STUDENTS' PERFORMANCE PREDICTION SYSTEM USING MULTI AGENT DATA MINING TECHNIQUE

Dr. Abdullah AL-Malaise[1], Dr. Areej Malibari[2] and Mona Alkhozae[3]

[1]Associate Professor, Department of Information System, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, KSA
[2]Assistant Professor, Department of Computer Sciences, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, KSA
[3]Department of Computer Sciences, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, KSA

ABSTRACT

A high prediction accuracy of the students' performance is more helpful to identify the low performance students at the beginning of the learning process. Data mining is used to attain this objective. Data mining techniques are used to discover models or patterns of data, and it is much helpful in the decision-making.
Boosting technique is the most popular techniques for constructing ensembles of classifier to improve the classification accuracy. Adaptive Boosting (AdaBoost) is a generation of boosting algorithm. It is used for the binary classification and not applicable to multiclass classification directly. SAMME boosting technique extends AdaBoost to a multiclass classification without reduce it to a set of sub-binary classification.
In this paper, students' performance prediction system using Multi Agent Data Mining is proposed to predict the performance of the students based on their data with high prediction accuracy and provide help to the low students by optimization rules.
The proposed system has been implemented and evaluated by investigate the prediction accuracy of Adaboost.M1 and LogitBoost ensemble classifiers methods and with C4.5 single classifier method. The results show that using SAMME Boosting technique improves the prediction accuracy and outperformed C4.5 single classifier and LogitBoost.

KEYWORDS

E-learning, Educational Data Mining, Classification, Ensemble of Classifier, Boosting, Adaboost, SAMME, Adaboost.M1, LogitBoost, Students' Performance Prediction, Multi-Agent System.

1. INTRODUCTION

There are increasing research interests in using data mining techniques in education. This emerging field called Educational Data Mining. It can be applied on the data related to the field of education. One of the educational problems that are solved with data mining is the prediction of students' academic performances. Prediction of students' performance is more beneficial for identifying the low academic performance students. Student retention is an indicator of academic performance and enrolment management of the university. The ability to predict a student's performance is very important in educational environments. Students' academic performance is based upon different factors like social, personal, Psychological and other environmental variables. A very promising tool to achieve this objective is the use of Data Mining. Data mining techniques are used to discover hidden patterns and relationships of data, which is very much helpful in decision-making. Now-a-days educational database is increased rapidly because of the large amount of data stored in it. The loyal students motivate the higher education systems; to know them well is by using valid management and processing of the students' database. Data mining approach provides valid information from existing student to manage relationships with upcoming students [1] [2] [3].

The most useful data mining techniques in e-learning is classification. It is a predictive data mining technique, makes prediction about data values using known results found from different data. C4.5 Decision tree algorithm has been successfully used in capturing knowledge and presents a powerful method of inferring classification models from a set of labelled examples [4]. Recently, there has been increasing interest in combining classifiers concept that is proposed for the performance improvement of individual classifiers. The integration algorithms approach is used to making decision more accurate, reliable and precise. One of a mechanism that is used to build an ensemble of classifiers is using different learning methods [5]. An ensemble of classifiers is a set of classifiers whose own decisions are combined to improve the performance of the overall system. The most popular techniques for constructing ensembles are Bagging and Boosting. These two techniques manipulate the training examples to generate multiple classifiers [6] [7].

The main objective of this paper is using data mining to predict the students' performance in the courses accurately. We build a Students' Performance Prediction System that can be able to predict the students' performance based on their academic result with high accuracy. We want also to prove if the boosting technique enhances the prediction accuracy or not by investigating the prediction accuracy of different of ensemble classifiers' methods and with single classifier method. We use Multi-Agent Data Mining technique in our system; each agent is responsible for a specific tasks.

2. BACKGROUND

2.1 ELECTRONIC LEARNING (E-LEARNING):

E-learning, as one of the main areas of e-services, has undergone intensive development as an inevitable result of the recent proliferation of internet technology. Traditional learning means restrict the student to certain learning methods, at a particular place and time whereas e-learning services create wider ranges for organizations and individuals who involved in learning and teaching. These environments facilitate the delivery of large parts of education through the use of tools and materials that are accessible directly to the learners' home or office, and at any time. In addition, the advancements in technology, which are used to enhancing the interactivity and media content of the web and the increasing quality of delivery platforms, create an ideal environment for the expansion of e-learning systems [8].

2.2 DATA MINING:

Data mining is the process of discovering various models, derived values and summaries from a given collection of data. It is important to realize that the problem of discovering or estimating dependencies from data or discovering new data is only one part of the general experimental procedure used by engineers, scientists and others who apply standard steps to draw conclusions from data [9]. The overall process of finding and interpreting patterns and models from data involves the repeated application of the following steps:

1. **Understand the application domain, the relevant previous knowledge and the goals of the end-user (formulate the hypothesis)** [9].
2. **Data Collection:** Determining how to find and extract the right data for modeling. First, we need to identify the different data sources are available. Data may be scattered in different data "silos," spreadsheets, files, and hard-copy (paper) lists [10].
3. **Data integration:** Integration of multiple data cubes, databases or files. A big part of the integration activity is to build a data map, which expresses how each data element in each data set must be prepared to express it in a common format and record structure [10].
4. **Data selection:** First of all the data are collected and integrated from all the various sources, and we select only the data which useful for data mining. Only relevant information is selected [9].
5. **Pre-processing:** The Major Tasks in Data Pre-processing are: Cleaning, Transformation and Reduction.
   - **Data cleaning:** Also called data cleansing. It deals with errors detecting and removing from data in order to improve the quality of data [11]. Data cleaning usually includes fill in missing values and identify or remove outliers.
   - **Data Transformation:** Data transformation operations are additional procedures of data pre-processing that would contribute toward the success of the mining process and improve data-mining results. Some of Data transformation techniques are Normalization, Differences and ratios and Smoothing [9].
   - **Data Reduction:** For large data sets, there is an increased likelihood that an intermediate, data reduction step should be performed prior to applying data-mining techniques. While large datasets have potential for better mining results, there is no guarantee that they will produce better knowledge than small datasets. Data reduction obtains a reduced dataset representation that is much smaller in volume, yet produces the same analytical results [9].
6. **Building the model:** In this step we choose and implement the appropriate data-mining task (ex. association rules, sequential pattern discovery, classification, regression, clustering, etc.), the data mining technique and the data mining algorithm(s) to build the model.
7. **Interpretation of the discovered knowledge (model /patterns):** The interpretation of the detected pattern or model reveals whether or not the patterns are interesting. This step is also called Model Validation/Verification and uses it to represent the result in a suitable way so it can be examined thoroughly [9].
8. **Decisions / Use of Discovered Knowledge:** It helps to make use of the knowledge gained to take better decisions.

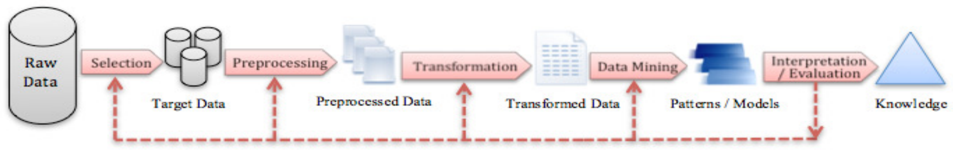


Figure 1: Data Mining process

2.3 EDUCATIONAL DATA MINING:

There are increasing research interests in using the data mining techniques in education. This emerging field called Educational Data Mining, concerns with developing methods that discover knowledge from data originating from educational environments. The data can be collected from operational and historical data reside in the databases of educational institutes. The student data can be academic or personal, and it can be collected from e-learning systems. Educational Data Mining uses different techniques such as Decision Trees, Naïve Bayes, Neural Networks, K- Nearest neighbor, and many others [12].

In the educational area, data mining is the process of transforming raw data compiled by education systems in useful information could be used to take informed decisions. It can be applied on the various data that are used in communicating with the stakeholders, student modeling, structuring the education domain, predicting the student grade, etc [13].

