EVALUATING CLASSIFICATION PROBLEMS USING PERFORMANCE METRICS

BY: Utpal Mishra (1609113120)

Under the supervision of

Dr. Dhiraj Pandey

Department of Information Technology

JSS ACADEMY OF TECHNICAL EDUCATION

C-20/1, C Block, Phase 2, Industrial Area, Sector 62, Noida, Uttar Pradesh 201301

2019-20

EVALUATING CLASSIFICATION PROBLEMS USING PERFORMANCE METRICS

BY: Utpal Mishra (1609113120)

Under the supervision of Dr. Dhiraj Pandey

Submitted to the Department of Information Technology in partial fulfillment of the requirements for the degree of Bachelor of Technology in Information Technology

Department of Information Technology

JSS ACADEMY OF TECHNICAL EDUCATION

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY

2019-20

CERTIFICATE

This is to certify that Project Report entitled **"EVALUATING CLASSIFICATION PROBLEMS USING PERFORMANCE METRICS"** which is submitted by **Utpal Mishra** in partial fulfilment of the requirements for the award of degree B. Tech. in Department of Information Technology of **Dr A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY**, is a record of the candidate's own work carried out by him under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Supervisor: Dr. Dhiraj Pandey

Date : 17th April 2020

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. I owe a special debt of gratitude towards ***Dr. Dhiraj Pandey*** *Department of Information Technology, JSS Academy of Technical Education, Noida for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant effort that our endeavours have seen the light of the day.*

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project.

Name : Utpal Mishra
Roll No.: 1609113120
Date : 17th April 2020

TABLE OF CONTENTS

CONTENT **PAGE NO.**

LIST OF TABLES

CONTENT **PAGE NO.**

LIST OF FIGURES

CONTENT **PAGE NO.**

LIST OF SYMBOLS

CONTENT PAGE NO.

LIST OF ABBREVIATIONS

ANN	**Artificial Neural Networks**
KNN	**K-Nearest Neighbor**
SVM	**Support Vector Machine**
NB	**Naive Bayes**
DT	**Decision Tree**
RF	**Random Forest**
DL	**Deep Learning**

ABSTRACT

One of the core tasks in building any machine learning model is to evaluate its performance. It's fundamental, and it's also really hard.

So how would one measure the success of a machine learning model? How would we know when to stop the training and evaluation and call it good? With this article, we'll try to answer these questions.
In the first section, we'll introduce what we mean by "ML Model Evaluation" and in subsequent sections, we'll discuss various evaluation metrics available about specific use cases to better understand these metrics.

CHAPTER 1
INTRODUCTION

While data preparation and training a machine learning model is a key step in the machine learning pipeline, it's equally important to measure the performance of this trained model. How well the model generalizes on the unseen data is what defines adaptive vs non-adaptive machine learning models.

By using different metrics for performance evaluation, we should be in a position to improve the overall predictive power of our model before we roll it out for production on unseen data.

Without doing a proper evaluation of the ML model using different metrics, and depending only on accuracy, can lead to a problem when the respective model is deployed on unseen data and can result in poor predictions.

This happens because, in cases like these, our models don't **learn** but instead **memorize**; hence, they cannot generalize well on unseen data. To get started, let's define these three important terms:

i. **Learning:** ML model learning is concerned with the accurate prediction of future data, not necessarily the accurate prediction of training/available data.

ii. **Memorization:** ML Model performance on limited data; in other words, overfitting on the known training dataset.

iii. **Generalization:** Can be defined as the capability of the ML model to apply learning to previously unseen data. Without generalization there's no learning, just memorization. But note that generalization is also goal specific —for instance, a well-trained image recognition model on zoo animal images may not generalize well on images of cars and buildings.

In the next section, we'll discuss the different evaluation metrics available that could help in the generalization of the ML model.

EVALUATION METRICS

The use of an evaluation matrix is one method of objectively evaluating a number of options against a number of criteria. These criteria are prioritised before the evaluation is made with greater weighting to those items of most importance. If there are criteria that absolutely must be met, two levels of evaluation matrices can be used. The first

level acts as a filter with each option evaluated against the mandatory criteria. Those options that meet every mandatory criterion go on to the second level to be evaluated against prioritised criteria.

After doing the usual Feature Engineering, Selection, and of course, implementing a model and getting some output in forms of a probability or a class, the next step is to find out how effective is the model based on some metric using test datasets.

