**A MINI PROJECT ON BREAST CANCER DATASET**

Submitted by

**BATCH NO: 4**

**16121A1285 P.DIVYA**

**16121A1286 P.LAVANYA**

**16121A1287 P.ALI ABBAS**

**16121A1288 P.ASHOK KUMAR**

**16121A1289 P.PRUDHVI KRISHNA**

**16121A1291 SIRISHA**

**16121A1292 DHARANEESH**

**16121A1293 FARHANA**

Under the Supervision of

**Mr. M. Thrilok Reddy,** *M.Tech*

Assistant Professor

Department of Information Technology



**INFORMATION TECHNOLOGY**

**Name of the lab: KNOWLEDGE ENGINEERING LAB** (16BT61232)

**SREE VIDYANIKETHAN ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE, and Accredited by NBA &amp NAAC)

aSree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

**2018-2019**

**CONTENTS**

**S. No Title Page No**

1. INTRODUCTION 3-4

2. BREAST CANCER DATA-SET

3 IMPLEMENTATION OF ALGORITHMS ON DATASET 6-27

1. **INTRODUCTION**

**Definition of data mining:**

In simple words, data mining is defined as a process used to extract usable data from a larger set of any raw data. It implies analyzing data patterns in large batches of data using one or more software. Data mining has applications in multiple fields, like science and research. As an application of data mining, businesses can learn more about their customers and develop more effective strategies related to various business functions and in turn leverage resources in a more optimal and insightful manner. This helps businesses be closer to their objective and make better decisions. Data mining involves effective data collection and warehousing as well as computer processing. For segmenting the data and evaluating the probability of future events, data mining uses sophisticated mathematical algorithms. Data mining is also known as Knowledge Discovery in Data (KDD).

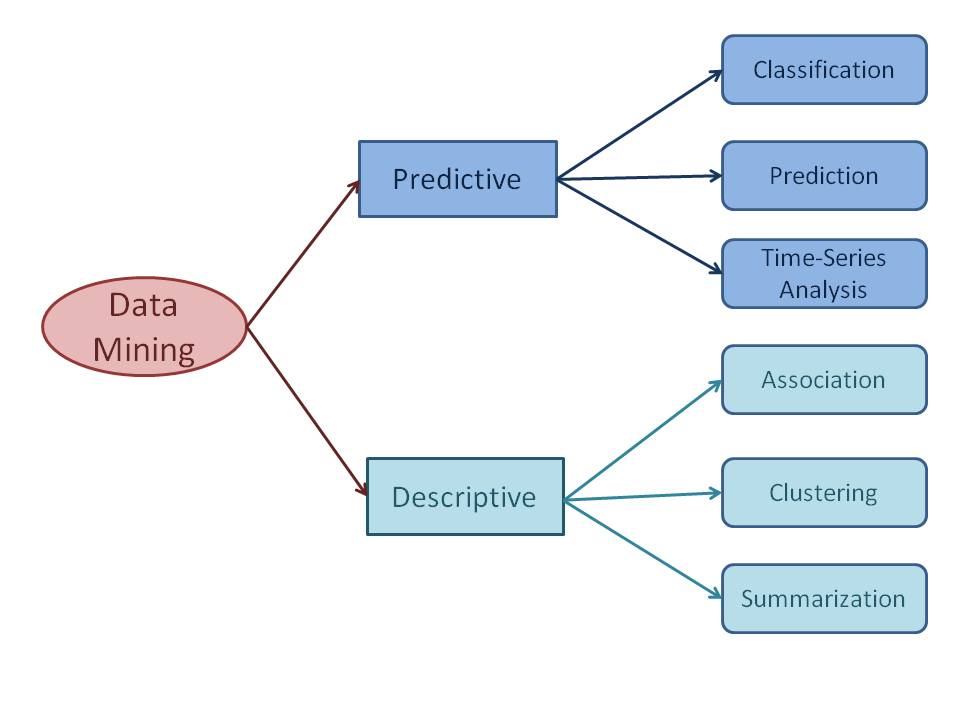
Alternative names of Data Mining:

* Knowledge discovery (mining) in databases (KDD),
* Knowledge extraction
* Data/pattern analysis
* Data archeology
* Data dredging
* Information harvesting
* Business intelligence

