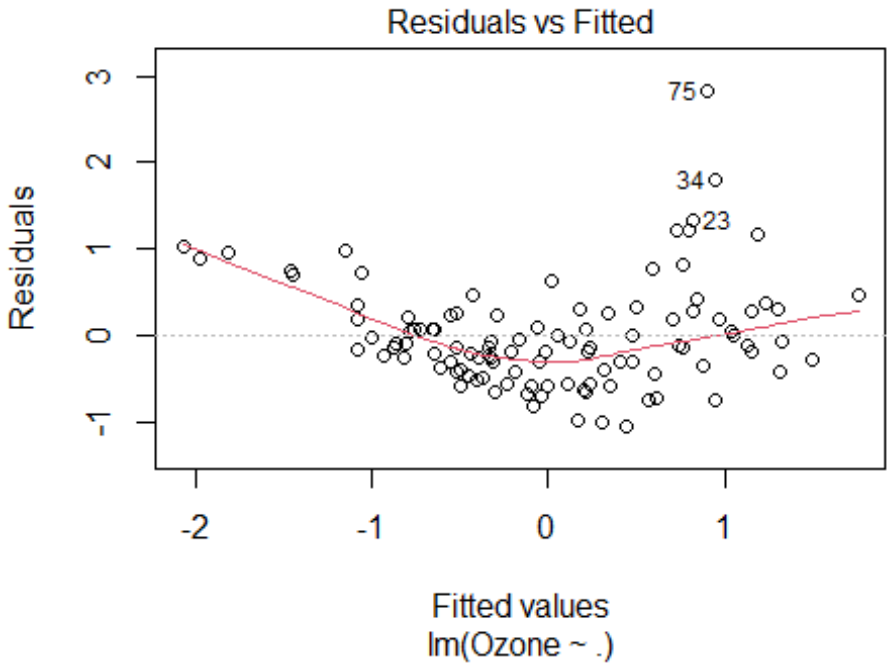
```
head(df)
```

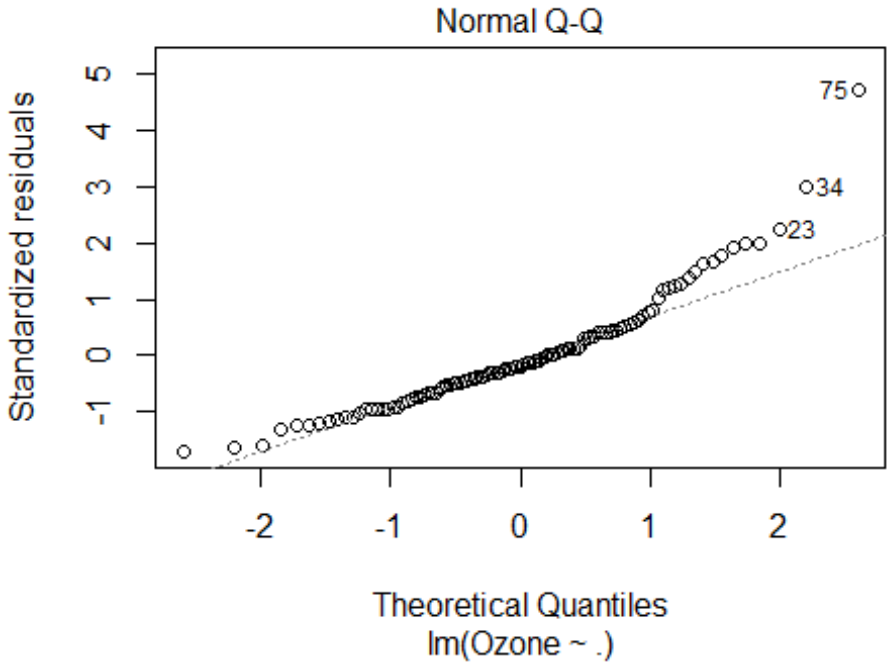
```
##         Ozone     Solar.R       Wind       Temp    Month        Day
## 1 -0.05623518  0.03578426 -0.7073526 -1.1184675 -1.46885 -1.7646514
## 2 -0.20517488 -0.75640516 -0.5399658 -0.5999406 -1.46885 -1.6473734
## 3 -0.92008547 -0.41532360  0.7433330 -0.3925298 -1.46885 -1.5300953
## 4 -0.74135783  1.38910784  0.4364572 -1.6369945 -1.46885 -1.4128172
## 5 -0.59241812  1.23507101 -0.3725790 -1.3258783 -1.46885 -1.0609830
## 6 -0.71156989 -0.96545514  1.0781066 -1.9481107 -1.46885 -0.9437049
```

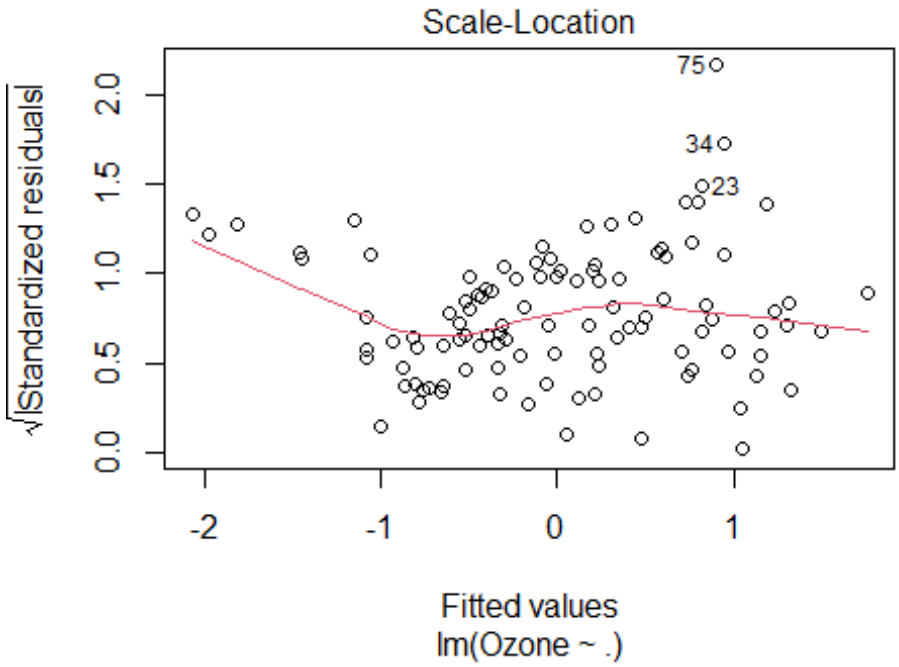
```
mdl = lm(Ozone~.,df)
summary(mdl)
```

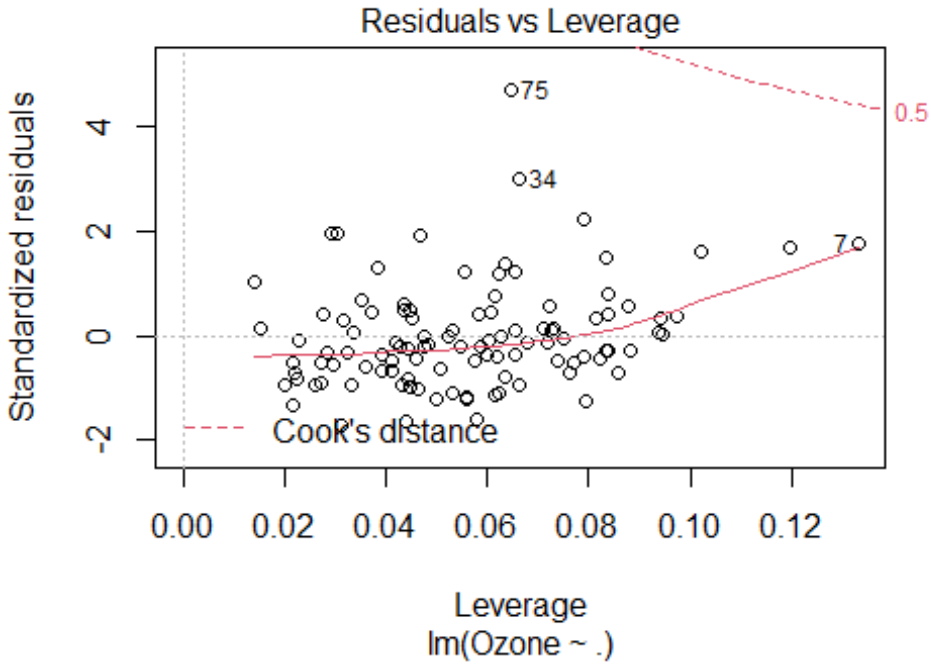
```
##
## Call:
## lm(formula = Ozone ~ ., data = df)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.0406 -0.3988 -0.1142  0.2547  2.8255 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.666e-16  6.013e-02   0.000   1.0000    
## Solar.R      1.284e-01  6.580e-02   1.951   0.0538 .  
## Wind        -3.548e-01  6.987e-02  -5.078 1.75e-06 ***
## Temp         5.474e-01  8.058e-02   6.793 7.75e-10 ***
## Month       -1.200e-01  6.834e-02  -1.756   0.0822 .  
## Day          6.397e-02  6.089e-02   1.051   0.2959    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.622 on 101 degrees of freedom
## Multiple R-squared:  0.6313, Adjusted R-squared:  0.6131 
## F-statistic: 34.59 on 5 and 101 DF,  p-value: < 2.2e-16

plot(mdl)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Ozone ~ .)
75
34
23
-1
0
1
2
3
-2
-1
0
1


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Ozone ~ .)
75
34
23
-1
0
1
2
3
4
5
-2
-1
0
1
2





Plots represent the residual vs fitted graph, QQ plot, scale location, and residuals vs leverage. The residual vs fitted plot shows that model is not a good fit because the red line deviates from the baseline at 0. QQ plot represents if the data is coming from the same

distribution or not and we see the slight deviation from the baseline at the lower and upper end. The leverage plot represents some outliers in the data.

2

abc)

