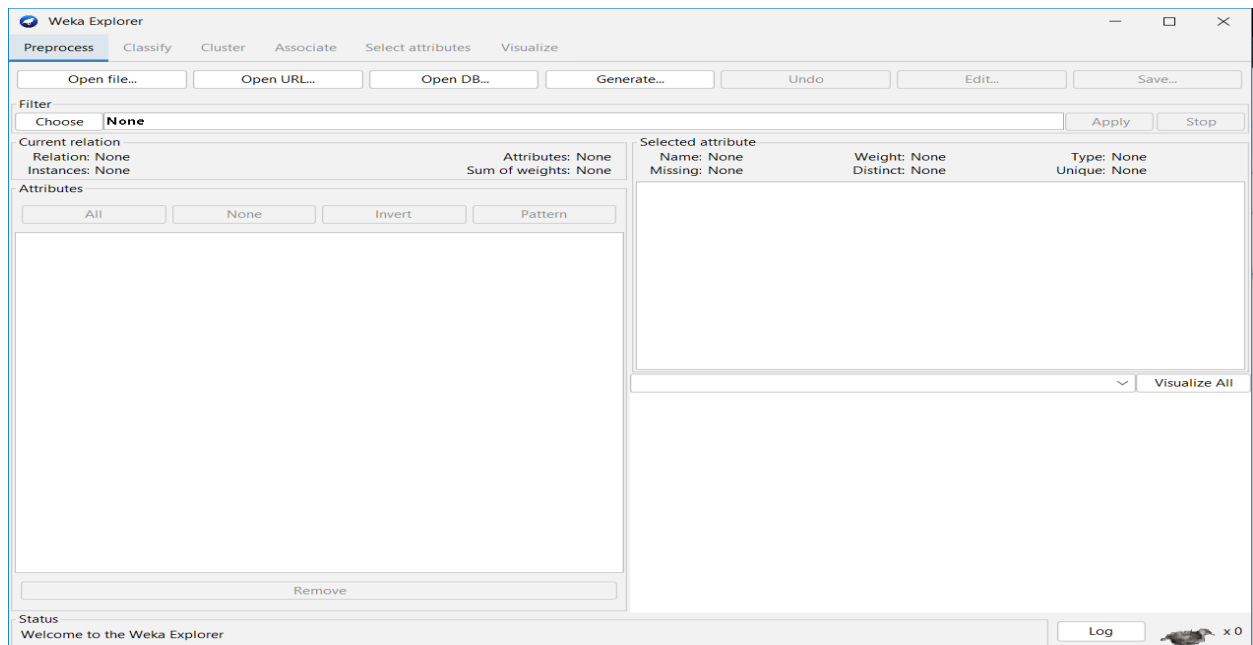