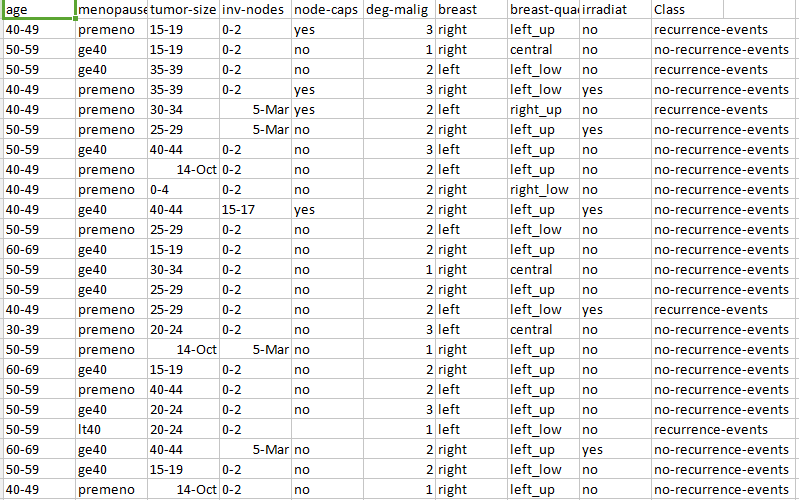
**Types of Data Mining Tasks:**

There are a number of data mining tasks such as classification, prediction, time-series analysis, association, clustering, summarization etc. All these tasks are either predictive data mining tasks or descriptive data mining tasks. A data mining system can execute one or more of the above specified tasks as part of data mining

Predictive data mining tasks come up with a model from the available data set that is helpful in predicting unknown or future values of another data set of interest. A medical practitioner trying to diagnose a disease based on the medical test results of a patient can be considered as a predictive data mining task. Descriptive data mining tasks usually finds data describing patterns and comes up with new, significant information from the available data set. A retailer trying to identify products that are purchased together can be considered as a descriptive data mining task.



1. **BREAST CANCER DATASET**

****

**3. IMPLEMENTATION OF ALGORITHMS**

In this data set we have applied two classification algorithms and one cluster algorithm. They are

**1.Classification algorithms:**

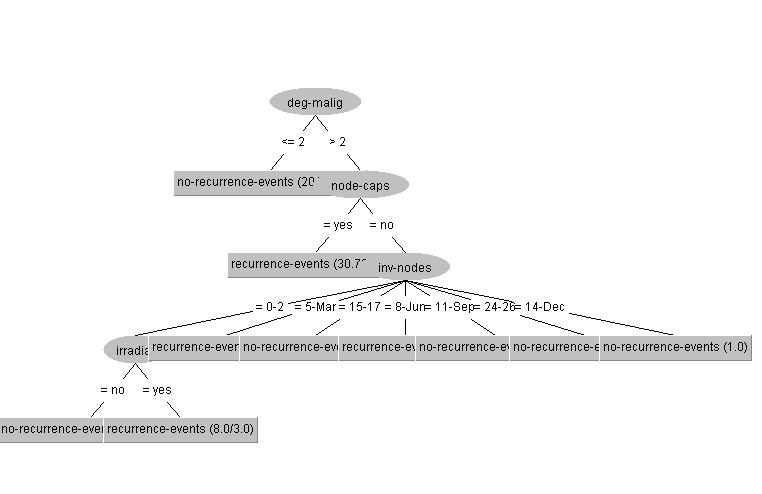
1. J48 algorithm (In WEKA software).
2. Naviebayes algorithm(In WEKA software ).
3. **Clustering algorithm:**
4. K-Mean Clustering Algorithm (In WEKA software).

**J48 algorithm:**

**Classification** is a **data mining** function that assigns items in a collection to target categories or classes. The goal of **classification** is to accurately predict the target class for each case in the **data**. For example, a **classification** model could be used to identify loan applicants as low, medium, or high credit risks.

Classification is the process of building a model of classes from a set of records that contain class labels. Decision Tree **Algorithm** is to find out the way the attributes-vector behaves for a number of instances. ... In the WEKA data mining tool, **J48** is an open source Java implementation of the C4.5 **algorithm**.

