Chapter - III

Classification with Zero-R Algorithm using Weka
### Chapter III

**CLASSIFICATION WITH ZERO-R ALGORITHM USING WEKA**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>78</td>
</tr>
<tr>
<td>3.2</td>
<td>Related Work</td>
<td>79</td>
</tr>
<tr>
<td>3.3</td>
<td>Introduction to Classifiers</td>
<td>80</td>
</tr>
<tr>
<td>3.4</td>
<td>Why Zero-R?</td>
<td>89</td>
</tr>
<tr>
<td>3.5</td>
<td>The Proposed Zero-R Algorithm</td>
<td>91</td>
</tr>
<tr>
<td>3.6</td>
<td>Experimental setup</td>
<td>92</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Procedure to Compile Using Java</td>
<td>93</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Discussion of new factor 'Accuracy (1-c)' for each individual attributes</td>
<td>97</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Result Analysis</td>
<td>98</td>
</tr>
<tr>
<td>3.7</td>
<td>Result and Discussion</td>
<td>106</td>
</tr>
<tr>
<td>3.8</td>
<td>Conclusion</td>
<td>107</td>
</tr>
</tbody>
</table>
3.1 Introduction

Data mining has various techniques to extract useful information in large amounts of data. Data mining in general falls into the following categories: classification patterns, association patterns, sequential patterns, and spatial-temporal patterns. The important feature of Data Mining algorithms is running time of an algorithm must be predictable and acceptable in large database.

KDD (Knowledge Discovery from Databases). KDD is the process of finding useful information and patterns in data. Data mining is the use of algorithms to extract the information and patterns derived by the KDD process. KDD is multistep process, the input to this process is the data and the output is the useful information desired by the users [1].

Agents

Agents are used to perform some action or activity on behalf of a user of a computer system. Agent refers to the entities which run in dynamic environment and have higher self-government capacity. Agent software is a type of computer program which simulates human intelligence behaviour. Agent should be able to learn from experience and to act autonomously to the ever changing task.

MAS: A group of agents can collectively and collaboratively form a Multi Agent System (MAS) to perform complex and lengthy tasks [1-7].

Weka is a collection of machine learning algorithms used for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.
There are many techniques for classification such as neural networks, Bayesian, decision tree, instance based learning, genetic algorithm, rough set, and fuzzy logic [3].

3.2 Related Work

By Azuraliza Abu Bakar, Zulaiha Ali Othman in paper “Agent Based Data Classification Approach for Data Mining”, an agent based approach is proposed to improve the rule application process. The proposed agents are embedded within the standard rule application techniques. The result shows the significant improvements in classification time and the number of matched rules with comparable classification accuracy. The process of classification involves applying the rules onto a set of unseen data. The experimental result shows that the proposed agent based technique or ClassifyAgent has significantly reduced the running time and the number of matched rules thus maintaining the accuracy of classification. This criterion is important in agent based data mining to obtain the good knowledge model from the complex and large database. The ClassifyAgent offers an alternative to achieving the data mining purpose of obtaining a good model with faster classification time from large database within reasonable timeframe. The experimental results indicate that in ClassifyAgent the best
performance can be achieved at minimal time and shorter path of the rule base
search [10].

By Cuong Tong, Dharmendra Sharma and Fariba Shadabi in paper titled as
“A Multi-Agents Approach to Knowledge Discovery”, they propose a real time
Data Mining and Multi-Agent System called DMMAS, modelling chronic disease
data. DMMAS approach employs data partitioning and multiple agents with
option to employ heterogeneous or homogenous data mining techniques,
distributing agent based processing for modelling and combining results from all
the agents to improve the efficiency. In this study, we proposed a real time data
mining cooperative multi-agents system called DMMAS. DMMAS is a multiagent
system with multiple miner agents and a combination agent as agent manager. The
proposed technique uses Data mining library Weka 3.5.7 and the experiment was
conducted using the UCI Pima Indian data set. The main goal the system is to
explore how data partitioning and multi agent approach can help to improve the
efficiency and also if possible the accuracy of chronic diseases management and
prediction tasks in real time [11].

