International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184 Volume 4, Number 4 (August 2015), pp.162-178

http://www.meacse.org/ijcar

MMS: Minimum Maximum Strategy for Classification and Testing

Mohammed Issam Younis
Computer Engineering Department, College of Engineering, University of Baghdad
Jadryah, Baghdad, Iraq

Abstract

This paper reviews the state-of-the-art and the art-of-the-practice of the classification machine learning algorithms. In addition, this paper proposes a novel input-output relation classification and testing strategy called Minimum Maximum Strategy (MMS). Internally, MMS derives the classification rules based on minimum-maximum values of attributes for each class till all entries in a data set are covered at least one. In doing so, MMS achieves 100% classification accuracy as well as mining the data set which facilitate building the classification model. Moreover, unlike other existing algorithm MMS generates instances for testing based on the boundary value analysis. As a proof of concept, MMS is used to build a classifier and test instances for the famous IRIS data set. Encouraging results are obtained from experimentations on the accuracy against well-known classification algorithms as well as the effectiveness of the test data generated by the MMS. Finally, it should be mentioned that all experiments are done using the WEKA machine learning tool.

Keywords: classification; Input-output relation; machine learning; data mining; boundary value analysis; evaluation metrics; evaluation matrix; learning; testing.

1. Introduction

Building an accurate and efficient classifiers for a data set is one of the active tasks of data mining and machine learning research. Usually, classification is a preliminary data analysis step for examining a set of cases (instances) to see if they can be grouped based on similarity to each other [1]. As such, given a classification and a partial observation, a statistical estimate of the unobserved attribute values and as the departure point for constructing new models, based on user's domain knowledge [1, 2]. The problem of classification is considered as NP-Complete problem (i.e., there is no unique solution). In addition, the prediction accuracy is considered as NP-Hard problem (i.e., there is no unique method that gives optimal results as far as the accuracy is concerned) [3-7]. For these reasons, many different types of classification techniques have been proposed, studied, and well explained in the literature. The classification algorithms can be classified according to their working methods into five categories: Rules, Bayesian, Decision Tree, Lazy, and Functions. In addition, a hybrid integration of these algorithms is also proposed [8-10].

The rule based classifier (e.g., Decision Table (DT) [11, 12], Decision Table/ Naive Bayes hybrid (DTNB) [12], JRip[13, 14], Fuzzy Unordered Rule Induction Algorithm (FURIA) [15], Part [16], Conjunctive Rule (CR)[17], ZeroR , OneR, Ordinal Learning Method (OLM) , Non-Nested Generalized Exemplars (NNGE), and Ripple-Down Rule learner (RIDOR) [5, 9]) uses a single attribute as the basis for its decisions and chooses the one that works best. Another simple technique is to use all attributes and allow them to make contributions to the decision that are equally important and independent of one another , the classification decision is based on the probability of statistical occurrence [9, 10].

Bayesian classifiers (e.g., NaiveBayes, Averaged N-Dependence Estimators (ANDE), and BayesNet) are a family of probabilistic classifiers based on applying Bayes' theorem with density estimators [18, 19].

NaiveBayes is a simple classifier that uses the normal distribution to model numeric attributes. NaiveBayes can use kernel estimation for improving the accuracy. BayesNet learns Bayesian nets by a learning algorithm for estimating the conditional probability tables of the network. Internally, the search is done using a selected algorithm among (K2, TAN, hill-climbing (HC), repeated hill-climbing(RHC), simulated annealing (SA), tabu search (TS), and genetic search (GS)) algorithms. The search algorithm can be set to do local or global optimization [8-10]. Averaged N-Dependence Estimators (e.g., A1DE and A2DE) achieves highly accurate classification by averaging over all of a small space of alternative naive-Bayes-like models, the algorithm has highly accurate classification on many classification problems [19].

Decision Trees (e.g., J48 (an open source Java implementation of C4.5 algorithm) [20-21], NavieREPTree, SimpleCart, Random Tree [6, 7], Best First Tree (BFT) [1, 4], A Hoeffding tree [22], Logical Analysis Data (LAD) tree [9], and Logistic Model Tree (LMT) [23]) are a non-parametric learning method used for classification based on simple decision rules inferred from the data features or using hybrid technique (e.g., NavieBayes Tree (NBT) which use NavieBayes classifier at the leaves).

Lazy Classifiers (e.g., Instance Based Learner (IBk) [24], kStar [24], and Locally Weighted Learning (LWL) [26]) store the training instances until the classification time. IBk is a k-nearest-neighbor (KNN) classifier. A variety of different search algorithms (linear search, kD-trees, ball trees, and cover trees) can be used to speed up the task of finding the nearest neighbors based on Euclidean function [9]. The kStar is a nearest-neighbor method with an entropy-based distance function. The LWL uses an instance-based algorithm to assign instance weights.

The functions category (e.g., sequential minimal-optimization (SMO) [27], Logistic [28], and MultilayerPerceptron (MLP) [4]) includes an assorted group of classifiers that can be written down as mathematical equations[9]. SMO implements the sequential minimal-optimization algorithm for training a classifier. Logistic uses regression functions to build a logistic regression model. MLP is a variant of multilayer feedforward neural network and can be trained using backpropagation.

The WEKA (Waikato Environment for Knowledge Analysis*)* workbench is a collection of state-of-the-art machine learning algorithms and data preprocessing tools. It includes all the algorithms described previously. The WEKA tool is developed using Java programming language and available free at the WEKA website [29]. In addition, WEKA enables the developers to integrate their algorithms within WEKA framework.