TYPES

When we talk about predictive models, we are talking either about a regression model (continuous output) or a classification model (nominal or binary output). The evaluation metrics used in each of these models are different.

i. Class Output

Algorithms like SVM and KNN create a class output. For instance, in a binary classification problem, the outputs will be either 0 or 1. But these algorithms are not well accepted by the statistics community. However, today we have algorithms which can convert these class outputs to probability.

ii. Probabilistic Output

Algorithms like Logistic Regression, Random Forest, Gradient Boosting, Adaboost etc. give probability outputs. Converting probability outputs to class output is just a matter of creating a threshold probability.

Different performance metrics are used to evaluate different Machine Learning Algorithms and are discussed here.

EVALUATION STRATEGIES

Metrics evaluates the quality of an engine by comparing engine's output (predicted result) with the original label (actual result). A engine serving better prediction should yield a higher metric score, the tuning module returns the engine parameter with the highest score. It is sometimes called loss function in literature, where the goal is to minimize the loss function.

During tuning, it is important for us to understand the definition of the metric, to make sure it is aligned with the prediction engine's goal.

In the classification template, we use Accuracy as our metric. Accuracy is defined as: the percentage of queries which the engine is able to predict the correct label.

Evaluation metrics are used to measure the quality of the statistical or machine learning model. **Evaluating** machine learning models or algorithms is essential for any project.

There are many different types of **evaluation metrics** available to test a model.

APPLICATIONS

If the evaluation matrix is being used to evaluate various design solutions, usually previously established requirements are used as the criteria. Since requirements are generally written in terms of “shall” or “should”, the mandatory criteria used in the first level of the evaluation are represented by the “shall” requirements while the “should” requirements are prioritised and used in the second level evaluation.

CHAPTER 2
EVALUATING METRICS

CROSS VALIDATION

Cross-validation is a statistical method used to estimate the skill of machine learning models. Cross Validation is a validation technique for the model to statistically examine the generalization pattern of the results on the independent dataset.

It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

To ensure that the model has analyzed and undestood the data pattern without noise or without being overfitted/ underfitting or with low bias, cross validation is required to statistically behold the stability of the model.

This model validation method provides a bit of flexibility over the splitting or groups or k-folds, which are as follow:

i. k-Fold (k=2):

It means the data is grouped into two i.e. the training and the test data. This type of grouping is opted if, we have enough data to make the model learn the pattern on a randomly trained training data. Any duplicacy and overlapping of grouped data should be avoid and final model - after testing - should be retrained on the complete dataset without any tuning in the hyperparameters.

ii. k-Fold (k=3):

This is comparatively a better approach then binary grouping the dataset as the dataset is bifurgated into three, the training data, the validation data and the test data. To evaluate the quality of model fitted on trained data, model is validated (prior to testing) on a new sample (validation dataset). This pattern is chosen if the data size is sufficient enough to be grouped as such.

iii. k-Fold (k):

For splitting the dataset, this is a prominent approach as the data available for to model the decipher the pattern is not never enough and model has to the problems of underfitting and increased loss.

In this method, the data is grouped into k folds and model is trained into k-1 times. Each time k-1 portion is trained and is validated over the remaining portion. Each time the

model is trained is validated on a new piece of data which significantly reduces the underfitting and the overfitting problem. This method is chosen for a small sized data as the model is free from a high bias or a high variance.

INFORMATION GAIN

Information gain is the reduction in entropy by transforming a dataset and is often used in training decision trees. Information gain is calculated by comparing the entropy of the dataset before and after a transformation.

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

ENTROPY

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. Entropy is a measure of disorder or uncertainty and the goal of machine learning models. The higher the entropy more the information content.

GINI INDEX

The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one. It favors larger partitions. Information Gain multiplies the probability of the class times the log (base=2) of that class probability. Information Gain favors smaller partitions with many distinct values.

CONFUSION MATRIX

A **confusion matrix** (Fig 1) is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

Statistically, a confusion matrix is an N X N matrix, where N is the number of classes being predicted.

Accuracy: the proportion of the total number of predictions that were correct.
Positive Predictive Value or Precision: the proportion of positive cases that were correctly identified.
Negative Predictive Value: the proportion of negative cases that were correctly identified.
Sensitivity or Recall: the proportion of actual positive cases which are correctly identified.
Specificity: the proportion of actual negative cases which are correctly identified.