Research has shown that over the past few years, data mining tools are
heavily used in healthcare spectrum. Agent based approach has become an
advanced trend in Knowledge discovery. The Classify Agent offers an alternative
to achieving the data mining purpose of obtaining a good model with faster
classification time from large database within reasonable timeframe. In Agent
Based Meta Model Agents are basic modelling entities that maintain a set of
properties and behaviours. By factoring agents, relationships, and behaviours into
separate components, more modular and expressive models can be created.
Research shows the knowledge discovery is using multi-agent approach for
quicker and reliable information retrieval [2-11].

3.3 Introduction to Classifiers

Classification is the most familiar and most popular data mining technique.
Classification application include medical diagnosis, image and pattern
recognition, loan approval, detecting faults in industry applications and classifying
financial market trends. Classification is the separation or ordering of objects (or things) into *classes*. If the classes are created without looking at data (non-empirically), the classification is called *apriori classification*. If the classes are created empirically (by looking at data), the classification is called *posteriori classification*.

**I] Bayes Classifier**

A *naive Bayes classifier* is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined.

Following are the subtypes of Bayes Classifier:

- AODE
- AODEsr
- BayesianLogisticRegression
- BayesNet
- ComplementNaiveBayes
- DMNBtext
- HNB
- NaiveBayes
- NaiveBayesMultinomial
- NaiveBayesMultinomialUpdatable
- NaiveBayesSimple
- NaiveBayesUpdatable
- WAODE

**II] Function Classifier**

Function algorithms are classified by type of mathematical equation that represents their relationship. Logistic regression is a well-known statistical technique that is used for modeling binary outcomes, such as 0 or 1. Logistic
Regression Models presents an overview of the full range of logistic models, including binary, proportional, ordered, partially ordered, and unordered categorical response regression procedures [1].

Following are the subtypes of Function Classifier:

- GaussianProcesses
- IsotonicRegression
- LeastMedSq
- LibLINEAR
- LibSVM
- LinearRegression
- Logistic
- MultilayerPerceptron
- PaceRegression
- PLScalculator
- RBFNetwork
- SimpleLogistic
- SMOreg
- VotedPerceptron
- Winnow

III] Lazy Classifier

Decision rules play an important role in KDD and data mining. Rule-based classifiers establish an accurate and interpretable model for data. In lazy learning methods new objects are classified without generalization step. For example, in kNN (k Nearest Neighbors) method, the decision of new object x can be made by taking a vote between k nearest neighbours of x [2].

In this classifier no offline learning, that is done during runtime, e.g., k-NN.

Following are the subtypes of Lazy Classifier:

- IB1
- IBk
- KStar
- LBR
- LWL

IV] Meta Classifier

Meta classifier used to develop the statistical model for given dataset to predict the accuracy. Bootstrap aggregating (bagging) and boosting are useful techniques to improve the predictive performance of tree models. Boosting is also
be useful in connection with many other models, e.g. for additive models with high-dimensional predictors; whereas bagging is most prominent for improving tree algorithms. Boosting is a very different method to generate multiple predictions (function estimates) and combine them linearly.

The meta-classifier approach is one of the simplest approaches to this problem. Given a base classifier, the approach is to learn a meta classifier that predicts the correctness of each instance classification of the base classifier. The source of the meta training data are the training instances. The meta label of an instance indicates reliable classification, if the instance is classified correctly by the base classifier; otherwise, the meta label indicates unreliable classification. The meta classifier plus the base classifier form one combined classifier. The classification rule of the combined classifier is to assign a class predicted by the base classifier to an instance if the meta classifier decides that the classification is reliable. Weka also offers a meta-classifier that takes a search algorithm and evaluator next to the base classifier. This makes the attribute selection process completely transparent and the base classifier receives only the reduced dataset.

