# Investigating the Impact of Various Classification Quality Measures in the Predictive Accuracy of ABC-Miner

Khalid M. Salama
School of Computing
University of Kent
Canterbury, CT2 7NF, UK
Email: kms39@kent.ac.uk

Alex A. Freitas
School of Computing
University of Kent
Canterbury, CT2 7NF, UK
Email: A.A.Freitas@kent.ac.uk

***Abstract*—Learning classifiers from datasets is a central problem in data mining and machine learning research. ABC-Miner is an Ant-based Bayesian Classification algorithm that employs the Ant Colony Optimization (ACO) meta-heuristics to learn the structure of Bayesian Augmented Naïve-Bayes (BAN) Classifiers. One of the most important aspects of the ACO algorithm is the choice of the quality measure used to evaluate a candidate solution to update pheromone. In this paper, we explore the use of various classification quality measures for evaluating the BAN classifiers constructed by the ants. The aim of this investigation is to discover how the use of different evaluation measures affects the quality of the output classifier in terms of predictive accuracy. In our experiments, we use 6 different classification measures on 25 benchmark datasets. We found that the hypothesis that different measures produce different results is acceptable according to the Friedman's statistical test.**

## I. Introduction

Data mining, a research field involving concepts and methods from artificial intelligence, machine learning and statistics, aims at discovering useful patterns in real-world datasets. Classification is one of the widely studied data mining tasks, in which the aim is to learn a model used to predict the class of unlabelled cases [1]. While the literature includes several approaches for tackling this problem, such as decision trees , artificial neural networks and classification rules [2], [1], the Bayesian approach for classification aims to model the (in)dependences relationships between the input domain variables given the class variable in a probabilistic network [3]. Bayesian network (BN) classifiers are used to predict the class of a case by computing the class with the highest posterior probability given the case's predictor attribute values, and learning effective BN classifiers – in terms of predictive accuracy – is our focus in this work.

ABC-Miner [4], recently introduced by the authors, is a classification algorithm that learns the structure of a Bayesian Augmented Naïve-Bayes (BAN) network using Ant Colony Optimization (ACO) – a global-search meta-heuristics for solving combinatorial optimization problems [5]. The Ant-based Bayesian Classification algorithm showed predictive effectiveness compared to other Bayesian classification algorithms, namely: Naïve-Bayes, Tree Augmented Naïve-Bayes (TAN) and General Baysian Network (GBN) [4]. Moreover, experiments also showed that the use of *accuracy* – a classification quality measure – as a quality evaluation measure during the algorithm's training phase is more effective than the use of conventional Bayesian scoring functions.

The motivation behind this work is based on the previous conclusion; since ABC-Miner showed classification effectiveness, we carry on extending the algorithm. In addition, one of the most important aspects of the ACO algorithm is the choice of the quality measure used to evaluate a candidate solution to update pheromone. In this paper, we explore the use of various classification quality measures for evaluating the BN classifiers constructed by the ants in the ABC-Miner algorithm. The aim of this investigation is to discover how the use of different evaluation measures affects the quality of the output classifier in terms of predictive accuracy. In our experiments, we explore the use of 6 different classification measures on 25 UCI repository [6] benchmark datasets.

The rest of the paper is organized as follows. In Section 2 we provide a background on Bayesian Network classifiers. In Section 3 we provide a brief overview on Ant Colony Optimization. We describe our Ant-based Bayesian Classification Algorithm in Section 4. In Section 5 we discuss the concepts of the classification quality measure and its related use as search heuristics, then we exhibit the measures used in the current work. Our experimental setup is described in Section 6, followed by the computational results in Section 7. Finally, we conclude with some general remarks and directions for future research in Section 8.

## II. Bayesian Networks Background

In the context of reasoning with uncertainty, Bayesian networks (BN) is a popular and statistically-sound type of method for modelling (in)dependence relationships between variables in a given data set[7]. A BN is a type of graphical model in the form of a directed acyclic graph (DAG), where nodes represent the variables (also called attributes or features) and edges represent probabilistic dependences between the

variables. Each node (variable) in a BN is associated with a conditional probability table, which specifies the probability for each value of that variable given each configuration of values of its parents in the network. Note that a Bayesian network is built to answer probabilistic queries about any node(s) in the network.

