DATA MINING CLASSIFICATION FOR BREAST CANCER PREDICTION

Agung Mulyo Widodo[1*]**,Nizirwan Anwar** [2]**, Bambang Irawan**[3]**, Lista Meria**[4]
[1,2,3] Dept. of Computer Science, Esa Unggul University, West Jakarta, 11510, Indonesia
[4] Dept. of Economic and Business, Esa Unggul University, West Jakarta, 11510, Indonesia
[*]e-mail : agung.mulyo@esaunggul.ac.id

Abstract

Many machine learning methods are used in constructing a predictive model. In this study, a study was conducted on the performance of the Bagging, IBk and Random forest classification algorithms using the same dataset to predict the diagnosis of breast cancer. The results obtained that the highest accuracy value was the IBk algorithm and the lowest was the Random forest algorithm, of the three algorithms.

Keywords : the Bagging, IBk, Random forest, classification algorithms, breast cancer predictive

1. Introduction

Breast cancer is cancer that forms in the cells of the breasts. Breast cancer can occur in both men and women, but it's far more common in women and very rare among men. Breast cancer is highly heterogeneous disease. It represents the second important cause of cancer death in women today. Breast cancer is very serious malignant tumor originating from the breast cells. There are two types of breast cancer namely benign and malignant. Benign breast cancer is non-invasive types of breast cancer is rarely a threat to life, occurs in the lining of milk ducts. In this type cancer doesn’t spread to neighboring tissues and remains in ducts that’s why called as ductal carcinoma. Malignant Breast cancer: This is invasive type of breast cancer begins in the lobules of the breast so named as lobular carcinoma. It spreads from where it began in the breast lobules to surrounding normal tissues and is a threat to life. Sometime they grow back even after removal. The symptoms of breast cancer is a lump in the breast or underarm, that persist after menstrual cycle. This lump is usually painless. Another symptom is retraction of nipple or discharge in nipple. Puckering of skin on the breast is also considered the symptom of breast cancer. Medical scientists consider that mammography screening as the most dependable method of early detection of breast cancer. Women may survive for long time, with an early detection of breast cancer. Since breast cancer is complex disease it is likely to be caused by a combination risk factors. Age, genetic factor and heredity are non-preventable risk factors associated with breast cancer. Preventable risk factors are overweight, hormone replacement therapy, alcohol and smoking. Other risk factors are radiation exposure, late pregnancy at older age, high bone density and post-menopausal hormone therapy.

Data Mining is the process of extracting hidden and useful information from large databases. There are many data mining techniques are available such as classification, clustering, association rule mining and so on. Classification is the process of classifying similar objects. There are many classification algorithms.

This experiment focuses on how data mining techniques are applied to predict breast cancer. It was carried out using the Bagging algorithm, the K-Means algorithm and the Rain Forest algorithm. Then the results are compared for the same dataset.

2. Classification Algorithm

The following describes some of the algorithms used in this experiment.

2.1 Bagging Algorithm

The Bagging Algorithm is a machine learning ensemble algorithm [1], proposed by Breiman in 1994, that enhances the accuracy of statistical classification methods. The Bagging Algorithm is derived from the "Bootstrap Aggregating" and has been applied frequently in fields such as biostatistics and remote sensing. It is used for information acquisition in artificial intelligence techniques. This algorithm helps to prevent over-learning because it reduces variance. Basically, it creates more than one training sample by generating different combinations of training data. The Bagging algorithm works with the small dimensions of the training datasets. Fig. 1 shows the working principle of this algorithm.

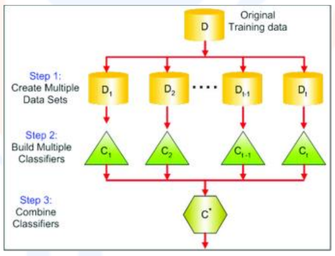

D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

Fig. 1 The working principle of Bagging Algorithm

The original training set is divided into *N* sub-clusters. Each of these sub-clusters is used as a training set and generates a classifier. These classifiers are combined by a unifying classifier. For this reason, this method has been termed, Bagging. To simplify, if it is assumed that the training data consists of *N* items, then a training data with *N* samples is replaced by a random selection of data from the training set. In this case, some of the samples are not included in some of the training data, while other samples are included in more than one training data. Each decision tree is trained with training data containing different samples produced in this way, and the result is determined by a majority vote.

2.2 IBk Algorithm

In Weka, the *K*-Nearest Neighbors (KNN) Algorithm is called the IBk Algorithm. IBk means instance-based learning with parameter *k* and is considered a classification algorithm. It can solve the classification problem [3] by using lazy classification techniques. In this algorithm, the distance between observations is determined [3] based on a local average calculation. This KNN algorithm is one of the supervised learning algorithms that uses learning techniques based on similarity. This non-parametric algorithm is a widely used instance-based learning algorithm and generates a prediction for a test case right on time [2]. In the *K*-Nearest Neighbor Algorithm, a new class of samples is determined by calculating the distance from the samples in the current sample (*k*-nearest neighbors) to a given *k* value of the sample. It uses distance measures, including Euclidean, Manhattan, Chebyshev, and Levenshtein (Edit Distance) distance functions to find *k* "close" samples. Because of its efficiency and productivity, Euclidian distance was used in this study with the IBk Algorithm in Weka, as shown in Eq. (1).

$$distance\ (X_1, X_2\ ) = \sqrt{\textstyle\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}) \qquad (1)$$

In classification with the *K*-Nearest Neighbor Algorithm, *k*, the KNN parameter, specifies the number of closest neighbors to be used when classifying a test sample and the result is determined by a majority vote, as shown in Fig. 2. Weka uses the "cross-validation" option to automatically select the best value.

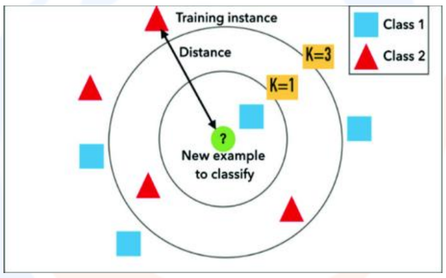


Fig. 2. Shows the working principle of IBk Algorithm

If the cross-validation option is not used, the selected *k* parameter might be too small or too large. Because of this, the *k* parameter is important to guard the KNN algorithm against a noisy dataset.

2.3 Random Forest (RF) Algorithm

The Random Forest Algorithm can be defined as a collection of tree type classifiers. Random forests were first presented in a proper study by Leo Breiman [4]. In his work, he describes a method for creating a forest of unrelated trees using a Classification and Regression Treelike (CART-like) procedure that is combined with randomized node optimization and bagging. In this algorithm, which was developed by Breiman [4], the goal is to combine the decisions of many multivariate trees, each of which is trained in different training clusters, rather than producing a single decision tree.

During the classification process, the Random Forest (RF) Algorithm aims to find the classification value using more than one decision tree. Instead of dividing each node into branches by using the best branch among all the variables, the Random Forest Algorithm divides each node by using the best variable among the randomly selected variables in each node. Each dataset is generated with displacement from the original dataset. Then, trees are developed using a random selection feature, but are not pruned [4]. This strategy makes the RF Algorithm unique and highly accurate. The RF Algorithm is also very fast, resistant to extreme adaptability, and can work with as many trees as desired.

The classification accuracy of the RF Algorithm depends on user-defined parameters such as *N* (number of trees) and *m* (number of variables / parameters used in each node). Therefore, the choice of the most appropriate parameter for the data increases the accuracy of classification. According to Breiman [4], the number of *m* variables taken as equal to the square root of the total number of *M* (number of overall variable) variables, generally gives a result that is closest to the optimal one. The RF Algorithm uses the Classification and Regression Tree (CART) algorithm to generate trees. In each node, branches are created according to the criteria of the CART algorithm (for example, the GINI index). The GINI Coefficient method is used to determine branching criteria [5]. Fig. 3 shows the working principle of the Random Forest Algorithm [4, 6].