OUTPUT:



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

J48 pruned tree

------------------

deg-malig <= 2: no-recurrence-events (201.0/40.0)

deg-malig > 2

| node-caps = yes: recurrence-events (30.72/7.72)

| node-caps = no

| | inv-nodes = 0-2

| | | irradiat = no: no-recurrence-events (36.0/11.0)

| | | irradiat = yes: recurrence-events (8.0/3.0)

| | inv-nodes = 5-Mar: recurrence-events (6.0/1.0)

| | inv-nodes = 15-17: no-recurrence-events (1.0)

| | inv-nodes = 8-Jun: recurrence-events (1.0)

| | inv-nodes = 11-Sep: no-recurrence-events (1.28)

| | inv-nodes = 24-26: no-recurrence-events (0.0)

| | inv-nodes = 14-Dec: no-recurrence-events (1.0)

Number of Leaves : 10

Size of the tree : 14

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 214 74.8252 %

Incorrectly Classified Instances 72 25.1748 %

Kappa statistic 0.2742

Mean absolute error 0.3601

Root mean squared error 0.4319

Relative absolute error 86.0709 %

Root relative squared error 94.4911 %

Total Number of Instances 286

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.282 0.055 0.686 0.282 0.400 0.317 0.629 0.466 recurrence-events

0.945 0.718 0.757 0.945 0.841 0.317 0.629 0.754 no-recurrence-events

Weighted Avg. 0.748 0.521 0.736 0.748 0.710 0.317 0.629 0.668

=== Confusion Matrix ===

a b <-- classified as

24 61 | a = recurrence-events

11 190 | b = no-recurrence-events

# Naive Bayes Classifiers:

# Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for plaing golf.

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

* Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
* Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

The fundamental Naive Bayes assumption is that each feature makes an:

* independent
* equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

* We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
* Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

**Note:** The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes’ theorem.

**Bayes’ Theorem**

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

where A and B are events and P(B) ? 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

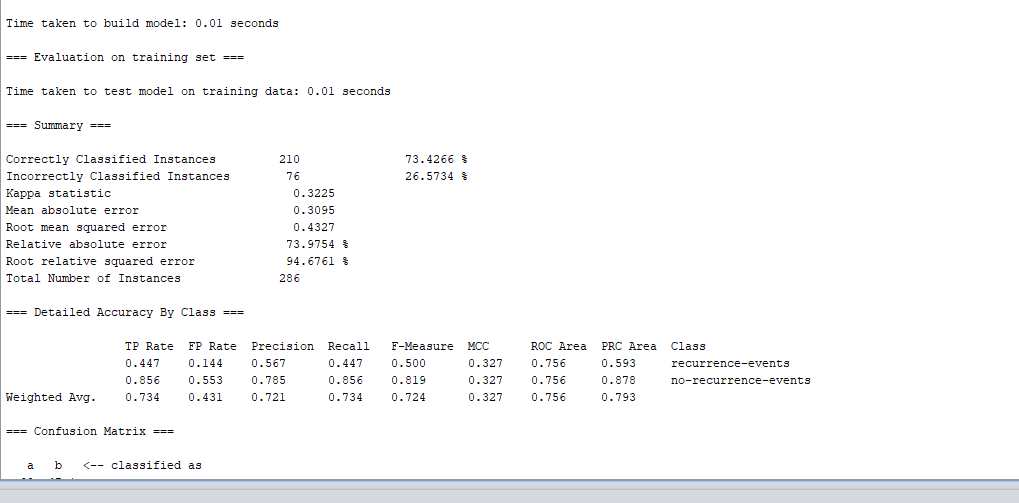
Now, with regards to our dataset, we can apply Bayes’ theorem in following way:

**PROCEDURE** **FOR NAIVE BAYES ALGORITHM IN WEKA:**

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose Diabetes data set and open file.
6. Click on classify tab and Choose weka🡪classifiers🡪bayes🡪naive bayes algorithm and select cross validation folds option.
7. Click on start button.