```
df <- read_csv("Lung_Capacity.csv")

## Rows: 725 Columns: 6

## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr (3): Smoke, Gender, Caesarean
## dbl (3): LungCap, Age, Height

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

str(df)

## spec_tbl_df [725 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ LungCap  : num [1:725] 6.47 10.12 9.55 11.12 4.8 ...
##  $ Age      : num [1:725] 6 18 16 14 5 11 8 11 15 11 ...
##  $ Height   : num [1:725] 62.1 74.7 69.7 71 56.9 58.7 63.3 70.4 70.5 59.2
...
##  $ Smoke    : chr [1:725] "no" "yes" "no" "no" ...
##  $ Gender   : chr [1:725] "male" "female" "female" "male" ...
##  $ Caesarean: chr [1:725] "no" "no" "yes" "no" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   LungCap = col_double(),
##   ..   Age = col_double(),
##   ..   Height = col_double(),
##   ..   Smoke = col_character(),
##   ..   Gender = col_character(),
##   ..   Caesarean = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>

colSums(is.na(df))

##   LungCap       Age    Height     Smoke    Gender Caesarean
##         1         2         0         0         0         0
```

Smoke, gender, and cesarean are the class variables in the dataset. smoke indicates the about if the participant smoke or not, gender tells us about the gender of the participant, and caesarean tells us about the lung operation.

```
table(df$Gender)

##
## female   male
##    358    367
```

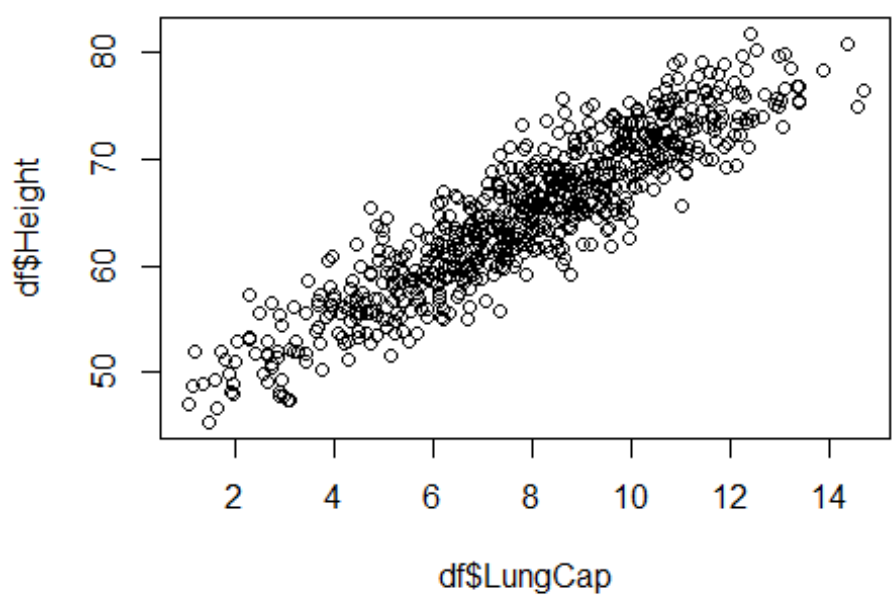
there are 358 females and 367 females.

g) Smoke, Caesarean, and gender should be factor format because they indicate the 2 different factors about the participants.