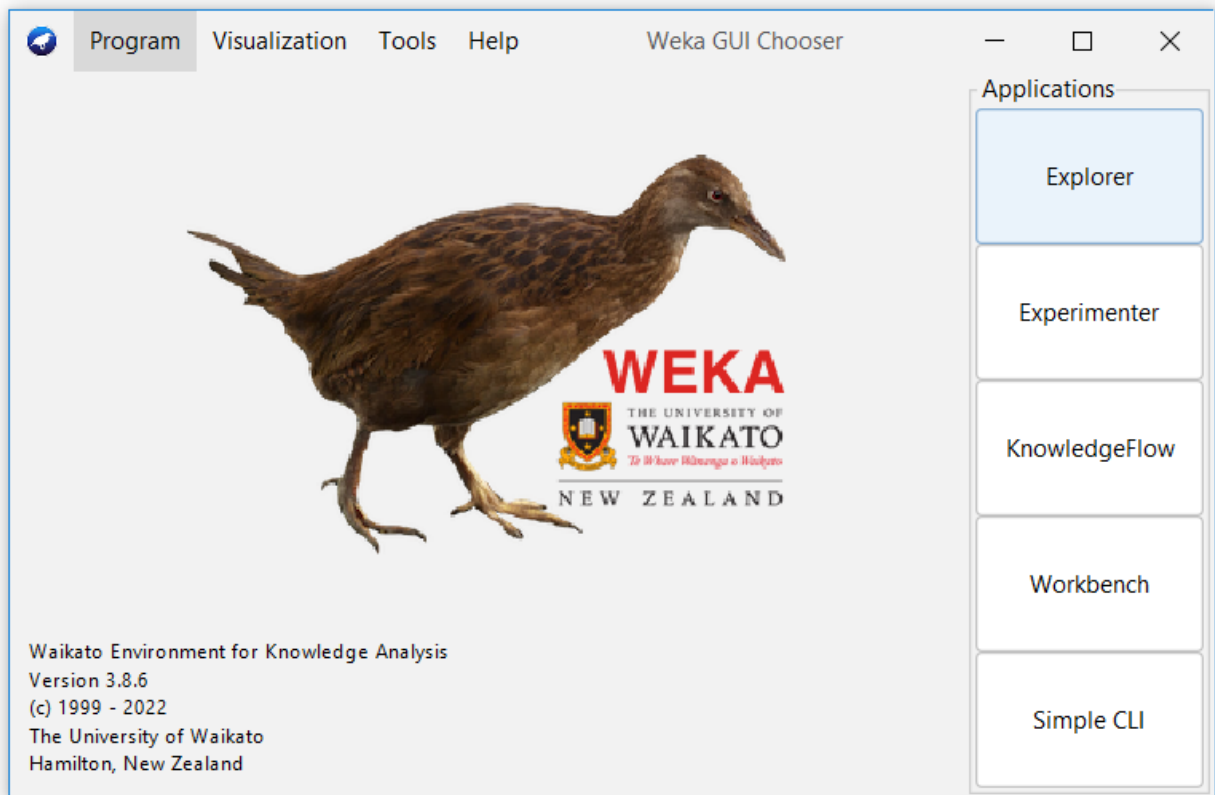
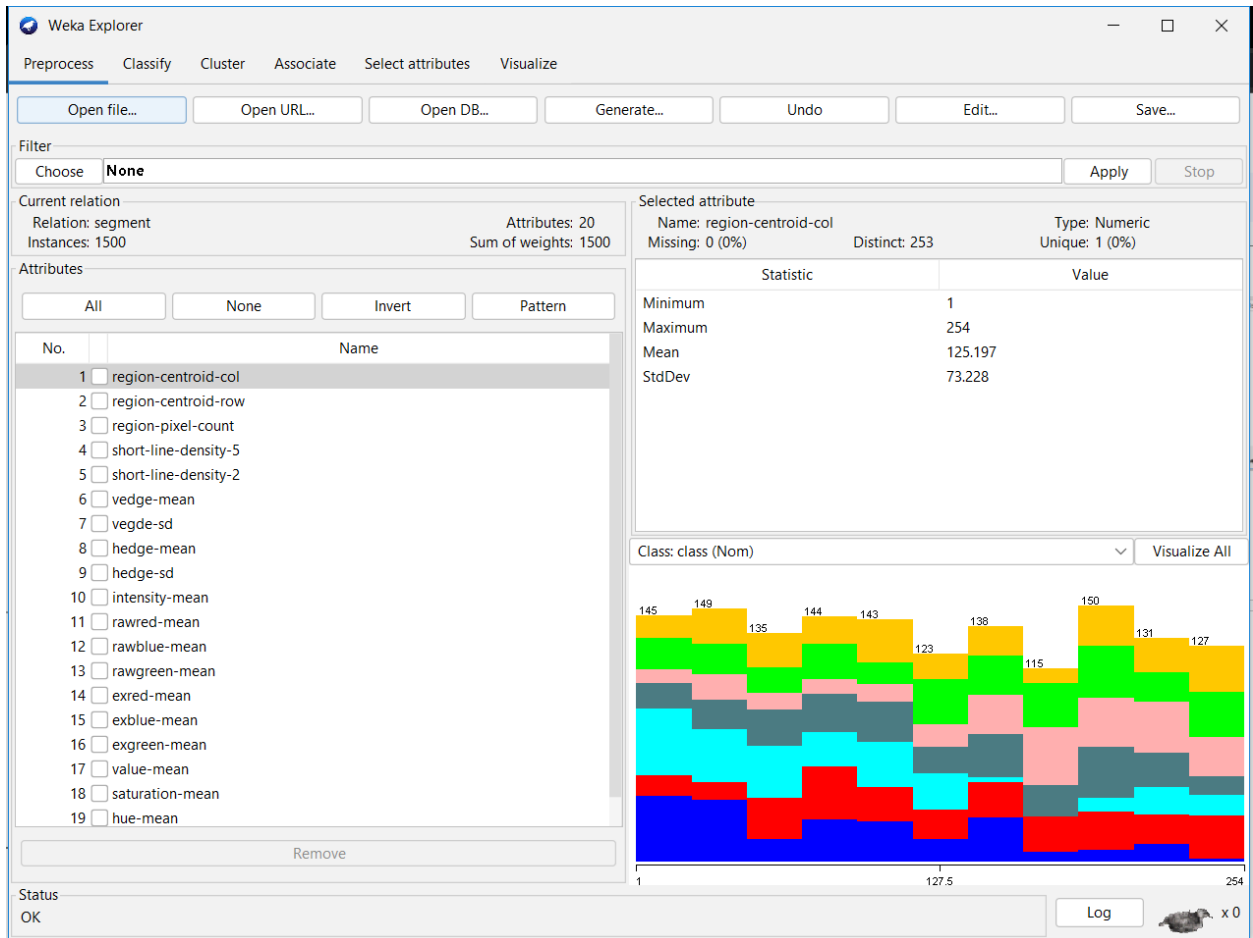