**OUTPUT:**=== Classifier model (full training set) ===

OUTPUT:



Class

Attribute recurrence-events no-recurrence-events

(0.3) (0.7)

=========================================================

age

40-49 28.0 64.0

50-59 26.0 72.0

60-69 18.0 41.0

30-39 16.0 22.0

70-79 2.0 6.0

20-29 1.0 2.0

[total] 91.0 207.0

menopause

premeno 49.0 103.0

ge40 36.0 95.0

lt40 3.0 6.0

[total] 88.0 204.0

tumor-size

15-19 8.0 24.0

35-39 8.0 13.0

30-34 26.0 36.0

25-29 19.0 37.0

40-44 7.0 17.0

14-Oct 2.0 28.0

0-4 2.0 8.0

20-24 17.0 35.0

45-49 2.0 3.0

50-54 4.0 6.0

9-May 1.0 5.0

[total] 96.0 212.0

inv-nodes

0-2 47.0 168.0

5-Mar 18.0 20.0

15-17 4.0 4.0

8-Jun 11.0 8.0

11-Sep 7.0 5.0

24-26 2.0 1.0

14-Dec 3.0 2.0

[total] 92.0 208.0

node-caps

yes 32.0 26.0

no 52.0 172.0

[total] 84.0 198.0

deg-malig

mean 2.3882 1.9055

std. dev. 0.721 0.6954

weight sum 85 201

precision 1 1

breast

right 37.0 99.0

left 50.0 104.0

[total] 87.0 203.0

breast-quad

left\_up 27.0 72.0

central 5.0 18.0

left\_low 36.0 76.0

right\_up 14.0 21.0

right\_low 7.0 19.0

[total] 89.0 206.0

irradiat

no 55.0 165.0

yes 32.0 38.0

[total] 87.0 203.0

Time taken to build model: 0.01 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances 210 73.4266 %

Incorrectly Classified Instances 76 26.5734 %

Kappa statistic 0.3225

Mean absolute error 0.3095

Root mean squared error 0.4327

Relative absolute error 73.9754 %

Root relative squared error 94.6761 %

Total Number of Instances 286

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.447 0.144 0.567 0.447 0.500 0.327 0.756 0.593 recurrence-events

0.856 0.553 0.785 0.856 0.819 0.327 0.756 0.878 no-recurrence-events

Weighted Avg. 0.734 0.431 0.721 0.734 0.724 0.327 0.756 0.793

=== Confusion Matrix ===

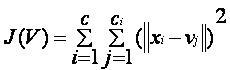
a b <-- classified as

38 47 | a = recurrence-events

29 172 | b = no-recurrence-events

**2.SIMPLE K-MEANS ALGORITHM**

k-means is  one of  the simplest unsupervised  learning  algorithms  that  solve  the well  known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as bary center of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of  this loop we  may  notice that the k centers change their location step by step until no more changes  are done or  in  other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

****

where,  
                           *‘||xi- vj||’* is the Euclidean distance between *xi* and *vj.*

*‘ci’* is the number of data points in *ith* cluster.

*‘c’* is the number of cluster centers.

**Algorithmic steps for k-means clustering**

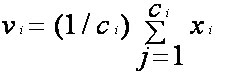
Let  X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select *‘c’* cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:



Where,*‘ci’* represents the number of data points in *ith* cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

**PROCEDURE FOR K-MEANS ALGORITHM IN WEKA:**

1. Open WEKA Tool.

2. Click on WEKA Explorer.

3. Click on Preprocessing tab button.

4. Click on open file button.

5. Choose WEKA folder in C drive.

6. Select and Click on data option button.

7. Choose diabetes data set and open file.

8. Click on cluster tab and Choose k-mean and select use training set test option.

9. Click on start button.

**OUTPUT:**

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

kMeans

======

Number of iterations: 3

Within cluster sum of squared errors: 1079.3135249855275

Initial starting points (random):

Cluster 0: 50-59,premeno,14-Oct,0-2,no,2,right,left\_up,no,no-recurrence-events

Cluster 1: 40-49,premeno,15-19,0-2,yes,3,right,left\_up,no,recurrence-events

Missing values globally replaced with mean/mode

Final cluster centroids:

Cluster#

Attribute Full Data 0 1

(286.0) (213.0) (73.0)