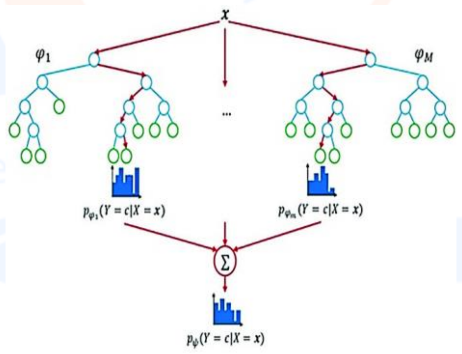


Fig. 3. Shows the working principle of Random Forest Algorithm

3. Method

To analyze the model prediction is used the WEKA. The steps are shown as below.

3.1 Preprocessing

The breast cancer dataset used in this work is collected from the website (http://tunedit.org/rsist of epo/UCI/breast-cancer.arff). This dataset consists of 286 instances and 10 Attributes, consist of 9 + the class attribute. The Attributes are shown below.

Table 1. Data set of Breast Cancer

Set-No	Attribut	Data Type	Data Missing
1	Age	Nominal	0
2	Menopause	Nominal	0
3	Tumor-Size	Nominal	0

4	Inv-Nodes	Nominal	0
5	Node-Caps	Nominal	8
6	Deg-Malig	Nominal	0
7	Breast	Nominal	0
8	Breast-Quad	Nominal	1
9	Irradiat	Nominal	0
10	Class	Nominal	0

Preprocessing is the steps that are taken before the data is processed. Pre-processing has an important role because no matter how good the technique used if the input data contains an error then the conclusions produced will also be wrong. The pre-processing stages include data cleaning, integration, transformation to data reduction. The preprocessing stage will also help data analysts understand the data. The more data understood, the better the model produced.

Back to WEKA, after selecting the Explorer menu the following display will appear, there are tabs for preprocess, classify, cluster, associate, select attributes and visualize. Select the **preprocess tab**.(fig. 4).

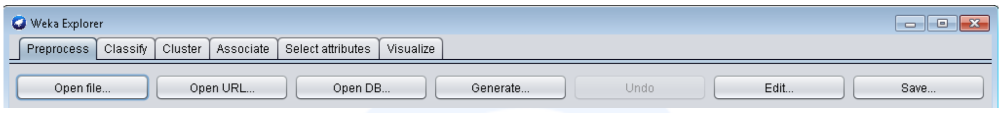
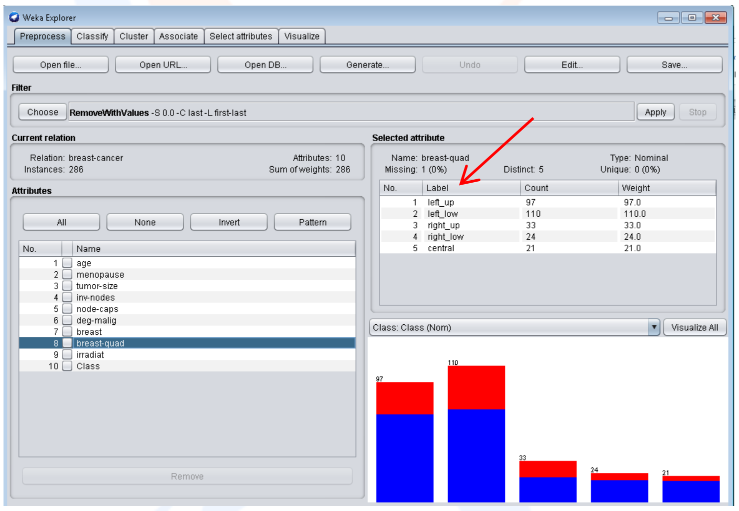


Fig. 4 Menu display

The preprocess tab is used to see at a glance about dataset information such as number of instances, attribute types, attribute content and histograms.

In the original dataset there is **1 (one) missing data** on the breast-quad attribute and **8 missing data on node-caps attribut**. Blank data can be caused by many things such as operator error, sensor error, or indeed the data does not exist.



RemoveWithValues -S 0.0 -C last -L first-last

Relation: breast-cancer
Instances: 286
Attributes: 10
Sum of weights: 286

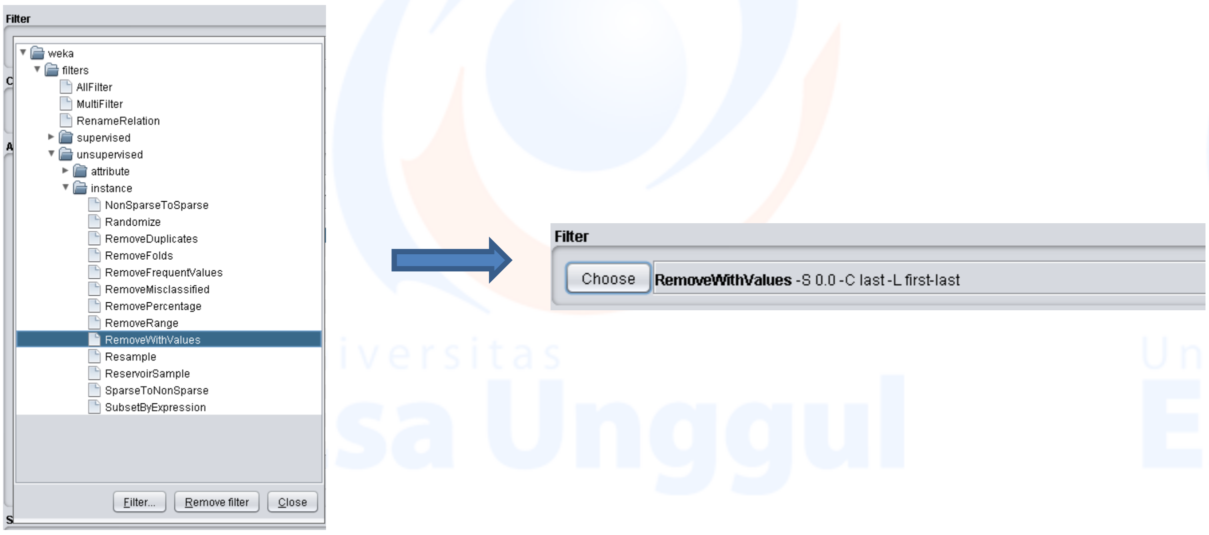
Name: breast-quad
Missing: 1 (0%)
Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	left_up	97	97.0
2	left_low	110	110.0
3	right_up	33	33.0
4	right_low	24	24.0
5	central	21	21.0

No.	Name
1	age
2	menopause
3	tumor-size
4	inv-nodes
5	node-caps
6	deg-malig
7	breast
8	breast-quad
9	irradiat
10	Class