The prediction of students' performance with high accuracy is more beneficial for identifying low academic achievement students in the beginning. Student retention is an indicator of academic performance and enrolment management of the university. The ability to predict the performance of a student is very important in educational environments. Students' academic performance is based upon different factors like personal, Psychological, social and other environmental variables. A very promising tool to achieve this objective is the use of Data Mining [2] [3]. One of the most useful data mining techniques in e-learning is classification. It is a predictive data mining technique, makes prediction about the values using known results found from different data [4].

2.4 C4.5 DECISION TREE ALGORITHM:

It is one of the most popular classification algorithms. It is an extension to ID3 developed by Quinlan [14]. C4.5 handle both continuous and discrete attributes. To handle the continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is less than or equal to it, and those are above the threshold. Also, it handles training data with missing attribute values. Missing attribute values are simply not used in gain and entropy calculations. Largest GainRatio is used by C4.5 for the purpose of splitting that ensures a larger than average information gain [15]. This procedure continues until each leaf node contains only observations from a single class or no gain in the information yielded by further splitting. For a decision tree using continuous attributes, data is partitioned into 2 outcomes at each node based on a threshold value for a single attribute. The threshold value with the greatest gain ratio value is selected at each node in the decision tree [9].

2.5 CONFUSION MATRIX:

The confusion matrix is a visualization tool commonly used to present performances of classifiers in classification tasks. It is used to show the relationships between real class attributes and that of predicted classes. The level of effectiveness of the classification model is calculated with the number of correct and incorrect classifications in each possible value of the variables being classified in the confusion matrix [16]. Given two classes the contingency or confusion matrix can be given as in Table 1 [17].

Table 1: Confusion Matrix

Confusion Matrix	Predicted Class		
Actual Class	Class	Class 1	Class 2
	Class 1	True Positive (TP)	False Negative (FN)
	Class 2	False Positive (FP)	True Negative (TN)

Where true positives (TP) mean the correct classifications of the positive examples; true negatives (TN) are the correct classifications of the negative examples; false positives (FP) represent the incorrect classification of the negative examples into the positive class, and false negatives (FN) are the incorrect classification of the positive examples into the negative class. There are several performance metrics available [17] [18].

1. The predictive **accuracy** of the classifier measures the proportion of correctly classified instances $= \mathrm{TP} + \mathrm{TN}/\mathrm{TP} + \mathrm{FP} \ + \mathrm{TN} \ + \mathrm{FN}$

2. **True Positive Rate (TPR or Recall or Sensitivity):** measures the percent of actual positive examples that are correctly classified $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$

3. **True Negative Rate (TNR or Specificity):** measures the percent of actual negative examples that are correctly classified $= \mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$

4. **Positive Predictive Value** (PPV): often called **Precision**, it is the percentage of the examples predicted to be positive that were correct $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$

5. **False Negative Rate (FNR):** The percentage of positive examples that were incorrectly classified $= \mathrm{FN}/(\mathrm{TP}+\mathrm{FN}) = 1-\mathrm{TPR}$

6. **False Positive Rate (FPR):** The percentage of negative example that were incorrectly classified $= \mathrm{FP}/(\mathrm{TN}+\mathrm{FP}) = 1-\mathrm{TNR}$

2.6 ENSEMBLE OF CLASSIFIERS:

An ensemble of classifiers is a set of classifiers whose own decisions are combined in some way to improve the performance of the overall system. It classifies new data points by taking a vote of their predictions [6] [7]. Ensemble classifiers are also known as multiple classifier systems, mixture of experts, and committee of classifiers [19].

The concept of ensemble learning is that no single classifier can claim to be uniformly superior to any other and that the integration of several single classifiers will enhance the performance of the final classifier (e.g. accuracy, reliability). Hence, ensemble classifiers are more accurate than the individual classifiers that make them up. The easiest approach to generate diverse classifiers is to manipulate the training data and run a base learner on the manipulated training data multiple times. Ensemble learning methods have been shown to be very successful in improving the classification accuracy of certain classifiers for artificial and real-world datasets [7].

One of the most popular techniques for constructing the ensembles of classifiers is **Boosting** [6] [7] [20].

Schapire introduce Boosting as a method to enhance the classification performance of a weak learning algorithm [23]. Boosting adaptively reweights the training data based on the error rate of the previous classifier. Boosting refers to a general and provably effective method that attempts to ‘boost’ the accuracy of any given learning algorithm. Although boosting is not algorithmically constrained, most boosting algorithms involve learning iteratively and adding weak classifiers to come up with a final strong classifier. Each added weak classifier is usually weighted according to its accuracy and trained with reweighted training data.

2.7 ADABOOST:

One of the earliest and best-known boosting algorithms is AdaBoost. Freund and Schapire introduced the Adaptive Boosting (AdaBoost) method as another method to influence the training data [22]. Initially, the algorithm assigns every instance xi in the training data with an equal weight. In each iteration i, the learning algorithm trained sequentially and tries to minimize the weighted error on the training set and returns the classifier Hi(x). The weighted error of Hi(x) is calculated and applied to update the weights on the instances of the training data xi. The weight of instance xi increases according to its impacts on the performance of the classifier that assigns a high weight for a misclassified instance xi and a low weight for a correctly classified instance. This allows for the current classifier to be focused on the most difficult patterns, that is, the ones that were not well classified by the previous classifiers. The final classifier H*(x) is constructed by a weighted vote of the individual Hi(x) according to its classification accuracy based on the weighted training set [7]. AdaBoost is designed for two-class classification and hence not directly applicable to multiclass problems.

2.8 MULTI CLASS CLASSIFICATION ALGORITHMS:

There are two approaches for extending binary Boosting algorithms to the multi-class case, depending on whether multi-class or binary weak-learners are used [24]. The most straightforward extension substitutes AdaBoost's binary weak-learners by multi-class ones, and this is the case of AdaBoost.M1, AdaBoost.M2 [21] LogitBoost [25] and SAMME [26].

The second approach transforms the original multi-class problem into a set of binary problems solved using binary weak-learners, each of which separates the set of classes in two groups [24]. AdaBoost.MH algorithm [27] is the most popular approach of this kind.

2.9 ADABOOST.M1:

AdaBoost.M1 is the first attempt to extend AdaBoost to multi-class classification problems and is very popular [22].

Adaboost.M1 assumes that the error of each weak classifier is less than 1/2, with respect to the distribution on which it was trained. This assumption is easily satisfied for two-class classification problems because the error rate of random guessing is 1/2, and it is harder to achieve in the multiclass case, where the random guessing error rate is (K − 1)/K. As a result, AdaBoost easily fail in the multi-class case and stopped [26].

```
1. Initialize the observation weights $w_i=1/n$, $i=1,2,\dots,n$.
2. For m=1 to M:
    (a) Fit a classifier $T^{(m)}(x)$ to the training data using weights $w_i$.
    (b) Compute $err^{(m)} = \sum_{i=1}^{n} w_i \, \mathbb{I}\,(c_i \neq T^{(m)}(x_i)) / \sum_{i=1}^{n} w_i$
        If ($err^{(m)}$<=0 || $err^{(m)}$>=0.5) return;
    (c) Compute $\alpha^{(m)} = \log \frac{1- err^{(m)}}{err^{(m)}}$
    (d) Set $w_i \leftarrow w_i . \exp(\alpha^{(m)} . \mathbb{I}\,(c_i \neq T^{(m)}(x_i)))$
    for i = 1,2,…, n.
    (e) Re-normalize $w_i$.
3. Output $C(x) = \arg\max \sum_{m=1}^{M} \alpha^{(m)} . \mathbb{I}\left(T^{(m)}(x) = k\right)$.
```

Figure 2: Adaboost.M1 Algorithm [26].

The weights of all dataset instances are then normalized so that the sum of these instances is equal to 1 and in order to keep this constraint, the weight of each instance is divided by the sum of the new weights.