======================================================================================

age 50-59 50-59 40-49

menopause premeno premeno premeno

tumor-size 30-34 25-29 30-34

inv-nodes 0-2 0-2 0-2

node-caps no no yes

deg-malig 2.049 1.8592 2.6027

breast left left left

breast-quad left\_low left\_low left\_low

irradiat no no no

Class no-recurrence-events no-recurrence-events recurrence-event

Time taken to build model (full training data) : 0.05 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 213 ( 74%)

1 73 ( 26%)

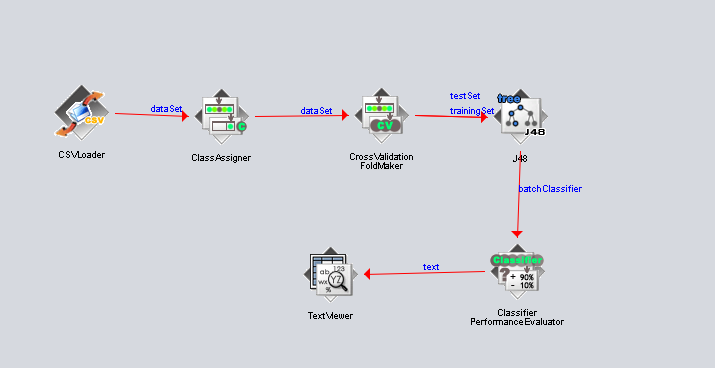
**Knowledge Flow:**

* The Knowledge Flow presents a data-flow inspired interface to WEKA. The user can select WEKA components from a tool bar, place them on a layout canvas and connect them together in order to form a knowledge-flow for processing and analyzing data.
* It has all of WEKA’s classifiers, filters, clusters, loaders and savers are available in the Knowledge Flow along with some extra tools.
* Data “flows” through modules: e.g., “data source” ->“filter” ->“classifier”-> “evaluator”.
* KF layouts can be saved and re-used later.

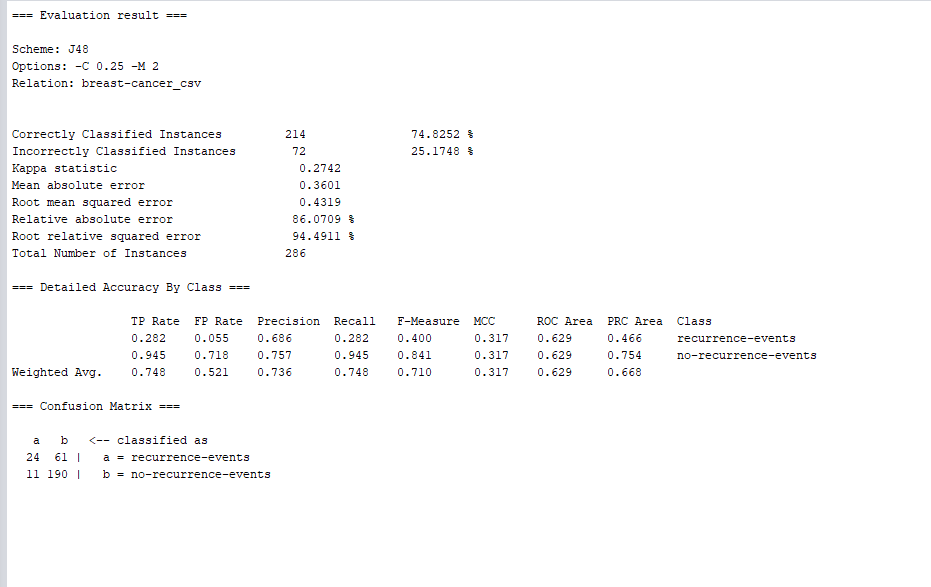
**Procedure to implement knowledge flow on breast cancer dataset:**

Setting up a flow to load an CSV file (batch mode) and perform a cross validation using algorithms:

1. Click on the Data Sources tab and choose CSVLoader from the toolbar.
2. Next place the CSVLoader component on the layout area by clicking somewhere on the layout.
3. Next specify an CSV file to load by first right clicking the mouse over the CSVLoader icon on the layout. A pop-up menu will appear. Select Configure under Edit in the list from this menu and browse to the location of your breast cancer CSV file.
4. Next click the Evaluation tab at the top of the window and choose the Class Assigner (allows you to choose which column to be the class) component from the toolbar. Place this on the layout.
5. Now connect the CSVLoader to the Class Assigner: first right click over the CSVLoader and select the dataset under Connections in the menu. A rubber band line will appear. Move the mouse over the Class Assigner component and left click - a red line labeled dataSet will connect the two components.
6. Next right click over the Class Assigner and choose Configure from the menu. This will pop up a window from which you can specify which column is the class in your data (last is the default).
7. Next grab a CrossValidationFoldMaker component from the Evaluation toolbar and place it on the layout. Connect the Class Assigner to the CrossValidationFoldMaker by right clicking over Class Assigner and selecting dataset from under Connections in the menu.
8. Next click on the Classifiers tab at the top of the window and scroll along the toolbar until you reach the required algorithm component in the trees or functions or rules section. Place a selected component on the layout.
9. Connect the CrossValidationFoldMaker to algorithm component TWICE by first choosing training Set and then testSet from the pop-up menu for the CrossValidationFoldMaker.
10. Next go back to the Evaluation tab and place a ClassifierPerformanceEvaluator component on the layout. Connect algorithm to this component by selecting the batchClassifier entry from the pop-up menu for algorithm component.
11. Next go to the Visualization toolbar and place a TextViewer component on the layout. Connect the ClassifierPerformanceEvaluator to the TextViewer by selecting the text entry from the pop-up menu for ClassifierPerformanceEvaluator.
12. Now start the flow executing by selecting Start loading from the pop-up menu for CSVLoader.
13. When finished you can view the results by choosing Show results from the pop-up menu for the TextViewer component.

