

Data Warehousing And Data Mining

-Venkata Varun Mangu
192111125

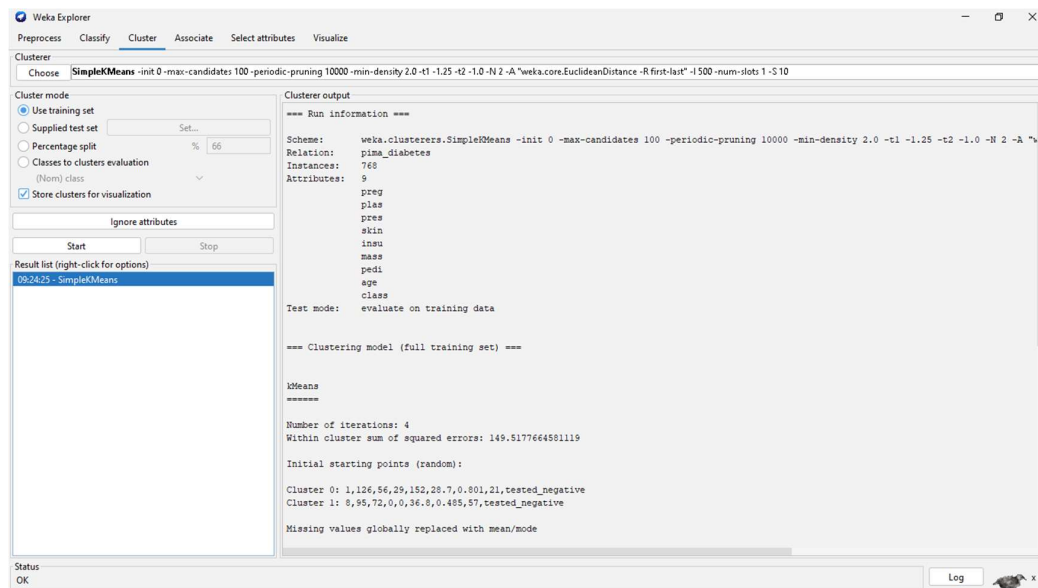
1.k-Means Clustering

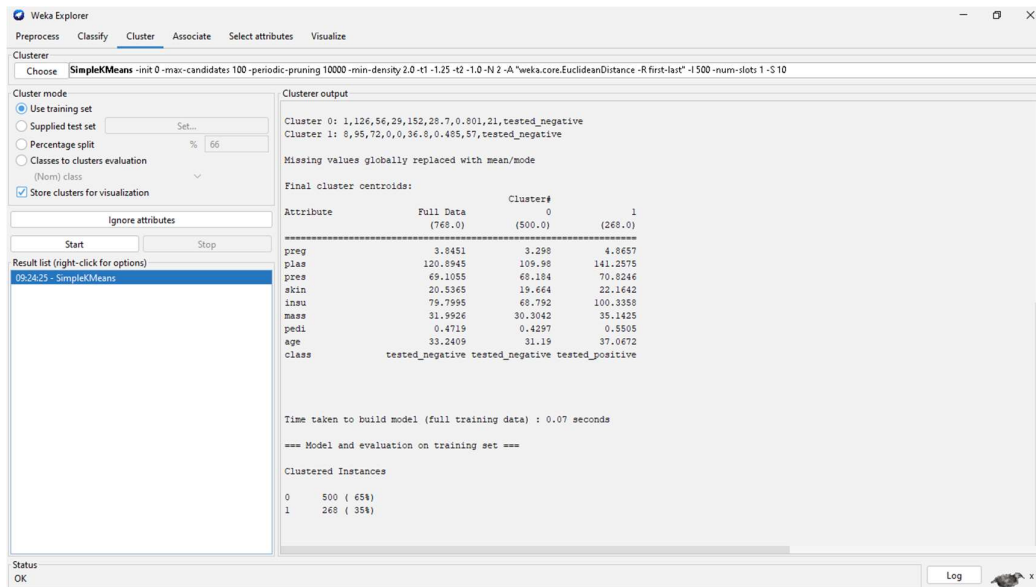
Data set:-diabetes

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:-





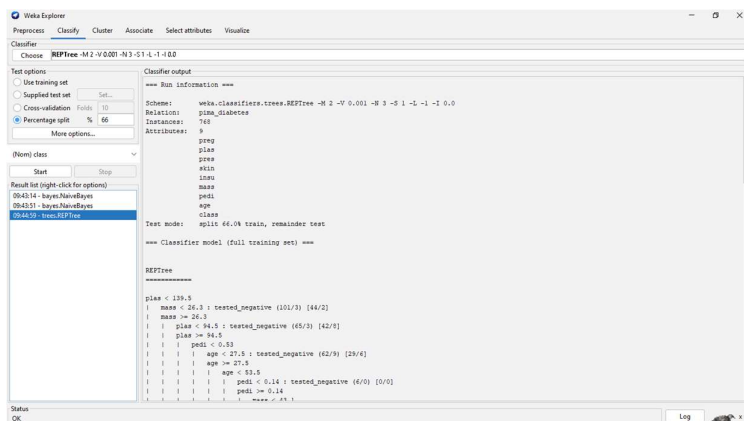
2.Decision Tree

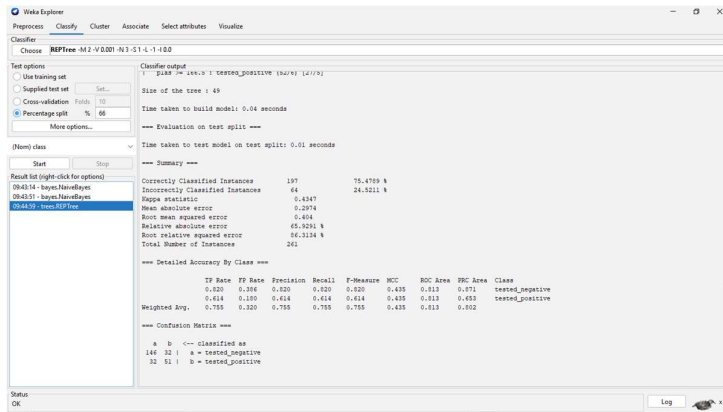
Dataset:-diabetes

ALGORITHM:-

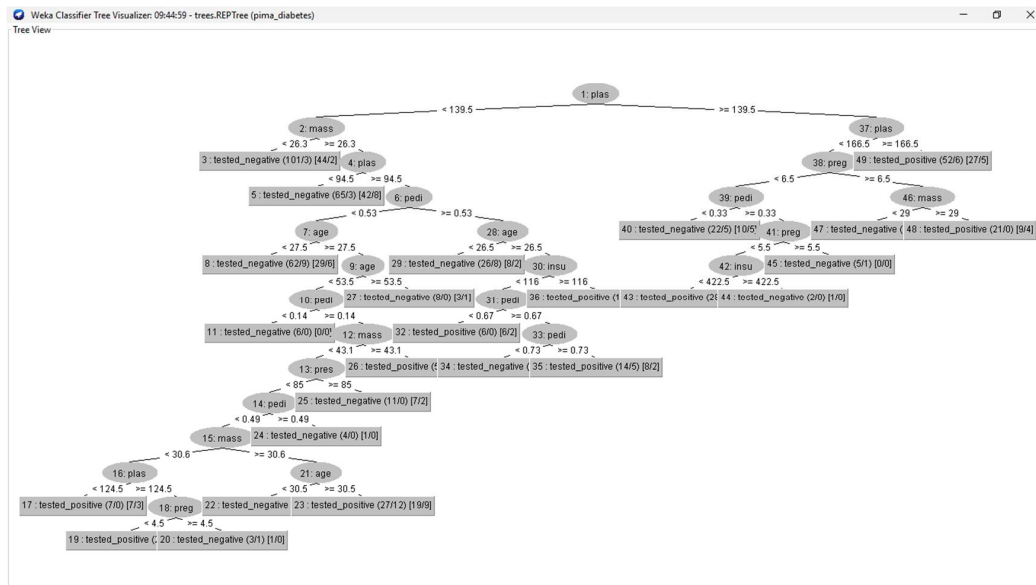
- 1.Determine the root node
- 2.Calculate Entropy($E = -\sum p_i \log_2 p_i$)
- 3.Calculate Entropy split for each attribute
- 4.Calculate Information Gain
($IG = \text{Entropy of parent node} - \text{sum of weighted entropy of child node}$)
5. Perform split
6. Perform further split until decision tree 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 likelihood tables by calculating the probabilities.
3. Use the bayes formula for calculating probabilities.

$$*P(A|B) = [P(B|A) P(A)] / P(B)^*$$
 where $P(B) \neq 0$
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:-

The screenshot shows the Weka Explorer interface with the NaiveBayes classifier selected. The test options are set to 'Percentage split' at 66%. The classifier output for the 'preg' attribute is displayed, showing the model's performance metrics.