After M rounds, an ensemble of classifiers (composite model) will be generated, which is then used to classify a new data. When a new instance X comes, it is classified through these steps [28].

```
Initialize weight of each class to 0;
  for i = 1 to k do
```

```
        Get weight $\alpha^{(i)}$ of classifier Ti ;
        Get class prediction for X from Ti: c = Ti(X);
        Add $\alpha^{(i)}$to weight for class c;
    endfor
  Return the class with the largest weight;
```

2.10 SAMME:

SAMME (Stagewise Additive Modeling using Multiclass Exponential Loss Function) algorithm extends the original AdaBoost algorithm to the multiclass case directly without reducing it to multiple two-class problems to solve Adaboost.M1 problem. For a K-class classification task, SAMME returns only one weighted classifier (rather than K) in each boosting iteration and the weak classifier only needs to be better than K-class random guessing (rather than 1/2) [26].

The solution of SAMME is compatible with the classification rule, so it is optimal in minimizing the misclassification error and only needs weak classifiers better than random guessing (correct probability larger than 1/K), rather than better than 1/2 as the two-class AdaBoost requires [26].

```
1. Initialize the observation weights $w_i=1/n$, $i=1,2,\ldots,n$.
2. For m=1 to M:
    (a) Fit a classifier $T^{(m)}(x)$ to the training data using weights $w_i$.
    (b) Compute $\mathrm{err}^{(m)} = \sum_{i=1}^{n} w_i \, \mathbb{I}\,(c_i \neq T^{(m)}(x_i)) / \sum_{i=1}^{n} w_i$
        (b.1) If ($\mathrm{err}^{(m)}$<=0 || $\mathrm{err}^{(m)}$>= (1 - (1/k))) return;
    (c) Compute $\alpha^{(m)} = \log \frac{1-\mathrm{err}^{(m)}}{\mathrm{err}^{(m)}} + \log(K-1)$
    (d) Set $w_i \leftarrow w_i . \exp(\alpha^{(m)} . \mathbb{I}\,(c_i \neq T^{(m)}(x_i)))$
    for i = 1,2,…, n.
    (e) Re-normalize $w_i$.
3. Output $C(x) = \arg\max \sum_{m=1}^{M} \alpha^{(m)} . \mathbb{I}\left(T^{(m)}(x) = k\right)$.
```

Figure 3: SAMME Algorithm [26].

The difference between SAMME and AdaBoost.M1 are in (b.1) and (c):

In the case of $K = 2$, it is equivalent to the original two-class AdaBoost because $\log(K - 1) = 0$. The term $\log(K - 1)$ is not artificial, it follows naturally from the multiclass generalization of the exponential loss in the binary case. Also, it makes SAMME Algorithm equivalent to fitting a forward stagewise additive model using the multi-class exponential loss function. In additional, if K=2, the algorithm returns to binary Adaboost. It is highly competitive with the best currently available multiclass classification methods, in terms of both practical performance and computational cost [26].

2.11 LOGITBOOST:

LogitBoost [25] performs additive logistic regression and generates models that maximize the probability of a class. In each iteration, the algorithm fits a regression model to a weighted training data. For a two-class problem, if the weak classifier minimizes the squared error, then the probability of the first class is maximized, and it can be extended to the multiclass as well. In general, AdaBoost optimizes the exponential loss whereas LogitBoost optimizes the probability [29]. LogitBoost relies on a binomial log-likelihood as a loss function, which is a more natural criterion in binary classification than the exponential criterion underlying the AdaBoost algorithm. LogitBoost can operate with Logit models, decision stumps, or decision trees as basic classifiers [30]. Decision Stump is a simplified Decision Tree introduced by Iba and Langley, having just one level and builds simple binary tree [31].

2.12 MULTI AGENT SYSTEM:

Agent as a computer system situated in an environment and is capable of autonomous action in this environment in order to meet its design objectives [32]. Agents are task-oriented, active, modeled to perform specific tasks and capable of autonomous action and decision-making. When combining multiple agents in one system to solve a problem, the system becomes a Multi-Agent System (MAS). These systems are comprised of agents that solve problems individually that are simpler than the overall problem. They can communicate and assist each other in achieving larger and more complex goals. Agents and data mining can work together to achieve required target. Data mining agents perform various functions of data mining. It is increasingly significant to develop better methods and techniques to organize the data for better decision-making processes [34]. Several agent development platforms and toolkits have been produced. Examples include AgentBuilder, AgentTool, ASL, Bee-gent, Grasshopper-2, MOLE, Open Agent Architecture, RETSINA, Zeus and JADE [33].

2.13 JAVA AGENT DEVELOPMENT ENVIRONMENT (JADE):

JADE (Java Agent Development Environment), is FIPA (Foundation for Intelligent Physical Agents) compliant middleware that enables development of applications based on the agent paradigm. JADE is fully implemented in the Java programming language. JADE simplifies the multi-agent systems implementation through a middleware that complies with the specifications of FIPA and through graphical tools that support the debugging and deployment phases. A JADE agent platform can be distributed across machines, which do not even need to share the same operating system, and the configuration can be controlled via a remote GUI [33].

3. RELATED WORK:

Paris et al. [5] compared the data mining methods accuracy to classifying students in order to predicting class grade of a student. These predictions are more useful for identifying the weak students and assisting administration to take remedial measures at initial stages to produce excellent graduate that will graduate at least with the second class upper.

Rathee and Mathur apply ID3, C4.5 and CART decision tree algorithms on the educational data for predicting a student's performance in the examination. All the algorithms are applied on the internal assessment data of the student to predict their academic performance in the final exam. The efficiency of various decision tree algorithms can be analyzed based on their accuracy and time taken to derive the tree. The predictions obtained from the system have helped the tutor to identify the weak students and improve their performance. C4.5 is the best algorithm among all the three because it provides better accuracy and efficiency than the other algorithms [35].

Minaei-Bidgoli, Kortemeyer and Punch applied data mining classifiers as a means of comparing and analyzing students' use and performance who have taken a technical course via the web. The results show that combining multiple classifiers leads to a significant accuracy improvement in a given data set. Prediction performance of combining classifiers is often better than a single classifier because the decision is relying on the combined output of several models [36].

Zhu et al. proposed SAMME algorithm using a multiclass exponential loss function. The numerical experiments have indicated that AdaBoost.MH in general performs very well, and

SAMME's performance is comparable with that of the AdaBoost.MH, and sometimes slightly better. Also, they discussed the computational cost of SAMME. SAMME generates only one K-class classifier in each iteration; thus, it is K times faster than AdaBoost.MH [26].

Bakar et al. proposed an agent based data classification approach, and it is based on creating agent within the main classification process. They show the use of agent within the classification theory, which will help to improve classification speed, and maintain the quality of knowledge. 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 [37].

4. STUDENTS' PERFORMANCE PREDICTION SYSTEM:

4.1 CONCEPTUAL MODEL AND ARCHITECTURE:

This system identify the factors which affects on the students' performance and helps students to understand their learning status through the predicted performance using the performance prediction model and students current learning usage data and views this performance to students with suitable messages to motivate and regulate their own behavior. The teacher also can view the predicted students' performance and help earlier in identifying weak students who need special attention and take the required procedure, such as sending an email.

The first step is historical students' data selection and pre-processing. The database has been considered as 3/3 initially in which 2/3 of the data after pre-processing has been used for training data to build the performance model and the remaining 1/3 data is used for testing purpose to evaluate the model.

Also, it selects and pre-processes the current students' data to predicts the current students' data and store the results with the current students' data and the best rules for performance optimization in the **Students Performance Information DB** to be ready for **Performance prediction Agent** to predict the current students' performance to view their current progress.

Rules Generator: traverses each path of the highest weight tree model to generate the tree rules and store it with its performance class in **Model and Rules DB.**

Model and Rules DB now contains the Model and the generated rules. There is no need to run the complete process of model building and evaluation again, and the system will provide the Model and Rules to **Performance Prediction Agent** directly to predict the current students' performance.

Messages DB: Contains Adaptive Messages according to students' learning status and current performance and it can affect students' learning behaviors and achievements.