Bayesian network classifiers are a specific type of probabilistic graphical model which is also represented in the form of a DAG, but, unlike BNs, Bayesian network classifiers are built to answer probabilistic queries about just one specific node: the class attribute. Hence, the class node has a special role in a Bayesian network classifier, and it is set as the parent of all other variables. A Bayesian network classifier is used to calculate the probability of each value (label) $c$ of the class variable $C$ given a case $\mathbf{x}$ (an instance of the input attributes $\mathbf{X} = \{X_1, X_2, ..., X_n\}$), and then assign to that case the class label with the highest probability, as in the following formulas:

$$C(\mathbf{x}) = \arg\max_{\forall c \in C} P(C = c | \mathbf{x} = x_1, x_2, ..., x_n), \quad (1)$$

letting $\mathbf{Pa}(X_i)$ be the set of parent predictor variables of $X_i$ in the network, according to the Bayes' Theorem:

$$\overbrace{P(c|x_1, x_2, ..., x_n)}^{\text{posterior probability}} \alpha \overbrace{P(c)}^{\text{prior probability}} \prod_{i=1}^{n} \overbrace{P(x_i|\mathbf{Pa}(X_i), c)}^{\text{likelihood}} \quad (2)$$

Broadly speaking, several types of BN classifiers have been proposed in the literature, varying from the simple Naïve-Bayes, to increasingly more complex types of Bayesian network classifiers like TAN, BAN, and GBN. Naïve-Bayes makes the very strong (and unrealistic) assumption that attributes are independent from each other given the class; which, in terms of a graphical model, means that each variable has a single parent node in the network: the class variable.

The aforementioned more complex types of BN classifiers mitigate Naïve-Bayes' limitation by allowing more complex types of variable dependence to be represented in the network [8], [9]. Hence, in a TAN structure, dependences between predictor attributes can be represented as a tree where each predictor attribute can have another predictor attribute as its parent, in addition to the class variable that is the parent of all attributes. In a BAN structure, dependences between predictor attributes can be represented by any DAG or by a DAG where a predictor attribute can have at most $k$-parents, again in addition to the class variable that is the parent of all attributes.

In contrast to the previous types of BN classifiers, in a GBN structure, the class variable can be either a parent or a child of any predictor attribute. In this case, the system initially builds a general-purpose BN (treating the class variable as any other node), and then it extracts the *Markov blanket* of the class node and use the set of nodes and edges in that Markov blanket as a Bayesian network classifier. For a review and comparison of various BN classifiers, see [8], [9].

The problem of learning a BN classifier from a given dataset $\mathbf{D}$ consists of two phases, namely learning the structure and the parameters of the network. Parameter learning is, by comparison, the simpler phase. In this phase, the conditional probability table (CPT) of each variable $X_i$ is often constructed by estimating, from the data, the likelihood of each value of that variable given each combination of values of its parent variables $\mathbf{Pa}(X_i)$ in the network structure $G$. The likelihood of the dataset $\mathbf{D}$ given a network $G$ is denoted by $P(\mathbf{D}|G)$. In the structure learning phase, one typically wants to find the network $G$ that maximizes $P(\mathbf{D}|G)$ for a given $\mathbf{D}$, which is the role of BN structure learning. A popular approach to this problem, in data mining and machine learning, is to specify a network scoring function, $f$, that evaluates the quality of each $G$ with respect to $\mathbf{D}$. Then one uses a search method to try to find the network structure with the best possible value of that scoring function, in the space of candidate network structures [8].

Most BN-classifier structure learning algorithms proposed in the literature are deterministic and greedy – adding or removing just one edge at a time to the current network and then evaluating the new network. Such greedy methods are likely to get trapped into local optima in the search space. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-complete, using a search method that guarantees to find the optimal network (based on exhaustive search) would be practical only for simple and small problems. Hence, in the case of large and complex problems, heuristic methods are needed to try to find a near-optimal solution in an acceptable time. In this context, stochastic meta-heuristic global search methods like ACO, which are less likely to get trapped into local optima, are worth exploring as a search method to build Bayesian network classifiers.

## III. Ant Colony Optimization

Inspired by the behaviour of natural ant colonies, Dorigo et al. [5], [10] have defined Ant Colony Optimization (ACO) as a meta-heuristic that can be applied to solve optimization problems, and ACO has been widely used in combinatorial optimization problems. The basic principle of ACO is that a population of artificial ants cooperates with each other to find the best path in a graph, representing a candidate solution to the target problem, analogously to the way that natural ants cooperate to find the shortest path between two points like their nest and a food source.

In ACO each artificial ant constructs a candidate solution to the target problem, represented by a path in a "construction graph". Ants cooperate via indirect communication, by depositing artificial pheromone (some type of information) on the nodes or edges of the construction graph. More precisely, an ant deposits pheromone on the nodes or edges of the path corresponding to its constructed solution, and the amount of pheromone deposited is proportional to the quality of that solution. The larger the amount of pheromone on a node or edge, the larger the probability that other ants will decide to visit that node or edge when constructing their solution, incorporating an aspect of global search into an ACO algorithm.