```
df$Smoke = as.factor(df$Smoke)
df$Gender = as.factor(df$Gender)
df$Caesarean = as.factor(df$Caesarean)
```

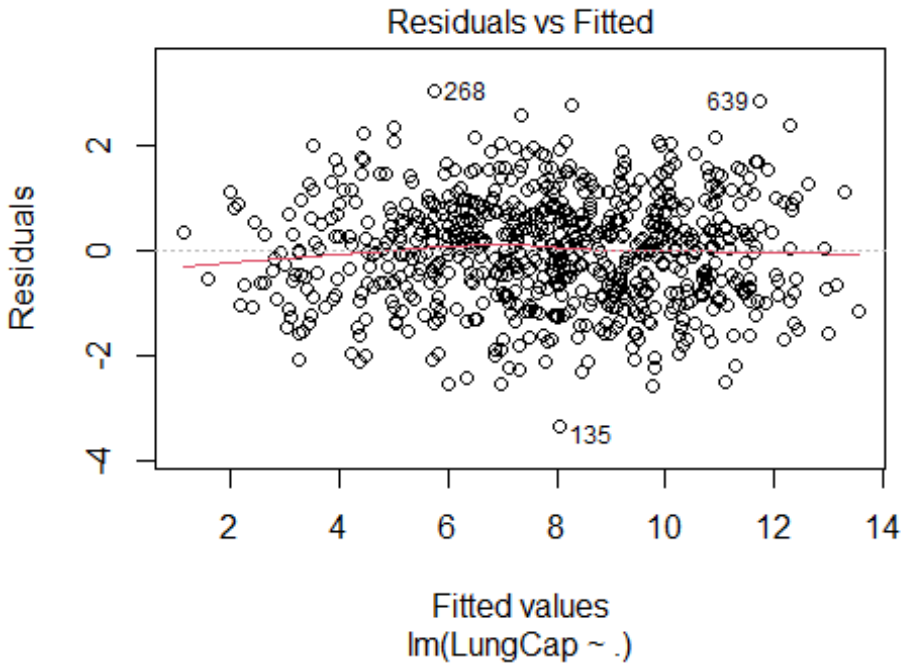
h-i)

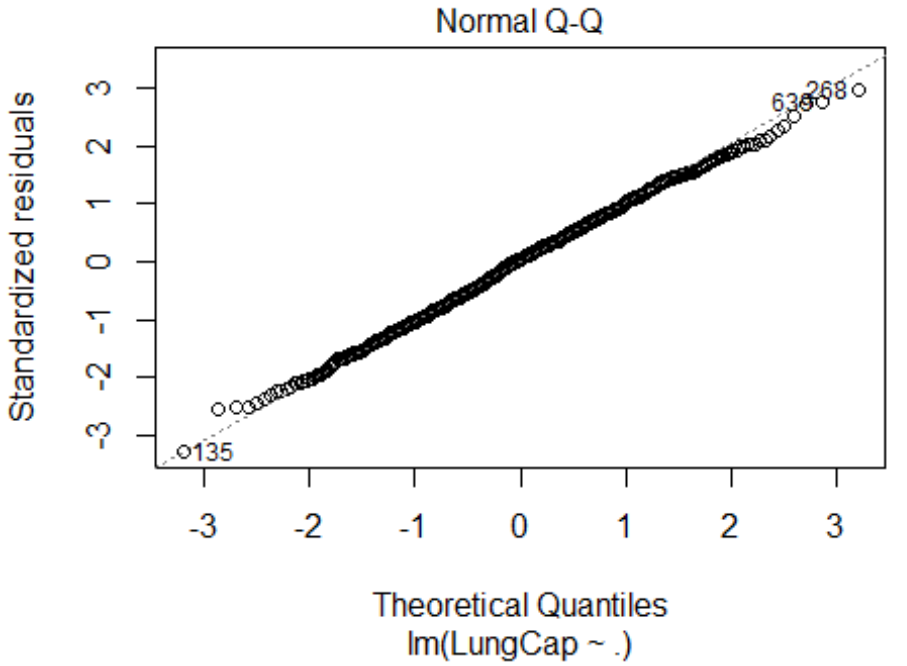
```
plot(df$LungCap,df$Height)
```

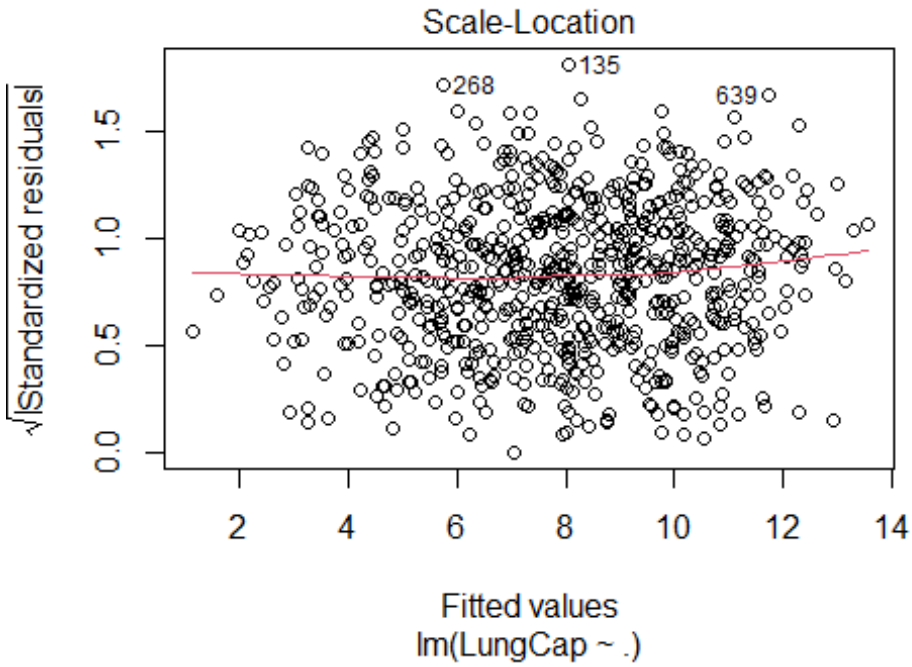


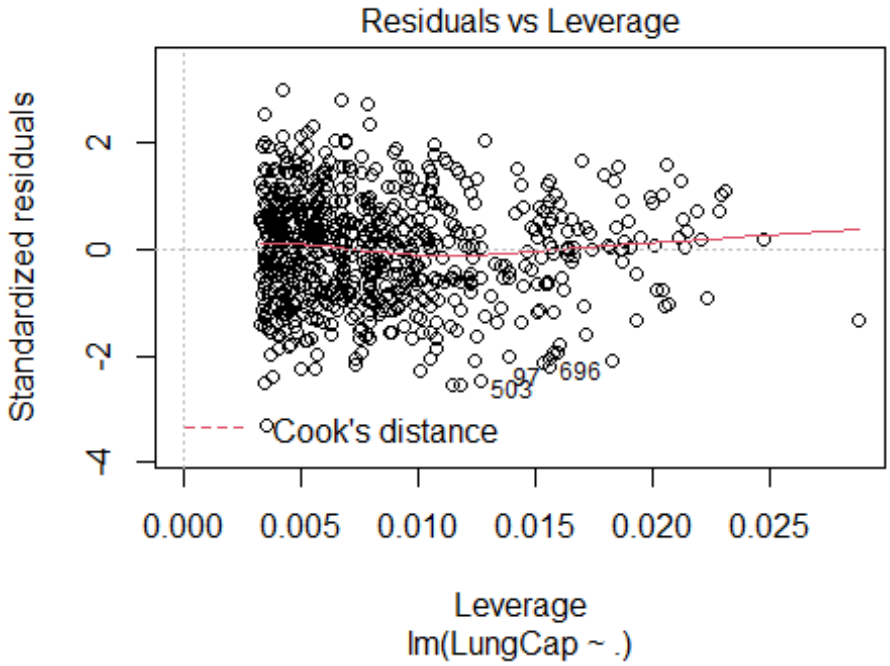

```
df=na.omit(df)
cor(df$LungCap,df$Height)

## [1] 0.911796
```

```r
mdll = lm(LungCap~.,df)
summary(mdll)

##
## Call:
## lm(formula = LungCap ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3420 -0.7186  0.0459  0.6942  3.0140
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -11.31201    0.46998 -24.069  < 2e-16 ***
## Age            0.16077    0.01807   8.898  < 2e-16 ***
## Height         0.26395    0.01005  26.277  < 2e-16 ***
## Smokeyes      -0.61101    0.12564  -4.863 1.42e-06 ***
## Gendermale     0.37542    0.07959   4.717 2.88e-06 ***
## Caesareanyes  -0.20224    0.09100  -2.222   0.0266 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.017 on 716 degrees of freedom
## Multiple R-squared:  0.8534, Adjusted R-squared:  0.8524
## F-statistic: 833.7 on 5 and 716 DF,  p-value: < 2.2e-16

plot(mdll)
```


Residuals vs Fitted
Residuals
Fitted values
lm(LungCap ~ .)
268
639
135
2
0
-2
-4
2
4
6
8
10
12
14


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(LungCap ~ .)
268
639
135
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3





Plots represent the residual vs fitted graph, QQ plot, scale location, and residuals vs leverage. Residual vs fitted plot shows that model is a good fit because the red line doesn't

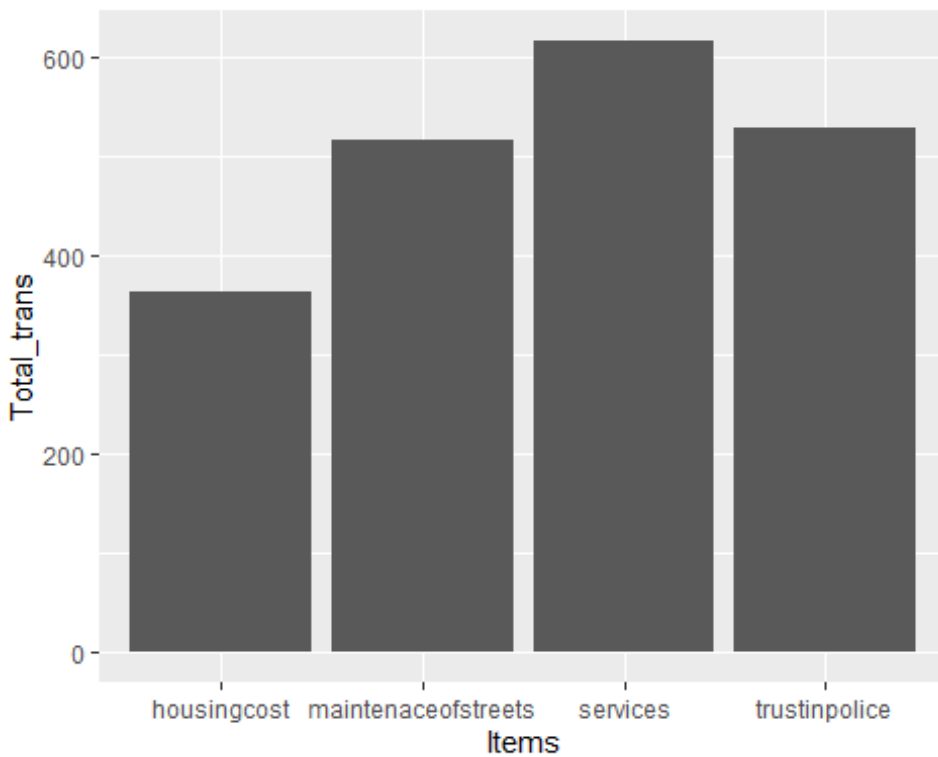
show deviate from the baseline at 0. QQ plot represents if the data is coming from the same distribution or not and we see it follows the same distribution. The leverage plot represents does not show any outliers in the data.

3

i-m-n-o)