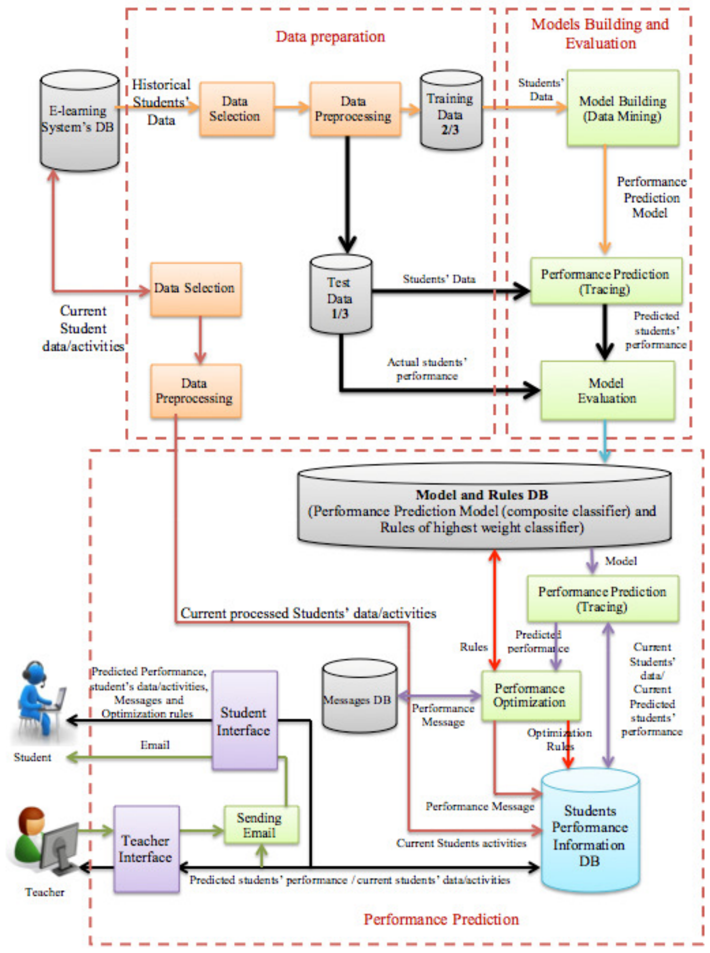

Data preparation
Models Building and Evaluation
E-learning System's DB
Historical Students' Data
Data Selection
Data Preprocessing
Training Data 2/3
Students' Data
Model Building (Data Mining)
Performance Prediction Model
Current Student data/activities
Data Selection
Test Data 1/3
Students' Data
Performance Prediction (Tracing)
Predicted students' performance
Actual students' performance
Data Preprocessing
Model Evaluation
Model and Rules DB
(Performance Prediction Model (composite classifier) and Rules of highest weight classifier)
Model
Performance Prediction (Tracing)
Current processed Students' data/activities
Rules
Predicted performance
Current Students' data/ Current Predicted students' performance
Predicted Performance, student's data/activities, Messages and Optimization rules
Student Interface
Messages DB
Performance Message
Performance Optimization
Optimization Rules
Student
Email
Sending Email
Performance Message
Students Performance Information DB
Teacher Interface
Current Students activities
Predicted students' performance / current students' data/activities
Teacher
Performance Prediction

Figure 4: System Architecture

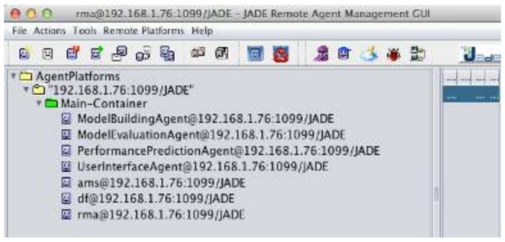
4.2 SYSTEM IMPLEMENTATION:

4.2.1 PROGRAMMING LANGUAGE, TOOLS AND LIBRARIES:

The system code is written in Java programming language (**NetBeans**). The functionalities are written using traditional java classes then these classes are converted into an agent-based behavior. The system is implemented in Java programming language to take advantage of existing library JADE and WEKA. **WEKA** (Waikato Environment for Knowledge Analysis) is an open source toolkit, and it consists of a collection of machine learning algorithms for data mining tasks [38]. We use **SAMME** boosting algorithm in our system and use **Adaboost.M1** and **LogitBoost** algorithm to evaluate the system. For **SAMME**, the default iterations number is 10. **C4.5** decision tree algorithm was used as a base classifier for **SAMME** and **Adaboost.M1**, and decision stump was used as the base classifier in **LogitBoost**. **JADE** is a platform using Java to establishing multi-Agent system with a series of Agent behaviors of Agent.

4.2.2 AGENTS COMMUNICATION:

Figure 5 shows Jade Agent platform, it contains one container: the “Main Container” and contains four agents: User Interface Agent, Model Building Agent, Model Evaluation Agent and Performance Prediction Agent, each agent responsible for specific tasks. User Interface Agent communicates with other Agents through sending request messages and receives the results. When sending a message needs only to specify the identity of the accepting Agent and the content.


rma@192.168.1.76:1099/JADE – JADE Remote Agent Management GUI
File Actions Tools Remote Platforms Help

```
▾ AgentPlatforms
  ▾ "192.168.1.76:1099/JADE"
    ▾ Main-Container
        ModelBuildingAgent@192.168.1.76:1099/JADE
        ModelEvaluationAgent@192.168.1.76:1099/JADE
        PerformancePredictionAgent@192.168.1.76:1099/JADE
        UserInterfaceAgent@192.168.1.76:1099/JADE
        ams@192.168.1.76:1099/JADE
        df@192.168.1.76:1099/JADE
        rma@192.168.1.76:1099/JADE
```

Figure 5: Multi Agent Platform

1. **User Interface Agent:** It is responsible for receiving the user request and sending this request with its required inputs to other agents to performs their tasks and receives the results to view it to the user. It is also responsible for gathering the information about students' data, attributes number, records number and the types of the data and other information.

2. **Model Building Agent:** It is responsible for building the Prediction Model, it receives the building request from **User Interface Agent** with the required data: Training Data, Iteration Number for boosting and send the result (Prediction Model) to **User Interface Agent,** and also generate the rules from the highest weight classifier and store it to be used later for **performance optimization**.

3. **Model Evaluation Agent:** It is responsible for Evaluate the Prediction Model, it receives the evaluation request from **User Interface Agent** with the required data: Prediction Model, Test Data and the actual students' performance to evaluate the **Model** and send the evaluation results to **User Interface Agent**.

4. **Performance Prediction Agent:** It is responsible for predicting the students' performance. First, it receives the prediction request from **User Interface Agent** with the required data: Prediction Model and students' Data to predict the students' performance and store it to be ready for prediction request from users. When the Performance Prediction Agent receives the user prediction request by User Interface Agent by student's ID or certain predicted performance, it selects the required student with the predicted performance and sends the prediction results to **User Interface Agent**, **Performance Prediction Agent.**

   For the optimization, **Performance Prediction Agent** select the suitable rules for optimize the low student's performance, view the reason for getting this low performance, motivate the student to get high performance by these rules and send these results to **User Interface Agent**.

4.2.3 SYSTEM IMPLEMENTATION STEPS:

A. Data Preparation:

Data preparation implemented manually, so the User Interface Agent will send the requests with the required data (inputs) to other agents and gets the results.

B. Data Collection:

Students' data were collected manually from EMES e-learning system of the computer skills (CPIT100) course, KAU, Jeddah, Saudi Arabia of session 2012-13. The course sections are: AAP = 79 female students and ABP = 76 female students.

C. Data Selection:

We select only the data is useful for data mining from EMES e-learning system and entered these data to an external Excel sheets.

The attributes selected for prediction:

- ✓ Student's name.
- ✓ Student's ID.
- ✓ Number of solved quizzes.
- ✓ Number of the correct answers of all quizzes.
- ✓ Number of submitted assignments.
- ✓ Number of the correct answers of all assignments.
- ✓ Total time of login hours the student was spent in the e-learning system.
- ✓ Number of pages the student was read.
- ✓ The student final grade class.
- ✓

D. Data Pre-processing:

1) **Data cleaning:** Data cleaning involves several of processes such as filling in the missing values, removing outliers, smoothing the noisy data and resolving inconsistencies.