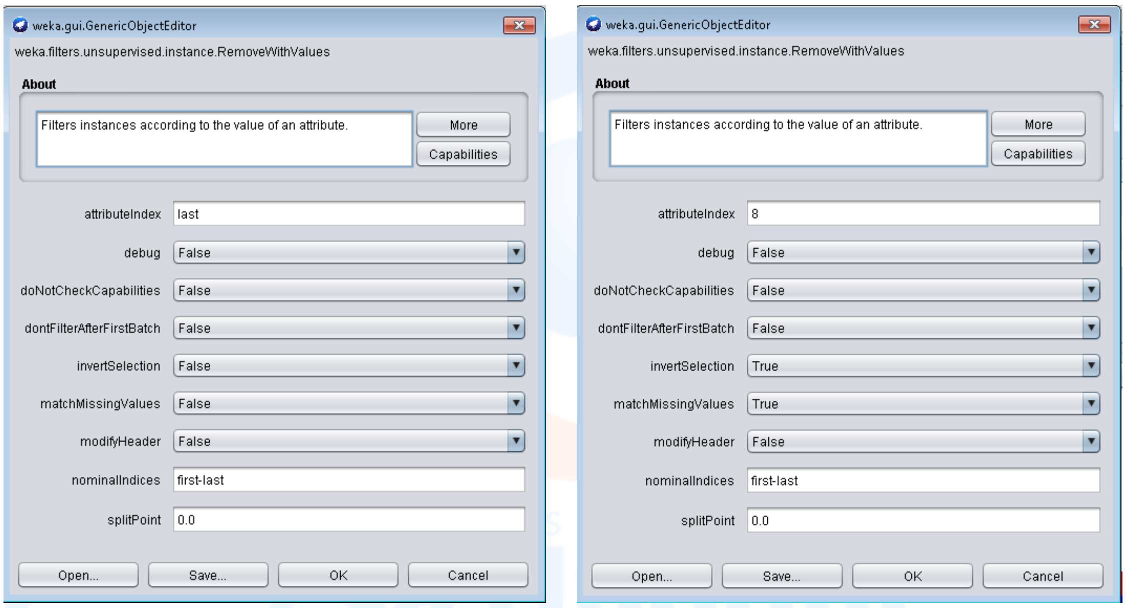
Fig. 5 Breast-Squad Attribute Display

This needs to be addressed because some machine learning techniques do not want to process blank data or can cause the resulting model to be inaccurate. One way to handle missing data is to delete rows, this can be done if there are not too many of them. In the filter section, click **Choose** dan filter **Filters** → **Unsupervised** → **Instance** → **RemoveWithValues**.



```
RemoveWithValues -S 0.0 -C last -L first-last
```

Fig. 6 Filter display



weka.filters.unsupervised.instance.RemoveWithValues

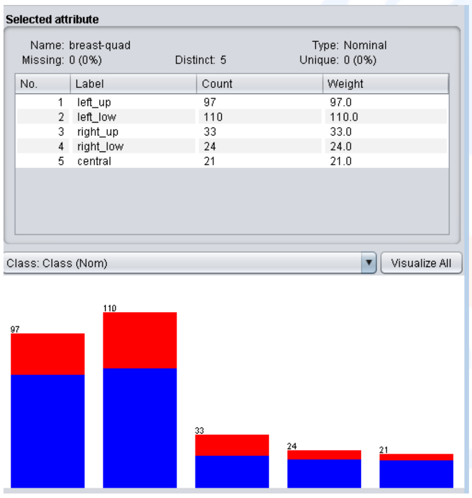
Filters instances according to the value of an attribute.

attributeIndex	last	8
debug	False	False
doNotCheckCapabilities	False	False
dontFilterAfterFirstBatch	False	False
invertSelection	False	True
matchMissingValues	False	True
modifyHeader	False	False
nominalIndices	first-last	first-last
splitPoint	0.0	0.0

Fig. 7 Filter processing display

Click the name of the filter to configure the filter. What needs to be changed is **attributeindex** to last (breast-quad attribute), **matchMissingValues** = True and **invertSelection** = True, then Click OK.

Click Apply, then the number of instances decreases by one and the number of missing values for the date attribute will be 0.

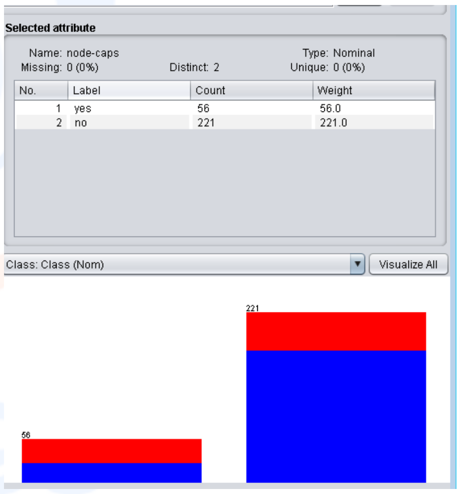

Selected attribute

Name: breast-quad　　Type: Nominal
Missing: 0 (0%)　　Distinct: 5　　Unique: 0 (0%)

No.	Label	Count	Weight
1	left_up	97	97.0
2	left_low	110	110.0
3	right_up	33	33.0
4	right_low	24	24.0
5	central	21	21.0

Class: Class (Nom)

Fig. 8 (a) Results of filter processing for breast-quad attributes


Selected attribute

Name: node-caps　　Type: Nominal
Missing: 0 (0%)　　Distinct: 2　　Unique: 0 (0%)

No.	Label	Count	Weight
1	yes	56	56.0
2	no	221	221.0

Class: Class (Nom)

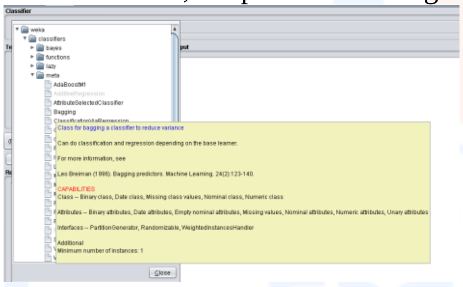
Fig. 8 (b) Results of filter processing for node-caps attributes

3.1 Classification Algorithm

3.1.1 Bagging Algorithm

After dataset is clear from data missing, it can be carried out the classification processing. The first step, is used Bagging algorithm, as follow.

- Select the **Classify** tab. In the Classifier section, there is the **Choose** button and the name of the machine learning algorithm that has parameters. Select the **Choose** button, a dropdown menu will appear. Click **meta**, and choose **Bagging** (a). In the text, it says " **Bagging** -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0 "(b). If the text is clicked, the parameter dialog will come out whose value can be adjusted (c).


Class for bagging a classifier to reduce variance

Can do classification and regression depending on the base learner.

For more information, see

Leo Breiman (1996). Bagging predictors. Machine Learning. 24(2):123-140.

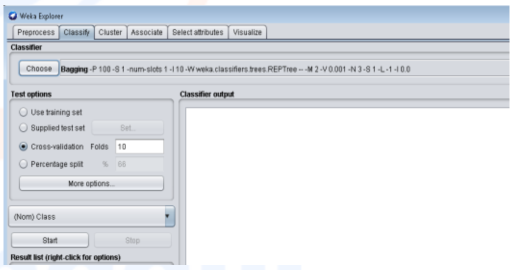
CAPABILITIES
Class -- Binary class, Date class, Missing class values, Nominal class, Numeric class

Attributes -- Binary attributes, Date attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- PartitionGenerator, Randomizable, WeightedInstancesHandler

Additional
Minimum number of instances: 1

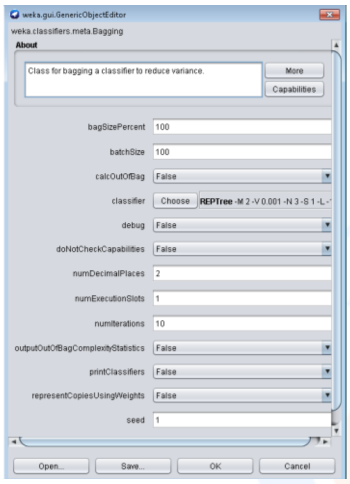
(a)


Classifier: Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Test options: Cross-validation　Folds 10

(Nom) Class

(b)


weka.gui.GenericObjectEditor

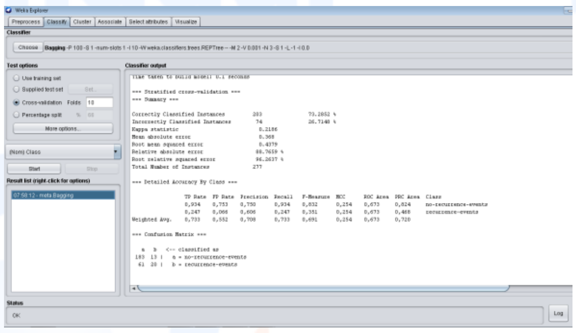
weka.classifiers.meta.Bagging

About

Class for bagging a classifier to reduce variance.