In addition, the probability of an ant choosing a node or edge also depends on the value of a heuristic function

associated with that node or edge that evaluates the desirability of nodes or edges in a local fashion. In order to design an ACO algorithm to solve a specific type of problem, the following components of the algorithm should be designed:

**Construction Graph** - This graph defines the search space to be explored by the ACO algorithm. Each ant incrementally constructs a candidate by visiting nodes and edges in the graph, corresponding to different combinations of components of a candidate solution.

**Heuristic Function** - This is a function that uses some problem-specific information to evaluate the quality of each candidate-solution component available in the construction graph. The value of this function influences an ant's choice of which components will be used for constructing its candidate solution.

**State Transition Rule** - This is a probabilistic transition rule that determines how each ant decides which search state (corresponding to a node or edge in the construction graph) will be visited next, in order to continue the construction of its candidate solution. This rule is based on both the heuristic function value $\eta$ and the pheromone amount $\tau$ associated with the candidate-solution components.

**Quality Evaluation Measure** - This is a problem-specific function used to evaluate the quality of a candidate solution constructed by an ant. The higher the quality of a solution constructed by an ant, the more pheromone will be deposited on the construction graph components used in that solution, which will encourage other ants to select those components in future iterations of the ACO algorithm.

**Pheromone Update Strategy** - This involves formulas to use for pheromone reinforcement and evaporation. Pheromone reinforcement is applied on construction graph components occurring in the constructed solution in proportion to that solution's quality, while pheromone evaporation is applied on the whole construction graph to avoid stagnation and premature convergence.

**Local Search** - An optional procedure to improve the quality of a constructed solution. This can be performed on each constructed candidate solution, or just on the best solution among the colony to reduce computational time.

ACO algorithms have been successful in solving several combinatorial optimization problems, including classification rule discovery [11], [12], [13], [14], [15] and general purpose BN construction [16], [17], [18]. However, ABC-Miner [4], recently introduced by the authors, is the first ACO algorithm to learn BN classifiers. This algorithm is briefly described in the next section, in order to make the paper more self-contained.

## IV. The ABC-Miner Algorithm

ABC-Miner is an ACO algorithm that learns a BN classifier by searching for the best possible Structure of a Bayesian network Augmented Naive Bayes (BAN) having at most $k$-dependences (parents) at each variable node [4]. Algorithm 1 outlines the ABC-Miner procedures.

**Algorithm 1** Pseudo-code of ABC-Miner.

```
Begin
BNC(gbest) = φ;
t = 1;
InitializePheromoneAmounts();
InitializeHeuristicValues();
repeat
    BNC(tbest) = φ;
    Q(tbest) = 0;
    for i = 1 → colony_size do
        BNC(i) = CreateSolution(ant(i));
        Q(i) = ComputeQuality(BNC(i));
        if Q(i) > Q(tbest) then
            BNC(tbest) = BNC(i);
            Q(tbest) = Q(i);
        end if
    end for
    PerformLocalSearch(BNC(tbest));
    UpdatePheromone();
    if Q(tbest) > Q(gbest) then
        BNC(gbest) = BNC(tbest);
        Q(gbest) = Q(tbest);
    end if
    t = t + 1;
until t = max_iterations or Convergence()
return BNC(gbest);
End
```

The construction graph consists of all the edges of the form $X \rightarrow Y$ where $X \neq Y$ and $X, Y$ belong to the set of predictor attributes in the dataset. These edges represent probabilistic attribute dependences in a BN classifier.

In essence, at each iteration, each ant incrementally constructs a candidate solution (i.e., a BN classifier). Then the quality of each candidate BN classifier is measured. The best solution produced in the colony at the current iteration undergoes local search, and then the BN classifier resulting from that local search is used to update the pheromone in the construction graph path corresponding to that classifier. After that, the system compares the quality of the current iteration's best solution $Q(tbest)$ with the quality of the global best solution $Q(gbest)$, in order to keep track of the best solution found along the entire search so far. This is repeated until the algorithm converges, or the predefined maximum number of iterations is reached.

Algorithm 2 shows the construction of a candidate BN classifier. An ant starts by considering a very simple BN classifier structure, namely a Naïve-Bayes structure, where each variable has just one parent, namely the class variable. Then the ant incrementally builds a more complex network, in the form of a Bayesian Augmented Naïve-Bayes (BAN) structure, by adding one edge at a time to the current network structure.