Classifier output

```
=== Run information ===
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: pima_diabetes
Instances: 768
Attributes: 9
  preg
  plas
  pres
  skin
  insu
  mass
  pedi
  age
  class
Test mode: split 66.0% train, remainder test

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

Naive Bayes Classifier

Class
Attribute  tested_negative (0.65)  tested_positive (0.35)
-----
preg
mean      3.4234      4.9795
std. dev. 3.0166      3.6027
weight sum      500      268
precision 1.0625      1.0625

plas
mean      109.9541     141.2581
std. dev. 26.1114     31.8728
weight sum      500      268
precision 1.0625      1.0625
```

The screenshot shows the Weka Explorer interface with the NaiveBayes classifier selected. The test options are set to 'Percentage split' at 66%. The classifier output for the 'plas' attribute is displayed, showing the model's performance metrics.

Classifier output

```
plas
mean      109.9541     141.2581
std. dev. 26.1114     31.8728
weight sum      500      268
precision 1.4741      1.4741

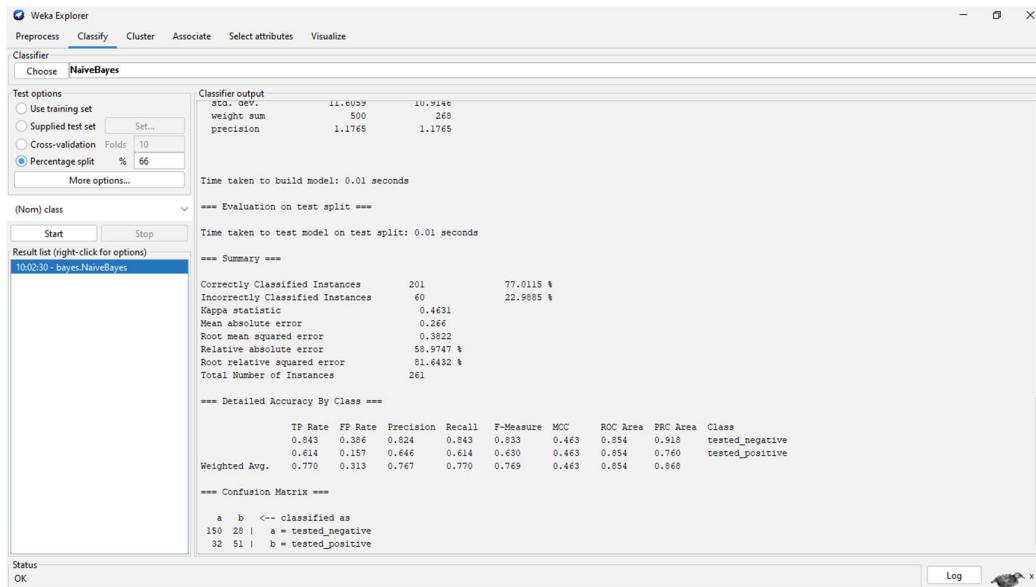
pres
mean      60.1397      70.718
std. dev. 17.9834     21.4094
weight sum      500      268
precision 2.6522      2.6522

skin
mean      19.8356      22.2824
std. dev. 14.8974     17.6992
weight sum      500      268
precision 1.98      1.98

insu
mean      68.8507      100.2812
std. dev. 98.828      138.4953
weight sum      500      268
precision 4.573      4.573

mass
mean      30.3009      35.1475
std. dev. 7.6833      7.2537
weight sum      500      268
precision 0.2717      0.2717

pedi
mean      0.4297      0.5504
std. dev. 0.2986      0.3715
weight sum      500      268
precision 0.0045      0.0045
```



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($A \cap B$)/support(A)*
- 8.calculate the confidence and convert it into percentage.
- 9.Finally,check which is more efficient.

Output:-

```

Weka Explorer
Preprocess  Classify  Cluster  Associate  Select attributes  Visualize

Associate
Choose  Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S 1.0 -c -1

Start  Stop
Result list (right-click for ...)
10:26:45 - Apriori

==== Run information ====

Scheme:      weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S 1.0 -c -1
Relation:    supermarket
Instances:   4627
Attributes:  217
              [list of attributes omitted]
==== Associator model (full training set) ====

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:(3.29)
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)

Status
OK
Log  x 0

```

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 growth using the ordered item set.
- 8.Then compute the conditionally pattern using FP growth.
- 9.Again find the conditionally frequency pattern.
- 10.Finally compute the FP growth 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]: 787 ==> [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