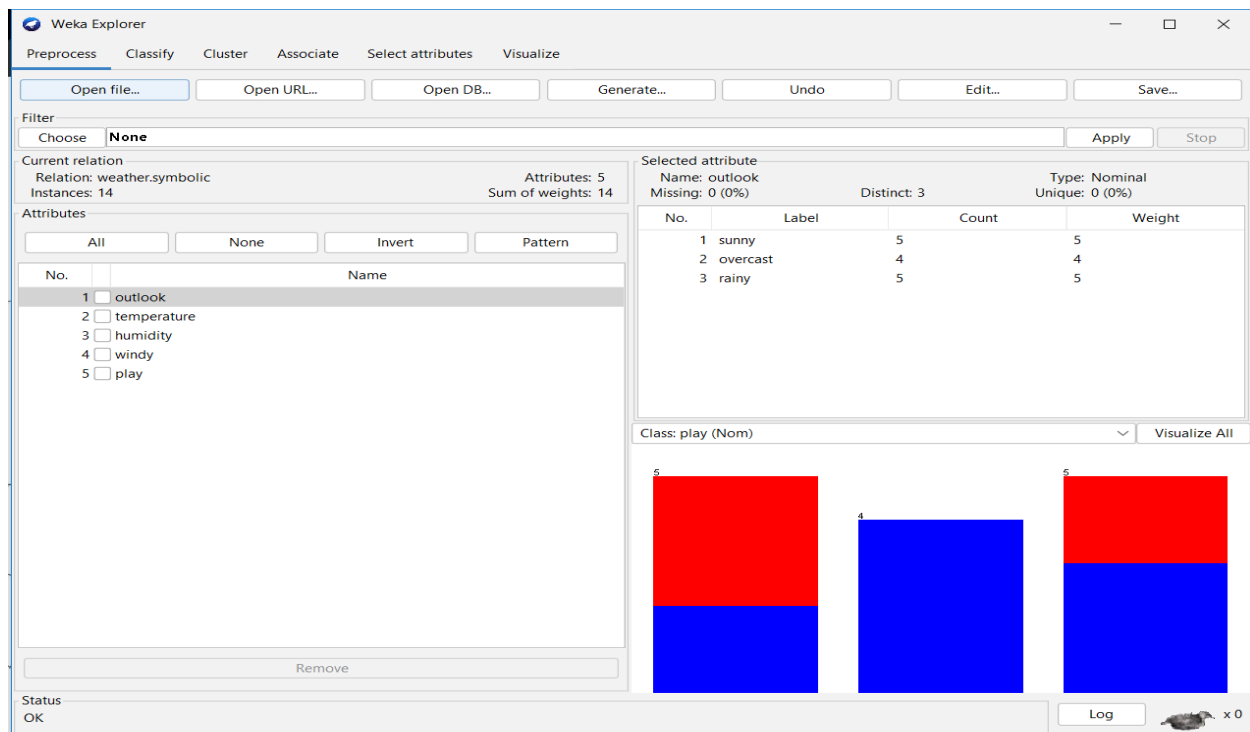