The selection of the edge to be added at each step is based on both the heuristic value (computed using conditional mutual

**Algorithm 2** Pseudo-code of Solution Creation Procedure.

**Begin** CreateSolution()
$BNC \leftarrow$ {Naïve-Bayes structure};
$k = ant.SelectMaxParents()$;
**while** $GetValidEdges() <> \phi$ **do**
    $\{i \rightarrow j\} = ant.SelectEdgeProbablistically()$;
    $BNC = BNC \cup \{i \rightarrow j\}$;
    $RemoveInvalidEdges(BNC, k)$;
**end while**
$BNC.LearnParameters()$;
**return** $BNC$;
**End**

information [4]) and the pheromone amount associated with each valid candidate edge that could be added at this step, using the following probabilistic state transition formula:

$$P_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_a \sum_b [\tau_{ab}(t)]^{\alpha} \cdot [\eta_{ab}]^{\beta}}, \quad (3)$$

where $P_{ij}$ is the probability of selecting the edge $i \rightarrow j$, $\tau_{ij}(t)$ is the amount of pheromone associated with edge $i \rightarrow j$ at iteration $t$ and $\eta_{ij}$ is the heuristic information for edge $i \rightarrow j$. The exponents $\alpha$ and $\beta$ are used to adjust the relative importance of the pheromone ($\tau$) and heuristic information ($\eta$), respectively, and are set using the "ants with personality" approach [14]. An edge is valid to be added to the current partial BN classifier if the inclusion of that edge in the classifier does not create a directed cycle and does not exceed the upper limit of $k$ parents for the current node (a limit chosen by the current ant). Once an edge is added to the current partial BN classifier, all the invalid edges are eliminated from the construction graph available for that ant. This process is repeated until no valid edges are available for that ant.

When the BN structure constructed by an ant is finished, the CPT (Conditional Probability Table) is computed for each variable, producing a complete BN classifier. Then the quality of the solution is evaluated and all the edges become available for constructing further candidate solutions.

The ABC-Miner algorithm evaluates the quality of the candidate constructed BN classifier during the training phase using *accuracy* [4], a conventional predictive measure, since the goal is to build a BN only for predicting the value of a specific class attribute, unlike conventional BN learning algorithms whose scoring function does not distinguish between the predictor and the class attributes.

In [4] it is reported that the ABC-Miner algorithm has obtained good predictive accuracy compared to other well-known conventional Bayesian classification algorithms, namely: Naïve-Bayes, TAN and GBN. Thus, it seems worth to further investigate the effectiveness of new variations of that algorithm. One of the most important aspects of an ACO algorithm for classification is the choice of the quality measure used to evaluate a candidate classifier and so for pheromone updating. In this context, we explore the use of different classification quality measures – besides `accuracy`, which was used originally by the algorithm – to investigate how the use of different evaluation measures affects the quality of the output BN classifier.

## V. Classifier Quality Measures

Measuring the predictive performance of a classification model (classifier) is based on the counts of the cases (validation cases in the training phase and test cases in the test phase) correctly and incorrectly predicted by the classifier. These counts are organized in a tabular structure known as a *confusion matrix*, as represented in Table I. The entries of the confusion matrix are defined as follows:

TABLE I
Confusion Matrix

| | | Predicted Class | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Class | Positive | **TP** | **FN** |
| | Negative | **FP** | **TN** |

**TP** The count of cases that belong to the positive class and are predicted as positive (true positives).
**FP** The count of cases that belong to the negative class and are predicted as positive (false positives).
**TN** The count of cases that belong to the negative class and are predicted as negative (true negatives).
**FN** The count of cases that belong to the positive class and are predicted as negative (false negatives).
**SM** The total count of cases (TP + FP + TN + FN).

For binary classification problems, where the class variable has exactly two values, only one confusion matrix is computed. However, in multi-class problems, where the class variable has more than two values, several matrices are computed, one for each class value considered as the positive class, with all the other classes grouped together to form the negative class.

One common approach for calculating the overall classifier quality from several confusion matrices is to calculate the quality on each class using a specific measure with each confusion matrix separately, and take the average of the qualities calculated across all the classes. Such an averaging approach is robust against the class imbalance problem, where in some datasets a class value has a dominant count of cases or where some class values have a much fewer count of cases. We employ this approach in our experiments.

Various classification quality evaluation measures are formulated using these elements of the confusion matrix, with different biases and quality aspects' importance. Several works aimed to study the effectiveness of these measures, yet in different classification contexts such as classification rule induction [19], [20], [21], [22], which highlighted the importance of rule quality measure chosen to be used to guide the search. We explore the effect of these various classifications quality evaluation measures in guiding the ACO search to construct effective Bayesian network classifiers.