This is the full classname of the meta-classifier:
weka.classifiers.meta.AttributeSelectedClassifier

Following are the subtypes of Meta Classifier:

<table>
<thead>
<tr>
<th>Package weka.classifiers.meta</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class Summary</strong></td>
</tr>
<tr>
<td><strong>AdaBoostM1</strong></td>
</tr>
<tr>
<td><strong>AdditiveRegression</strong></td>
</tr>
<tr>
<td><strong>AttributeSelectedClassifier</strong></td>
</tr>
<tr>
<td><strong>Bagging</strong></td>
</tr>
<tr>
<td><strong>ClassificationViaRegression</strong></td>
</tr>
<tr>
<td><strong>CostSensitiveClassifier</strong></td>
</tr>
<tr>
<td><strong>CVParameterSelection</strong></td>
</tr>
<tr>
<td>Decorate</td>
</tr>
<tr>
<td>------------------------------</td>
</tr>
<tr>
<td>FilteredClassifier</td>
</tr>
<tr>
<td>Grading</td>
</tr>
<tr>
<td>LogitBoost</td>
</tr>
<tr>
<td>MetaCost</td>
</tr>
<tr>
<td>MultiBoostAB</td>
</tr>
<tr>
<td>MultiClassClassifier</td>
</tr>
<tr>
<td>MultiScheme</td>
</tr>
<tr>
<td>OrdinalClassClassifier</td>
</tr>
</tbody>
</table>

V] MI Classifier (Mul-Instance Classifier)

Multi-instance (MI) classification is a supervised learning technique, but differs from normal supervised learning:
- it has multiple instances in an example
- only one class label is observable for all the instances in an example

Classifiers

Multi-instance classifiers were originally available through a separate software package, Multi-Instance Learning Kit (= MILK). But due to the introduction of the relational attribute in the ARFF format, they became part of Weka in version 3.5.3 (developer version only).

These classifiers can now be found in the following package:

weka.classifiers.mi
Data Format

The data format for multi-instance classifiers is fairly simple:

- **bag-id** - nominal attribute; unique identifier for each bag
- **bag** - relational attribute; contains the instances of an example
- **class** - the class label for the examples

Weka offers two filters to convert from flat file format (or propositional format), which is normally used in supervised classification, to multi-instance format and vice versa:

- `weka.filters.unsupervised.attribute.PropositionalToMultiInstance`
- `weka.filters.unsupervised.attribute.MultiInstanceToPropositional`

Following are the subtypes of MI Classifier:

- CitationKNN
- MDD
- MIBoost
- MIDD
- MIEMDD
- MILR
- MINND
- MIOptimalBall
- MISMO
- MISVM
- MIWrapper
- SimpleMI

VI] Misc Classifier

In this type of classifier those various classifiers are included that do not fit in any another category.

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FLR</strong></td>
</tr>
<tr>
<td>Fuzzy Lattice Reasoning Classifier (FLR) v5.0</td>
</tr>
<tr>
<td>The Fuzzy Lattice Reasoning Classifier uses the notion of Fuzzy Lattices for creating a Reasoning Environment.</td>
</tr>
<tr>
<td>The current version can be used for classification using numeric predictors.</td>
</tr>
<tr>
<td>For more information see: 1.</td>
</tr>
</tbody>
</table>
## VII]  Rule Classifier

In Rule classifier algorithms can be used for classification of datasets with nominal class labels. Conjugative rule generates the initial rule set and prune two variants of each rule from the randomized data by using grow phase and prune phase procedures. Only one variant is generated from an empty rule while the other is generated, by adding antecedents to the original rule. After all the rules in have been examined and if there are still residual positives, then more rules are generated [3].

Following are the subtypes of Rules Classifier:

- **ConjunctiveRule** This class implements a single conjunctive rule learner that can predict for numeric and nominal class labels.

- **DecisionTable** Class for building and using a simple decision table majority classifier.

- **DecisionTable.hashKey** Class providing keys to the hash table
Classification with Zero-R Algorithm using Weka / 87

- **JRip** This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which is proposed by William W.

- **M5Rules** Generates a decision list for regression problems using separate-and-conquer.

- **OneR** Class for building and using a 1R classifier.

- **PART** Class for generating a PART decision list.

- **Prism** Class for building and using a PRISM rule set for classification.

- **Ridor** The implementation of a RIpple-DOwn Rule learner.

- **Rule** Abstract class of generic rule

- **RuleStats** This class implements the statistics functions used in the propositional rule learner, from the simpler ones like count of true/false positive/negatives, filter data based on the ruleset, etc.