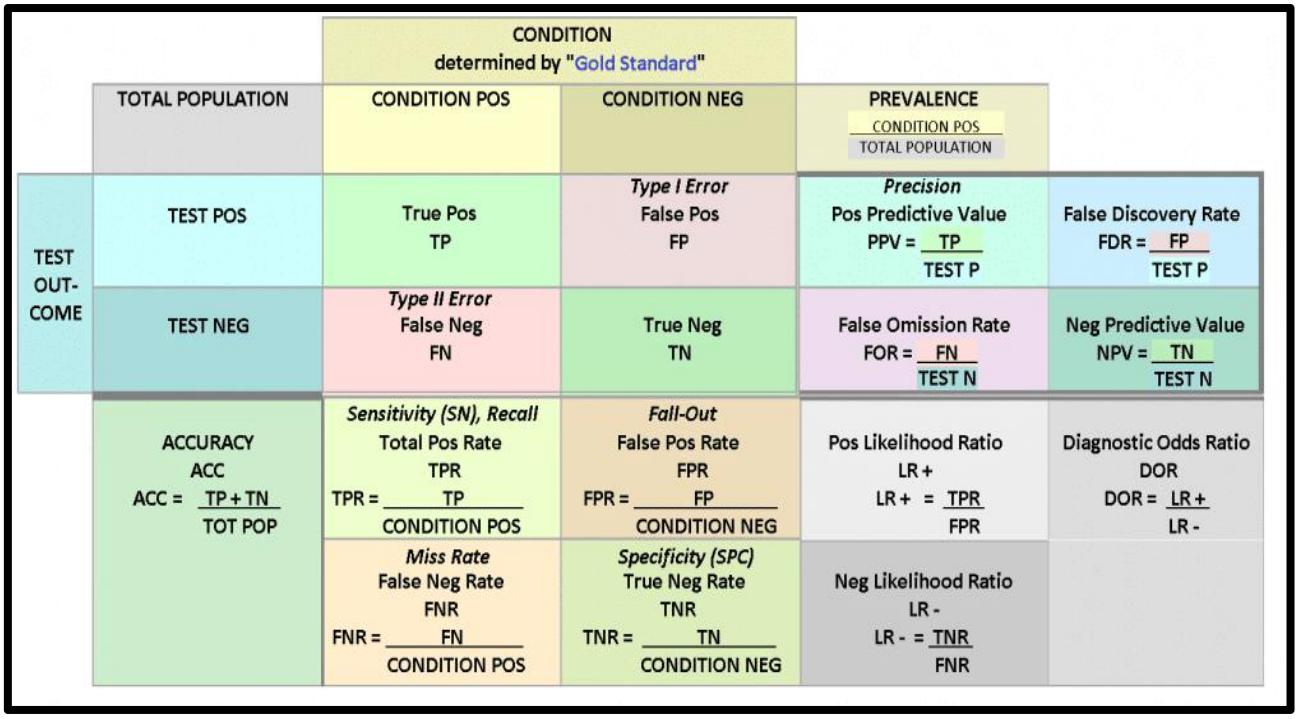

Fig 1: Confusion Matrix

F1-SCORE

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall:

$$F1 = 2/ (recall^{-1} + precision^{-1})$$

$$F1 = \frac{2*(recall.precision)}{(recall + precsion)}$$

F1 score conveys the balance between the precision and the recall.

ROC CURVE

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve (Fig 2).

It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

The ROC curve is a useful tool for a few reasons:

i. The curves of different models can be compared directly in general or for different thresholds.
ii. The area under the curve (AUC) can be used as a summary of the model skill.