The following presents the 6 quality measures explored in our work.

`Accuracy` (Equation 4) - The baseline measure which is used in the original ABC-Miner [4] to evaluate the candidate constructed Bayesian network classifiers. Accuracy measures the ratio of the count of correctly classified cases (TP + TN) over the sum of counts of all the cases.

$$Accuracy = \frac{TP + TN}{SM} \tag{4}$$

`F-measure` (Equation 6) - Widely used in the context of information retrieval and text classification systems. It calculates a harmonic mean between precision and recall.

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F - measure = \frac{2.Precision.Recall}{Precision + Recall} \tag{6}$$

`Sensitivity × Specificity` (Equation 7) - Used in the Ant-Miner [13] and in the $c$Ant-Miner$_{PB}$ [20] classification rule discovery algorithms . Sensitivity measures the ratio of the count of true positives to the count of all the positive cases, and the specificity measures the ratio of the count of true negatives to the count of all the negative cases.

$$Sensitivity \times Specificity = \frac{TP}{TP + FN}.\frac{TN}{TN + FP} \tag{7}$$

`Jaccard Coefficient` (Equation 8) - It calculates the similarity between sample sets. In the classification context, it measures the accuracy only with respect to the true positive count, neglecting the true negatives. This measure has been used in [20], [19] as a classification rule quality function, where it was one of the best performing functions.

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{8}$$

`M-estimate` (Equation 9) - a parametric function with the parameter $m$, where was set to 22.4 as found to be the best value by Janssen and Fürnkranz in [22].

$$M - estimate = \frac{TP + m.\frac{TP}{SM}}{TP + FP + m} \tag{9}$$

`Klösgen` (Equation 10) - When $\omega$ = 0, the measure acts as the precision (Equation 5). As $\omega$ increases the measure acts in a manner similar to recall (Equation 5). The transition from one to the other is not linear, and as $\omega$ increases further it starts acting as coverage. An $\omega$ value of 0.43 was found to be optimum by Janssen and Fürnkranz in [22] and is used in our experiments.

$$Klosgen = \left(\frac{TP + FP}{SM}\right)^{\omega} . \left(\frac{TP}{TP + FP} - \frac{TP + FN}{SM}\right) \tag{10}$$

It is crucial to emphasize that there are two different quality evaluation operations. The first is the one used on the testing phase to evaluate the classification quality of the final output classifier on a test set *unseen* during training, which is fixed to evaluate several algorithms or several variations of an algorithm. The second is the one used during the training phase for evaluating each candidate classifier constructed by an ant on a validation set (which is basically the training set).

Candidate solutions are evaluated during the training phase to perform pheromone update for guiding the search to build a high quality classifier as a final output. In the experiments, the former is fixed to `accuracy` (Equation 4) in the testing phase, while the latter used the various aforementioned measures in the training phase to examine their effectiveness with respect to maximizing the `accuracy` test measure. The next section describes the experimental methodology in detail.

## VI. Experimental Setup

In our experiments, we modified ABC-Miner in two ways. First, we remove the heuristic component from the probabilistic state transition formula (Equation 3). The reason behind this is to isolate the effect of the quality measure and to make sure that only the evaluation function will affect the ants' decisions and guide the search through pheromone amounts deposited. Second, we only perform the local search on the final converged-on solution $BNC(gbest)$ (i.e. BAN classifier structure) rather than performing local search on each iteration's best solution $BNC(tbest)$, in order to reduce the algorithm's computational time.

The performance of classification quality measures was evaluated using 25 public-domain datasets from the well-known UCI (University of California at Irvine) dataset repository [6]. Datasets containing continuous attributes were discretized in a pre-possessing step, using C4.5-Disc [1]. The main characteristics of the datasets are shown in Table III, and the parameter settings are shown in Table II, which are the same default values used in [4].

TABLE II
Parameter Settings Used in Experiments

| Parameter | Value |
|---|---|
| `max_iterations` | 100 |
| `colony_size` | 10 |
| `conv_ iterations` | 10 |
| `max_parents` | 3 |

The experiments were carried out using a well-known *stratified* 10-fold cross-validation procedure, which works as follows. First, the target dataset is divided into 10 mutually exclusive partitions (or folds), with approximately the same number of cases in each partition. Then, for each of the 6 different quality measures used for candidate-solution evaluation and pheromone updating in this work, a version of ABC-Miner using that measure is run 10 times, where each time a different partition is used as the test set and the other 9 partitions are merged and used as the training set.