- **Zero-R** Class for building and using a 0-R classifier.

VIII] Tree Classifier

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making.

Decision trees are a classic way to represent information from a machine-learning algorithm, and offer a fast and powerful way to express structures in data. Decision Tree uses the concept of information gain to make a tree of classificatory decisions with respect to a previously chosen target classification. Decision tree provides information produced by data mining techniques that can be represented in many different ways.
Decision trees used in data mining are of two main types:

- **Classification tree** analysis is when the predicted outcome is the class to which the data belongs.
- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

The term **Classification And Regression Tree (CART)** analysis is an umbrella term used to refer to both of the above procedures. J48 builds decision trees from a set of training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets.

**Decision Tree Advantages**

Amongst other data mining methods, decision trees have various advantages:

- **Simple to understand and interpret.** People are able to understand decision tree models after a brief explanation.
- **Requires little data preparation.** Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- **Able to handle both numerical and categorical data.** Other techniques are usually specialised in analysing datasets that have only one type of variable. Ex: relation rules can be used only with nominal variables while neural networks can be used only with numerical variables.
- **Uses a white box model.** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is difficult to understand.
- **Possible to validate a model using statistical tests.** That makes it possible to account for the reliability of the model.
- **Robust.** Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

- **Performs well with large data in a short time.** Large amounts of data can be analysed using standard computing resources.

**Package weka.classifiers.trees**

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADTree</strong></td>
</tr>
<tr>
<td><strong>DecisionStump</strong></td>
</tr>
<tr>
<td><strong>Id3</strong></td>
</tr>
<tr>
<td><strong>J48</strong></td>
</tr>
<tr>
<td><strong>LMT</strong></td>
</tr>
<tr>
<td><strong>M5P</strong></td>
</tr>
<tr>
<td><strong>NBTree</strong></td>
</tr>
<tr>
<td><strong>RandomForest</strong></td>
</tr>
<tr>
<td><strong>RandomTree</strong></td>
</tr>
<tr>
<td><strong>REPTree</strong></td>
</tr>
<tr>
<td><strong>UserClassifier</strong></td>
</tr>
</tbody>
</table>

### 3.4 Why Zero-R?

There are number of classification methods. Most popular methods are Decision Tree, Naive Bayes techniques and rule-based classification. A Decision Tree results in a flowchart like tree structure where each node denotes an attribute value and each branch represents an outcome of the test. The tree leaves represent the classes. There are many advantages of decision tree. DTs certainly are easy to use and efficient. Rules can be generated are that are easy to interpret and understand. They are suitable for large databases because the tree size is independent of the database size. Each tuple in the database must be filtered through the tree. This takes time proportional to the height of the tree, which is fixed. Trees can be constructed for data with many attributes. Disadvantages also
Classification with Zero-R Algorithm using Weka

exist for DT algorithms, first they do not easily handle continuous data. These attribute domains must be divided into categories to be handled. Handling missing data is difficult because correct branches in the tree could not be taken. Since the DT is constructed from the training data, overfitting may occur. This can be overcome via tree pruning. Finally, correlations among attributes in the database are ignored by the DT process.

Bayesian classification is quite different from decision tree approach. In Bayesian classification we have a hypothesis that the given data belongs to a particular class. We then calculate the probability for the hypothesis to be true. The approach requires only one scan of the whole data. Assuming the contribution by all attributes is independent and that each contributes equally to the classification problem. Naive Bayes classification is based on Bayes rule of conditional probability as expression \( P(A/B) \) stands for the probability that event A will happen, given that event B has already happened. For example, A and B may be probabilities,

\[
P(A/B) \text{ then is the probability of passing A when we know that B has been passed.}
\]

Now here is Bayes theorem:

\[
P(A/B) = P(B/A) \frac{P(A)}{P(B)} \text{ as  }
\]

\[
P(A/B) = P(A&B)/P(B)
\]

and

\[
P(B/A) = P(A&B)/P(A) \text{ Dividing first equation by second gives Bayes Theorem.}
\]

As compared with other classifiers Rule classifier algorithms are based on different Rules Methods. In this classifier of Weka most simple and easy to understand algorithm is Zero-R algorithm. Algorithms are powerful technique for solution of various combinatorial or optimization problems.