In general, the classification contains two phases: a training phase to train the classifier followed by testing phase (to evaluate the classifier). However, there is no guaranty that the trained classifier will give adequate accuracy on the data set due to the aforementioned NP-Hard problem. In addition, there is a possibility to predict the output for some instances not in the trained data set which requires test case data generation sampling strategy. Fortunately, WEKA provides a virtualized experimental environment with auto-generated metrics to evaluate the algorithms involve: summary of the evaluation, confusion matrix, and classification details per class. The summary of evaluation reports the accuracy, Kappa statistic (KS), Mean Absolute Error, Root Mean Squared Error (RMSE), Relative Absolute Error (RAE), and Root Relative Squared Error (RRSE). The confusion matrix summary the results of classification by class, from which the details are derived include: true positive (TP) rate, false positive (FP) rate, true negative (TN) rate, false negative (FN) rate, Precision, Recall, F-Measure, Matthews correlation coefficient (MCC), receiver operator characteristic (ROC) Area, and Programmatic Risk Classification (PRC) Area [9][29]. An ideal classifier should have 100% accuracy, KS=1.0, MAE=0, RMSE=0, RAE=0, RRSE=0, a confusion matrix with non-zero diagonal elements, and zero non-diagonal elements, which implies TP=TN= Precision=Recall=F-Measure= MCC=ROC Area=PRC Area =1, and FP=FN=0 [8-10]. As such, achieving an ideal classifier is considered a challenging task.

Boundary Value Analysis (BVA) is widely used as a black box test case generation sampling strategy in software and hardware testing. BVA is based on minimum and maximum values for attributes in the System Under Test (SUT) [30]. However, the problem in the classification differs from test case generation for testing because the data set may have missing values and thus cannot be adopted directly for learning phase. On the other hand, as a black box sampling strategy, it could be adapted to generate test case data generation for the testing phase in the classification problem.

Fix and build from earlier works and motivated by achieving ideal classifier challenge , this paper proposes a novel input-output relation classification strategy called Minimum Maximum Strategy (MMS) to build an expert system that capable to analyze and classify the data set perfectly. In addition, the proposed MMS can generate test instances for evaluation purposes. This paper is organized as follows. Section 2 highlights the proposed MMS strategy. Section 3 gives the summary of the IRIS data set. Section 4 gives an illustrative example on applying MMS on Iris data set. Section 5 discuss integrating the derived rules on WEKA tool. Section 6 discuss the derivation of test cases from the derived rules using BVA. Section 7 evaluates the MMS_IRIS classifier and compares it against the reviewed algorithms. Finally, Section 8 states the conclusion and gives some recommendations for future work.

2. The Proposed MMS Strategy

In order to facilitate the classification, it is required to reverse the rule of BVA (i.e., given data set then analyze the boundary values to make a decision rules for the output).

The MMS strategy derives the classification rules based on the input-output relation (IOR). First, it copies the entire data set into a converge data set (Called Pi). The Pi data set separates the inputs and outputs attributes. Next, the output attributes are divided according to the class values (i.e., for each output). After that, the derivation of classification rules starts in which MMS counts the number of instances per class and determines the minimum and maximum values for each input attribute per output values. The classification rules are built iteratively based on the minimum and maximum boundary values inside the data set. Then eliminates these entries in the Pi data set. This process is done iteratively until Pi data set is empty (i.e., 100 % coverage criteria is satisfied). Fig. 1 shows the MMS strategy to derive the classification rules.

```
While (Pi) is not empty
  {
    Reports the count for each class
    Determine Minimum-Maximum values for each input attributes
   for each input attribute
       for each class
             {
             If there is an inference rule based on unique classification with the corresponding  attributes
                 { write the rule, based on the derived rule
                   eliminates the instances from the data set
                 }//if
             } //for each class
 }// while
```

Fig. 1 The MMS Classification Rules Derivation.

3. The IRIS Data Set Summary

The IRIS data set is taken from the University of California at Irvine (UCI) data sets, and is freely available on the UCI's website [31]. The IRIS data set has 150 instances, three IRIS plants (Setosa, Versicolour, and Virginica), and 4 real-valued attributes as tabulated in Table 1. each class has 50 instances. As such, the class distribution is 33.3% for each of the three classes.

Table 1 The IRIS Data Set Summary.

Attributes Names	**Attributes Minimum, Maximum Values**	
	Min	**Max**
Sepal Length (SL) in cm	4.3	7.9
Sepal Width (SW) in cm	2.0	4.4
Petal Length (PL) in cm	1.0	6.9
Petal Width (PW) in cm	0.1	2.5

4. IRIS Classification Rules Derivation

This section gives an illustrative example by considering the Iris data set described in section 3, followed the steps described in section 2. Here, MMS determines the boundary values (i.e., minimum and maximum values of the inputs for each class separately) as summarized in Table 2.

Table 2 The Initial PI for the IRIS Data Set.

Attributes Names	**Attributes Minimum Maximum Values / Class**					
	Setosa (50)		**Versicolor (50)**		**Virginica (50)**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
SL in cm	4.3	5.8	4.9	7.0	4.9	7.9
SW in cm	2.3	4.4	2.0	3.4	2.2	3.8
PL in cm	1.0	1.9	3.0	5.1	4.5	6.9
PW in cm	0.1	0.6	1.0	1.8	1.4	2.5

Now applying the classification rules derivation, at the first iteration, the following rules are derived:
If (SL<4.9) → Setosa
If(SL>7.0)→ Virginica
If(SW>3.8)→ Setosa
If(SW<2.2)→ Versicolor
If(PL<=1.9)→ Setosa
If (3.0<=PL<4.5)→ Versicolor

If(PL>5.1)→ Virginica
If(PW<=0.6)→ Setosa
If (1.0<=PW<1.4)→ Versicolor
If(PW>1.8)→ Virginica

These derived rules eliminate 124 instances from the Pi data set. The updated Pi data set is summarized in Table 3. It should be mentioned that the data set is mined to the residual data set (i.e., by eliminating the covered instances by the derived rules). In addition, the Setosa class is eliminated (i.e., achieves 100% coverage for the Setosa classification). Moreover, new class occurrence, minimum, and maximum values are determined during the second iteration.