```
library(readxl)
#setwd('C:/Users/klx/Desktop/Rweka/')
df <- read_excel("happy(1).xlsx")
summary(df)

##   Happiness            service      housingOfCost   SchoolQuality
##  Length:286         Min.   :1.000   Min.   :1.000   Min.   :1.000
##  Class :character   1st Qu.:4.000   1st Qu.:2.000   1st Qu.:3.000
##  Mode  :character   Median :5.000   Median :3.000   Median :3.000
##                     Mean   :4.315   Mean   :2.538   Mean   :3.266
##                     3rd Qu.:5.000   3rd Qu.:3.000   3rd Qu.:4.000
##                     Max.   :5.000   Max.   :5.000   Max.   :5.000
##                     NA's   :143     NA's   :143     NA's   :143
##  TrustInPloice   maintenaceOfStreets SocialCommunity
##  Min.   :1.000   Min.   :1.000       Min.   :1.000
##  1st Qu.:3.000   1st Qu.:3.000       1st Qu.:4.000
##  Median :4.000   Median :4.000       Median :4.000
##  Mean   :3.699   Mean   :3.615       Mean   :4.217
##  3rd Qu.:4.000   3rd Qu.:4.000       3rd Qu.:5.000
##  Max.   :5.000   Max.   :5.000       Max.   :5.000
##  NA's   :143     NA's   :143         NA's   :143

trans = transactions(df)

## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7 not logical or factor. Applying def
ault
## discretization (see '? discretizeDF').

## Warning in discretize(x = c(NA, 3, NA, 3, NA, 5, NA, 5, NA, 5, NA, 5, NA,
: The calculated breaks are: 1, 4, 5, 5
##   Only unique breaks are used reducing the number of intervals. Look at ?
discretize for details.

## Warning in discretize(x = c(NA, 4, NA, 3, NA, 5, NA, 5, NA, 5, NA, 5, NA,
: The calculated breaks are: 1, 4, 5, 5
##   Only unique breaks are used reducing the number of intervals. Look at ?
discretize for details.

trans
```

```
## transactions in sparse format with
##  286 transactions (rows) and
##  18 items (columns)

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.1.1

df=na.omit(df)
Total_trans  = c(sum(df$service),sum(df$housingOfCost),sum(df$TrustInPloice),
sum(df$maintenaceOfStreets))
Items = c('services','housingcost','trustinpolice','maintenaceofstreets')
c=data.frame(Total_trans,Items)
c

##   Total_trans               Items
## 1         617            services
## 2         363         housingcost
## 3         529       trustinpolice
## 4         517 maintenaceofstreets

ggplot(c,aes(x=Items,y=Total_trans))+geom_col()
```



q)

```
rul1 <- apriori(trans, parameter = list(supp = 0.05, conf = 0.8,minlen=2))

## Apriori
##
## Parameter specification:
```

```
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.8    0.1    1 none FALSE            TRUE       5    0.05      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 14
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[18 item(s), 286 transaction(s)] done [0.00s].
## sorting and recoding items ... [17 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [228 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

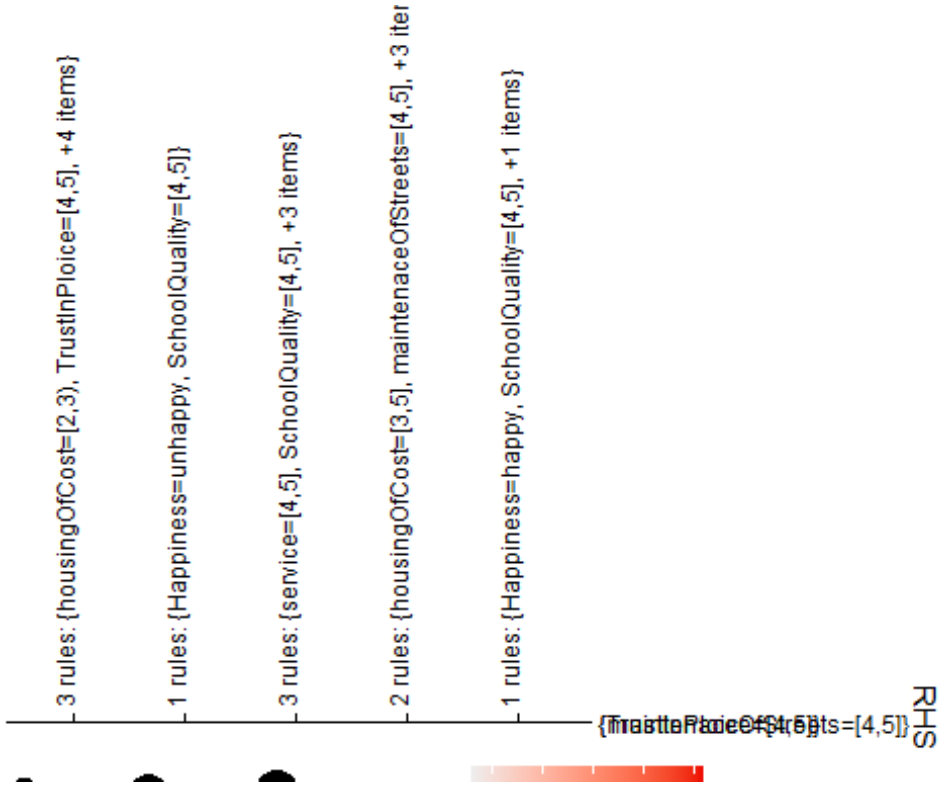
```
inspect(head(sort(rul1, by="lift"), 5))
```

```
##     lhs                        rhs                            support conf
idence   coverage     lift count
## [1] {Happiness=happy,
##      SchoolQuality=[4,5],
##      TrustInPloice=[4,5],
##      SocialCommunity=[4,5]} => {maintenaceOfStreets=[4,5]} 0.06993007  0.9
523810 0.07342657 3.026455    20
## [2] {Happiness=happy,
##      service=[4,5],
##      SchoolQuality=[4,5],
##      TrustInPloice=[4,5],
##      SocialCommunity=[4,5]} => {maintenaceOfStreets=[4,5]} 0.06293706  0.9
473684 0.06643357 3.010526    18
## [3] {housingOfCost=[2,3),
##      TrustInPloice=[4,5],
##      SocialCommunity=[4,5]} => {maintenaceOfStreets=[4,5]} 0.05594406  0.9
411765 0.05944056 2.990850    16
## [4] {Happiness=unhappy,
##      SchoolQuality=[4,5]}   => {TrustInPloice=[4,5]}       0.05244755  0.8
823529 0.05944056 2.934337    15
## [5] {SchoolQuality=[4,5],
##      TrustInPloice=[4,5],
##      SocialCommunity=[4,5]} => {maintenaceOfStreets=[4,5]} 0.10489510  0.9
090909 0.11538462 2.888889    30
```

```
subrules1 <- head(sort(rul1, by="lift"), 10)
```

r-s-t)

```r
df= na.omit(df)
subrules1 <- head(sort(rul1, by="lift"), 10)

plot(subrules1, method="grouped",  control=list(k=5))
```



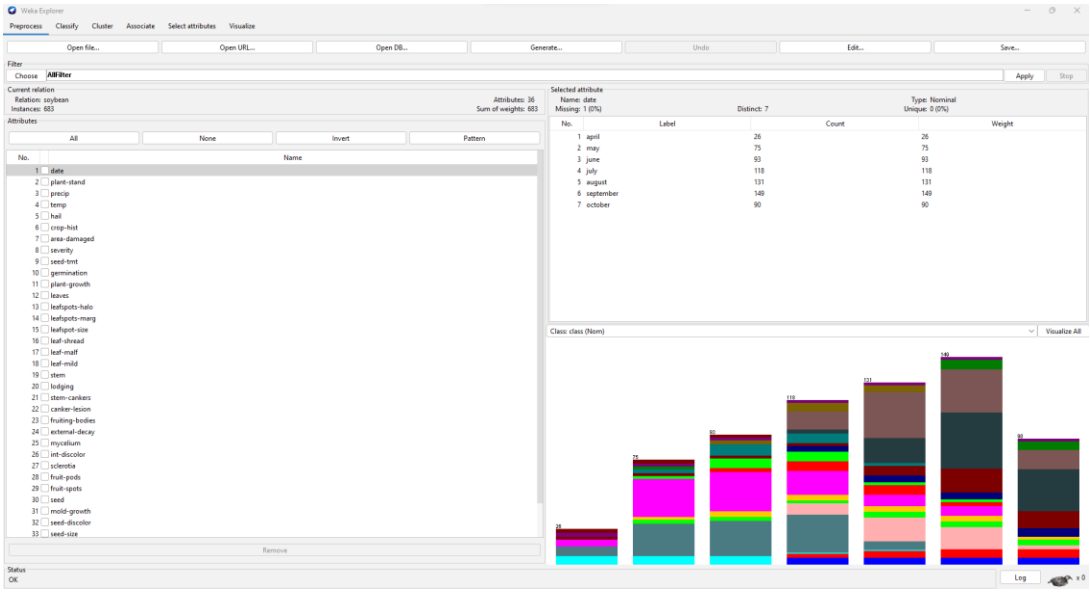

```r
plot(subrules1, method="graph")
```

Part 2

2

b.