2) **Data Transformation:** The data is from different resource Different sections' teachers and Different number of quizzes and assignments in each course section. So we take the percentage of these attributes. The attributes are:

   All solved quizzes percentage = number of solved quizzes / number of all quizzes *100
   Percentage of the correct answers of the all quizzes = number of the correct answers of all quizzes / number of quizzes answers *100
   All submitted assignments percentage = number of submitted assignments / number of all assignments *100
   Percentage of the correct answers of the all assignments = number of the correct answers of all assignments / number of assignments answers *100

   Smoothing: Rounding the transformed attributes values. If the value for the attributes is 0.93, then the smoothed values will be = 1.

3) **Data Reduction:**
   i. **Column-reduction techniques (Features /Attributes reduction):** We choose only the attributes that are used for building the prediction model.
      To build the prediction model we choose:
      - ✓ All solved quizzes percentage.
      - ✓ Percentage of the correct answers of the all quizzes.
      - ✓ All submitted assignments percentage.
      - ✓ Percentage of the correct answers of the all assignments.
      - ✓ Total time of login hours the student was spent in the e-learning system.
      - ✓ Number of pages the student was read.
      - ✓ The student final grade class.

   ii. **Feature-discretization techniques (Values Reduction):**
   We reduce the values manually, based on our priori knowledge about the attributes. So:
      If the student's grade >=90 then the performance class will be High.
      If the student's grade >=70 & < 90 then the performance class will be Medium.
      If the student's grade <70 then the performance class will be Low.

E. Data Formatting: Convert the students' data in the Excel file to ARFF file (Attribute-Relation File Format).

F. Data Partitioning: The data will be divided into training and testing data. The training data is applied to build the model while the testing data is used to verify the model.

G. Performance Prediction Model Building: Build the performance prediction model from Training Data using data mining technique. We use the Classification task, decision tree method (C4.5 with SAMME boosting technique) to build the performance model and generate and store the rules for performance optimization.

H. Performance Prediction Model Evaluation: Evaluate the prediction model by Test Data using the confusion matrix and calculates the prediction accuracy and store the evaluated Model.

I. Students' Performance Prediction: Predict the students' performance and optimize the low predicted students' performance and store students' performance and students' activities/data to be ready for performance prediction request.

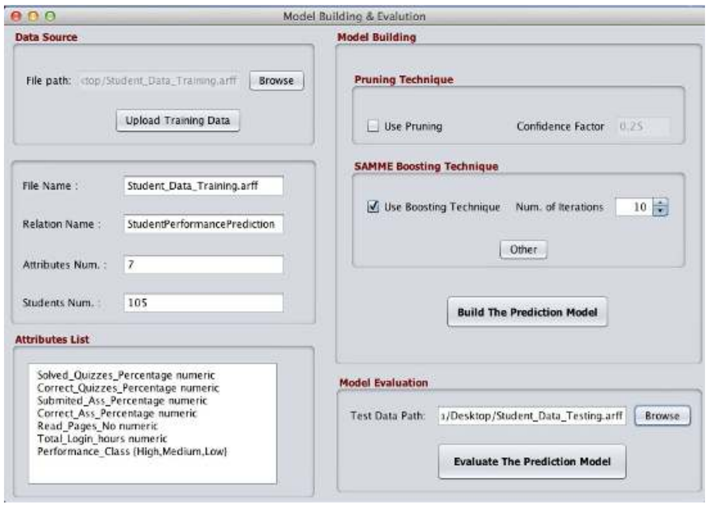
1) Students' Performance Prediction Model Building and Evaluation:

a) Build the Model: We select the pre-processed **Training Data** and click on **Upload Training Data**.

When click on **Upload Training Data**, the system shows the training data description, these data are file name; relation name, attributes number, students' number and attributes list with its data type.

We choose **Use Boosting Technique** (**SAMME**) with 10 iterations and click on **Build The Prediction Model**. A composite model (10 classifiers) is built and each classifier has a weight that indicates its accuracy, the system traverses each path of the highest weight classifier to generate the rules and store it with its performance class to be used for performance optimization**.**

b) Evaluate the Model: We select the pre-processed **Test Data** and click on **Evaluate the Prediction Model.**



File path: ctop/Student_Data_Training.arff

File Name : Student_Data_Training.arff

Relation Name : StudentPerformancePrediction

Attributes Num. : 7

Students Num. : 105

```
Solved_Quizzes_Percentage numeric
Correct_Quizzes_Percentage numeric
Submited_Ass_Percentage numeric
Correct_Ass_Percentage numeric
Read_Pages_No numeric
Total_Login_hours numeric
Performance_Class {High,Medium,Low}
```

☐ Use Pruning   Confidence Factor 0.25

☑ Use Boosting Technique   Num. of Iterations 10

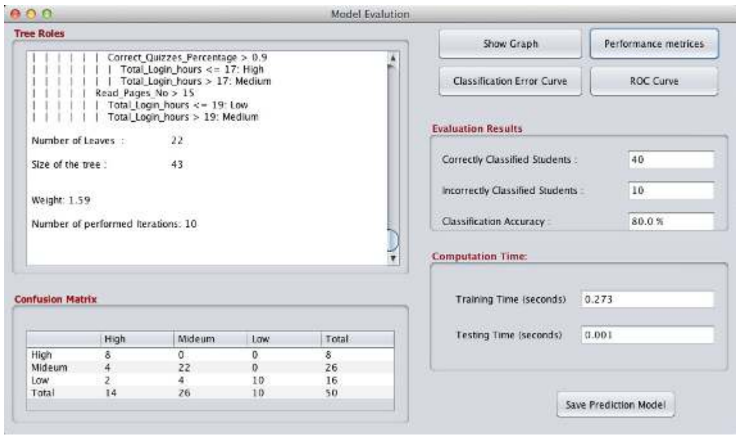
Test Data Path: ı/Desktop/Student_Data_Testing.arff

Figure 6: Model Building and Evaluation Interface

The system evaluates the model by tracing the prediction model with the processed test data, compares the predicted student performance with actual student performance and evaluates the model by the **Confusion Matrix**. The evaluation window in Figure 7 shows the model rules, confusion matrix result, evaluation results and computation time.

For computational time, we compute the start time and end time for building model and model testing time processes by stopwatch (Java Timer Class) and subtract start time from end time and divided by 1000 because the stopwatch returns the time in Milliseconds.

Duration = end Time – start Time
Time in seconds = Duration / 1000



Tree Roles

```
| | | | | | Correct_Quizzes_Percentage > 0.9
| | | | | | | Total_Login_hours <= 17: High
| | | | | | | Total_Login_hours > 17: Medium
| | | | | Read_Pages_No > 15
| | | | | | Total_Login_hours <= 19: Low
| | | | | | Total_Login_hours > 19: Medium

Number of Leaves  :        22

Size of the tree :         43


Weight: 1.59

Number of performed Iterations: 10
```

Confusion Matrix

	High	Mideum	Low	Total
High	8	0	0	8
Mideum	4	22	0	26
Low	2	4	10	16
Total	14	26	10	50

Evaluation Results

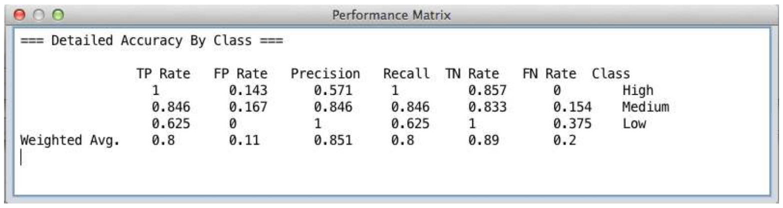
Correctly Classified Students :	40
Incorrectly Classified Students :	10
Classification Accuracy :	80.0 %

Computation Time:

Training Time (seconds)	0.273
Testing Time (seconds)	0.001

Figure 7: Model Building and Evaluation Results

View the performance metric by click on **Performance metrics** button and the results is based on the confusion matrix results.