In WEKA Zero-R classifier simply predicts the majority class in the training data. Although it makes little sense to use this scheme for prediction, it can be useful for determining a baseline performance as a benchmark for other learning schemes.
In WEKA Zero-R is a simple classifier. Zero-R is a trivial classifier, but it gives a lower bound on the performance of a given dataset which should be significantly improved by more complex classifiers. As such, it is a reasonable test on how well the class can be predicted without considering the other attributes. It can be used as a Lower Bound on Performance. Any learning algorithm in WEKA is derived from the abstract weka.classifiers.

3.5 The Proposed Zero-R Algorithm

The Proposed Algorithm called of PrepZero-R shows that we are removing lower bound values 0 and checking the results how it affects the class value. Following are the steps of PrepZero-R algorithm shown in fig.3.1

In figure 3.1, in first step, we are selecting classifier Zero-R, simultaneously checking for its capabilities, if it does not satisfy its condition capabilities then directly Exit. If it is capable then build classifier. In next step getting Instance values, then set the Lower Boundary value for instance, remove Lower Boundary value form calculation, finally calculate Zero-R class value then Exit.
3.6 Experimental setup

The proposed study uses Data mining library Weka 3.6.5 and we introduce new term Accuracy \((1-e)\) which gives better results in error. The experiment was conducted using the UCI Pima Indian data set [14]. The dataset contains 768 instances of Pima Indian heritage females who were diagnosed for diabetes. There were 268 instances diagnosed with diabetes [11]. There are 8 attributes and the
diagnostic result (diabetes negative or diabetes positive) in the data set. The attributes are as follows:

- Number of times pregnant
- Plasma glucose concentration
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)
- Body mass index, diabetes
- Pedigree function
- Age (years)
- Test result (as class variable (0 or 1))

The experiment is to measure time and accuracy of a classification on the UCI Pima Indian dataset [14]. The target variable is Class Variable (number 9) in the dataset. Class variable value is mutually exclusive, either diabetes negative or diabetes positive.

In our experiment we have used five attributes as follows:

- Number of times pregnant
- Plasma glucose concentration
- Serum insulin
- Diabetes pedigree function
- Test result

### 3.6.1 Procedure to Compile Using Java

Procedure to Compile, Run and Add New Algorithms in Weka


2. Install it in E:\Weka3.4

3. Create a folder in D:\ drive called mytmp

4. Open E:\Weka3.4 folder we can view all the extracted from weka-3-4-6jre.exe; 25.4 MB

5. We will find a file called weka-src.java, Copy and paste it in D:\mytmp

6. Extract all the files using WinZip in to D:\mytmp

7. We will find the folders like lib, src, meta-inf, test and build.xml file
8. Create a folder in D:\ drive called tmp

9. Open Net beans IDE and click on File\New Project, New Project dialog box appears
   a. Categories: General
   b. Projects: Java Application
   c. Click on next button

10. In the Name and location appears
    a. Project Name: weka
    b. Project Location: D:\tmp
    c. Project Folder: D:\tmp\weka
    d. Put tick mark in Set as Main project, create main class
    e. Fill the text box with "weka.gui.Main"
    f. Click on Finish button

11. We can view the created file now in Net beans with Source package, test package and libraries

12. Go to D:\mytemp\src\main\java and copy the folder weka and paste it in D:\tmp\weka\src.
    a. A dialog appears that folder name is existing, overwrite yes or cancel
    b. Click on yes to all button.

13. Now switch to Net beans and click on Weka project and Source Package, we can observe that all the earlier files such as classification, gui, clustering, filters etc are dumped in to it.

14. Click on Build Menu\Build Main Project, weka project will be compiled.

15. Click on Run Menu \Run Main Project, weka project will be executed,

16. Step1-15 are used to compile and run existing weka file, step 16-30 are used to compile add new algorithm, recompile, re run and finally view the added algorithm in to gui chooser main window of weka.