bagSizePercent	100
batchSize	100
calcOutOfBag	False
classifier	REPTree -M 2 -V 0.001 -N 3 -S 1 -L -
debug	False
doNotCheckCapabilities	False
numDecimalPlaces	2
numExecutionSlots	1
numIterations	10
outputOutOfBagComplexityStatistics	False
printClassifiers	False
representCopiesUsingWeights	False
seed	1

(c)


Classifier: Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Test options: Cross-validation　Folds 10

(Nom) Class

Result list: 07:58:12 - meta.Bagging

Classifier output

```
Time taken to build model: 0.1 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         203               73.2852 %
Incorrectly Classified Instances        74               26.7148 %
Kappa statistic                          0.2186
Mean absolute error                      0.368
Root mean squared error                  0.4379
Relative absolute error                 88.7659 %
Root relative squared error             96.2637 %
Total Number of Instances              277

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,934    0,753    0,750      0,934    0,832      0,254    0,673     0,824     no-recurrence-events
                 0,247    0,066    0,606      0,247    0,351      0,254    0,673     0,468     recurrence-events
Weighted Avg.    0,733    0,552    0,708      0,733    0,691      0,254    0,673     0,720

=== Confusion Matrix ===

   a   b   <-- classified as
 183  13 |   a = no-recurrence-events
  61  20 |   b = recurrence-events
```

Status: OK

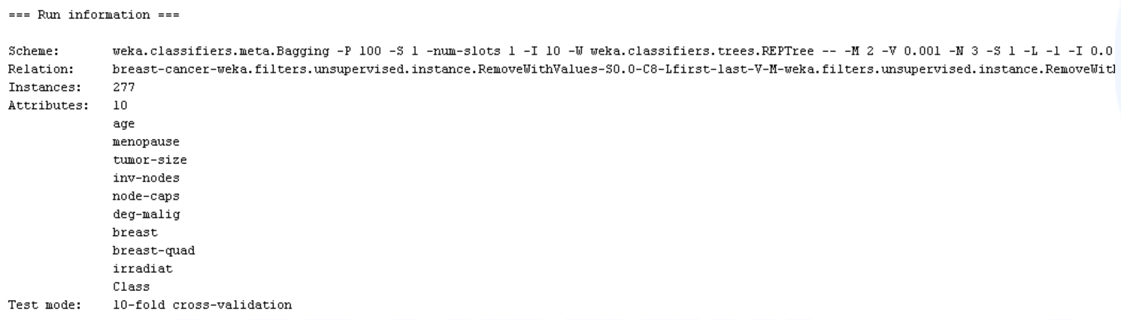
(d)

Fig. 9 The Bagging Algorithm Processing Display

- Before starting the learning process (or learning) by pressing the Start button, make sure the class attributes are appropriate. In our dataset, the class attribute is **no-recurrence events**. In addition, choose the appropriate test options. Here are the details of the test options :
  - **Use training set**: the model evaluates how well it predicts the class of all data instances. This is rather dangerous because it can cause accuracy to be artificially high.
  - **Supplied test set**: the model evaluates how well it predicts the class of test data instances that are loaded from separate files. This test data is in the form of ARFF and of course it must be the same attribute as the training data.
  - **Cross validation**: the model is evaluated by cross-validation, with the number of folds according to user input (default: 10). For example if the training data have 1000 rows and 10 folds are set (ten cross validation), then for the first batch, rows 1-100 are used for testing and rows 101-1000 are used for training. For the second batch 101-200 for testing and 1-100 plus 201-1000 for training. And so on in turns until batch 10. Accuracy of each batch is calculated and the accuracy of the model is the average of all batches. Cross validation is more commonly used.
  - **Percentage split**: the model evaluates how well it predicts the class of instances which are certain% of all data (hold-out). Large % according to user input (default: 66). Training data is broken up into two according to percentage, the first part is testing data, the second part is training data.

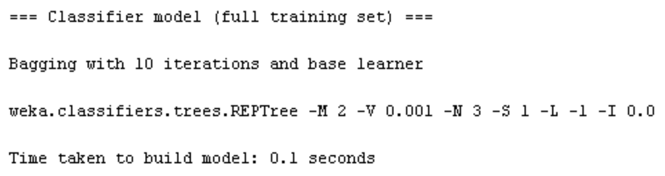
  Select **cross validation** with **Folds 10**. After all settings are correct, click **Start**, and see the results in the **Classifier output** as follows.

  - **Run information**: the list of learning information includes the scheme (learning algorithm and its parameters), the name of the relationship (defined in the arff file), number of instances, number and list of attributes, and test options.



```
=== Run information ===

Scheme:       weka.classifiers.meta.Bagging -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Relation:     breast-cancer-weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C8-Lfirst-last-V-M-weka.filters.unsupervised.instance.RemoveWit
Instances:    277
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
Test mode:    10-fold cross-validation
```

  - **Classifier model (full training set)**: learning model that results from all training data in text representation.



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

Bagging with 10 iterations and base learner

weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0

Time taken to build model: 0.1 seconds
```

  - **Summary**: list of performance statistics that show how accurately the classifier model can predict the actual class of each instance according to test options. For the 10-fold cross validation test options selected, there are 277 instances tested in 10 iterations. In the results shown, there were 203 instances (73,29%) that were correctly predicted by the class, and 74 other instances (26,72%) were incorrectly predicted.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         203               73.2852 %
Incorrectly Classified Instances        74               26.7148 %
Kappa statistic                          0.2186
Mean absolute error                      0.368
Root mean squared error                  0.4379
Relative absolute error                 88.7659 %
Root relative squared error             96.2637 %
Total Number of Instances              277
```

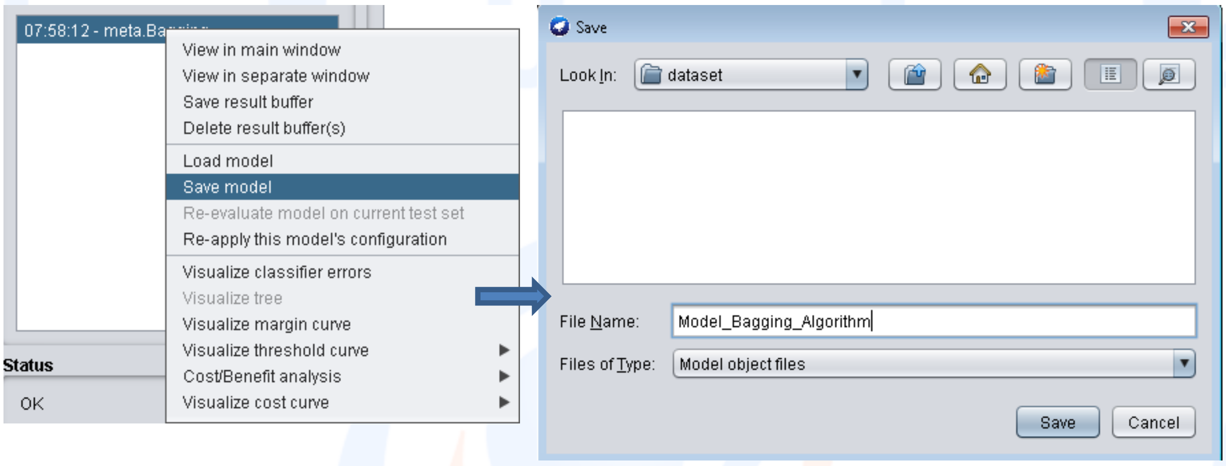
- **Detailed accuracy by class**: a more detailed measure of performance at the class level.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,934    0,753    0,750      0,934    0,832      0,254    0,673     0,824     no-recurrence-events
                 0,247    0,066    0,606      0,247    0,351      0,254    0,673     0,468     recurrence-events
Weighted Avg.    0,733    0,552    0,708      0,733    0,691      0,254    0,673     0,720
```

- **Confusion matrix**: a matrix that shows how many instances are predicted to each class (per column) and the number of instances that correspond to the actual label (per row). In this example there are 244 instances that are predicted or classified as Yes. Of these 244 instances, 183 instances are labeled Yes (also called True Positive), and 61 instances are labeled No (also called False Positive). The Correctly Classified Instances value in Summary 213 (73,29%) is a (183 + 20) value from this confusion matrix. Conversely, the Uncorrectly Classified Instances value in Summary 74 (26,72%) is a (61 + 13) value from this confusion matrix