I uploaded the data in weka using weka explorer. In weka explorer, the open file button helped to import the dataset in weka.



c.

The dataset contains 36 categorical variables, where some of them are ordinal and the sum of them are nominal. Class variable tells us about the different kinds of soybean seeds.

Variable class is our class variable which contains the following categories,

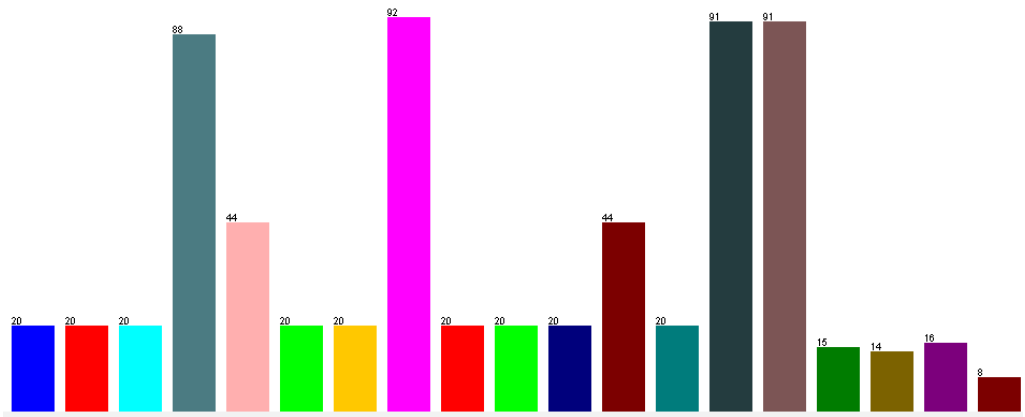
1. Diaporthe-stem-canker
2. Charcoal-rot
3. Rhizoctonia-root-rot
4. Phytophthora-rot
5. Brown-stem-rot
6. Powdery-mildew
7. Downy-mildew
8. Brown-spot
9. Bacterial-blight
10. Bacterial-pustule
11. Purple-seed-stain
12. Anthracnose
13. Phyllosticta-leaf-spot
14. Alternaria leaf-spot
15. Frog-eye-leaf-spot
16. Diaporthe-pod-&-stem-blight
17. Cyst-nematode
18. 2-4-d-injury
19. Herbicide-injury

Selected attribute

Name: class
Missing: 0 (0%)
Distinct: 19
Type: Nominal
Unique: 0 (0%)

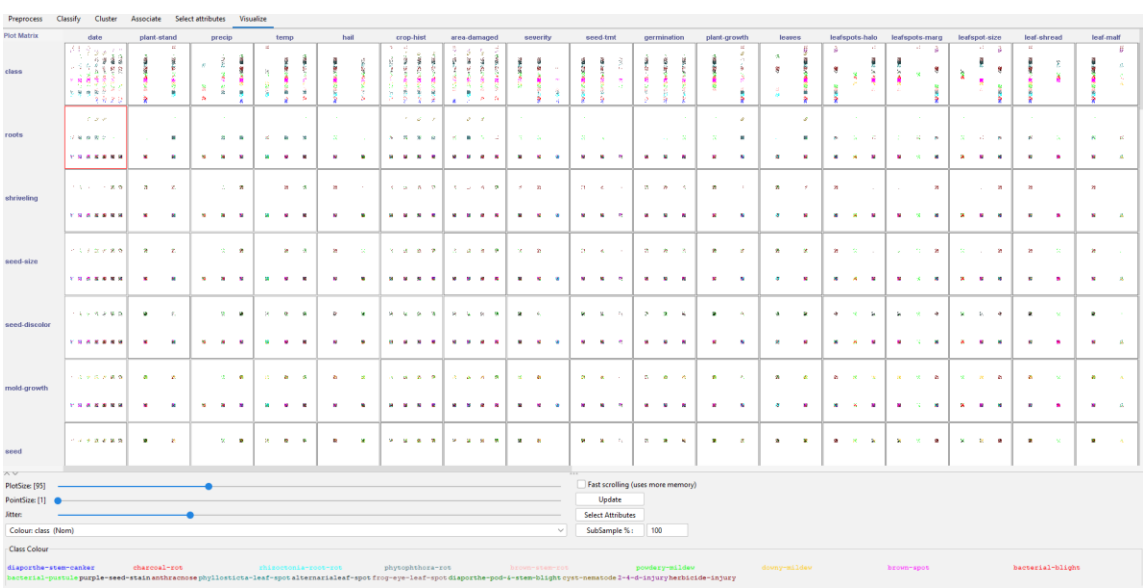
No.	Label	Count	Weight
1	diaporthe-stem-canker	20	20
2	charcoal-rot	20	20
3	rhizoctonia-root-rot	20	20
4	phytophthora-rot	88	88
5	brown-stem-rot	44	44
6	powdery-mildew	20	20
7	downy-mildew	20	20
8	brown-spot	92	92
9	bacterial-blight	20	20
10	bacterial-pustule	20	20
11	purple-seed-stain	20	20
12	anthracnose	44	44
13	phyllosticta-leaf-spot	20	20
14	alternarialeaf-spot	91	91
15	frog-eye-leaf-spot	91	91
16	diaporthe-pod-&-stem-blight	15	15
17	cyst-nematode	14	14

Class: class (Nom) Visualize All

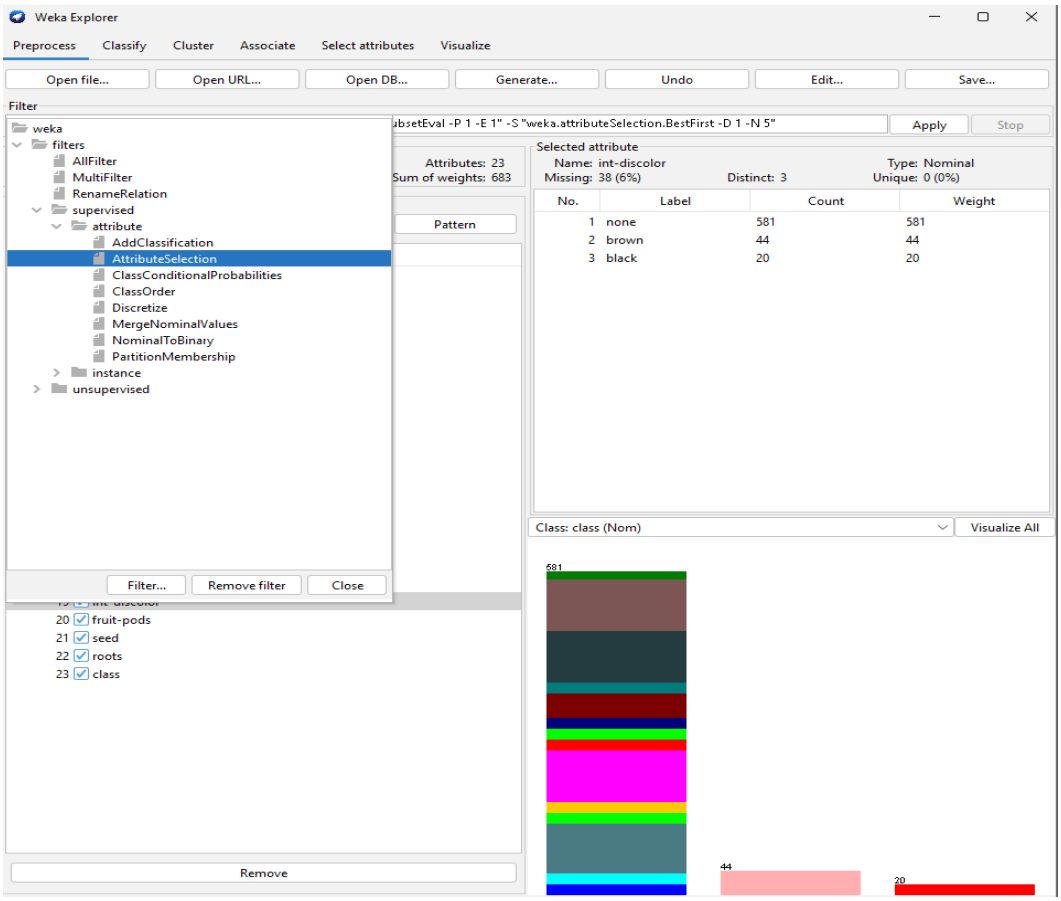


d.

visualization tab color code the points across all the variables.



e.