17. Go to D:\tmp\weka\src\weka\clusterers and select SimpleKMeans.java
    a. Copy SimpleKMeans.java in the same folder (i.e.D:\tmp\weka\src\weka\clusterers) it appears as Copy of SimpleKMeans.java
b. Rename this Copy of SimpleKMeans.java as Improved Simple K-means java

18. Switch to Net Beans and double click on source package, and open weka.clusters we can see the ImprovedSimpleKmeans.java file in that package.

19. Double click on this file, the source code will be opened
   a. Modify the class name, constructor name and wherever you feel it is appropriate as ImprovedSimpleKmeans
   b. Compile this file separately by pressing f9.
   c. Run this file separately by pressing shift+f6

20. Now as a whole Compile and Run the Weka project again as mentioned in steps 14, 15, since new algorithm is added.

21. We need to make Weka aware that clusters package is added with new algorithm, in order to be able to use it in the GUI as well: to do so add the appropriate superclass/interface key into the weka/gui/GenericProperitesCreator.props file.
   a. Double-click on weka.gui package
   b. We can view GenericProperitesCreator.props text file.
   c. Double-click on GenericProperitesCreator.props text file
   e. Save the file
   f. Compile and run the weka project again as mentioned steps 15-25.

22. Now we have to make a java archive files for both java code and java class files. Because our new algorithm is in D:\tmp\weka\src\weka\clusters folder but original execution has to be done in E:\Weka3.4. So, we have to make a jar file with the same file name java-src.java and replace the existing file in E:\weka3.4
a. Cmd

b. Go to directory D:\tmp\weka\src\weka

c. Type the command as follows: `jar -cvf weka-src.jar *.*`

d. Now we can observe `weka-src.jar` file existing in the directory `D:\tmp\weka\src\weka`

e. Copy the file and paste it in E:\weka3.4 folder

f. It will ask whether to replace the existing file, Click on Ok button

23. We have to copy the `weka.class` file into E:\weka 3.4. By default NetBeans builds the class file as `weka.class` and archives it into `weka.jar`.

a. The class file is located in `D:\tmp\weka\dist` folder as `java.jar`

b. Copy the file and paste it in E:\weka3.4 folder

c. It will ask whether to replace the existing file, Click on Ok button

24. Switch to E:\weka3.4 folder and double click on the `weka3.4` (with console) shortcut to execute weka software and check whether our new algorithm is added or not.

25. Now we can observe that Improved Simple K-Means is added in `weka.cluster` package along with Simple K-Means algorithm and is functioning properly.
3.6.2 Discussion of new factor ‘Accuracy (1-e)’ for each individual attributes

The summation of Mean absolute error (e) and accuracy should be 100 for the cross validation in Weka.

Thus,

\[ \text{Accuracy} + \text{Error} = 100\% \]

So \[ \text{Accuracy} = 100 - (\text{Error}) \]

\[ \therefore \text{ the term Accuracy} = "(1-e)" \]

Thus, the result introduces the new term Accuracy (1-e).

For example: In result analysis, considering attribute number of times pregnant, the addition of Mean absolute error (e) and accuracy is 100 as

\[ 2.7745 + 97.2255 = 100. \]

The same logic is applicable to remaining all attributes.
3.6.3 Result Analysis

The following figures are the snapshot of results:
Zero-R for Plasma Gluc Conc Original

=== Run information ===

Scheme: weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances: 768
Attributes: 5

No of times Pregnant
Plasma Gluc Conc
serum insulin
Diab Pedigree func
Test Result

Test mode: 10-fold cross-validation

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

Zero-R predicts class value: 120.89453125

Time taken to build model: 578705 nano seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient -0.0853
Mean absolute error(e) 25.2014
Accuracy (1-e) 74.7986 %
Root mean squared error 31.9791
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 768

Zero-R for Plasma Gluc Conc Removing 0 values with Accuracy

=== Run information ===

Scheme: weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances: 768
Attributes: 5

No of times Pregnant
Plasma Gluc Conc
serum insulin
Diab Pedigree func
Test Result

Test mode: 10-fold cross-validation

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

Zero-R predicts class value: 120.53881278538813
Time taken to build model: 462559 nano seconds