```
=== Confusion Matrix ===

   a   b   <-- classified as
 183  13 |   a = no-recurrence-events
  61  20 |   b = recurrence-events
```

To save the model, click the result list to be saved, then right-click and "Save Model".



Look In: dataset

File Name: Model_Bagging_Algorithm

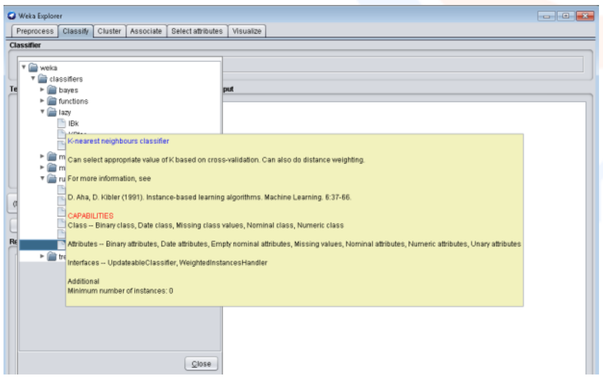
Files of Type: Model object files

Fig. 10 Save Model for Bagging Algorithm display

3.1.2 IBk Algorithm

Furthermore, the second step, is used IBk algorithm, as follow.

- Select the **Classify** tab. In the Classifier section, there is the **Choose** button and the name of the machine learning algorithm that has parameters. Select the **Choose** button, a dropdown menu will appear. Click **lazy**, and choose **Bagging** (a). In the text, it says " **IBk** -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" "(b). If the text is clicked, the parameter dialog will come out whose value can be adjusted (c).



K-nearest neighbours classifier

Can select appropriate value of K based on cross-validation. Can also do distance weighting.

For more information, see

D. Aha, D. Kibler (1991). Instance-based learning algorithms. Machine Learning. 6:37-66.

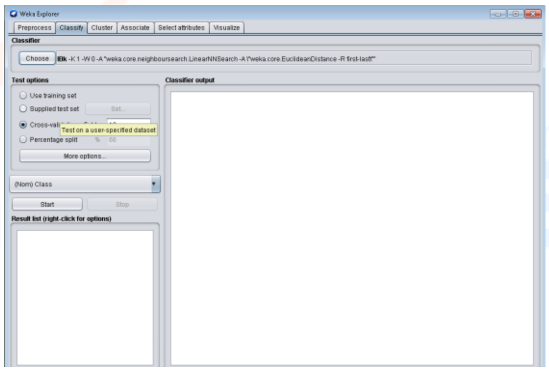
CAPABILITIES
Class -- Binary class, Date class, Missing class values, Nominal class, Numeric class

Attributes -- Binary attributes, Date attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- UpdateableClassifier, WeightedInstancesHandler

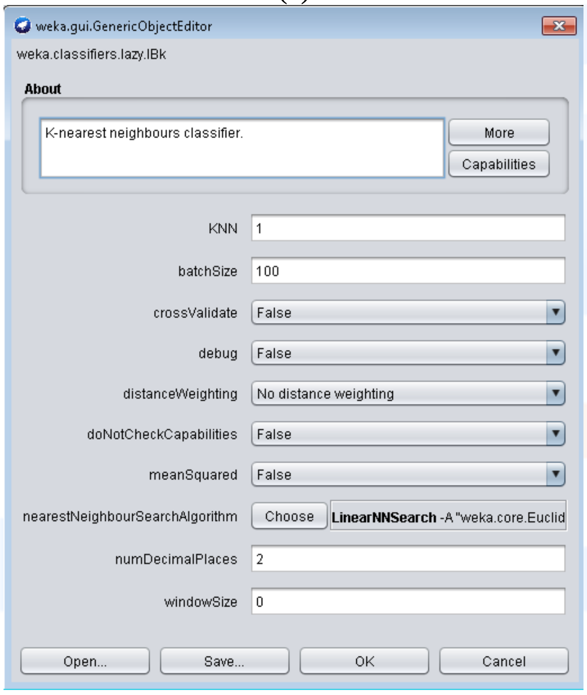
Additional
Minimum number of instances: 0

(a)



IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

(b)

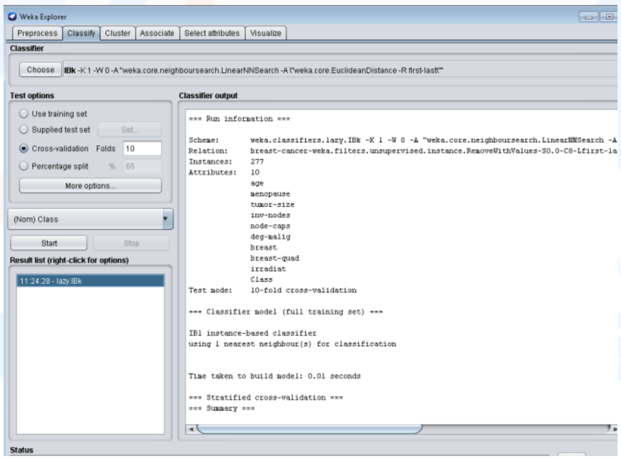


weka.classifiers.lazy.IBk

K-nearest neighbours classifier.

KNN	1
batchSize	100
crossValidate	False
debug	False
distanceWeighting	No distance weighting
doNotCheckCapabilities	False
meanSquared	False
nearestNeighbourSearchAlgorithm	Choose **LinearNNSearch** -A "weka.core.Euclid
numDecimalPlaces	2
windowSize	0

(c)



IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Folds 10, % 66

11:24:28 - lazy.IBk

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
Relation:     breast-cancer-weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C8-Lfirst-la
Instances:    277
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
Test mode:    10-fold cross-validation

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

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===
```

(d)

Fig. 11 The IBk Algorithm Processing Display

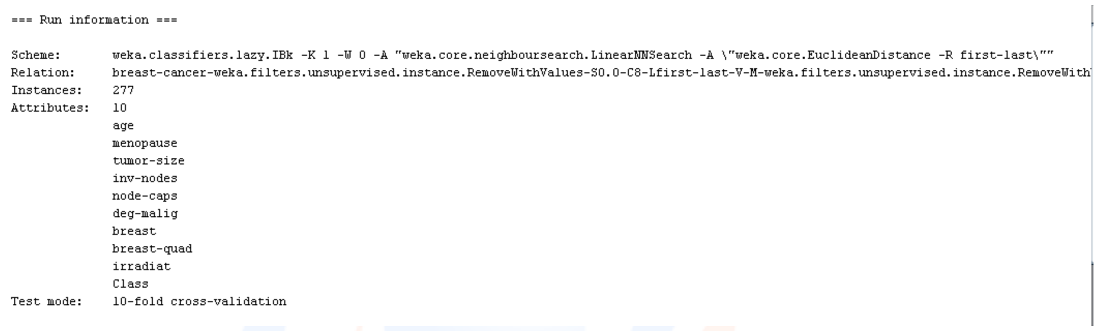
- Before starting the learning process (or learning) by pressing the Start button, make sure the class attributes are appropriate. In our dataset, the class attribute is **no-recurrence events**. In addition, choose the appropriate test options. Here are the details of the test options :
  - **Use training set**: the model evaluates how well it predicts the class of all data instances. This is rather dangerous because it can cause accuracy to be artificially high.
  - **Supplied test set**: the model evaluates how well it predicts the class of test data instances that are loaded from separate files. This test data is in the form of ARFF and of course it must be the same attribute as the training data.
  - **Cross validation**: the model is evaluated by cross-validation, with the number of folds according to user input (default: 10). For example if the training data have 1000 rows

and 10 folds are set (ten cross validation), then for the first batch, rows 1-100 are used for testing and rows 101-1000 are used for training. For the second batch 101-200 for testing and 1-100 plus 201-1000 for training. And so on in turns until batch 10. Accuracy of each batch is calculated and the accuracy of the model is the average of all batches. Cross validation is more commonly used.