BUILD DATA WAREHOUSE AND EXPLORE WEKA

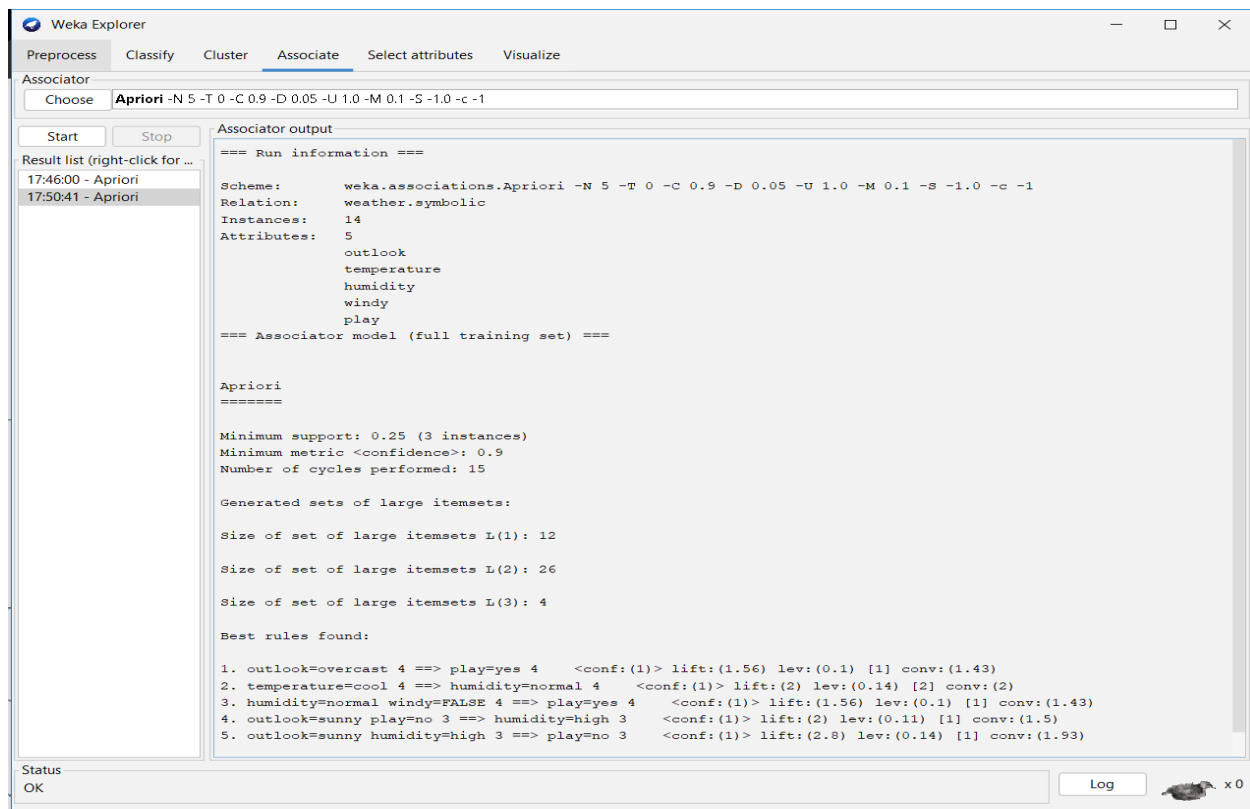




Perform data preprocessing task and demonstrate performing association rule mining on data sets



Association rule



Demonstrate of classification rule process on weka data set using naive bayes algorithm

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose **NaiveBayes**

Test options:
☐ Use training set
☐ Supplied test set (Set...)
☐ Cross-validation (Folds: 10)
☒ Percentage split (%: 70)
 More options...

(Nom) play
 Start Stop

Result list (right-click for options)
 17:54:44 - bayes.NaiveBayes

Classifier output

```

=== Run information ===

Scheme:      weka.classifiers.bayes.NaiveBayes
Relation:    weather.symbolic
Instances:   14
Attributes:  5
  outlook
  temperature
  humidity
  windy
  play
Test mode:   split 70.0% train, remainder test

=== Classifier model (full training set) ===

Naive Bayes Classifier

Attribute      Class      yes      no
                (0.63) (0.38)
=====
outlook
  sunny         3.0      4.0
  overcast      5.0      1.0
  rainy         4.0      3.0
  [total]       12.0     8.0

temperature
  hot           3.0      3.0
  mild          5.0      3.0
  cool          4.0      2.0
  [total]       12.0     8.0

humidity
  high          4.0      5.0
  normal        7.0      2.0
  [total]       11.0     7.0
  
```

Status: OK Log x 0

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose **NaiveBayes**

Test options:
☐ Use training set
☐ Supplied test set (Set...)
☐ Cross-validation (Folds: 10)
☒ Percentage split (%: 70)
 More options...

