Data Warehousing And Data Mining

-Venkata Varun Mangu
192111125

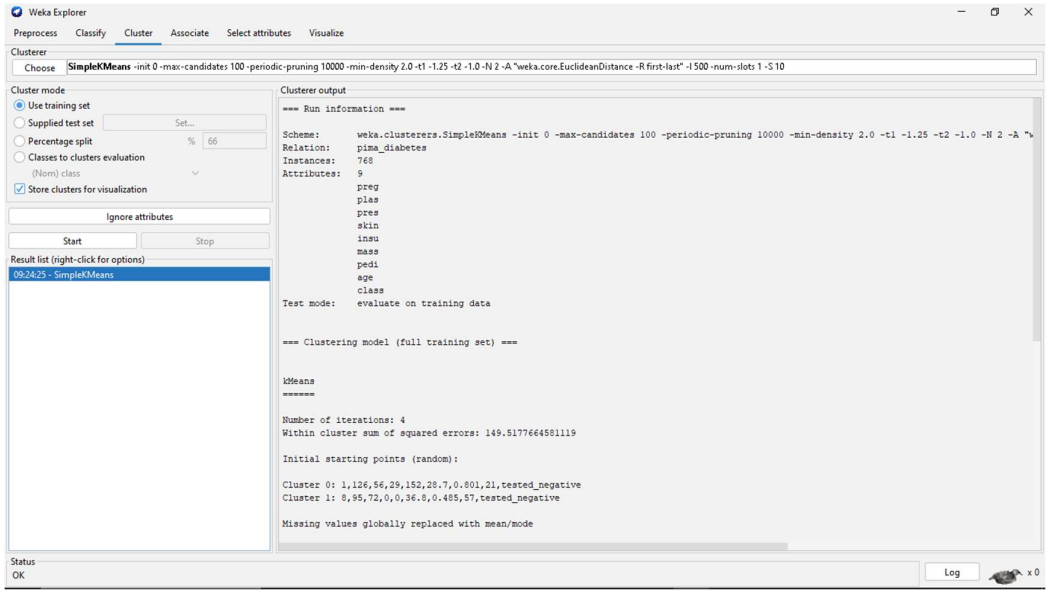
1.k-Means Clustering

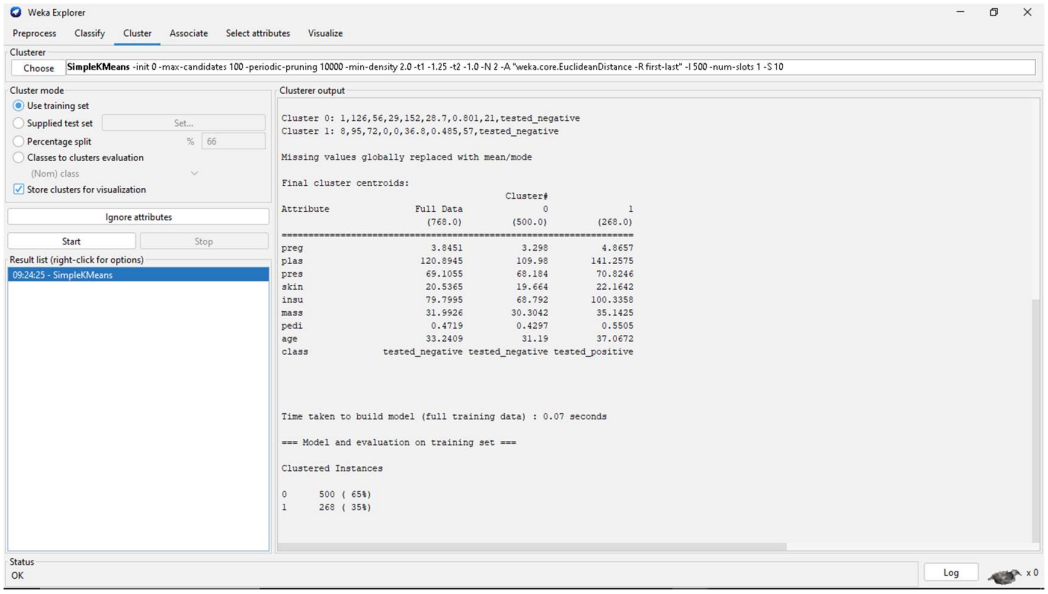
Data set:-diabetes

ALGORITHM:-

1. Provide K number of clusters
2. Select the K data points into the clusters
3. Now it will generate the cluster centroids
4. Iterate the following steps to find the final step
5. Sum of squared distance between the data points and centroids
6. Assign each data point to cluster until all clusters
7. At last compute the centroids for the clusters taking average of all data points of clusters