- **Percentage split**: the model evaluates how well it predicts the class of instances which are certain% of all data (hold-out). Large % according to user input (default: 66). Training data is broken up into two according to percentage, the first part is testing data, the second part is training data. In this case, its used 100% dataset as training data.

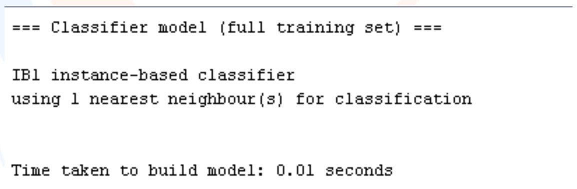
Select **cross validation** with **Folds 10**. After all settings are correct, click **Start**, and see the results in the **Classifier output (d)** as follows.

- **Run information**: the list of learning information includes the scheme (learning algorithm and its parameters), the name of the relationship (defined in the arff file), number of instances, number and list of attributes, and test options.



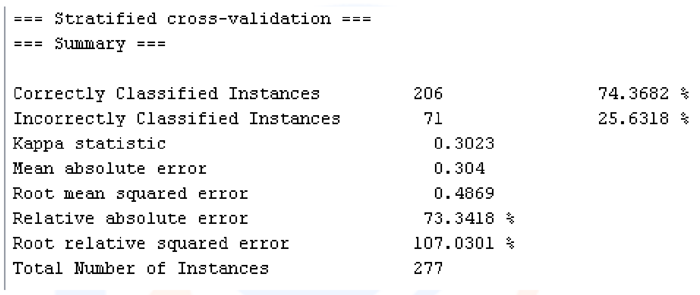
```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     breast-cancer-weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C8-Lfirst-last-V-M-weka.filters.unsupervised.instance.RemoveWith
Instances:    277
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
Test mode:    10-fold cross-validation
```

- **Classifier model (full training set)**: learning model that results from all training data in text representation.



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

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0.01 seconds
```

- **Summary**: list of performance statistics that show how accurately the classifier model can predict the actual class of each instance according to test options. For the 10-fold cross validation test options selected, there are 277 instances tested in 10 iterations. In the results shown, there were 206 instances (74,37%) that were correctly predicted by the class, and 71 other instances (25,63%) were incorrectly predicted.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         206               74.3682 %
Incorrectly Classified Instances        71               25.6318 %
Kappa statistic                          0.3023
Mean absolute error                      0.304
Root mean squared error                  0.4869
Relative absolute error                 73.3418 %
Root relative squared error            107.0301 %
Total Number of Instances              277
```

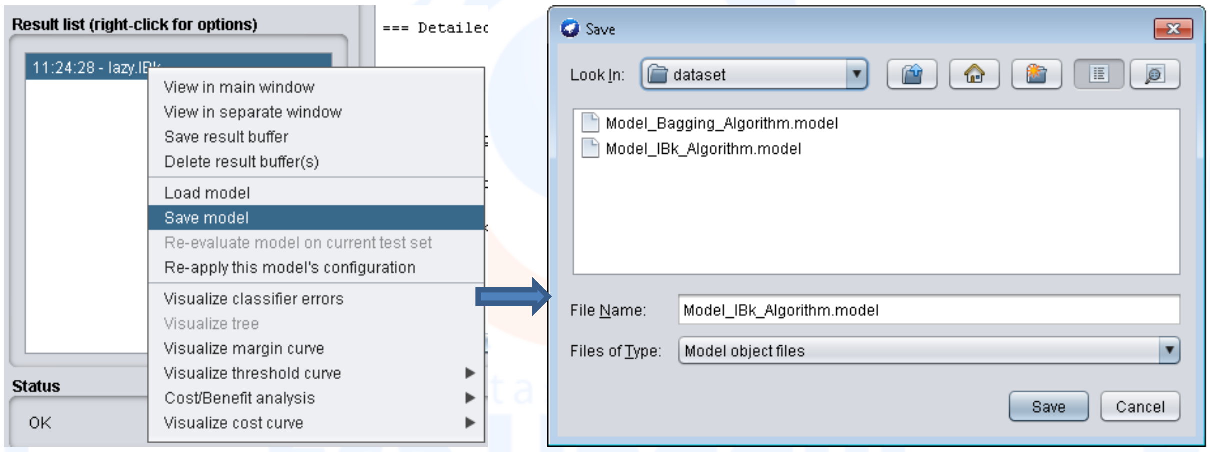
- **Detailed accuracy by class**: a more detailed measure of performance at the class level.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,898    0,630    0,775      0,898    0,832      0,317    0,652     0,788     no-recurrence-events
                 0,370    0,102    0,600      0,370    0,458      0,317    0,652     0,498     recurrence-events
Weighted Avg.    0,744    0,475    0,724      0,744    0,723      0,317    0,652     0,703
```

- **Confusion matrix**: a matrix that shows how many instances are predicted to each class (per column) and the number of instances that correspond to the actual label (per row). In this example there are 206 instances that are predicted or classified as Yes. Of these 206 instances, 176 instances are labeled Yes (also called **True Positive**), and 51 instances are labeled No (also called **False Positive**). The Correctly Classified Instances value in Summary 206 (73,29%) is a (176 + 30) value from this confusion matrix. Conversely, the Uncorrectly Classified Instances value in Summary 71 (25,63%) is a (20 + 51) value from this confusion matrix