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

Correlation coefficient  -0.0828
Mean absolute error(e)  25.1687
Accuracy (1-e)  74.8313 %
Root mean squared error  31.984
Relative absolute error  99.8703 %
Root relative squared error  100.0153 %
Total Number of Instances  768

Zero-R for Serum Insulin Original

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
   No of times Pregnant
   Plasma Gluc Conc
   serum insulin
   Diab Pedigree func
   Test Result

Test mode:10-fold cross-validation

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

Zero-R predicts class value: 79.79947916666667

Time taken to build model: 462559 nano seconds

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

Correlation coefficient  -0.1149
Mean absolute error(e)  84.6632
Accuracy (1-e)  15.3368 %
Root mean squared error  115.3473
Relative absolute error  100 %
Root relative squared error  100 %
Total Number of Instances  768
Zero-R for Serum Insulin Removing 0 values with Accuracy

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
   No of times Pregnant
   Plasma Gluc Conc
   serum insulin
   Diab Pedigree func
Test Result
Test mode:10-fold cross-validation

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

Zero-R predicts class value: 79.48249619482496
Time taken to build model: 437346 nano seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient                      -0.1058
Mean absolute error (e)                        84.5618
Accuracy (1-e)                                 15.4382 %
Root mean squared error                        115.3279
Relative absolute error                        99.8803 %
Root relative squared error                    99.9831 %
Total Number of Instances                      768

Zero-R for Diab Pedigree Func Original

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
   No of times Pregnant
   Plasma Gluc Conc
   serum insulin
   Diab Pedigree func
Test Result
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===
Zero-R predicts class value: 0.4718763020833327

Time taken to build model: 523181 nano seconds

=== Cross-validation ===

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient</td>
<td>-0.1222</td>
</tr>
<tr>
<td>Mean absolute error(e)</td>
<td>0.2457</td>
</tr>
<tr>
<td>Accuracy (1-e)</td>
<td>99.7542 %</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.3317</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>100 %</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>100 %</td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>768</td>
</tr>
</tbody>
</table>

Zero-R for Diab Pedigree Func Removing 0 values with Accuracy

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
  No of times Pregnant
  Plasma Gluc Conc
  serum insulin
  Diab Pedigree func
  Test Result
Test mode:10-fold cross-validation

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

Zero-R predicts class value: 0.4636042617960428

Time taken to build model: 17540777 nano seconds

=== Cross-validation ===

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient</td>
<td>-0.1009</td>
</tr>
<tr>
<td>Mean absolute error(e)</td>
<td>0.2458</td>
</tr>
<tr>
<td>Accuracy (1-e)</td>
<td>99.7542 %</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.3317</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>99.2693 %</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>99.9888 %</td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>768</td>
</tr>
</tbody>
</table>
Zero-R for Test Result Original

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
No of times Pregnant
Plasma Gluc Conc
serum insulin
Diab Pedigree func
Test Result
Test mode:10-fold cross-validation

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

Zero-R predicts class value: 0.3489583333333333

Time taken to build model: 421213 nano seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient -0.0908
Mean absolute error(e) 0.4548
Accuracy (1-e) 99.5452 %
Root mean squared error 0.4771
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 768

Zero-R for Test Result Removing 0 values with Accuracy

=== Run information ===

Scheme:weka.classifiers.rules.Zero-R
Relation: Data with Titles
Instances:768
Attributes:5
No of times Pregnant
Plasma Gluc Conc
serum insulin
Diab Pedigree func
Test Result
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===
Zero-R predicts class value: 0.350076103500761

Time taken to build model: 3184273 nano seconds

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient</td>
<td>-0.0846</td>
</tr>
<tr>
<td>Mean absolute error (e)</td>
<td>0.4552</td>
</tr>
<tr>
<td>Accuracy (1-e)</td>
<td>99.5448 %</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.4772</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>100.0977 %</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>100.0277 %</td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>768</td>
</tr>
</tbody>
</table>

In graphical form the results for five attributes that we have used in our experiment are as follows by using Visualize All option of Weka.