Output:-



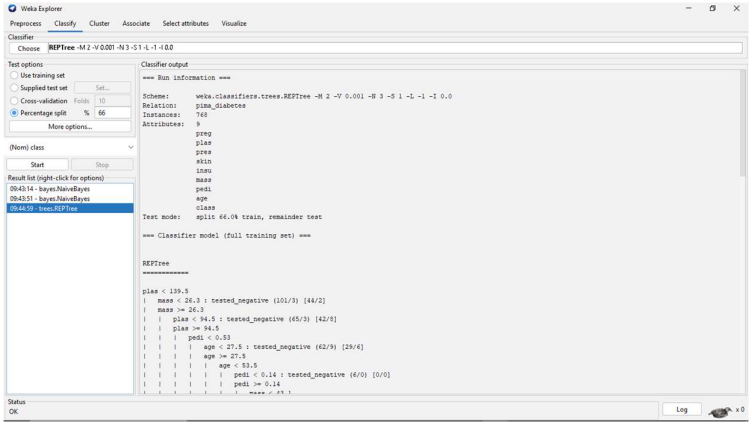


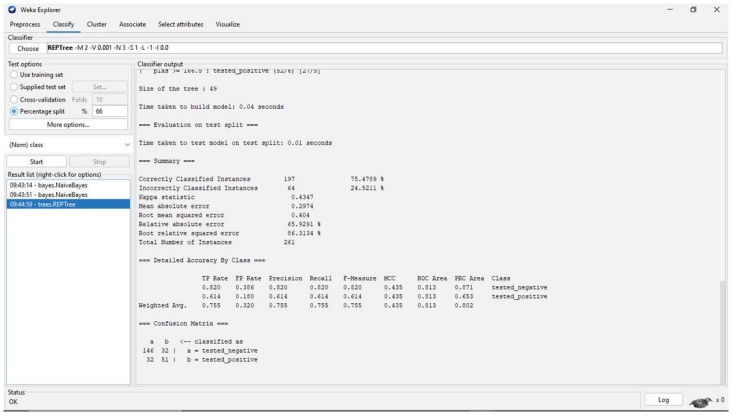
2.Decision Tree

Dataset:-diabetes

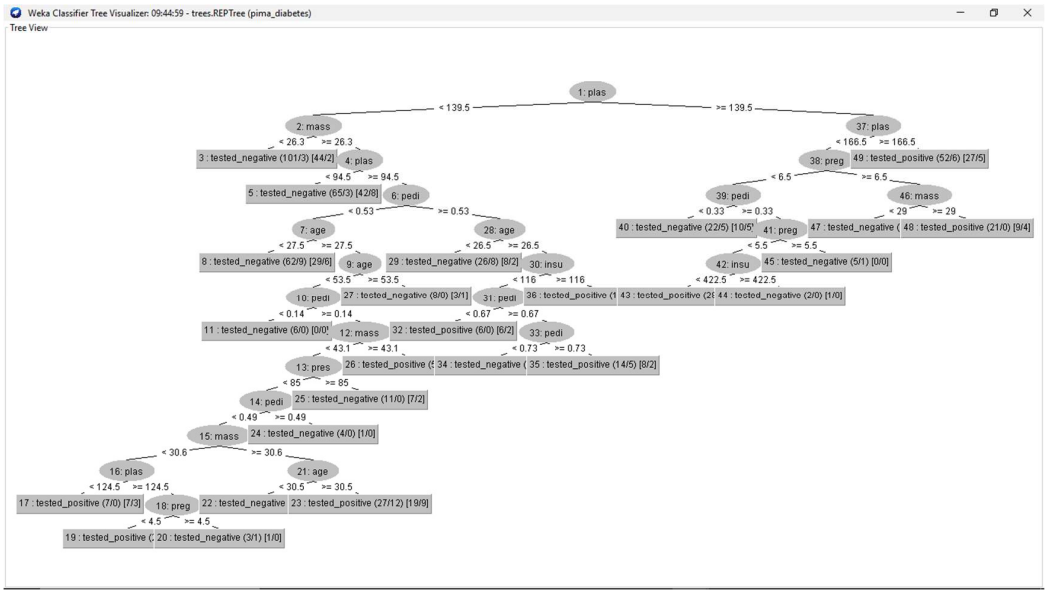
ALGORITHM:-
1.Determine the root node
2.Calculate Entropy(E=-∑ pi log2 pi)
3.Calculate Entropy split for each attribute
4.Calculate Information Gain
(IG=Entropy of parent node-sum of weighted entropy of child node)
5. Perform split
6. Perform further split until decision tree formed
7. Compute decision tree