```
=== Confusion Matrix ===

   a   b   <-- classified as
 176  20 |   a = no-recurrence-events
  51  30 |   b = recurrence-events
```

To save the model, click the result list to be saved, then right-click and "Save Model".



File Name: Model_IBk_Algorithm.model

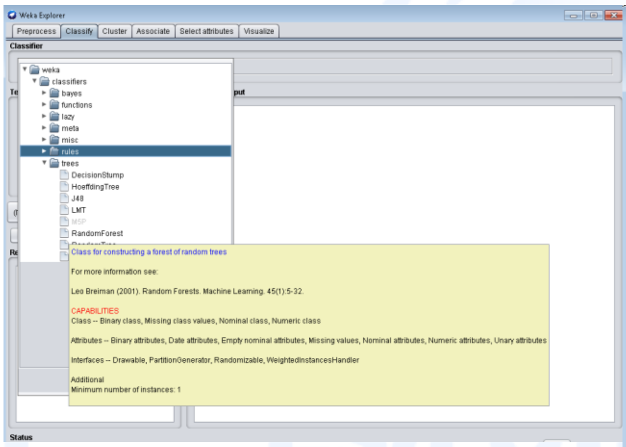
Files of Type: Model object files

Fig. 12 Save Model for IBk Algorithm display

3.1.3 RandomForest Algorithm

The last, third step, is used Random Forest (RF) algorithm, as follow.

- Select the **Classify** tab. In the Classifier section, there is the **Choose** button and the name of the machine learning algorithm that has parameters. Select the **Choose** button, a dropdown menu will appear. Click **trees**, and choose **RandomForest** (a). In the text, it says " **RandomForest** -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1 "(b). If the text is clicked, the parameter dialog will come out whose value can be adjusted (c).



Class for constructing a forest of random trees

For more information see:

Leo Breiman (2001). Random Forests. Machine Learning. 45(1):5-32.

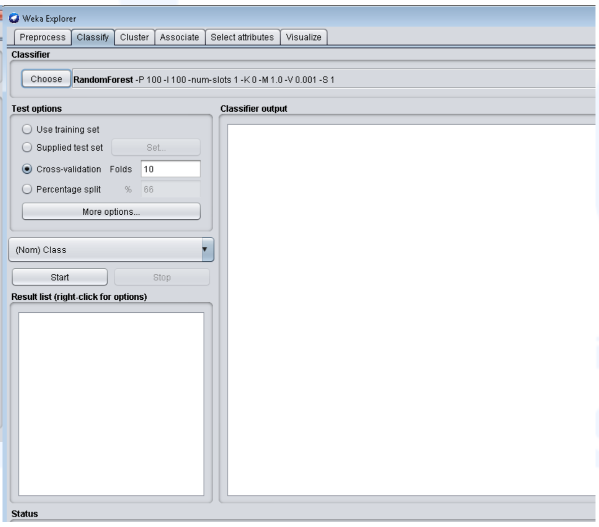
CAPABILITIES
Class -- Binary class, Missing class values, Nominal class, Numeric class

Attributes -- Binary attributes, Date attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- Drawable, PartitionGenerator, Randomizable, WeightedInstancesHandler

Additional
Minimum number of instances: 1

(a)

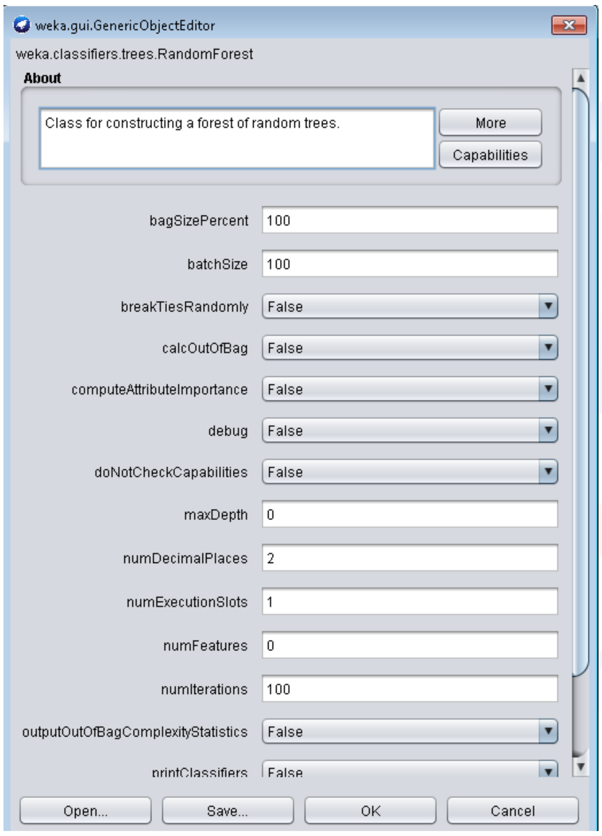


RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Cross-validation Folds 10

% 66

(b)

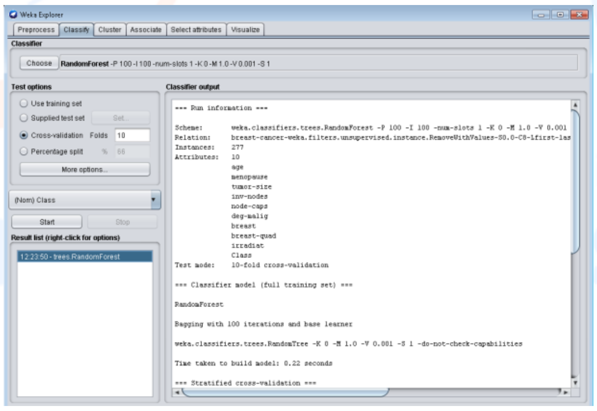


weka.classifiers.trees.RandomForest

Class for constructing a forest of random trees.

bagSizePercent	100
batchSize	100
breakTiesRandomly	False
calcOutOfBag	False
computeAttributeImportance	False
debug	False
doNotCheckCapabilities	False
maxDepth	0
numDecimalPlaces	2
numExecutionSlots	1
numFeatures	0
numIterations	100
outputOutOfBagComplexityStatistics	False
printClassifiers	False

(c)



RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Cross-validation Folds 10

12:23:50 - trees.RandomForest

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001
Relation:     breast-cancer-weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C8-Lfirst-las
Instances:    277
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
Test mode:    10-fold cross-validation

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

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.22 seconds

=== Stratified cross-validation ===
```

(d)

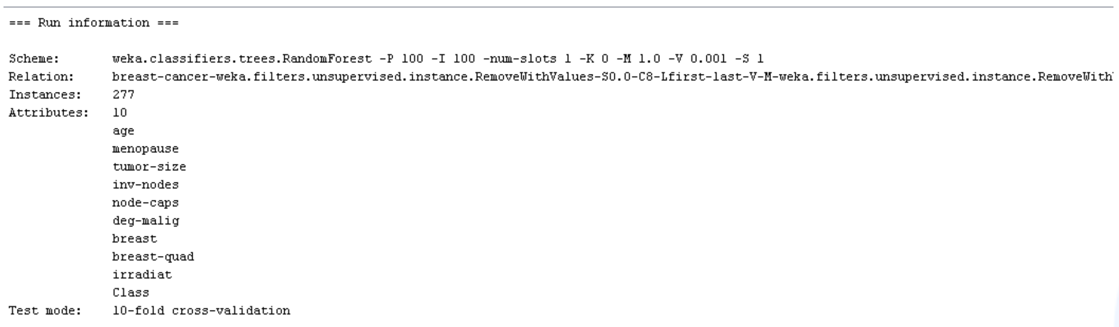
Fig. 13 The RandomForest (RF) Algorithm Processing Display

- Before starting the learning process (or learning) by pressing the Start button, make sure the class attributes are appropriate. In our dataset, the class attribute is **no-recurrence events**. In addition, choose the appropriate test options. Here are the details of the test options :
  - **Use training set**: the model evaluates how well it predicts the class of all data instances. This is rather dangerous because it can cause accuracy to be artificially high.
  - **Supplied test set**: the model evaluates how well it predicts the class of test data instances that are loaded from separate files. This test data is in the form of ARFF and of course it must be the same attribute as the training data.

- **Cross validation**: the model is evaluated by cross-validation, with the number of folds according to user input (default: 10). For example if the training data have 1000 rows and 10 folds are set (ten cross validation), then for the first batch, rows 1-100 are used for testing and rows 101-1000 are used for training. For the second batch 101-200 for testing and 1-100 plus 201-1000 for training. And so on in turns until batch 10. Accuracy of each batch is calculated and the accuracy of the model is the average of all batches. Cross validation is more commonly used.
- **Percentage split**: the model evaluates how well it predicts the class of instances which are certain% of all data (hold-out). Large % according to user input (default: 66). Training data is broken up into two according to percentage, the first part is testing data, the second part is training data. In this case, its used 100% dataset as training data.

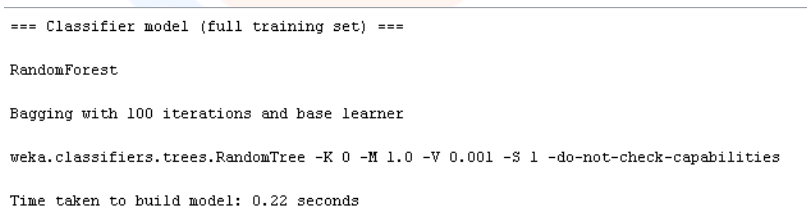
Select **cross validation** with **Folds 10**. After all settings are correct, click **Start**, and see the results in the **Classifier output (d)** as follows.