Table 3 The Residues PI for the IRIS Data Set at the Second Iteration.

Attributes Names	**Attributes Minimum Maximum Values / Class**					
	Setosa (0)		**Versicolor (18)**		**Virginica (8)**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
SL in cm	-	-	5.4	7.0	4.9	6.3
SW in cm	-	-	2.2	3.4	2.2	3.0
PL in cm	-	-	4.5	5.1	4.5	5.1
PW in cm	-	-	1.4	1.8	1.5	1.8

At the second iteration, the following rules are derived:

If (SL<5.4) → Virginica
If(SL>6.3)→ Versicolor
If(SW>3.0)→ Versicolor
If (PW<1.5)→ Versicolor

Similarly, these rules eliminate 13 instances from the Pi data set. It should be mentioned that there is no prediction rule for the Petal Length attributes, since the residues classes have the same boundary exactly. The updated Pi data set is summarized in Table 4.

Table 4 The Residues PI for the IRIS Data Set at the Third Iteration.

Attributes Names	**Attributes Minimum Maximum Values / Class**					
	Setosa (0)		**Versicolor (6)**		**Virginica (7)**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
SL in cm	-	-	5.4	6.3	5.9	6.3
SW in cm	-	-	2.2	3.0	2.2	3.0
PL in cm	-	-	4.5	5.1	4.8	5.1
PW in cm	-	-	1.5	1.6	1.5	1.8

At the third iteration, the following rules are derived:

If(SL<5.9)→ Versicolor
If(PL<4.8)→ Versicolor
If (PW>1.6) → Virginica

These rules eliminates 9 instances from the Pi data set. The updated Pi data set is summarized in Table 5.

Table 5 The Residues PI for the IRIS Data Set at the Fourth Iteration.

Attributes Names	**Attributes Minimum Maximum Values / Class**					
	Setosa (0)		**Versicolor (2)**		**Virginica (2)**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
SL in cm	-	-	6.0	6.3	6.0	6.3
SW in cm	-	-	2.5	2.7	2.2	2.8
PL in cm	-	-	4.9	5.1	5.0	5.1
PW in cm	-	-	1.5	1.6	1.5	1.5

During the fourth iteration, the following rules are derived:

If (SW<2.5) → Virginica
If (SW>2.7) → Virginica
If(PL<5.0)→ Versicolor
If (PW>1.5) → Versicolor

These rules eliminates the remaining four instances, thus the Pi data set now is empty as shown in Table 6. Thus, all instances are covered for all outputs. Since the first two identification rules removes the Virginca class from the Pi data set, the last two rules can be replaced by unconditional inference rule as follows:
→ Versicolor.

Table 6 The Empty PI for the IRIS Data Set after the Fourth Iteration.

Attributes Names	**Attributes Minimum Maximum Values / Class**					
	Setosa (0)		**Versicolor (0)**		**Virginica (0)**	
	Min	**Max**	**Min**	**Max**	**Min**	**Max**
SL in cm	-	-	-	-	-	-
SW in cm	-	-	-	-	-	-
PL in cm	-	-	-	-	-	-
PW in cm	-	-	-	-	-	-

The next section explains the mapping and integrating the classification rules to build a classifier for the IRIS data set using WEKA tools.

5.Constructing and Integrating the MMS_IRIS Classifier with WEKA Tool

Like others machine learning algorithms, MMS starts by learning phase to derive the classification rules as explained in the previous section. The MMS compiles the generated rules into a dedicated classification model. The mapping involves two steps. First generate the source code that is compatible with WEKA tools. The second phase compiles the generated source code into a class file using Java compiler. The mapping is done by the MMS automatically, and the resulted source code is shown in Fig. 2.

The classify function is required by the WEKA tool , it takes an object array argument and returns an integer that represents the index to the output array (in our case Setosa, Versicolor, and Virginica) which has the index values 0, 1, and 2 respectively. First, the classify function casts the object array into their corresponding real values. Next, each iteration is mapped to a function called $level_{n-1}$ (where n is the number of iteration). Thus the first, second, ..., nth iterations are mapped to function name Level0, Level1, ..., $level_{n-1}$ respectively. Each classification rule is mapped to if statement and return the index of the corresponding output. If all rules are not taken, the function returns the decision from the next level and so on.

6. Constructing Test Data Set

Unlike other classification algorithms, MMS supports test data generation based on applying the simple BVA on the derived rules, the resulted test suite is called MMS_IRIS_Testing for short. The idea is to pick a value in the range of the decision rule for a certain attribute and fix other attributes values in their range. For clarity, consider the first derived rule (i.e., If (SL<4.9) →Setosa) two test cases can be derived:

instance 1: 4.3, 2.3, 1.0, 0.1, Iris-setosa

instance 2: 4.6, 4.4, 1.9, 0.6, Iris-setosa

Referring to Table 2, the minimum SL is 4.3 which is the first value in the first test case, and the second value is chosen from the range (4.3, 4.9). The second, third and forth attributes values for the first and second test cases are chosen from minimum and maximum attributes values respectively. The complete test case suite is shown in Fig. 3. It should be mentioned that the generated test suite is not a subset of the IRIS data set; thus, it can be used for testing and evaluating other IRIS' classifiers.

7. Evaluation and Discussion

In order to evaluate the MMS_IRIS classifier and make a fair comparison with other classification algorithms, a series of experiments is conducted to meet the following intertwined objectives:

- To investigate whether or not the MMS_IRIS supports the ideal classification condition.
- To evaluate and compare MMS_IRIS against other families.
- To investigate the effectiveness of the derived MMS_IRIS_Testing.