Figure 3.2 Graphical Representation of Experimental Results
Table 3.1

Comparison of Results

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Class Value</th>
<th>Accuracy</th>
<th>Time (Nano Sec)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of times pregnant</td>
<td>3.8450520833333335</td>
<td>4.494672754946728</td>
<td>97.2255</td>
<td>97.0705</td>
</tr>
<tr>
<td>Plasma glucose concentration</td>
<td>120.8945313</td>
<td>120.5388128</td>
<td>74.7986</td>
<td>74.8313</td>
</tr>
<tr>
<td>Serum insulin</td>
<td>79.79947917</td>
<td>79.48249619</td>
<td>15.3368</td>
<td>15.4382</td>
</tr>
<tr>
<td>Diabetes pedigree function</td>
<td>0.471876302</td>
<td>0.463604262</td>
<td>99.7524</td>
<td>99.7542</td>
</tr>
</tbody>
</table>
3.7 Result and Discussion

The experiment was conducted using the UCI Pima Indian data set [5]. The dataset contains 768 instances of Pima Indian heritage females who were diagnosed for diabetes.

The diagnostic result (diabetes negative or diabetes positive) in the data set. The five attributes are as follow: number of times pregnant, plasma glucose concentration, serum insulin, diabetes pedigree function and finally the test result.

The experiment is to measure time and accuracy of a classification on the UCI Pima Indian dataset [6]. Class variable value is mutually exclusive, either diabetes negative or diabetes positive. There are four standard methods for Data Mining: association, classification, clustering techniques and prediction.

In Table 3.1 the class value for each attribute is compared in Zero-R and PrepZero-R algorithm. The Accuracy is measured in Time (Nano Seconds) is comparison with Zero-R and PrepZero-R algorithm and the difference is shown in the last column in Table 3.1.

For most medical applications the logical rules are not precise but vague and the uncertainty is present both in premise and in the decision. For this kind of application a good methodology is the rule representation from decision-tree method, which is easily understood by the user [4].

The experimental result shows that we are removing lower bound values 0 and checking the results how it affects the class value. As shown in table1 the proposed algorithm has significantly reduced the running time and Accuracy. This criterion is important in agent based data mining to obtain the good knowledge model from the complex and large database. This classifier simply predicts the majority class in the training data. It makes little sense to use this scheme for prediction; it can be useful for determining a baseline performance as a benchmark for other learning schemes. Zero-R tests how well the class can be predicted without considering other attributes. It can be used as a Lower Bound on Performance and introduced new factor “Accuracy (1-e)” for each individual attribute.
3.8 Conclusion

Data mining and multi-agent approach has been used successfully in the development of large complex systems. Agents are used to perform some action or activity on behalf of a user of a computer system. The study proposes an agent based algorithm PrepZero-R using Zero-R algorithm in Weka.

There are number of classification methods. Most popular methods are Decision Tree, Naive Bayes techniques and rule-based classification. Decision trees do not easily handle continuous data. Also Naive Bayes classification is based on Bayes rule of conditional probability as expression $P(A/B)$. As compared with other classifiers Rule classifier algorithms are based on different Rules Methods. In this classifier of Weka most simple and easy to understand algorithm is Zero-R algorithm. Algorithms are powerful technique for solution of various combinatorial or optimization problems. Zero-R is a simple and trivial classifier, but it gives a lower bound on the performance of a given dataset which should be significantly improved by more complex classifiers. The Proposed Algorithm called PrepZero-R has significantly reduced time taken to build the model than Zero-R algorithm by removing the Lower Bound Values 0 while preprocessing and comparing the result with class values. To conclude the preprocessing approach for knowledge discovery shows significant increase in performance. This criterion is important in Agent Based data mining to obtain the good knowledge from complex and large databases. Also proposed study introduced new factor “Accuracy (1-e)” for each individual attribute.
REFERENCES

www.ijcaonline.org/volume12/number10/pxc3872322.pdf


www.cvc.uab.es/~petia/maya%20sacr3.pdf


ieeeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4631677

ieeeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4631677


[14] Plamena Andreeva, Maya Dimitrova1, Petia Radeva, “Data Mining Learning Models And Algorithms For Medical Applications”.


doi.ieee Computer Society.org/10.1109/ICCIMA.2003.1238106