The predictive performance associated with each quality measure is computed as the average value of the `accuracy`

TABLE III
DATASETS USED IN THE EXPERIMENTS

| Dataset | Cases | Attributes | Classes |
|---|---|---|---|
| balance scale | 625 | 4 | 3 |
| breast cancer (wisconsin) | 286 | 9 | 2 |
| car evaluation | 1,728 | 6 | 4 |
| chess (rook vs. pawn) | 3,196 | 36 | 2 |
| contraceptive method choice | 1,473 | 9 | 3 |
| statlog credit (australian) | 690 | 14 | 2 |
| statlog credit (german) | 1,000 | 20 | 2 |
| dermatology | 366 | 33 | 6 |
| ecoli | 336 | 8 | 8 |
| glass | 214 | 10 | 7 |
| hayes-roth | 160 | 4 | 3 |
| heart (cleveland) | 303 | 12 | 3 |
| heart (statlog) | 270 | 13 | 2 |
| ionosphere | 351 | 34 | 2 |
| iris | 150 | 4 | 3 |
| monks | 432 | 6 | 2 |
| nursery | 12,960 | 8 | 5 |
| page Blocks | 5,473 | 10 | 5 |
| post-operative patient | 90 | 8 | 3 |
| soybean | 307 | 35 | 19 |
| spect heart | 267 | 22 | 2 |
| tic-tac-to | 958 | 9 | 2 |
| voting records | 435 | 16 | 2 |
| wine | 178 | 13 | 3 |
| yeast | 1,484 | 8 | 10 |

on the test set across the 10 runs. In addition, we ran ABC-Miner with each of the 6 quality measures 10 times – using a different random seed to initialize the search each time – for each cross-validation fold. So, the `accuracy` associated with each quality measure is actually averaged over 100 values (10 cross-validation folds times 10 runs per fold).

## VII. COMPUTATIONAL RESULTS

Table IV reports the mean and the standard error $(mean \pm standard\ error)$ of predictive accuracy values obtained by 10-fold cross validation for the 25 datasets, where the highest accuracy for each dataset is underlined and shown in bold face. The last row shows the average rank of each quality measure in terms of predictive accuracy. The average rank for a given quality measure $m$ is obtained by first computing the rank of $m$ on each dataset individually. The individual ranks are then averaged across all datasets to obtain the overall average rank. Note that the lower the value of the rank, the better the quality measure.

As shown, `Klösgen` achieved the highest predictive accuracy amongst all quality measures in 13 datasets, while `M-estimate` achieved the highest accuracy in 5 datasets, `Jaccard` and Sensitivity $\times$ Specificity each achieved the highest accuracy in 3 datasets, `accuracy` in 2 datasets and `F-measure` in none.

According to the overall rankings of the quality measures used in our experiments, 4 of them outperformed the `accuracy` baseline quality measure that has been originally used in the ABC-Miner algorithm [4]. `Klösgen` obtained the best overall averaging raking with a value of **1.82**, followed by Sensitivity $\times$ Specificity that obtained an overall averaging ranking with a value of **3.54**. `M-estimate` and `Jaccard` came in the third and the fourth places respectively, with overall average ranking values **3.62** and **3.68** respectively. `accuracy`, the baseline measure, obtained the fifth place with overall average ranking with a value of **3.96**. `F-measure` obtained the worst rank with a value of **4.38**.

The non-parametric Friedman statistical test [23] with the Holm's post-hoc test was applied to the average rankings (last row in Table IV). The test showed that `Klösgen`, the best performing measure, is statistically better than all of the other 5 measures used in our experiments with a significance level of **5%**.

TABLE IV
PREDICTIVE ACCURACY % ($mean \pm standard\,error$) RESULTS.