It should be mentioned that all the experiments are done using a laptop with Windows 7 Installed, WEKA version 3.7.12, and Intel Core I7 CPU.

```
public class IRISMinMaxClassifier {
public static int classify(Object[] i) throws Exception {
       // cast the input attributes to its corresponding values
       //the function returns 0,1,2 for  Setosa,Versicolor,Virginica
      double sepalLength=(Double)i[0];
      double sepalWidth=(Double) i[1];
      double petalLength=(Double)i[2];
      double petalWidth=(Double) i[3]  ;
      return  Level0(sepalLength,sepalWidth,petalLength,petalWidth); // call Level0
      } //classify
  static int Level0(double sepalLength,double sepalWidth,double petalLength,double petalWidth) {
  if(sepalLength<4.9) return 0
  if(sepalLength>7.0) return 2;
  if(sepalWidth>3.8) return 0;
  if(sepalWidth<2.2) return 1;
  if(petalLength<=1.9) return 0;
  if(petalLength<4.5 && petalLength>=3.0) return 1;
  if(petalLength>5.1) return 2;
  if(petalWidth<=0.6) return 0;
  if( petalWidth<1.4 && petalWidth>=1.0) return 1;
  if(petalWidth>1.8) return 2;
  return Level1(sepalLength,sepalWidth,petalLength,petalWidth);
    } // Level0
  static int Level1(double sepalLength,double sepalWidth,double petalLength,double petalWidth) {
  if(sepalLength<5.4) return 2 ;
  if(sepalLength>6.3) return 1;
  if(sepalWidth>3.0) return 1;
  if(petalWidth<1.5) return 1;
  return Level2(sepalLength,sepalWidth,petalLength,petalWidth);
   }// Level1
static int Level2(double sepalLength,double sepalWidth,double petalLength,double petalWidth) {
if(sepalLength<5.9) return 1 ;
if(petalLength<4.8) return 1;
if(petalWidth>1.6) return 2;
 return Level3(sepalLength,sepalWidth,petalLength,petalWidth);
  }//Level2
static int Level3(double sepalLength,double sepalWidth,double petalLength,double petalWidth) {
if(sepalWidth<2.5) return 2 ;
if(sepalWidth>2.7) return 2 ;
return 1;
  }// Level3
}// Class
```

Fig. 2 The Auto-Generated Java Source Code by the MMS for IRIS Classifier.

```
@RELATION iris
@ATTRIBUTE sepallength  REAL
@ATTRIBUTE sepalwidth   REAL
@ATTRIBUTE petallength  REAL
@ATTRIBUTE petalwidth   REAL
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
@DATA
4.3,2.3,1.0,0.1,Iris-setosa
4.6,4.4,1.9,0.6,Iris-setosa
7.1,2.2,4.5,1.4,Iris-virginica
7.1,3.8,6.9,2.5,Iris-virginica
4.3,4.3,1.0,0.1,Iris-setosa
4.8,4.1,1.9,0.6,Iris-setosa
4.9,2.0,3.0,1.0,Iris-versicolor
7.0,2.1,1.0,1.8,Iris-versicolor
4.3,2.3,1.0,0.1,Iris-setosa
4.8,4.8,1.9,0.6,Iris-setosa
4.9,2.2,3.0,1.0,Iris-versicolor
7.0,3.4,4.4,1.8,Iris-versicolor
4.9,2.2,5.5,1.4,Iris-virginica
7.0,3.8,6.9,2.5,Iris-virginica
4.3,2.3,1.0,0.3,Iris-setosa
4.8,4.8,1.9,0.5,Iris-setosa
4.9,2.2,4.5,1.0,Iris-versicolor
7.0,3.4,5.1,1.3,Iris-versicolor
4.9,2.2,4.5,1.9,Iris-virginica
7.0,3.8,5.0,2.5,Iris-virginica
4.9,2.2,4.5,1.5,Iris-virginica
5.3,3.0,5.1,1.8,Iris-virginica
6.4,2.2,4.5,1.4,Iris-versicolor
7.0,3.4,5.1,1.8,Iris-versicolor
5.4,3.1,4.5,1.4,Iris-versicolor
6.3,3.4,5.1,1.8,Iris-versicolor
5.4,2.2,4.5,1.4,Iris-versicolor
6.3,3.0,5.1,1.4,Iris-versicolor
5.4,2.2,4.5,1.5,Iris-versicolor
5.8,3.0,5.1,1.6,Iris-versicolor
6.0,2.2,4.5,1.5,Iris-versicolor
6.3,3.0,4.7,1.6,Iris-versicolor
5.9,2.2,4.8,1.7,Iris-virginica
6.3,3.0,5.1,1.8,Iris-virginica
6.0,2.2,5.0,1.5,Iris-virginica
6.3,2.4,5.1,1.5,Iris-virginica
6.0,2.8,5.0,1.5,Iris-virginica
6.3,2.8,5.1,1.5,Iris-virginica
6.0,2.5,4.9,1.5,Iris-versicolor
6.3,2.7,4.9,1.6,Iris-versicolor
6.0,2.5,5.0,1.6,Iris-versicolor
6.3,2.7,5.1,1.6,Iris-versicolor%%%
```

Fig. 3 The Auto-Generated Test Cases by the MMS in WEKA Format.

7.1 MMS_IRIS Classifier Evaluation

The MMS_IRIS classifier evaluation consists of two phases. In the first phase, the MMS_IRIS classifier is evaluated under the exhaustive testing (ET) (i.e., use the full data set; with 150 instances, for testing). In the first phase, the classifier MMS_IRIS is evaluated under the MMS_IRIS_Testing (i.e., use 42 test cases). Tables 7 till 9 and Tables 10 till 12 give the summary of evaluation, confusion matrix, and classification details per class for the MMS_IRIS classifier under the ET and the MMS_IRIS_Testing, respectively.