In pre-processing tab, we can select the attribute selection filter and apply it using the apply button which helps us to select useful variables for supervised machine learning model implementation.

f.

I used InfoGainAttributeEval to rank all the variables in the data.

Attribute selection output

```
               canker-lesion
               fruiting-bodies
               external-decay
               int-discolor
               fruit-pods
               seed
               roots
               class
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 23 class):
        Information Gain Ranking Filter

Ranked attributes:
 1.1517   16 canker-lesion
 1.0129   12 leafspot-size
 0.8684   11 leafspots-halo
 0.8437   20 fruit-pods
 0.6918   15 stem
 0.6715    1 date
 0.6265    9 plant-growth
 0.5853    2 precip
 0.5392   22 roots
 0.5245   19 int-discolor
 0.4829   18 external-decay
 0.4808    5 area-damaged
 0.4241    3 temp
 0.4133   21 seed
 0.3614   14 leaf-mild
 0.3568   10 leaves
 0.3517   17 fruiting-bodies
 0.3106    6 severity
 0.2465   13 leaf-malf
 0.0784    4 hail
 0.0742    7 seed-tmt
 0.0554    8 germination

Selected attributes: 16,12,11,20,15,1,9,2,22,19,18,5,3,21,14,10,17,6,13,4,7,8 : 22
```

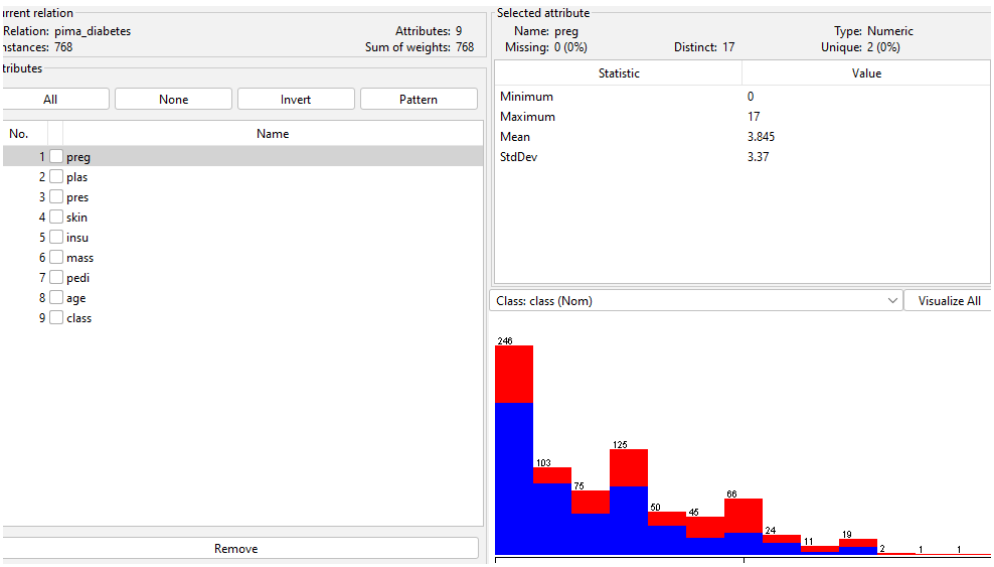
g.

Each attribute got a rank value which help us to find out the importance of the feature. Canker lesion is the most important feature among all others. Leafspot size is the second most important feature. Hail, seed-TMT, and germination is the least important feature among all with rank score of 0.0784, 0.0742, and 0.0554 respectively.

3

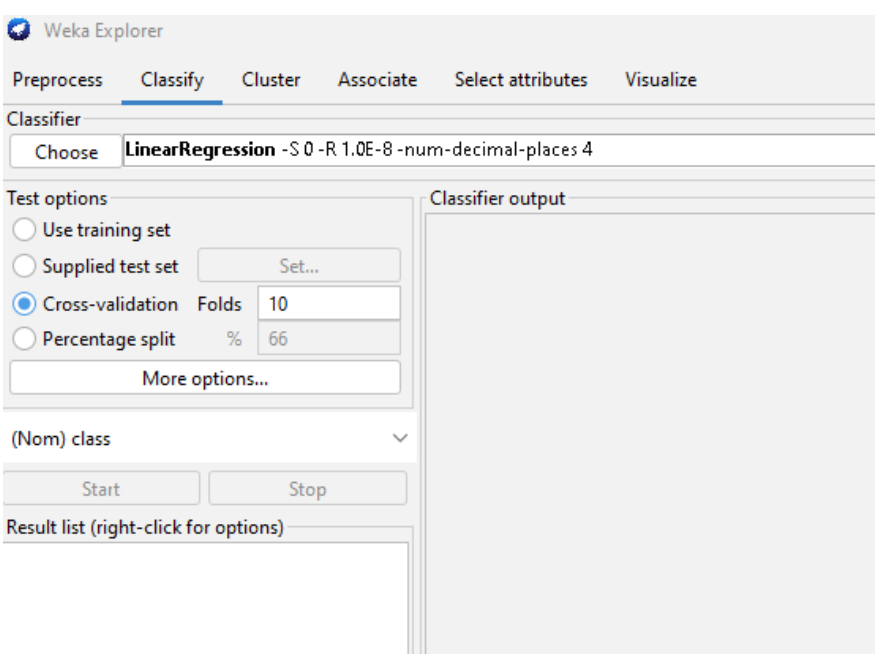
B

I uploaded the data in weka using weka explorer. In the weka explorer, the open file button helped to import the dataset in weka.