OUT PUT:-





Decision tree:-



3.Bayesain Classification:-

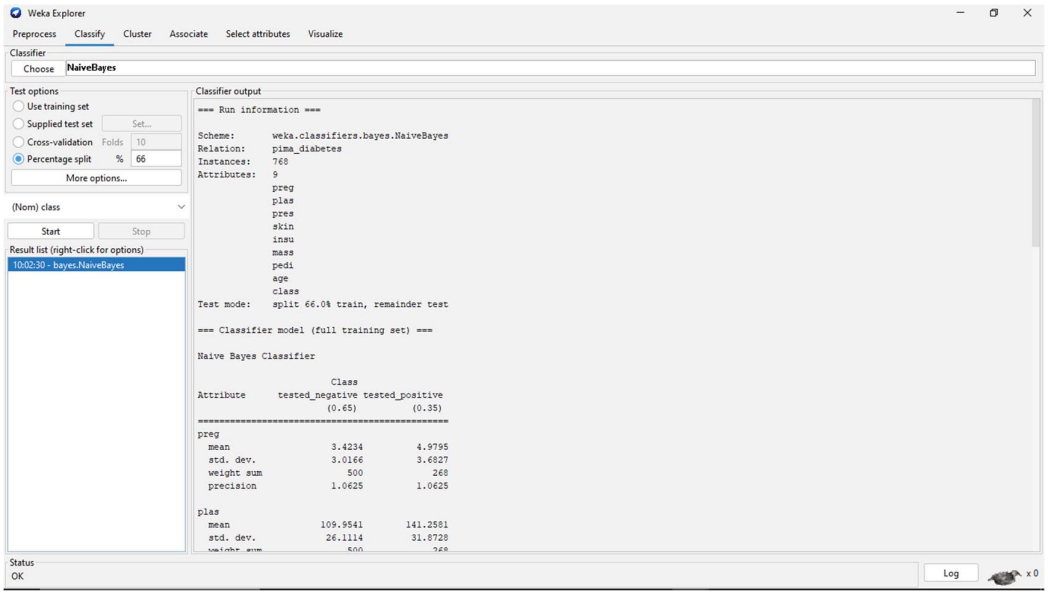
Data set:-diabetes

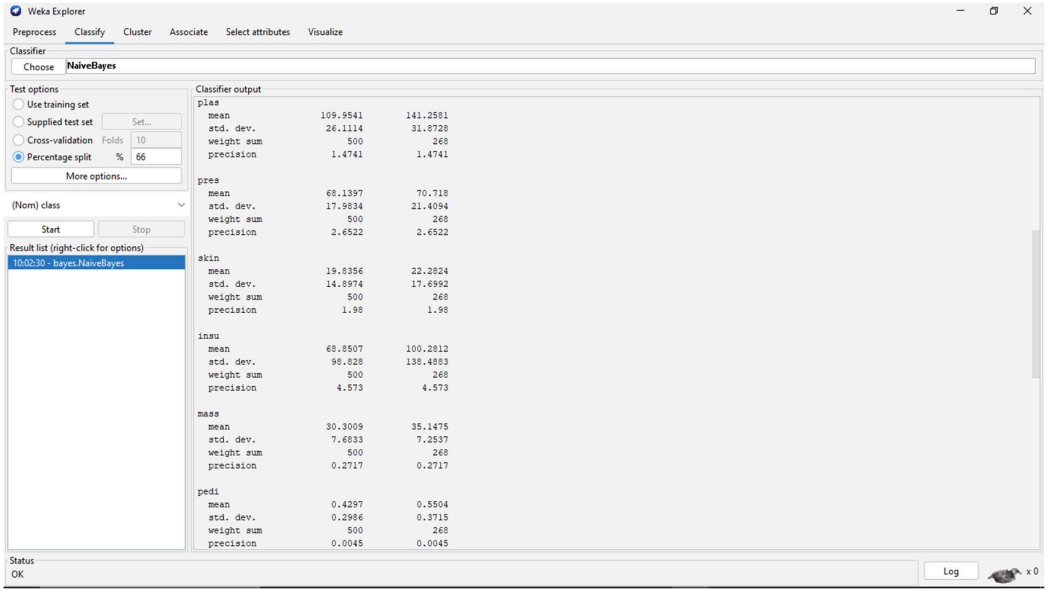
ALGORITHM:-

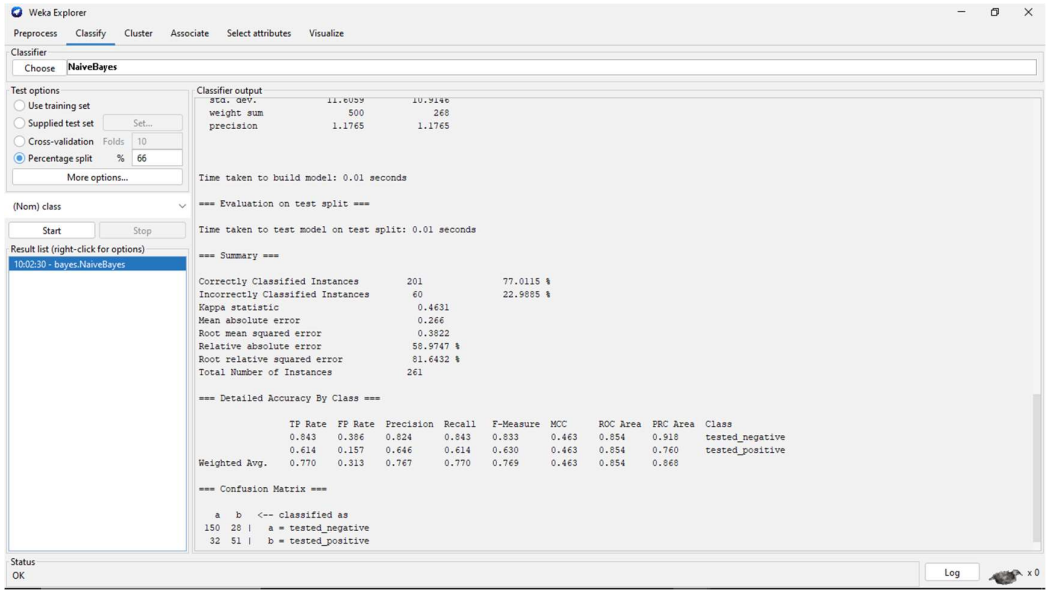
1.Convert given dataset into frequency table.
2.Construct livelihood tables by calculating the probabilities.
3.Use the bayes formula for calculating probabilities.
P(A|B) = [P(B|A) P(A)]/ P(B), where P(B) ≠ 0
4.now calculate the probability for all possible choices.
5.Then compare all the outputs.