Table 7 Summary of Evaluation for the MMS_IRIS Classifier under the ET.

Correctly Classified	**Accuracy%**	**KS**	**MAE**	**RMSE**	**RAE%**	**RRSE%**
150	100	1.0	0	0	0	0

Table 8 The Confusion Matrix for the MMS_IRIS Classifier under the ET.

```
a  b  c     <-- classified as
50  0  0  |  a = Iris-setosa
0 50  0   |  b = Iris-versicolor
0  0 50   |  c = Iris-virginica
```

Table 9 Classification Details per Class for the MMS_IRIS Classifier under the ET.

TP Rate	**FP Rate**	**TN Rate**	**FN Rate**	**Precision**	**Recall**	**F-Measure**	**MCC**	**ROC Area**	**PRC Area**	**Class**
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-setosa
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-versicolor
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-virginica

Table 10 Summary of Evaluation for the MMS_IRIS Classifier under the MMS_IRIS_Testing.

Correctly Classified	**Accuracy%**	**KS**	**MAE**	**RMSE**	**RAE%**	**RRSE%**
42	100	1.0	0	0	0	0

Table 11 The Confusion Matrix for the MMS_IRIS Classifier under the MMS_IRIS_Testing.

```
a  b  c     <-- classified as
8  0  0   |  a = Iris-setosa
0 20  0   |  b = Iris-versicolor
0  0 14   |  c = Iris-virginica
```

Table 12 Classification Details per Class for the MMS_IRIS Classifier under the MMS_IRIS_Testing.

TP Rate	FP Rate	TN Rate	FN Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-setosa
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-versicolor
1.0	0	1.0	0	1.0	1.0	1.0	1.0	1.0	1.0	Iris-virginica

Referring to Tables 7 till 12, it is clear that MMS_IRIS classifier meet the ideal condition under both ET and MMS_IRIS_Testing as far as all evaluation metrics is concerned. From another perspective, MMS_IRIS_Testing data set is a good sampling strategy that can be used as a testing data set for evaluation purposes. In addition, the MMS_IRIS_Testing data set is not necessary to be a subset of the original data set due to sampling behavior of black box strategy. As such, it can be used to test the prediction accuracy for other IRIS classifiers.

7.2 Comparison

In order to make a fair comparison, it should be mentioned here that none of the reviewed algorithms achieved the ideal condition using cross-validation training. However, it is unfair to compare MMS_IRIS classifier with other existing algorithms in this way, due to the fact that MMS builds the classifier using the full data set. As such, the evaluation considers two steps. The first step is to run exhaustive learning/ testing (ELT) (i.e., use the full data set for both learning and testing) to elect the most accurate algorithms in each classification family. In doing so, the candidate classifiers for each family can do accurate classification with invariant data set (i.e., no test case values from outside the IRIS data set). In addition, this experiment will report the ideal classifiers for the IRIS data set when an exact classification is desired. In the second phase, the selected classifiers in the first step are tested again under the MMS_IRIS_Testing data set to judge their behavior to classify the untrained instances which expect to be more accurate than the first phase.

7.2.1 Comparison based on ELT

Tables 13 till 17 give a summary of the evaluation for the rule-based, Bayesian, decision tree, lazy, and function-based classifiers respectively for the IRIS data set based on ELT. the dashed rows show the elected classifier based on accuracy obtained. The last column shows the rank for each classifier relative to its' corresponding family.

Referring to Table 13, the NNGE classifier has achieved the ideal condition and has the first rank in this family. OLM and FURIA are in the second place. DTNB, JRip, PART, OneR, DT, Ridor, CR, and ZeroR classifiers have the ranks 3,4,5,6,7,8,9, and 10 respectively.

Table 13 Summary of Evaluation for the Rule Based Family under the ELT.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Relative Rank
CR	100	66.667	0.5	0.2222	0.3334	50	70.7186	9
DT	144	96	0.94	0.0683	0.1582	15.3665	33.5636	7
DTNB	146	97.3333	0.96	0.025	0.1246	5.632	26.4375	3
FURIA	147	98	0.97	0.0133	0.1155	3	24.4949	2
JRip	146	97.3333	0.96	0.0329	0.1283	7.4074	27.2166	4
NNGE	**150**	**100**	**1.0**	**0**	**0**	**0**	**0**	**1**
OLM	147	98	0.97	0.0133	0.1155	3	24.4949	2
OneR	144	96	0.94	0.0267	0.1633	6	34.641	6
PART	146	97.3333	0.96	0.0338	0.1301	7.6122	27.5902	5
Ridor	143	95.3333	0.93	0.0311	0.1764	7	37.4166	8
ZeroR	50	33.3333	0	0.4444	0.4714	100	100	10

Table 14 Summary of Evaluation for the Bayesian Family under the ELT.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Relative Rank
A1DE	143	95.3333	0.93	0.0343	0.1362	7.7107	28.9019	8
A2DE	**145**	**96.6667**	**0.95**	**0.0344**	**0.1298**	**7.7313**	**27.5445**	**1**
BayesNet_GGS	144	96	0.94	0.0304	0.1368	6.8301	29.0144	3
BayesNet_GHC	144	96	0.94	0.0304	0.1368	6.8301	29.0144	3
BayesNet_GK2	142	94.6667	0.92	0.0331	0.1545	7.4367	32.7793	11
BayesNet_GRHC	144	96	0.94	0.0304	0.1368	6.8301	29.0144	3
BayesNet_GSA	144	96	0.94	0.0304	0.1368	6.8301	29.0144	3
BayesNet_GTAN	144	96	0.94	0.0324	0.1340	7.2820	28.4244	4
BayesNet_GTS	144	96	0.94	0.0304	0.1368	6.8301	29.0144	3
BayesNet_LGS	144	96	0.94	0.0505	0.1372	11.3609	29.1108	7
BayesNet_LHC	142	94.6667	0.92	0.0331	0.1545	7.4367	32.7793	11
BayesNet_LK2	142	94.6667	0.92	0.0331	0.1545	7.4367	32.7793	11
BayesNet_LRHC	142	94.6667	0.92	0.0331	0.1545	7.4367	32.7793	11
BayesNet_LSA	144	96	0.94	0.0411	0.1365	9.2371	28.9596	6
BayesNet_LTAN	143	95.3333	0.93	0.0436	0.1395	9.8165	29.6022	9
BayesNet_LTS	142	94.6667	0.92	0.0327	0.1509	7.3553	32.0031	10
NaiveBayes_NKE	144	96	0.94	0.0324	0.1495	7.2883	31.7089	5
NaiveBayes_KE	**145**	**96.6667**	**0.95**	**0.0356**	**0.1376**	**8.0029**	**29.1798**	**2**