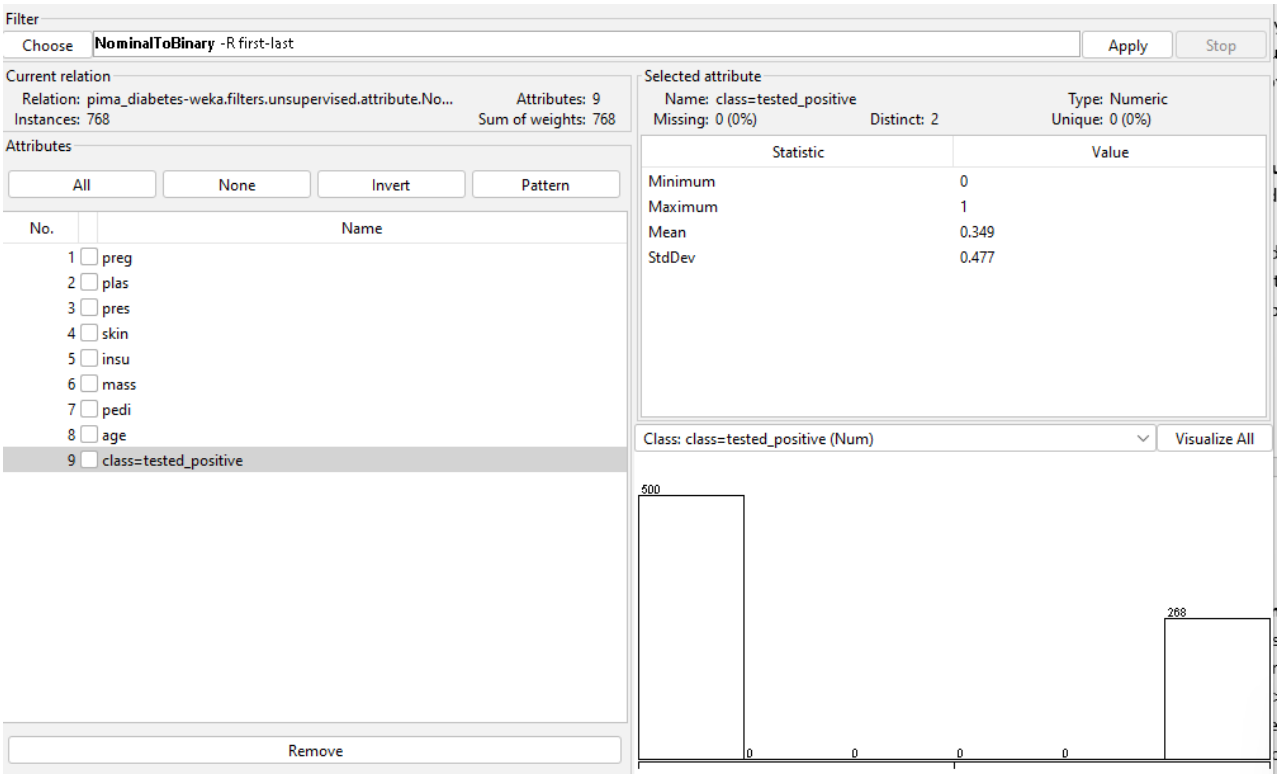
C

Since our class variable is not in numeric format, we can not implement the linear regression model. Option for a linear regression model is disabled until and unless we change the format of the class attribute.



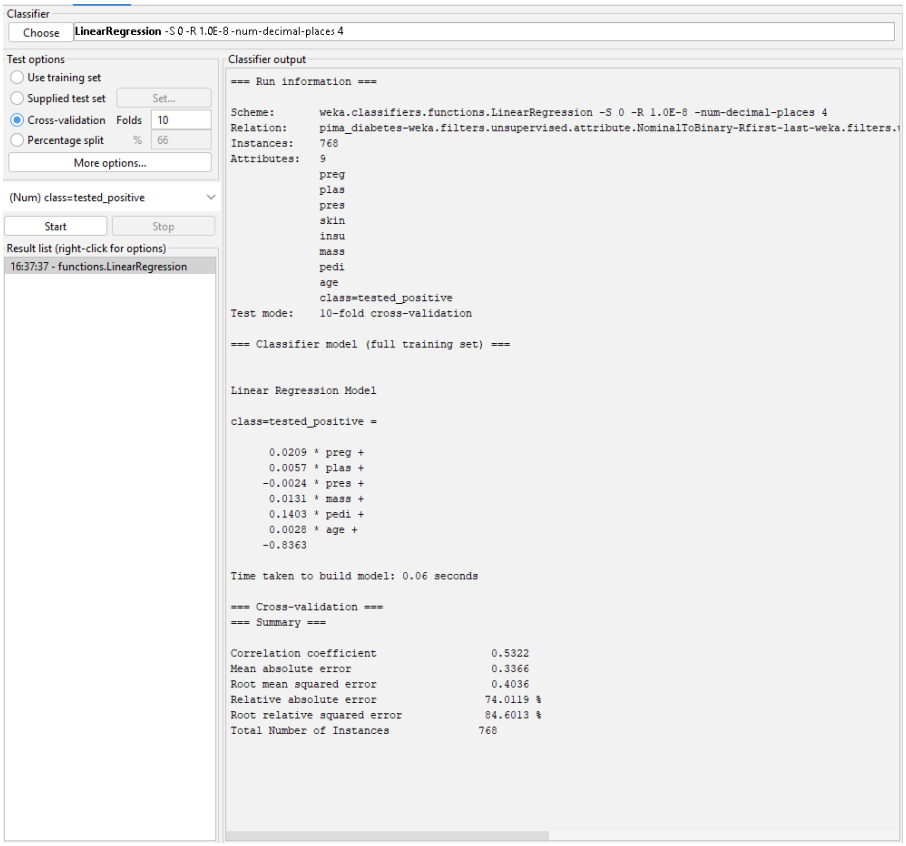
d-e)

I changed the filter by setting class(norm) to no class by scrolling the tab above the graph and implemented the filter NominalToBinary to convert the classes in number of 0 and 1.



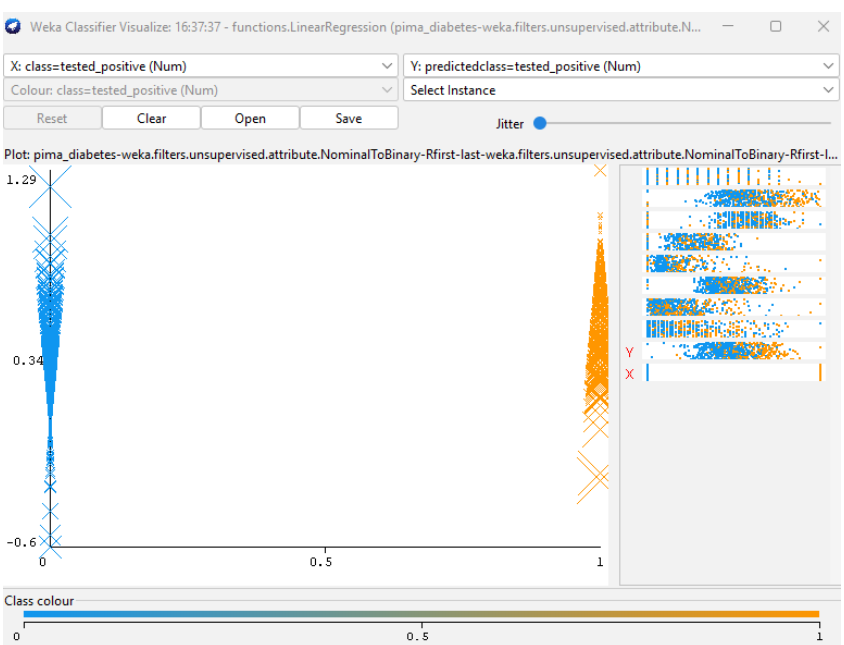
f-g)

Liner regression can be implemented in classifier tab using choose button - > function - > linear regression. Model output is as follow,



h)

We can visualize the classification error right clicking on the implemented model.



i)

By specifying plan text in output prediction, we got the following table. Which represent the actual values and the predicted value of the model.

Classifier output

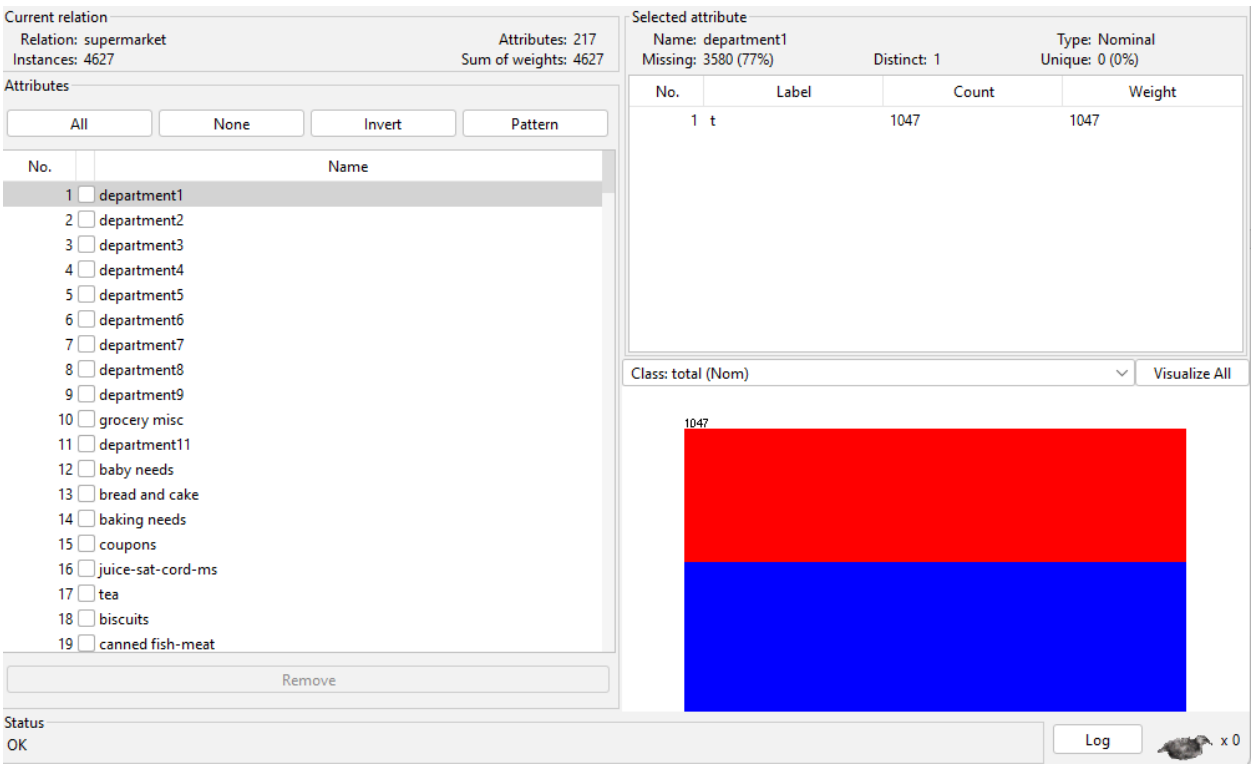
```
=== Predictions on test data ===
```