| Dataset | accuracy | F-measure | Sens×Spec | Jaccard | M-estimate | Klösgen |
|---|---|---|---|---|---|---|
| bal | **77.48 ± 0.9** | 77.35 ± 0.9 | 77.32 ± 0.8 | 77.32 ± 0.9 | 77.32 ± 0.8 | 77.32 ± 0.9 |
| bcw | 95.45 ± 0.9 | 95.46 ± 0.9 | 95.46 ± 0.9 | 95.13 ± 1.2 | 95.54 ± 0.9 | **95.76 ± 0.8** |
| car | 98.15 ± 0.3 | 97.42 ± 0.5 | 98.90 ± 0.3 | 98.58 ± 0.3 | 97.64 ± 0.5 | **99.48 ± 0.5** |
| chess | 93.65 ± 0.9 | 93.48 ± 0.9 | 94.62 ± 0.9 | 93.09 ± 1.2 | 93.08 ± 0.5 | **95.21 ± 0.5** |
| cmc | 61.86 ± 0.6 | 59.95 ± 0.5 | 61.76 ± 0.5 | 62.13 ± 0.6 | 62.20 ± 0.5 | **64.48 ± 0.6** |
| crd-a | 87.42 ± 0.9 | 87.75 ± 0.6 | 87.90 ± 0.6 | 87.21 ± 0.9 | 87.45 ± 0.9 | **88.08 ± 0.9** |
| crd-g | 80.57 ± 0.9 | 79.90 ± 1.2 | 79.11 ± 0.9 | **82.97 ± 1.2** | 79.13 ± 0.9 | 81.51 ± 0.9 |
| drm | 98.95 ± 0.9 | 98.96 ± 0.9 | 99.02 ± 0.6 | 98.82 ± 0.9 | 99.02 ± 0.5 | **99.22 ± 0.6** |
| ecoli | 88.40 ± 0.8 | 88.34 ± 0.8 | 87.64 ± 0.8 | 88.28 ± 0.6 | **88.47 ± 0.6** | 87.89 ± 0.5 |
| glass | 82.65 ± 1.2 | 82.65 ± 1.2 | 82.55 ± 1.2 | **82.94 ± 1.4** | 82.55 ± 1.2 | 82.65 ± 0.9 |
| hay | 87.62 ± 3.1 | 87.41 ± 2.8 | 88.51 ± 2.4 | 87.41 ± 2.8 | **88.62 ± 2.8** | 88.60 ± 3.1 |
| hrt-c | 82.07 ± 2.8 | 79.96 ± 3.1 | 80.31 ± 2.8 | 82.96 ± 2.4 | 81.33 ± 2.8 | **84.12 ± 2.4** |
| hrt-s | 91.22 ± 1.8 | 91.88 ± 1.4 | **92.27 ± 1.2** | 92.01 ± 2.2 | 91.35 ± 1.8 | 92.14 ± 1.8 |
| iono | 95.32 ± 0.3 | 95.18 ± 0.5 | 95.32 ± 0.3 | 95.40 ± 0.3 | 95.25 ± 0.5 | **95.61 ± 0.3** |
| iris | **96.11 ± 0.6** | 95.89 ± 0.6 | 95.89 ± 0.6 | 95.89 ± 0.5 | 95.64 ± 0.8 | 95.89 ± 0.6 |
| mnk | 61.90 ± 0.9 | 61.46 ± 0.9 | 62.34 ± 1.2 | 62.52 ± 1.2 | 62.27 ± 0.9 | **63.65 ± 0.9** |
| nurs | 97.13 ± 0.9 | 96.88 ± 0.9 | 98.22 ± 0.9 | 97.53 ± 0.8 | 96.81 ± 0.8 | **98.77 ± 0.8** |
| pb | 98.58 ± 0.9 | 98.63 ± 0.8 | 98.00 ± 0.9 | 98.30 ± 0.8 | **98.74 ± 0.8** | 98.63 ± 0.8 |
| pop | 75.79 ± 0.9 | 78.20 ± 0.9 | **77.96 ± 0.9** | 77.24 ± 0.9 | **77.96 ± 0.9** | 78.93 ± 0.8 |
| soy | 58.02 ± 1.2 | 57.89 ± 1.2 | 59.66 ± 1.4 | 58.94 ± 0.8 | 58.80 ± 1.2 | **60.64 ± 1.2** |
| spect | 89.51 ± 1.2 | 89.50 ± 1.8 | 88.58 ± 1.4 | 89.50 ± 1.4 | 89.97 ± 1.2 | **90.90 ± 1.2** |
| ttt | 79.82 ± 0.3 | 71.29 ± 0.3 | 83.19 ± 0.6 | 84.58 ± 0.5 | 84.72 ± 0.3 | **86.10 ± 0.5** |
| vot | 96.67 ± 1.2 | 96.38 ± 0.8 | 96.47 ± 0.8 | **96.87 ± 0.8** | 97.38 ± 1.2 | 97.17 ± 0.8 |
| wine | 97.22 ± 1.6 | 97.51 ± 1.6 | **98.42 ± 1.2** | 97.50 ± 1.2 | 98.21 ± 0.9 | 98.15 ± 1.2 |
| yst | 62.69 ± 1.2 | 62.71 ± 1.2 | 63.59 ± 1.4 | 63.16 ± 1.2 | **63.62 ± 0.8** | 63.54 ± 1.2 |
| **Average Rank** | **3.96** | **4.38** | **3.54** | **3.68** | **3.62** | **1.82** |

## VIII. CONCLUSION

ABC-Miner is a recently introduced algorithm that employs the ACO meta-heuristics to construct Bayesian network classifiers with the structure of a BAN. This paper has explored the effect of using 6 different classification quality measures for evaluating the candidate BN classifiers constructed by the ants and updating pheromone during the training phase of ABC-Miner. The effect of using these quality measures in the training phase was assessed according to their effectiveness in producing classifiers with a high predictive performance in the testing phase, using `accuracy` as a fixed predictive performance evaluator.