6.Determine the probability which is more efficient by checking outputs.
7.Finally,compute the probability using bayesian classification

Output:-







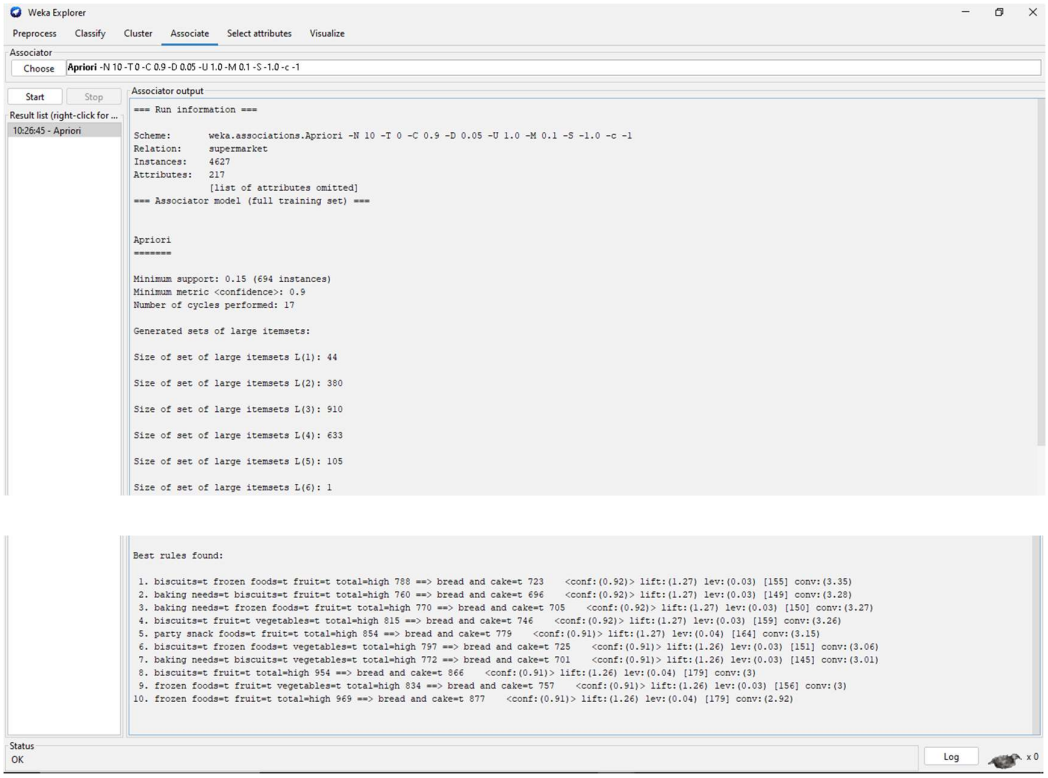
4.Apriori:-

Dataset:-Super market

ALGORITHM:-

1.firstly,convert the given transactional database into an frequency table.
2.Assign any minimum support to the frequency table,in which contains item sets and suppor count.
3.The item sets and support count is combinely called as candidate set.
4.Now,check the support count with the minimum support.
5.Remove the support count which is less than minimum support and write the remaining item sets in descending order.
6.Again checking by combining two itemsets.
7.iterate the steps until the support count should be equal to minimum support.
Confidence=support(A∩B)/support(A)
8.calculate the confidence and convert it into percentage.
9.Finally,check which is more efficient.