Referring to Table 14, no classifier in the Bayesian family achieves the ideal condition. The BayesNet classifier is trained by setting the searching algorithm to global (G) and then local (L) search. In general, BayesNet classifier is performed better using global optimization search algorithms. Similarly, the NaiveBayes classifier is trained without and with kernel estimation. the NaiveBayes classifier is performed better when enabling the kernel estimation. However, changing these parameters change the evaluation slightly. The A2DE and NaiveBayes_KE have the same accuracy and are given the first and second relative rank respectively. The third rank is given for BayesNet_GGS , BayesNet_GHC, BayesNet_GRHC, BayesNet_GSA, and BayesNet_GTS. BayesNet_GTAN, NaiveBayes_NKE, BayesNet_LSA, BayesNet_LGS, A1DE, BayesNet_LTAN, and BayesNet_LTS are given the relative ranks 4 till 10 respectively. Finally, the eleventh rank is given to BayesNet_GK2, BayesNet_LHC, BayesNet_LK2, and BayesNet_LRHC.

Table 15 Summary of Evaluation for the Decision Tree Family under the ELT.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Relative Rank
BFT	147	98	0.97	0.0206	0.1014	4.6250	21.5058	4
HoeffdingTree	144	96	0.94	0.0350	0.1486	7.8697	31.5185	7
J48	147	98	0.97	0.0233	0.1080	5.2482	22.9089	5
LADTree	**150**	**100**	**1**	**0.0088**	**0.024**	**1.9712**	**5.0861**	**2**
LMT	148	98.6667	0.98	0.0196	0.0921	4.4065	19.5468	3
NBT	145	96.6667	0.95	0.0578	0.1427	13.0110	30.2671	6
RandomTree	**150**	**100**	**1**	**0**	**0**	**0**	**0**	**1**
REP Tree	144	96	0.94	0.0490	0.1566	11.0306	33.2123	8
Simple Cart	147	98	0.97	0.0233	0.1080	5.2482	22.9089	5

Referring to Table 15, the Random Tree classifier has achieved the ideal condition and has the first rank in the decision tree family. The LAD Tree classifier has 100% accuracy with non-zero errors and given the second place. LMT and BFT have the ranks 3, and 4 respectively. Both J48 and Simple Cart has the fifth relative rank. Finally, NBT, Hoeffding Tree, and REP Tree classifiers have the relative ranks 6,7, and 8 respectively.

Table 16 Summary of Evaluation for the Lazy Family under the ELT.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Relative Rank
IBk, k=1	**150**	**100**	**1.00**	**0.0085**	**0.0091**	**1.9219**	**1.9335**	**2**
IBk, k=2	146	97.3333	0.96	0.0198	0.0883	4.4445	18.7331	4
IBk, k=3	145	96.6667	0.95	0.0235	0.1088	5.2910	23.0838	5
KStar	**150**	**100**	**1.00**	**0.0062**	**0.0206**	**1.3992**	**4.3621**	**1**
LWL	147	98	0.97	0.0765	0.1636	17.2085	34.7114	3

Referring to Table 16, no classifier in the lazy family achieves the ideal condition due to non-zero errors. the IBK with KNN=1 gives better accuracy than KNN=2 and KNN=3 for the IRIS data set. The classifiers KStar, IBk_1, LWL, IBK_2, and IBK_3 have the relative ranks 1,2,3,4, and 5 respectively.

Table 17 Summary of Evaluation for the Functions Family under the ELT.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Relative Rank
MLP	**148**	**98.6667**	**0.98**	**0.0248**	**0.0911**	**5.5779**	**19.3291**	**2**
Logistic	**148**	**98.6667**	**0.98**	**0.0196**	**0.0921**	**4.4065**	**19.5468**	**1**
SMO	145	96.6667	0.95	0.2296	0.2854	51.6667	60.5530	3

Referring to Table 17, no classifier in the functions family achieves the ideal condition or 100% accuracy. The Logistic, MLP, and SMO classifiers have the relative ranks 1,2, and 3 respectively.

7.2.2 Comparison based on MMS_IRIS_Testing Data Set

In this section, the elected classifiers are tested again using the MMS_IRIS_Test data set. Table 18 shows the summary of evaluation for the elected classifiers under the MMS_IRIS_Testing with 42 test cases (Fig. 3). Table 19 gives the miss-predicted instances for further analysis.

Table 18 Summary of Evaluation for the Elected Classifiers under the MMS_IRIS_Testing.