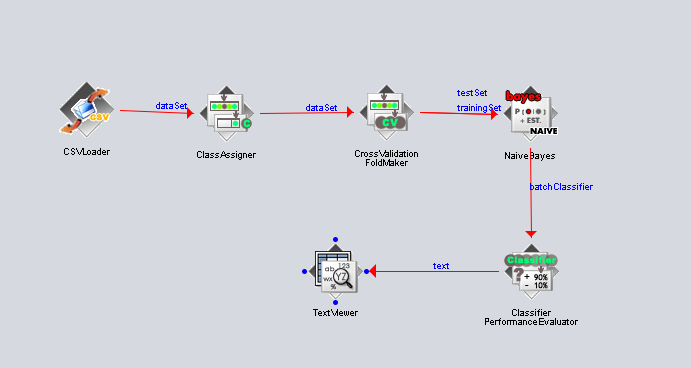
**OUTPUT:**

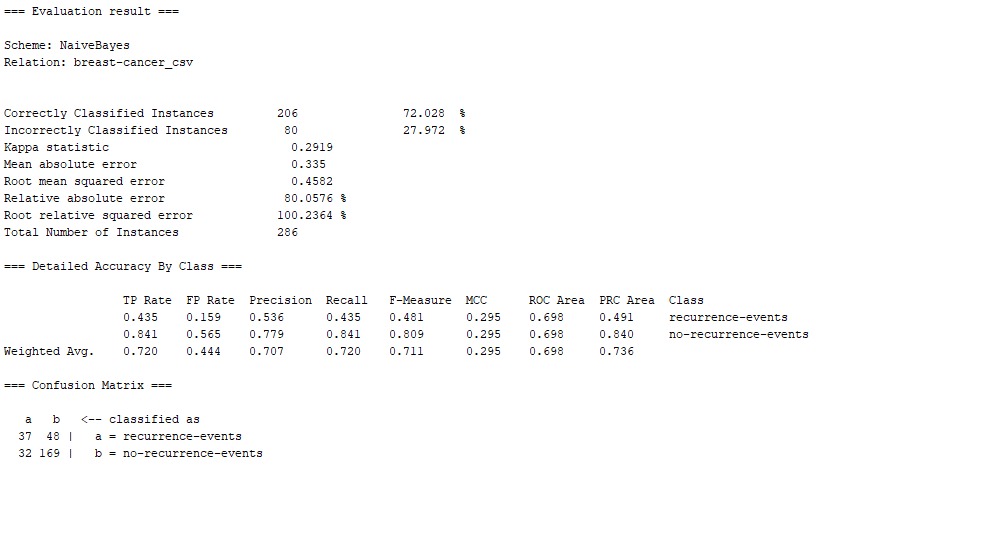
== Evaluation result ===

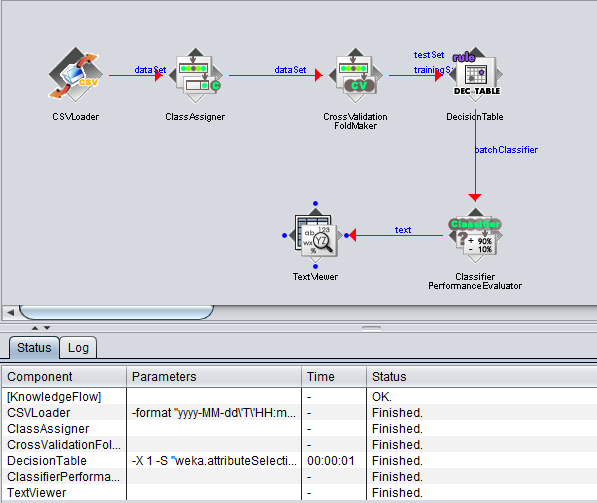


**J48 algorithm:**

**OUTPUT:**

****

****

****

**Performing Association rule mining by using Apriori Algorithm on ‘Breast cancer’ Dataset using R.**

**Introduction to R:-**

* R is a programming language and software environment for statistical analysis, graphics representation and reporting.
* R was created by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.
* R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

**The important features of R −**

* R is a well-developed, simple and effective programming language which includes conditionals, loops; user defined recursive functions and input and output facilities.
* R has an effective data handling and storage facility,
* R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
* R provides a large, coherent and integrated collection of tools for data analysis.
* R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

Association mining is usually done on transactions data from a retail market or from an online e-commerce store. Since most transactions data is large, the apriori algorithm makes it easier to find these patterns or rules quickly.

A rule is a notation that represents which item/s is frequently bought with what item/s. It has an *LHS* and an *RHS* part and can be represented as follows:

**itemset A => itemset B**

This means, the item/s on the right were frequently purchased along with items on the left.

Since association mining deals with transactions, the data has to be converted to one of class transactions, made available in R through the **arules** pkg. This is a necessary step because the **apriori()** function accepts transactions data of class transactions only.

The **apriori()**generates the most relevant set of rules from a given transaction data. It also shows the support, confidence and lift of those rules. These three measures can be used to decide the relative strength of the rules.

To view the transactions, use the **inspect()** function instead.

The **eclat()** takes in a transactions object and gives the most frequent items in the data based the support you provide to the supp argument. The **maxlen** defines the maximum number of items in each **item set** of frequent items.

**OUTPUT:**

> **library(arules)**

**> tn<-read.transactions(file="E:/forestfires.csv",rm.duplicates= FALSE,format = "single",sep=",",cols=c(1,2))**

**> tn**

transactions in sparse format with

10 transactions (rows) and

8 items (columns)

**>frequentItems <- eclat (tn, parameter = list(supp = 0.22, maxlen = 3))**

Eclat

parameter specification:

tidLists support minlen maxlen target ext

FALSE 0.22 1 3 frequent itemsets FALSE

algorithmic control:

sparse sort verbose

7 -2 TRUE

Absolute minimum support count: 2

create itemset ...

set transactions ...[8 item(s), 10 transaction(s)] done [0.00s].

sorting and recoding items ... [4 item(s)] done [0.00s].

creating bit matrix ... [4 row(s), 10 column(s)] done [0.00s].

writing ... [14 set(s)] done [0.00s].

Creating S4 object ... done [0.00s].