Output:-



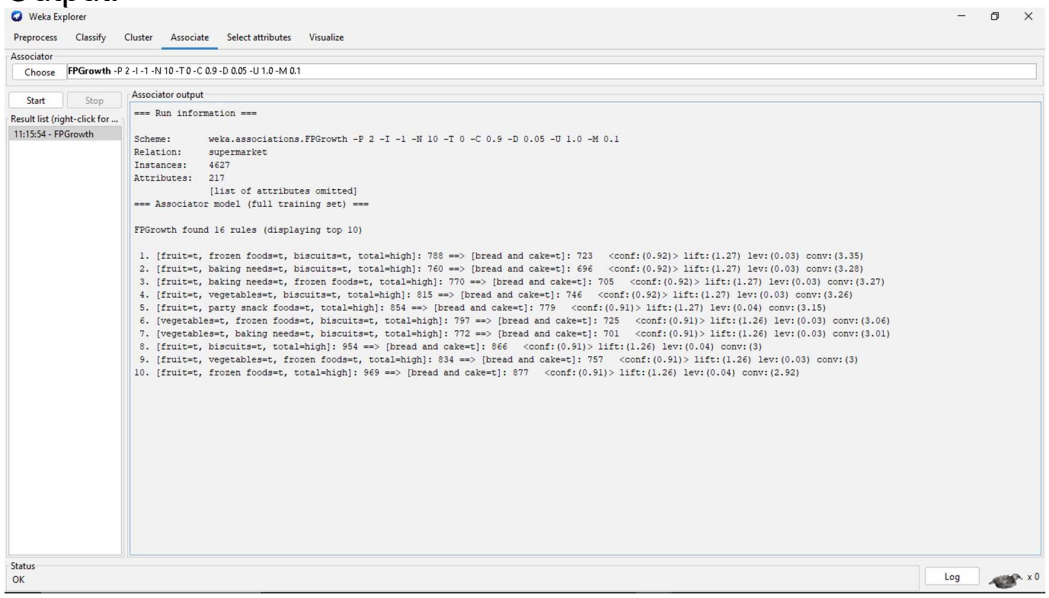
5.FP GROWTH

Dataset:-super market

ALGORITHM:-

1.firstly,convert the given transactional database into an frequency table.
2.Assign any minimum support to the frequency table,in which contains itemsets and support count.
3.The item sets and support count is combinely called as candidate set.
4.Now,check the support count with the minimum support.
5.Remove the support count which is less than minimum support and write remaining items in descending order.
6.Find the ordered item set using frequency table.
7.Construct the FP gowth using the ordered item set.
8.Then compute the conditionally pattern using FP grpowth.
9.Again find the conditionally frequency pattern.
10.Finally compute the FP gtowth algorithm.

Output:-


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Start
Stop
Result list (right-click for ...
11:15:54 - FPGrowth
Associator output
=== Run information ===
Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation: supermarket
Instances: 4627
Attributes: 217
[list of attributes omitted]
=== Associator model (full training set) ===
FPGrowth found 16 rules (displaying top 10)
1. [fruit=t, frozen foods=t, biscuits=t, total=high]: 788 ==> [bread and cake=t]: 723 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.35)
2. [fruit=t, baking needs=t, biscuits=t, total=high]: 760 ==> [bread and cake=t]: 696 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.28)
3. [fruit=t, baking needs=t, frozen foods=t, total=high]: 770 ==> [bread and cake=t]: 705 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.27)
4. [fruit=t, vegetables=t, biscuits=t, total=high]: 815 ==> [bread and cake=t]: 746 <conf:(0.92)> lift:(1.27) lev:(0.03) conv:(3.26)
5. [fruit=t, party snack foods=t, total=high]: 854 ==> [bread and cake=t]: 779 <conf:(0.91)> lift:(1.27) lev:(0.04) conv:(3.15)
6. [vegetables=t, frozen foods=t, biscuits=t, total=high]: 797 ==> [bread and cake=t]: 725 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.06)
7. [vegetables=t, baking needs=t, biscuits=t, total=high]: 772 ==> [bread and cake=t]: 701 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3.01)
8. [fruit=t, biscuits=t, total=high]: 954 ==> [bread and cake=t]: 866 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(3)
9. [fruit=t, vegetables=t, frozen foods=t, total=high]: 834 ==> [bread and cake=t]: 757 <conf:(0.91)> lift:(1.26) lev:(0.03) conv:(3)
10. [fruit=t, frozen foods=t, total=high]: 969 ==> [bread and cake=t]: 877 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(2.92)
Status
OK
Log
x 0