- **Run information**: the list of learning information includes the scheme (learning algorithm and its parameters), the name of the relationship (defined in the arff file), number of instances, number and list of attributes, and test options.



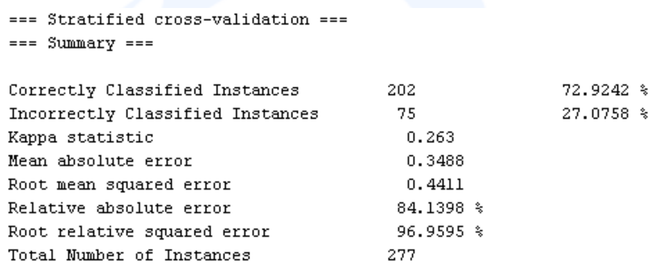
```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation:     breast-cancer-weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C8-Lfirst-last-V-M-weka.filters.unsupervised.instance.RemoveWith
Instances:    277
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
Test mode:    10-fold cross-validation
```

- **Classifier model (full training set)**: learning model that results from all training data in text representation.



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

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.22 seconds
```

- **Summary**: list of performance statistics that show how accurately the classifier model can predict the actual class of each instance according to test options. For the 10-fold cross validation test options selected, there are 277 instances tested in 10 iterations. In the results shown, there were 202 instances (72,92%) that were correctly predicted by the class, and 75 other instances (27,08%) were incorrectly predicted.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         202               72.9242 %
Incorrectly Classified Instances        75               27.0758 %
Kappa statistic                          0.263
Mean absolute error                      0.3488
Root mean squared error                  0.4411
Relative absolute error                 84.1398 %
Root relative squared error             96.9595 %
Total Number of Instances              277
```

- **Detailed accuracy by class**: a more detailed measure of performance at the class level.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,888    0,654    0,767      0,888    0,823      0,276    0,681     0,812     no-recurrence-events
                 0,346    0,112    0,560      0,346    0,427      0,276    0,681     0,470     recurrence-events
Weighted Avg.    0,729    0,496    0,706      0,729    0,707      0,276    0,681     0,712
```

- **Confusion matrix**: a matrix that shows how many instances are predicted to each class (per column) and the number of instances that correspond to the actual label (per row). In this example there are 202 instances that are predicted or classified as Yes. Of these 202 instances, 174 instances are labeled Yes (also called **True Positive**), and 28 instances are labeled No (also called **False Positive**). The Correctly Classified Instances value in Summary 202 (72,92%) is a (174 + 28) value from this confusion matrix. Converselly, the Uncorrectly Classified Instances value in Summary 75 (27,08%) is a (22 + 53) value from this confusion matrix

```
=== Confusion Matrix ===

   a   b   <-- classified as
 174  22 |   a = no-recurrence-events
  53  28 |   b = recurrence-events
```

To save the model, click the result list to be saved, then right-click and "Save Model".

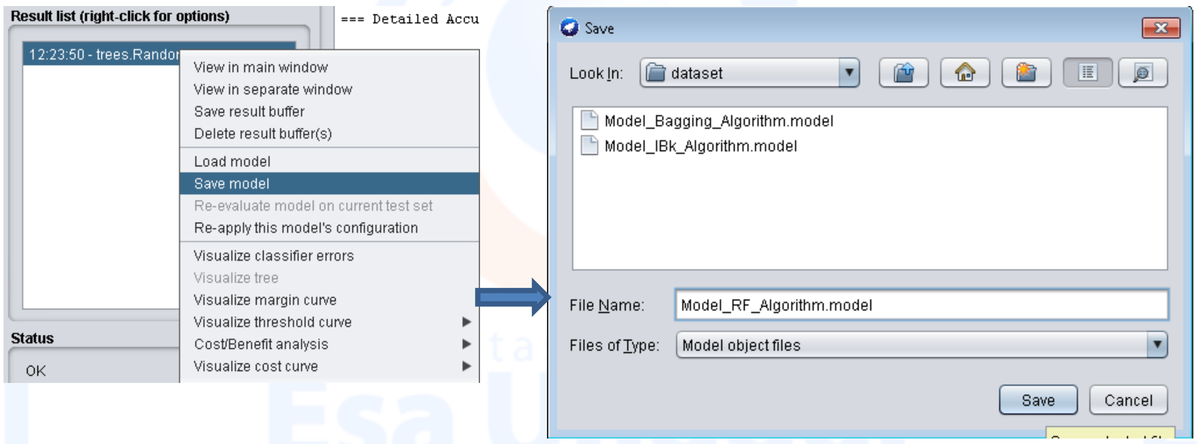


Fig. 14 Save Model for RF Algorithm display

4. Performance Measure of Classifiers

In our experiment, data is supplied to classifier of Bagging, Algorithm, IBk Algorithm and Random Forest Algorithm to classify the data. The classifiers performance is evaluated through Confusion Matrix. It is used for measuring the performance of classifiers. In the confusion matrix, correctly classified instances are calculated by sum of diagonal elements TP (True Positive) and TN (True Negative) and others as well as FP (false positive) and FN (False Negative) are called incorrectly classified instances. **Accuracy** is defined as the ratio of correctly classified instances to total number of instances in the breast cancer dataset.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

There are totally 286 records in the Breast cancer dataset , After pre-processing, the number of instances will be 277 which is used in the classification process. Among these 277 instances, 196 instances which are belongs to non-recurrent event and 81 instances are belongs to recurrent event. Confusion matrix of each algorithms, is described below.

Table 1. Confusion matrix of Bagging Algorithm

Class Target	No-recurrence-events	Recurrence-events
No-recurrence-events	183	13
Recurrence-events	61	20

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{183+20}{183+20+61+13} = 0{,}7329 = 73{,}29\%$$

Table 2. Confusion matrix of IBk Algorithm

Class Target	No-recurrence-events	Recurrence-events
No-recurrence-events	176	20
Recurrence-events	51	30

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{176+30}{176+30+51+20} = 0{,}7437 = 74{,}37\%$$

Table 3. Confusion matrix of RandomForest Algorithm

Class Target	No-recurrence-events	Recurrence-events
No-recurrence-events	174	22
Recurrence-events	53	28

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{174+28}{183+20+61+13} = 0{,}7292 = 72{,}92\%$$

From the accuracy calculation above, is seen that the IBk algorithm has the better accuracy than the other, namely, 74,37 %. Meanwhile the Bagging algorithm has an accuracy, 73,29 % and the Randomforest algorithm has 72,92%.

5. Conclusion

Diagnosis of disease is a very challenging task in the field of health care. Many data mining techniques are used in decision making process. In our experiment, we have applied Bagging, IBk (K-Means) and RandomForest data mining classification techniques which are used to classify the recurrent and non-recurrent breast cancer disease. The performance of classifiers are evaluated through the confusion matrix in terms of accuracy. The IBk Algorithm gives 74,37 % which is providing better accuracy than other Algorithm. As a future work the same technique is used to apply for other disease datasets such as Diabetes, Heart disease, Lung cancer etc.

References

[1] Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123 https://doi.org/10.1007/bf00058655

[2} Aydin, E. A. & Keles, M. K. (2017). Breast cancer detection using K-nearest neighbors data mining method obtained from the bow-tie antenna dataset. *International Journal of RF and Microwave Computer-Aided Engineering*, *27*(6). https://doi.org/10.1002/mmce.21098

[3] Altman, N. S. (1992). An Introduction to Kernel and NearestNeighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175. https://doi.org/10.2307/2685209

[4] Breiman, L. (2001). Random forests. *Mach Learn*, 45, 5-32. https://doi.org/10.1023/A:1010933404324

[5] Mather, P. M. (2004). *Computer processing of remotely sensed images: An introduction*. West Sussex: Joh Wiley &Sons.

[6] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Mach Learn*, 63, 3-42. https://doi.org/10.1007/s10994-006-6226-1