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

                 TP Rate   FP Rate   Precision   Recall  TN Rate   FN Rate  Class
                   1         0.143     0.571      1        0.857     0        High
                   0.846     0.167     0.846      0.846    0.833     0.154    Medium
                   0.625     0         1          0.625    1         0.375    Low
Weighted Avg.      0.8       0.11      0.851      0.8      0.89      0.2
```

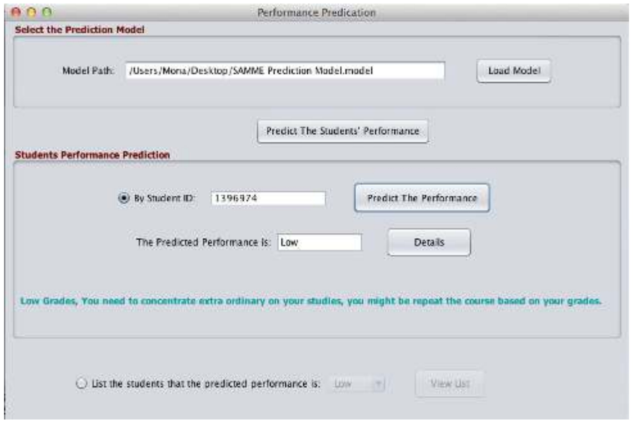
Figure 8: Performance metrics Results

Show Graph: used for visualize the tree for a single classifier (use C4.5 without boosting).
ROC Curve (Receiver Operating Characteristics): False Positive Rate (FPR) vs. True Positive Rate (TPR).

Classification Error Curve: in each boosting iteration, SAMME algorithm build a classifier then test this classifier to calculates the incorrectly classified students to get the total error weights of this classifier and calculate the classifier's accuracy weight, the curve show the incorrectly classified students in each iteration (number of iterations vs. test error).
We can save the prediction model to be used later for performance prediction.

2) **Current Students' Performance Prediction:**
First we load the Prediction Model and click on predict the Students' Performance, the system fetch all required students' data from the excel sheet and put it in a list and convert it to arff format to be suitable for tracing these data on the prediction model to predicts the students' performance and store the performance result it to be ready for student/teacher request.

When predict the performance by student ID, the system gets the predicted performance from the stored students' data. The system views the student's predicted performance with the suitable messages according to the predicted student's performance.


Performance Predication
Select the Prediction Model
Load Model
Predict The Students' Performance
Students Performance Prediction
Predict The Performance
Details
View List

Model Path: /Users/Mona/Desktop/SAMME Prediction Model.model

(●) By Student ID: 1396974

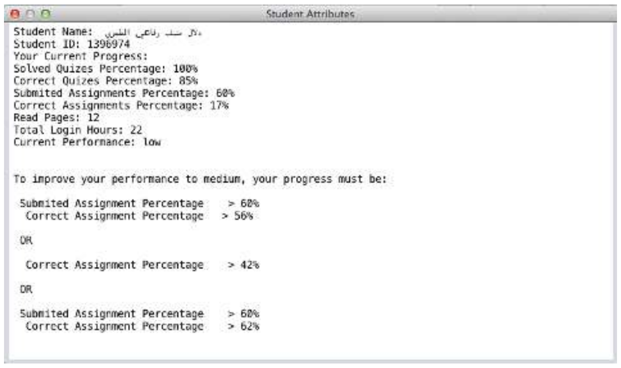
The Predicted Performance is: Low

Low Grades, You need to concentrate extra ordinary on your studies, you might be repeat the course based on your grades.

○ List the students that the predicted performance is: Low

Figure 9: Student's Performance Prediction by Students' ID (Low performance)

When click on **Details** button, the system view more details about the reason for getting this performance and how it can be optimize the performance by fetching the rules from the rules file to gain higher performance and shows the student how can achieve this high performance by motivational messages, example, if the student get Low performance, the system fetch the Medium rules only.



```
Student Name:  دلال سند رفاعي اللبري
Student ID: 1396974
Your Current Progress:
Solved Quizes Percentage: 100%
Correct Quizes Percentage: 85%
Submited Assignments Percentage: 60%
Correct Assignments Percentage: 17%
Read Pages: 12
Total Login Hours: 22
Current Performance: low


To improve your performance to medium, your progress must be:

 Submited Assignment Percentage    > 60%
  Correct Assignment Percentage   > 56%

 OR

  Correct Assignment Percentage    > 42%

 OR

 Submited Assignment Percentage    > 60%
  Correct Assignment Percentage    > 62%
```

Figure 10: Student's Performance Prediction Details (Low Performance)

- The teacher can view the students' predicted performance by select a certain predicted performance to view a list of students have this performance; the system fetch from the stored data all students have the same performance and the teacher can send an email to these students.

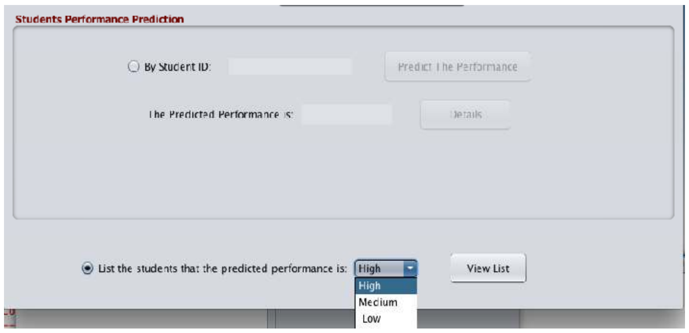

Students Performance Prediction
By Student ID:
Predict The Performance
The Predicted Performance is:
Details
List the students that the predicted performance is:
High
High
Medium
Low
View List

Figure 11: View Student's Performance Prediction List by selecting a certain Performance

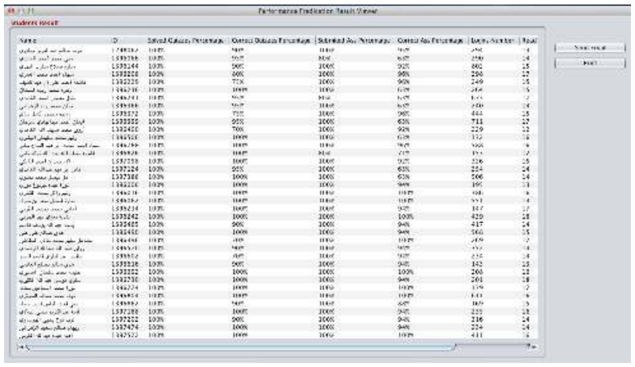


Figure 12: Students' Performance Prediction List

5. EXPERMENTAL RESULTS:

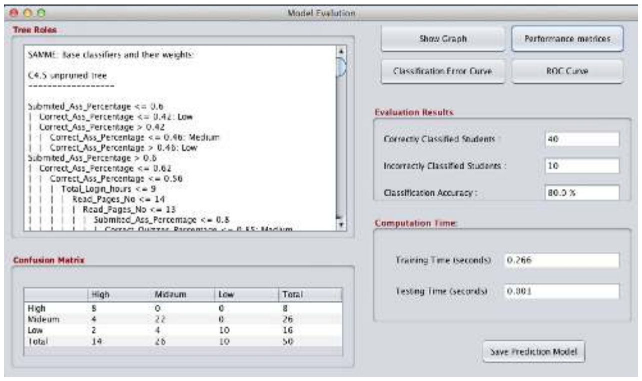
The experiment was conducted using the **EMES** e-learning system dataset. The dataset contains 155 students from two sections of the computer skills (CPIT100) course, KAU University, Jeddah, Saudi Arabia of session 2012-13.

The dataset is divided into two parts: training and testing data, 105 students selected for training data to create the prediction model and the remaining is used for testing data used for model evaluation. The classification algorithm used is C4.5 with SAMME Boosting technique. Also, we used C4.5 as single classifier and Adaboost.M1 and LogitBoost boosting technique to evaluate our system. The results are recorded for comparison purpose.

The experiment is to measure classification accuracy, performance metrics, classification error curve, Roc curve, the computational time to complete the classification tasks (training time for model building and the testing time for model evaluation). Also, we measure the effect of increasing the iterations number on the prediction accuracy. There are no special experiments or standards to measure the multi-agent system efficiency, because our design goal for multi-agent system is only to divide the classification tasks, each agent responsible for a particular set of tasks.

We start building the performance prediction model using SAMME boosting technique with 10 iteration and C4.5 as the base classifier. After the process finished, it returns the results: confusion matrix, performance metrics, correctly classified students, incorrectly classified students, accuracy percentage, Roc curve, classification Error curve, Model Building Time and

Model Testing Time). Figure 13 shows the system results after build and evaluate the prediction model.



Tree Roles