Empirical evaluation on 25 UCI datasets has shown that the performance of different quality measures varies substantially across different datasets. However, the `Klösgen` measure has obtained the best overall average predictive performance, outperforming the other measures at a statistical significance level of 5%. One possible research direction is to try to combine the use of several quality measures in the same learning procedure, which is left to future work.

## REFERENCES


[1] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2010.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2000.

[3] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, "Bayesian Network Classifiers," *Machine Learning*, pp. 131–163, 1997.

[4] K. M. Salama and A. A. Freitas, "ABC-Miner: an Ant-based Bayesian Classification Algorithm," *International Conference on Swarm Intelligence (ANTS)*, pp. 2677–2694, 2012.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*. The MIT Press, 2004.

[6] UCI Repository of Machine Learning Databases. Retrieved Oct 2011 from, URL:www.ics.uci.edu/ mlearn/MLRepository.html.

[7] R. Daly, Q. Shen, and S. Aitken, "Learning bayesian networks: Approaches and issues," *Knowledge Engineering Reviews*, vol. 26, no. 2, pp. 99–157, 2011.

[8] J. Cheng and R. Greiner, "Learning bayesian belief network classifiers: Algorithms and system," *14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pp. 141–151, 2001.

[9] L. Jiang, D. Wang, Z. Cai, and X. Yan, "Survey of improving naïve-bayes for classification," *3rd International Conference on Advanced Data Mining and Applications (ADMA)*, pp. 134–145, 2007.

[10] M. Dorigo and T. Stützle, *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*, ser. International Series in

Operations Research & Management Science. Springer New York, 2003, vol. 57.

[11] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization." *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 651–665, 2007.

[12] D. Martens, B. Baesens, and T. Fawcett, "Editorial survey: swarm intelligence for data mining," *Machine Learning*, vol. 82, no. 1, pp. 1–42, 2011.

[13] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 321–332, 2002.

[14] K. M. Salama, A. Abdelbar, and A. A. Freitas, "Multiple Pheromone Types and Other Extensions to the Ant-Miner Classification Rule Discovery Algorithm," *Swarm Intelligence*, vol. 5, no. 3–4, pp. 149–182, 2011.

[15] K. M. Salama, A. M. Abdelbar, F. E. Otero, and A. A. Freitas, "Utilizing Multiple Pheromones in an Ant-based Algorithm for Continuous-Attribute Classification Rule Discovery." *Applied Soft Computing*, vol. 13, no. 1, pp. 667–675, 2012.

[16] L. M. de Campos, J. M. Fernandez-Luna, J. A. Gamez, and J. M. Puerta, "Ant colony optimization for learning Bayesian networks," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 291–311, 2002.

[17] P. C. Pinto, A. Nägele, M. Dejori, T. A. Runkler, and Ao, "Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 767–779, 2009.

[18] Y. Wu, J. McCall, and D. Corne, "Two novel Ant Colony Optimization approaches for Bayesian network structure learning," *IEEE International Congress on Evolutionary Computation (CEC 2010)*, pp. 1–7, 2010.

[19] K. M. Salama and A. M. Abdelbar, "Exploring Different Rule Quality Evaluation Functions in ACO-based Classification Algorithms," *IEEE Swarm Intelligence Symposium*, pp. 1–8, 2011.

[20] M. Medland and F. Otero, "A Study of Different Quality Evaluation Functions in the cAnt-Miner(PB) Classification Algorithm," *4ht International Conference on Genetic and Evolutionary Computation Conference (GECCO 2012)*, pp. 49–56, 2012.

[21] J. Furnkranz and P. Flach, "ROC 'n Rule Learning - Towards a Better Understanding of Covering Algorithms," *Machine Learning,*, pp. 39–77, 2005.

[22] F. Janssen and J. Furnkranz., "On the quest for optimal rule learning heuristics," *Machine Learning*, pp. 343–379, 2010.

[23] S. Garca and F. Herrera, "An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.