(Nom) play
 Start Stop

Result list (right-click for options)
 17:54:44 - bayes.NaiveBayes

Classifier output

```

windy
TRUE          4.0      4.0
FALSE         7.0      3.0
[total]       11.0     7.0

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances      2          50 %
Incorrectly Classified Instances    2          50 %
Kappa statistic                    0
Mean absolute error                 0.4913
Root mean squared error             0.5426
Relative absolute error             98.2684 %
Root relative squared error        102.942 %
Total Number of Instances          4

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          1.000    1.000    0.500     1.000    0.667      ?       0.500    0.750    yes
          0.000    0.000    ?         0.000    ?         ?       0.500    0.583    no
Weighted Avg.   0.500    0.500    ?         0.500    ?         ?       0.500    0.667

=== Confusion Matrix ===

a b  <-- classified as
2 0 | a = yes
2 0 | b = no
  
```

Status: OK Log x 0

Demonstrate performing Regression on datasets

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'SimpleLinearRegression' classifier is chosen. The 'Test options' section shows 'Cross-validation' selected with 10 folds. The 'Result list' on the left shows several entries, with the most recent one selected. The 'Classifier output' pane displays the following information:

```
=== Run information ===

Scheme:      weka.classifiers.functions.SimpleLinearRegression
Relation:    cpu
Instances:   209
Attributes:  7
             MYCT
             MMIN
             MMAX
             CACH
             CHMIN
             CHMAX
             class
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

Linear regression on MMAX

0.01 * MMAX - 34

Predicting 0 if attribute value is missing.

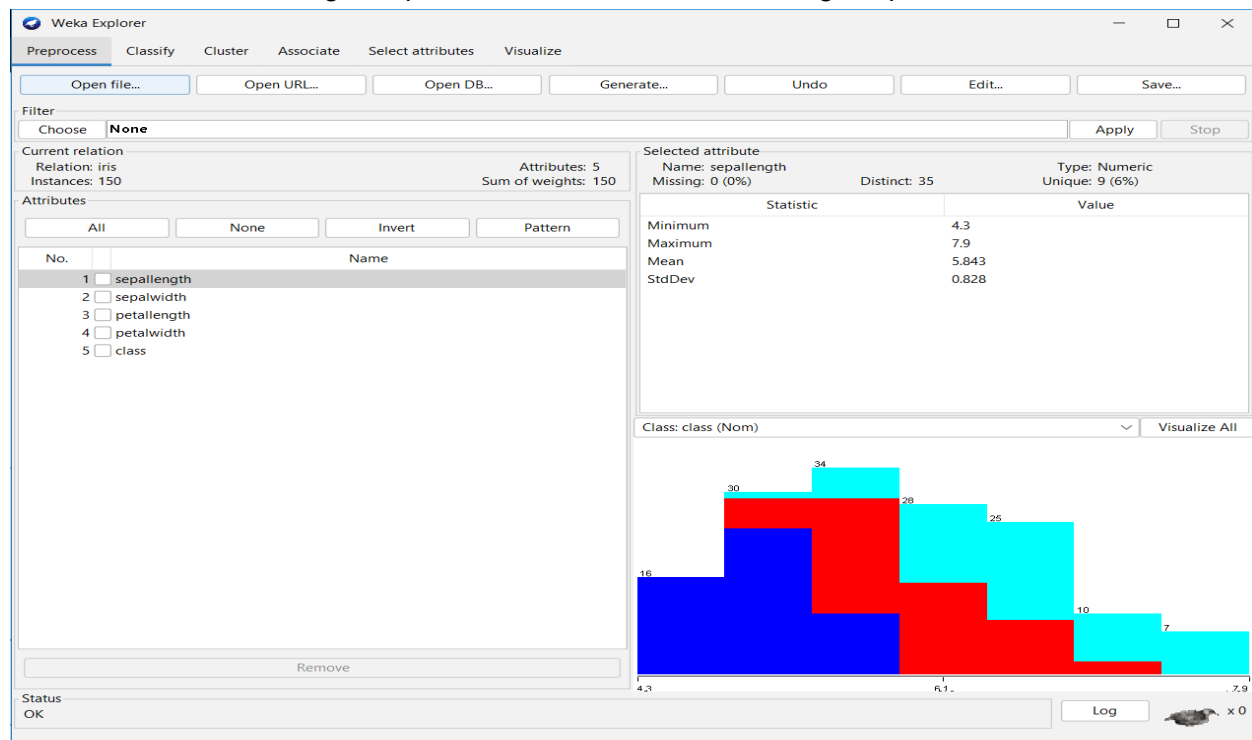
Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient      0.7844
Mean absolute error         53.8054
Root mean squared error     99.5674
Relative absolute error     55.908 %
Root relative squared error  61.8997 %
Total Number of Instances   209
```

The status bar at the bottom shows 'Status OK' and a 'Log' button.