Algorithm	Correctly Classified	Accuracy%	KS	MAE	RMSE	RAE%	RRSE%	Rank
MMS_IRIS	**42**	**100**	**1.0**	**0**	**0**	**0**	**0**	**1**
NNGE	35	83.3333	0.74	0.1111	0.3333	26.5152	72.9695	**3**
A2DE	29	69.0476	0.51	0.2156	0.3862	51.4453	84.5474	10
NaiveBayes_KE	29	69.0476	0.49	0.2102	0.3979	50.1715	87.0986	9
LADTree	34	80.9524	0.70	0.1320	0.3115	31.5083	68.1907	5
RandomTree	33	78.5714	0.66	0.1429	0.3780	34.0909	82.7396	8
IBk, k=1	35	83.3333	0.74	0.1176	0.3302	28.0749	72.2789	4
KStar	34	80.9524	0.70	0.1478	0.2780	35.2763	60.8531	7
MLP	37	88.0952	0.81	0.1233	0.2688	29.4317	58.8504	2
Logistic	34	80.9524	0.70	0.1442	0.3029	34.4065	66.3160	6

Referring to Table 18, Only, the MMS_IRIS achieves the ideal condition and has the first rank. Surprisingly, even though MLP is not achieved the ideal nor 100% accuracy under ELT, MLP has the second rank even. The NNGE, IBk (k=1), LADTree, Logistic, KStar, Random Tree, NaiveBayes_KE, and A2DE classifiers have the ranks 3 till 10 respectively. It is clear that the derived MMS_IRIS_Testing can classify the prediction accuracy significantly, and gives better evaluation than ELT method.

Referring to Table 19, the third instance int the MMS_IRIS_Testing is the most critical test case. Only, MMS_IRIS classifier predicts it correctly. The other miss-predicted test cases are not unique among the other classifiers and thus has nominal behavior. The reason behind this miss prediction of instance 3 is the interaction between the boundary values for the Sepal Width, Petal Length, and Petal width attributes for Virginica class with other classes when fixing the Sepal Length value at minimum boundary value (Table 2). Moreover, when change the instance 3 to nominal values, all the elected classifiers predict it correctly, which is an indication that they are performed better in nominal values than boundary values. As such, their accuracy is higher when tested them under ELT than in IRIS_MMS_Testing data set.

Table 19 The Miss-predicted Instances for the Elected Classifiers in the MMS_IRIS_Testing.

Family	Algorithm	Correctly Classified Instances	Incorrectly Classified Instances	Miss-predicted instances
MMS	**MMS_IRIS**	**42**	**0**	**-**
Rules	**NNGE**	35	7	**3,** 12,13, 21,24,26,28
Bayes	**A2DE**	29	13	**3,**8,12,13,18,19,21,24,26,36, 38, 39,41
Bayes	**NaiveBayes_KE**	29	13	**3,**12,13 ,19,21,24,26,33, 35,36,37,38,42
DT	**LADTree**	34	8	**3,**8,11,21,24 ,26,28, 33
DT	**RandomTree**	33	9	**3,**8,12,18 ,19,21,24,26,28
Lazy	**IBk, k=1**	35	7	**3,**28,30,36,37,40,42
Lazy	**KStar**	34	8	**3,**24,26,30,36,37,40,42
Functions	**MLP**	37	5	**3,**37,38,41,42
Functions	**Logistic**	34	8	**3,**8,26,30,36,37,40,42

8. Conclusion

This paper has demonstrated and stressed the NP_Complete and NP_Hard problems in the modeling and evaluation of the classifiers. As a result, there is a need to derive an ideal classifier in a systematic manner . In doing so, the MMS has been proposed for modeling an ideal classifier and generates test data set based on BVA and IOR. A case study regarding the IRIS data set is conducted. The practical results can be tackled into two intertwined perspectives.

The first perspective said for an ideal classifier it is required to learn from an exhaustive data set and predict the entire set precisely (i.e., 100% instances coverage during learning and testing phases). As such, only the MMS_IRIS, NNGE, and Random Tree classifiers have achieved the ideal conditions under assumption no test instances outside the IRIS data set (i.e., invariant instances).

From prediction accuracy point of view, it is desired to generate test instances for testing the prediction of the classifier. As such a sampling strategy is required to derive new critical instances that is not necessary be a subset of learning set (i.e., variant instances). From this perspective, the proposed MMS has adopted BVA to generate the MMS_IRIS_Testing data set. By subjecting the most accurate algorithms based on ELT again to test under MMS_IRIS_Testing the prediction accuracy has been decreased significantly except for the MMS_IRIS classifier which has kept the ideal condition invariant. In addition, even though some algorithms achieved 100% accuracy under ELT (with 150 instances), they are more sensitive for prediction new instances. For instance, as far as the accuracy is concerned, the MLP, NNGE, and Random Tree, under ELT has scored 98.6667% , 100%,and 100% respectively; however, under MMS_IRIS_Testing (with merely 42instances) has scored 88.0952%, 83.3333%, and 78.5714 respectively. As such, the derived MMS_IRIS_Testing has three advantages: first it reduced the testing data set in a systematic manner. Second, it is more appropriate for testing the prediction accuracy than ELT approach. Third, the MMS_IRIS has identified a critical instance (instance 3 in Fig. 3) that is recommended to be in the IRIS data set as well as learning data set for the classification algorithms.

Our future work involves studying the reduction of decision rules by considering the number of the covered instances and the sensitivity of the attributes in a greedy manner. Finally, sampling strategies like BVA, combinatorial interaction testing are adopted widely for software and hardware testing, adopting them in machine learning is not widespread. However, the practical results obtained in this paper are promising. For this reason, further research, methodologies, and experimentations are a forthcoming stream in machine learning to adopt these sampling strategies.

9. References

[1] V. Chauraisa and S. Pal, "A Novel Approach for Breast Cancer Detection using Data Mining Techniques", International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Vol. 2, Issue 1, pp. 2456- 2465, January 2014.

[2] V. Chauraisa and S. Pal, "Data Mining Techniques: To Predict and Resolve Breast Cancer Survivability", International Journal of Computer Science and Mobile Computing (IJCSMC), Vol. 3, Issue. 1, pp.10-22, January 2014.