```
SAMME: base classifiers and their weights:

C4.5 unpruned tree
------------------

Submited_Ass_Percentage <= 0.6
|   Correct_Ass_Percentage <= 0.42: Low
|   Correct_Ass_Percentage > 0.42
|   |   Correct_Ass_Percentage <= 0.46: Medium
|   |   Correct_Ass_Percentage > 0.46: Low
Submited_Ass_Percentage > 0.6
|   Correct_Ass_Percentage <= 0.62
|   |   Correct_Ass_Percentage <= 0.56
|   |   |   Total_Login_hours <= 9
|   |   |   |   Read_Pages_No <= 14
|   |   |   |   |   Read_Pages_No <= 13
|   |   |   |   |   |   Submited_Ass_Percentage <= 0.8
|   |   |   |   |   |   |   Correct_Quizzes_Percentage <= 0.55: Medium
```

Confusion Matrix

	High	Midum	Low	Total
High	8	0	0	8
Mideum	4	22	0	26
Low	2	4	10	16
Total	14	26	10	50

Evaluation Results

Correctly Classified Students : 40

Incorrectly Classified Students : 10

Classification Accuracy : 80.0 %

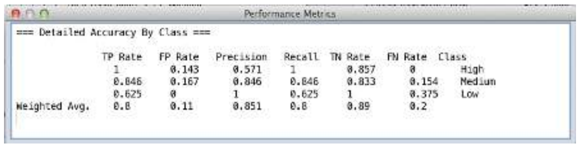
Computation Time:

Training Time (seconds) 0.266

Testing Time (seconds) 0.001

Figure 13: Model Building and Evaluation Results

Figure 14 shows the performance prediction results according to the confusion matrix for each class High, Medium and Low, True Positive Rate (TPR), True Negative Rate (TNR), False Positive (FPR), False Negative Rate (FNR), Precision and Recall.



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

                 TP Rate   FP Rate   Precision   Recall  TN Rate   FN Rate  Class
                   1         0.143     0.571      1        0.857     0          High
                   0.846     0.167     0.846      0.846    0.833     0.154      Medium
                   0.625     0         1          0.625    1         0.375      Low
Weighted Avg.      0.8       0.11      0.851      0.8      0.89      0.2
```

Figure 14: Performance Metrics Results

Classification Error Curve in Figure 15 shows the incorrectly classified students in each iteration.

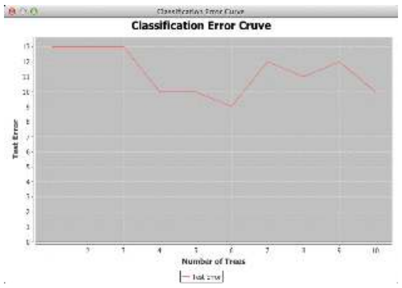


Figure 15: Classification Error Curve

Roc Curve in Figure 16 shows False Positive Rate (FPR) vs. True Positive Rate (TPR) in the High prediction performance.

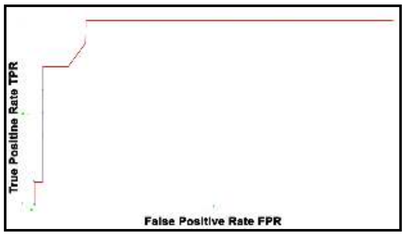


Figure 16: Roc Curve

Performance Comparison Results: Table 2 shows C4.5, SAMME, Adaboost.M1 and LogitBoost comparison results about correctly and incorrectly classified students, prediction accuracy and model building and evaluation computational time. For SAMME, Adaboost.M1 and LogitBoost 10 iterations was performed. SAMME and Adaboost.M1 have the same prediction accuracy result but different only in model building time, Adaboost.M1 take less time than SAMME boosting technique. LogitBoost got lower prediction accuracy and C4.5 build the prediction model faster than others because it is build only one classifier. Table 3 shows the performance prediction results.

Table 2: Prediction Accuracy and Computational Time Results

Algorithm	Correctly classified students	Incorrectly classified students	Prediction Accuracy	Model Building Time (Sec.)	Model Testing Time (Sec.)
SAMME Boosting Technique	40	10	**80 %**	0.266	0.001
Adaboost.M1 Boosting Technique	40	10	**80 %**	0.262	0.001
LogitBoost boosting technique	25	25	50 %	0.132	0.001
C4.5	37	13	74 %	**0.094**	0.001

Table 3: C4.5, SAMME, Adaboost.M1 and LogitBoost Performance metrics results

Algorithm	Class	TPR	FPR	TNR	FNR	Precision	Recall
C4.5 decision tree	**High**	0.875	0.119	0.881	0.125	0.583	0.875
	Medium	0.885	0.333	0.667	0.115	0.742	0.885
	Low	0.438	0	1	0.563	1	0.438
SAMME Boosting Technique	**High**	1	0.143	0.857	0	0.571	1
	Medium	0.846	0.167	0.833	0.154	0.846	0.846
	Low	0.625	0	1	0.375	1	0.625
Adaboost.M1 Boosting Technique	**High**	1	0.143	0.857	0	0.571	1
	Medium	0.846	0.167	0.833	0.154	0.846	0.846
	Low	0.625	0	1	0.375	1	0.625
LogitBoost boosting technique	**High**	0.75	0.31	0.69	0.25	0.316	0.75
	Medium	0.577	0.5	0.5	0.423	0.556	0.577
	Low	0.25	0	1	0.75	1	0.25

Boosting Iterations Effects: we increased the iterations number to 60. Figure 17 shows the Prediction Accuracy Curves for SAMME, Adaboost.M1 and LogitBoost during 60 boosting iterations.

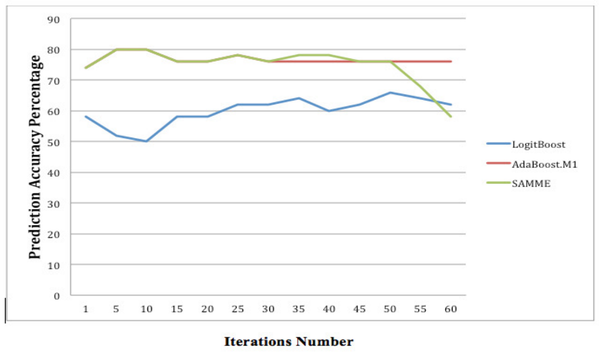


Figure 17: Prediction Accuracy Curves for SAMME, Adaboost.M1 and LogitBoost

- SAMME and Adaboost.M1 outperformed the single classifier C4.5 and the Sub-binary boosting technique “LogitBoost”.
- In Model Building Time, Adaboost.M1 take less time than SAMME boosting technique but the prediction model building in C4.5 is faster than others because it is build only one classifier.
- In SAMME, the highest weight classifier in the prediction model (composite model) is 2.94 but in Adaboost.M1 is equal to 2.25. The highest weight classifier indicates the classifier accuracy.
- When increasing the iterations number, the prediction accuracy of SAMME and Adaboost.M1 was decreased but in LogitBoost the accuracy is increased.
- SAMME and Adaboost.M1 have the same prediction accuracy percentage (80 %) in the 5th and 10th iteration, only in the 35th and 40th iteration, SAMME got the higher accuracy percentage (78 %) and highest weight classifier in the model is 2.94.
- In Adaboost.M1, the maximum number of iterations that were performed is 40 and highest weight classifier in the model is 2.25. Adaboost.M1 stopped in the 40th iteration because the total weight of incorrectly classified students of training data is equal to 0.5.
- When the iterations number increased in LogitBoost, we got a highest accuracy (66 %) in the 50th iteration.