**> inspect(frequentItems)**

items support count

[1] {3,4,6} 0.5 5

[2] {3,5,6} 0.5 5

[3] {4,5,6} 0.7 7

[4] {4,6} 0.7 7

[5] {5,6} 0.7 7

[6] {3,6} 0.5 5

[7] {3,4,5} 0.7 7

[8] {3,4} 0.7 7

[9] {3,5} 0.7 7

[10] {4,5} 0.9 9

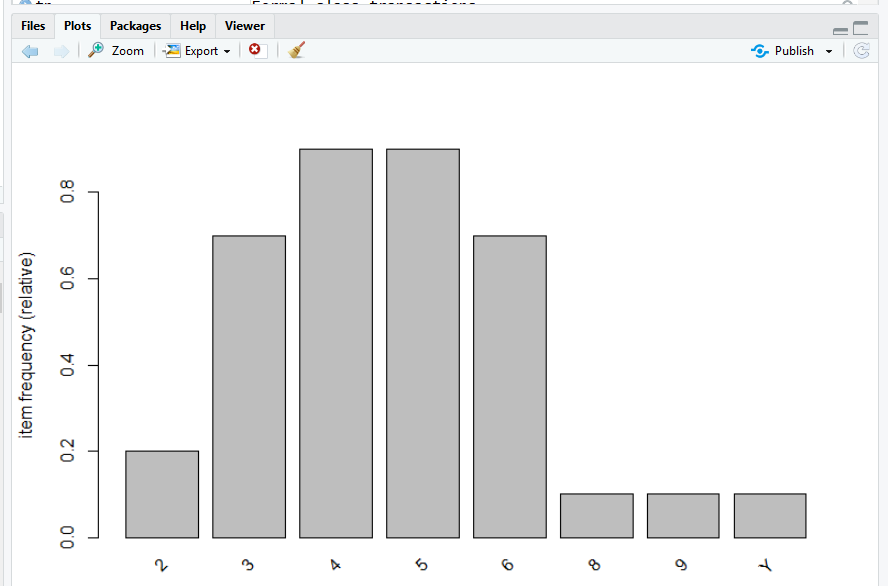
[11] {4} 0.9 9

[12] {5} 0.9 9

[13] {3} 0.7 7

[14] {6} 0.7 7

> **itemFrequencyPlot(tn)**



**> rules <- apriori (tn, parameter = list(supp = 0.22, conf = 0.7))**

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext

0.7 0.1 1 none FALSE TRUE 5 0.22 1 10 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 2

set item appearances ...[0 item(s)] done [0.00s].

set transactions ...[8 item(s), 10 transaction(s)] done [0.00s].

creating transaction tree ... done [0.00s].

checking subsets of size 1 2 3 4 done [0.00s].

writing ... [32 rule(s)] done [0.00s].

creating S4 object ... done [0.00s].

**> rules\_conf <- sort (rules, by="confidence", decreasing=TRUE**)

**>rules <- apriori(tn, parameter = list (supp = 0.22, conf = 0.7, maxlen=3))**

Apriori

Parameter specification:

confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext

0.7 0.1 1 none FALSE TRUE 5 0.22 1 3 rules FALSE

Algorithmic control:

filter tree heap memopt load sort verbose

0.1 TRUE TRUE FALSE TRUE 2 TRUE

Absolute minimum support count: 2

set item appearances ...[0 item(s)] done [0.00s].

set transactions ...[8 item(s), 10 transaction(s)] done [0.00s].

sorting and recoding items ... [4 item(s)] done [0.00s].

creating transaction tree ... done [0.00s].

checking subsets of size 1 2 3 done [0.00s].

writing ... [28 rule(s)] done [0.00s].

creating S4 object ... done [0.00s]

**> inspect(rules)**

lhs rhs support confidence lift count

[1] {} => {6} 0.7 0.7000000 1.000000 7

[2] {} => {3} 0.7 0.7000000 1.000000 7

[3] {} => {4} 0.9 0.9000000 1.000000 9

[4] {} => {5} 0.9 0.9000000 1.000000 9

[5] {6} => {3} 0.5 0.7142857 1.020408 5

[6] {3} => {6} 0.5 0.7142857 1.020408 5

[7] {6} => {4} 0.7 1.0000000 1.111111 7

[8] {4} => {6} 0.7 0.7777778 1.111111 7

[9] {6} => {5} 0.7 1.0000000 1.111111 7

[10] {5} => {6} 0.7 0.7777778 1.111111 7