inst#	actual	predicted	error
1	0	0.325	0.325
2	0	0.308	0.308
3	0	0.11	0.11
4	0	0.069	0.069
5	1	-0.274	-1.274
6	1	0.642	-0.358
7	1	0.289	-0.711
8	1	0.449	-0.551
9	1	0.411	-0.589
10	1	0.371	-0.629
11	0	-0.016	-0.016
12	0	0.22	0.22
13	0	0.927	0.927
14	1	0.369	-0.631
15	1	0.293	-0.707
16	0	0.62	0.62
17	0	0.277	0.277
18	0	0.328	0.328
19	0	0.413	0.413
20	0	0.047	0.047
21	1	0.825	-0.175
22	0	0.447	0.447
23	0	0.413	0.413
24	0	-0.231	-0.231
25	1	0.248	-0.752
26	1	0.601	-0.399
27	1	0.35	-0.65
28	0	0.501	0.501
29	1	0.472	-0.528

4:

a-b)

I uploaded the data in weka using weka explorer. In weka explorer, open file button helped to import the dataset in weka.



c)

Data set contain a lot of attributes with name of supermarket items. Each items represent the transections of specific product. I visualize the attribute tea, biscuits, and haircare. Red color represents the high and blue color represent the low category from total attribute which is our class variable.

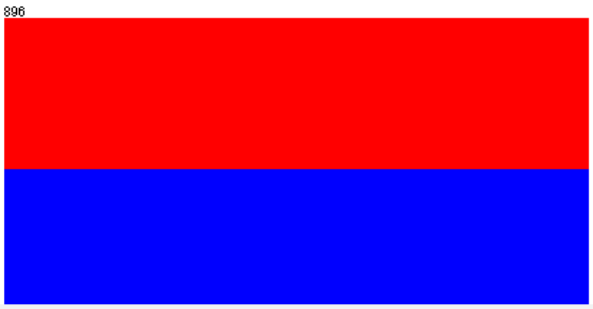
Selected attribute

Name: tea | Type: Nominal

Missing: 3731 (81%) | Distinct: 1 | Unique: 0 (0%)

No.	Label	Count	Weight
1	t	896	896

Class: total (Nom) | Visualize All



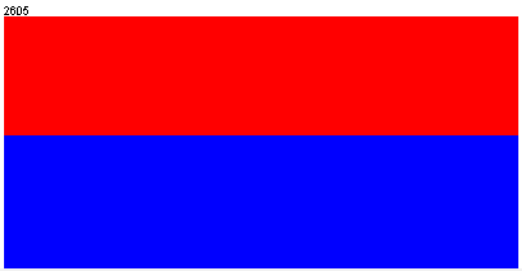
Selected attribute

Name: biscuits | Type: Nominal

Missing: 2022 (44%) | Distinct: 1 | Unique: 0 (0%)

No.	Label	Count	Weight
1	t	2605	2605

Class: total (Nom) | Visualize All



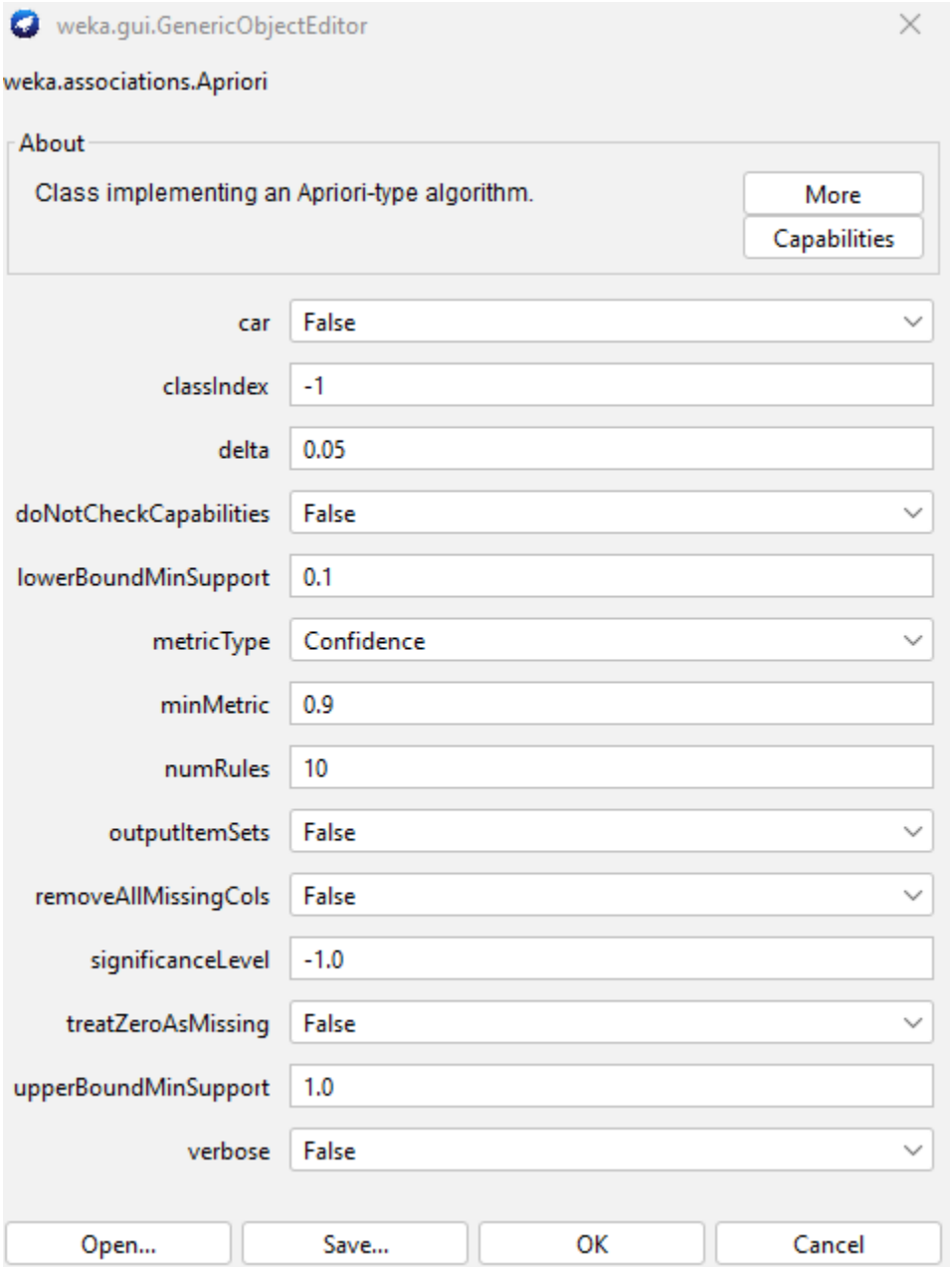
d-e)

Associate feature to help us find the association between the different attributes. Following picture represent the output of aprori algorithm.

```
Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:

 1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
 2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696    <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
 3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705    <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)
```

f)

weka helps us to increase and decrease the rule by adjusting number of rules confidence and other properties.



g-h)

following table represent the association of the items.

```
Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:

 1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
 2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696    <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
 3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705    <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)
```

If we take a look at the first rule, we can see

biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)

which shows us that there are 788 times biscuits, frozen foods, and fruit are sold together and are in high class. With all these we can also see bread and cake sold 723 times. Lift value is also greater than 1 which shows strong association of bread and cake biscuits, frozen food and fruits.

So, we can predict that,

If someone is buying biscuits, frozen foods, and fruits, he most probably buy the bread and cake as well.