Demonstrate of clustering rule process on data set iris.arff using simple k-mean



=== Run information ===

Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance" -R first-last -I 500 -num-slots 1 -S 10

Relation: iris

Instances: 150

Attributes: 5

sepalength

sepalwidth

petallength

petalwidth

class

Test mode: split 66% train, remainder test

=== Clustering model (full training set) ===

kMeans

=====

Number of iterations: 3

Within cluster sum of squared errors: 7.817456892309574

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor

Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

Cluster 2: 6.9,3.1,5.1,2.3,Iris-virginica

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Cluster#			
	Full Data (150.0)	0 (50.0)	1 (50.0)	2 (50.0)
=====				
=====				
sepalength	5.8433	5.936	5.006	6.588
sepalwidth	3.054	2.77	3.418	2.974
petallength	3.7587	4.26	1.464	5.552
petalwidth	1.1987	1.326	0.244	2.026
class	Iris-setosa	Iris-versicolor	Iris-setosa	Iris-virginica

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on test split ===

kMeans

=====

Number of iterations: 3

Within cluster sum of squared errors: 5.344375826509504

Initial starting points (random):

Cluster 0: 4.8,3,1.4,0.3,Iris-setosa

Cluster 1: 6.7,3.1,4.7,1.5,Iris-versicolor

Cluster 2: 5.6,3,4.1,1.3,Iris-versicolor

Missing values globally replaced with mean/mode

Final cluster centroids:

Cluster#

Attribute	Full Data (99.0)	0 (35.0)	1 (33.0)	2 (31.0)
-----------	---------------------	-------------	-------------	-------------

=====

sepalwidth	5.8313	5.0514	6.6061	5.8871
sepalwidth	3.0586	3.4543	2.9576	2.7194
petalwidth	3.6848	1.4771	5.5697	4.171
petalwidth	1.1657	0.2571	1.9879	1.3161
class	Iris-setosa	Iris-setosa	Iris-virginica	Iris-versicolor

Time taken to build model (percentage split) : 0 seconds

Clustered Instances

0	15 (29%)
1	17 (33%)
2	19 (37%)

Visualization:



Write a program of Apriori algorithm using any programming language (implemented in Python)

```
#import required library
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import association_rules, apriori

#read the data from csv
df = pd.read_csv('bread_basket.csv')

# cleaning the item column
df['Item'] = df['Item'].str.strip()
df['Item'] = df['Item'].str.lower()

#reset index and group together
transactions_str = df.groupby(['Transaction',
'Item'])['Item'].count().reset_index(name='Count')

# making a mxn matrice where m=transaction and n=items and each row
represents whether the item was in the transaction or not
my_basket = transactions_str.pivot_table(index='Transaction',
columns='Item', values='Count', aggfunc='sum').fillna(0)

# making a function which returns 0 or 1
# 0 means item was not in that transaction, 1 means item present in that
transaction
def encode(x):
    if x<=0:
        return 0
    if x>=1:
        return 1

# applying the function to the dataset
my_basket_sets = my_basket.applymap(encode)

# using the 'apriori algorithm' with min_support=0.01 (1% of 9465)
# It means the item should be present in atleast 94 transaction out of
9465 transactions only when we considered that item in frequent itemset
frequent_items = apriori(my_basket_sets, min_support = 0.01, use_colnames
= True)
```

```

# now making the rules from frequent itemset generated above
rules = association_rules(frequent_items, metric = "lift", min_threshold =
1)
rules.sort_values('confidence', ascending = False, inplace = True)

# arranging the data from highest to lowest with respect to 'confidence'
rules.sort_values('confidence', ascending=False)

```

Output:

index	antecedents	consequents	support	confidence	lift
30	frozenset({'toast'})	frozenset({'coffee'})	0.023666138	0.704402516	1.472431495
28	frozenset({'spanish brunch'})	frozenset({'coffee'})	0.010882198	0.598837209	1.2517655
19	frozenset({'medialuna'})	frozenset({'coffee'})	0.03518225	0.569230769	1.189878364
23	frozenset({'pastry'})	frozenset({'coffee'})	0.047543582	0.552147239	1.154168202
1	frozenset({'alfajores'})	frozenset({'coffee'})	0.019651347	0.540697674	1.130234869