[3] B. F. Chimieski1 and R. D. R. Fagundes, "Association and Classification Data Mining Algorithms Comparison over Medical Data Sets", Journal of Health Informatics (JHI), Vol. 5, No.2, pp.44-51, 2013.

[4] N.M. Isa, W. M. Fahmi, and W. Mamat, "Clustered-Hybrid Multilayer Perceptron Network for Pattern Recognition Application", Applied Soft Computing, Volume 11, Issue 1, pp. 1457-1466, January 2011.

[5] C. L. Devasena, T. Sumathi, V.V. Gomathi, and M. Hemalatha, "Effectiveness Evaluation of Rule Based Classifiers for the Classification of Iris Data Set", Bonfring International Journal of Man Machine Interface, Vol. 1, Special Issue, pp. 5-9, December 2011.

[6] T.Sridevi and A.Murugan, "A Novel Feature Selection Method for Effective Breast Cancer Diagnosis and Prognosis", International Journal of Computer Applications (IJCA), Vol. 88, No.11, pp. 28-33, February 2014.

[7] S. Kalmegh , "Analysis of WEKA Data Mining Algorithm REPTree, SimpleCart and RandomTree for Classification of Indian News", International Journal of Innovative Science, Engineering & Technology (IJISET), Vol. 2, Issue 2, pp. 438-446, February 2015.

[8] T. Hastie, R. Tibshirani, and J. Friedman, "Data Mining, Inference, and Prediction", 2nd Edition, Springer-Verlag, 2009.

[9] I. H. Witten, E. Frank, and M. A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques", Third Edition, The Morgan Kaufmann Series in Data Management Systems, January 2011.

[10] G. James, D. Witten, T. Hastie, and R. Tibshirani , "An Introduction to Statistical Learning: with Applications in R", 1st Edition, Springer Texts in Statistics, August 2013.

[11] R. Kohavi, "The Power of Decision Tables", Proceedings of the 8th European Conference on Machine Learning, Springer, pp. 174-189, 1995.

[12] S. Sheng, C. X. Ling, "Hybrid Cost-sensitive Decision Tree, Knowledge Discovery in Databases", Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Computer Science (LNCS) Vol. 3721, Springer, 2005.

[13] W. W. Cohen, “Fast Effective Rule Induction”, Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann , California, USA, pp.115-123, July 1995.
[14] W. S., S. Asad, and M. A. Khan, “Feature Subset Selection using Association Rule Mining and JRip Classifier”, International Journal of Physical Sciences, Vol. 8, No. 18, pp. 885-896, May 2013.
[15] J. Huhn and E. Hullermeier, “FURIA: An Algorithm for Unordered Fuzzy Rule Induction”, Data Mining and Knowledge Discovery , Springer, Vol. 19, Issue 3, pp. 293-319, December 2009.
[16] E. Frank, I. H. Witten, “Generating Accurate Rule Sets Without Global Optimization”, Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, pp. 144-151, 1998.
[17] K. Vorontsov and A. Ivahnenko, “Tight Combinatorial Generalization Bounds for Threshold Conjunction Rules”, Proceedings of the 4th International Conference on Pattern Recognition and Machine Intelligence (PReMI’11), Lecture Notes in Computer Science (LNCS), Vol. 6744, , pp 66-73, July 2011.
[18] R. Caruana and A. Niculescu-Mizil, “An Empirical Comparison of Supervised Learning Algorithms”, Proceedings of the 23th International Conference on Machine Learning, ACM, pp. 161-168, June 2006.
[19] K.M. Ting and H. Salem, “Learning by Extrapolation from Marginal to Full-multivariate Probability Distributions: Decreasingly Naive Bayesian Classification”, Machine Learning. Vol. 86, No. 2, pp.233-272, 2012.
[20] T.R. Patil and S. S. Sherekar, “Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification”, International Journal Of Computer Science And Applications, Vol. 6, No.2, pp. 256-261, April 2013.
[21] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, “Decision Tree Analysis on J48 Algorithm for Data Mining”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 6, pp. 1114-1119, June 2013.
[22] I. Zliobaite, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, “Evaluation Methods and Decision Theory for Classification of Streaming Data with Temporal Dependence”, Machine Learning, Springer, Vol. 98, Issue 3, pp 455-482 , March 2015.
[23] N. Landwehr, M. Hall, and E. Frank, “Logistic Model Trees”, Machine Learning, Springer, Vol. 59, Issue 1-2, pp 161-205, May 2005.
[24] A. Beygelzimer, S. Kakade, and J. Langford, “Cover Trees for Nearest Neighbor”, Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104, 2006.
[25] J. G. Cleary and L. E. Trigg, “K*: An Instance-based Learner Using an Entropic Distance Measure”, Proceedings of the 12th International Conference on Machine Learning, pp. 108-114, 1995.
[26] E. Frank, M. Hall, and B. Pfahringer, “Locally Weighted Naive Bayes”, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, pp. 249-256, 2003.
[27] C. Changand C. Lin, “LIBSVM: A Library for Support Vector Machines”, ACM Transactions on Intelligent Systems and Technology , Vol. 2, No. 3, pp. 1-27, 2011.
[28] J. Platt, “Probabilistic “Outputs For Support Vector Machines and Comparisons to Regularized Likelihood Methods”, Advances in Large Margin Classifiers, Vol. 10, No. 3, pp. 61–74, 1999.
[29] WEKA 3 Web Site, “Data Mining Software in Java”, available at: http://www.cs.waikato.ac.nz/ml/weka, last accessed on 25 July 2015.
[30] K. Z. Zamli, M. I. Younis, S. A. C. Abdullah, Z. H. C. Soh, “Software Testing”, 2nd Edition, Open University of Malaysia, April 2009.
[31] UCI’s Web Site, “IRIS Data Set” available at: https://archive.ics.uci.edu/ml/data sets/Iris, last accessed on 25 July 2015.