6. CONCLUSIONS

We applied Multi Agent Data Mining Technique in **EMES** e-learning system for students’ performance prediction and optimization. It showed how useful data mining in an educational system in particularly to predict the performance of a student accurately. Particularly, we obtained the prediction model using **SAMME** boosting technique. Also, we used the rules extracted from the prediction model for performance optimization. We examine the effects of ensemble methods in the performance prediction accuracy.

When compare the system with the single classifier C4.5, SAMME outperformed the single classifier **C4.5**. Also, we compare SAMME with Adaboost.M1 and **LogitBoost** boosting technique, **SAMME** and **Adaboost.M1** have the same accuracy percentage, only in the **35**th and **40**th iteration, **SAMME** got the **higher** accuracy percentage. **SAMME** and **Adaboost.M1** outperformed Sub-binary boosting technique “**LogitBoost**”.

REFERENCES

[1] E. Osmanbegović and M. Suljić, 'Data mining approach for predicting student performance', Economic Review, vol 10, iss 1, 2012.

[2] M. Sukanya, S. Biruntha, S. Karthik and T. Kalaikumaran, 'Data Mining: Performance Improvement in Education Sector using Classification and Clustering', in International Conference on Computing and Control Engineering (ICCCE), 2012.

[3] B. Bhardwaj and S. Pal, 'Data Mining: A prediction for performance improvement using classification', International Journal of Computer Science and Information Security (IJCSIS), vol 9, iss 4, 2011.

[4] C. Chen, Y. Chen and C. Liu, 'Learning performance assessment approach using web-based learning portfolios for e-learning systems', IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, vol 37, iss 6, pp. 1349--1359, 2007.

[5] I. Paris, L. Affendey and N. Mustapha, 'Improving academic performance prediction using voting technique in data mining', World Academy of Science, Engineering and Technology, vol 4, pp. 820--823, 2010.

[6] C. Wang, 'New ensemble machine learning method for classification and prediction on gene expression data'. pp.3478--3481, 2006.

[7] Y. Liu, 'Drug Design by Machine Learning: Ensemble Learning for QSAR Modeling', in the Fourth International Conference on Machine Learning and Applications (ICMLA’05), 2005.

[8] M. Alkhattabi, D. Neagu and A. Cullen, 'Assessing information quality of e-learning systems: a web mining approach', Computers in Human Behavior, vol 27, iss 2, pp. 862--873, 2011.

[9] M. Kantardzic, Data mining: Concepts, models, methods, and algorithms, John Wiley & Sons, 2003.

[10] R. Nisbet, J. Elder and G. Miner, Handbook of statistical analysis and data mining applications, 1st ed. Amsterdam: Academic Press/Elsevier, 2009.

[11] E. Rahm and H. Do, 'Data cleaning: Problems and current approaches', IEEE Data Eng. Bull., vol 23, iss 4, pp. 3--13, 2000.

[12] S. Yadav, B. Bharadwaj and S. Pal, 'Data mining applications: A comparative study for predicting student's performance', in International Journal of Innovative Technology & Creative Engineering (IJITCE), vol 1, iss 12, pp. 13--19, 2012.

[13] S. Kumar and M. Vijayalakshmi, 'Mining Of Student Academic Evaluation Records in Higher Education', in International Conference on Recent Advances in Computing and Software Systems (RACSS), pp. 67 – 70, 2012.

[14] J. Quinlan, C4.5: Programs for Machine Learning. San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

[15] G. Agrawal and H. Gupta, 'Optimization of C4.5 Decision Tree Algorithm for Data Mining Application', International Journal of Emerging Technology and Advanced Engineering, vol 3, iss 3, pp. 341-345, 2013.

[16] J. Thongkam, G. Xu and Y. Zhang, 'AdaBoost algorithm with random forests for predicting breast cancer survivability', in International Joint Conference on Neural Networks, pp. 3062--3069, 2008.

[17] S. Shanthi and R.G. Ramani, 'Gender specific classification of road accident patterns through data mining techniques', in IEEE International Conference On Advances In Engineering, Science And Management (ICAESM -2012), pp.359--365, 2012.

[18] A. Tan and D. Gilbert, 'Ensemble machine learning on gene expression data for cancer classification', Applied Bioinformatics, vol 2, iss 3, pp.75--83, 2003.

[19] B. Verma and A. Rahman, 'Cluster-oriented ensemble classifier: impact of multicluster characterization on ensemble classifier learning', IEEE Transactions on Knowledge and Data Engineering, vol 24, iss 4, pp. 605--618, 2012.

[20] Y. Freund and R. Schapire, 'A short introduction to boosting', Journal of Japanese Society For Artificial Intelligence, vol 14, iss 5, pp. 771--780, 1999.

[21] Y. Freund and R. Schapire, 'A decision-theoretic generalization of on-line learning and an application to boosting', Journal of Computer and System Sciences, vol 55, iss 1, pp.119--139, 1997.

[22] Y. Freund and R. Schapire, 'Experiments with a new boosting algorithm', In the 13th International Conference on Machine Learning, pp. 148--156, 1996.

[23] R. Schapire, 'The strength of weak learnability', Machine learning, vol 5, iss 2, pp.197--227, 1990.

[24] A. Fernandez-Baldera and L. Baumela, 'Multi-class boosting with asymmetric binary weak-learners', Pattern Recognition, vol 47, iss 5, pp. 2080--2090, 2014.

[25] J. Friedman, T. Hastie, and R. Tibshirani, 'Additive logistic regression: a statistical view of boosting', The Annals of Statistics, vol 28, iss 2, pp. 337-- 407, 2000.

[26] J. Zhu, H. Zou, S. Rosset and T. Hastie, 'Multi-class adaboost', Statistics and Its Interface, vol 2, pp. 349--360, 2009.

[27] R. Schapire and Y. Singer, 'Improved boosting algorithms using confidence-rated prediction', Machine Learning, vol 37, iss 1, pp. 297--336, 1999.

[28] S. Pramanik, U. Chowdhury, B. Pramanik and N. Huda, 'A comparative study of bagging, boosting and C4.5: The recent improvements in decision tree learning algorithm', Asian J. Inf. Tech, vol 9, iss 6, pp. 300--306, 2010.

[29] Y. Krishnaraj and C. Reddy, 'Boosting methods for protein fold recognition: an empirical comparison', In International Conference on Bioinformatics and Biomedicine, pp. 393--396, 2008.

[30] J. Wu, W. Zhang and R. Jiang, 'Comparative study of ensemble learning approaches in the identification of disease mutations', In 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010), vol 6, pp. 2306--2310, 2010.

[31] W. Iba and P. Langley, 'Induction of one-level decision trees', In the Ninth International Workshop on Machine Learning, pp. 233--240, 1992.

[32] M. Wooldridge, An Introduction to Multiagent Systems, 1st ed. Chichester: John Wiley & Sons, 2002.

[33] K. Albashiri, 'An investigation into the issues of multi-agent data mining', Ph.D, University of Liverpool, 2010.

[34] S. Srinivasan, J. Singh and V. Kumar, 'Multi-agent based decision Support System using Data Mining and Case Based Reasoning', International Journal of Computer Science Issues, vol 8, iss 4, pp. 340--349, 2011.

[35] A. Rathee and R.P. Mathur, 'Survey on Decision Tree Classification algorithms for the Evaluation of Student Performance', International Journal of Computers & Technology, vol 4, iss 2, pp. 244--247, 2013.

[36] B. Minaei-Bidgoli, G. Kortemeyer, and W. F. Punch, 'Enhancing Online Learning Performance: An Application of Data Mining Method', In 7th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2004), 2004.

[37] A. Bakar, Z. Othman, A. Hamdan, R. Yusof and R. Ismail, 'Agent based data classification approach for data mining', In International Symposium on Information Technology, vol 2, pp. 1--6, 2008.

[38] I. Witten, E. Frank and M. Hall, Data Mining: Practical machine learning tools and techniques, 3rd ed. Burlington, Mass.: Morgan Kaufmann Publishers, 2011.