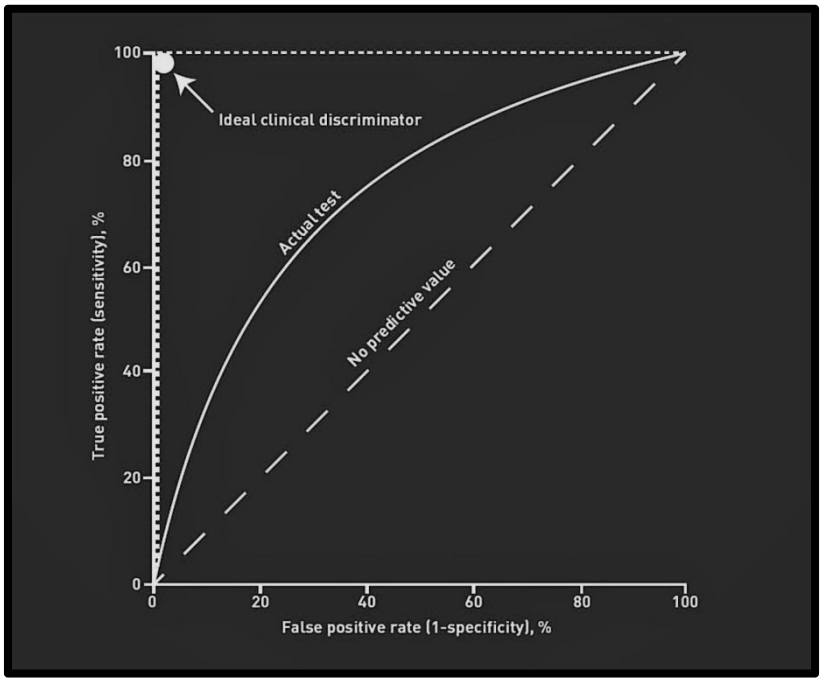

Fig 2: ROC Curve

ACTIVATION FUNCTIONS

These functions (Fig 3) comes into play while palying with Network and hold its importance by imparting non-linearity to the model which would otherwise have been a Polynomial Regression Model with linear dependent relationship between feature variables and target variables.

With the functionality of being a complex computational, the model learn better through non-linearity also eligible to comprise and incorporate any non-linear functionality into the network of artificial neurons (Universal Function Approximator) and deal effectively with high-dimensional data. Activation functions are applied after evaluating the equation of hypothesis *h(x)* at each opertional node.

$$y_i = Ø(h(x))$$
$$y_i = Ø(w*x + b)$$

{ y_i = output at i^{th} node
Ø() = activation function
w = weight matrix (variable node)
x = input matrix (constant node)
b = bias (variable node) }

i. Sigmoid Function:

$$Ø(x) = \frac{1}{1 + e^{-ax}} \quad ; 0 < Ø(x) < 1$$

{ Ø() = activation function
a = constant (usully a=1)
x = input matrix(constant node) }

Sigmoid functions is an easy and simple interpretable function but sticks to Vanishing Gradient problem due to slower convergence and a non-zero (0.5) centric function, resulting in harder optimization. The Sigmoid Formula is shown above.

ii. Hyperbolic Tangent (tanh(x)):

Formula for Hyperbolic Tangent is shown below:

$$Ø(x) = tanh(x) \quad ; 0 < tanh(x) < 1$$

{ Ø() = activation function
x = input matrix(constant node) }

A better performanc is shown by Tan(h) function which ranges from -1 to 1 thus, making an easy optimization and zero centric function but still encounters Vanishing Gradient problem.

iii. Rectified Linear Unit (ReLu):

As the network goes throh backpropagation while dealing with a large dataset and using a complex computational function for mapping the hidden layers, the gradient diminishes to update the parameters as we move closer to the input as the error becomes smaller and smaller resulting in slower rate of convergence.

In a such a multi-layered network, ReLu stand out, resolving the Vanishing Gradient problem and thus, being the widely used activation.

Formlua for ReLu function is representted below:

$$\emptyset(x) = max(0,x) \;\; ; \; 0 < \emptyset(x) < \infty$$

{ Ø() = activation function
x = input matrix(constant node) }

iv. Leaky ReLu:

This activation functions counters few drawbacks of ReLu which were of being limited to hidden layers and deactivating neurons or simply, resulting in dead neurons.

Leaky ReLu function is illustrated below:

$$\emptyset(x) = max(0.1x,x) \;\; ; \; -\infty < \emptyset(x) < \infty$$

{ Ø() = activation function
x = input matrix(constant node) }

v. Randomized Leaky ReLu (RLReLu):

Randomized Leaky ReLu function is illustrated below:

$$\emptyset(x) = max(0,x) + a*min(0,x)) \; ; \; -\infty < \emptyset(x) < \infty$$

{ Ø() = activation function
x = input matrix(constant node)
a = randomized variable }

vi. Softmax

Softmax Fucntion (usually at last layer) is used to find the probability of ‘n’ unique events to determine the class of the input. The outcome is based on the probability distribution ranging from 0 to 1 and determining the probaility of each class with taget class holding the maximum probability, comparatively in a multi-class classification problem.

Softmax Function formula is as shown:

$$\emptyset(x_i) = x_i \,/\, ({}^{k}\Sigma_{j=1}\, x_j)$$

{ Ø() = activation function

x = input matrix(constant node) }

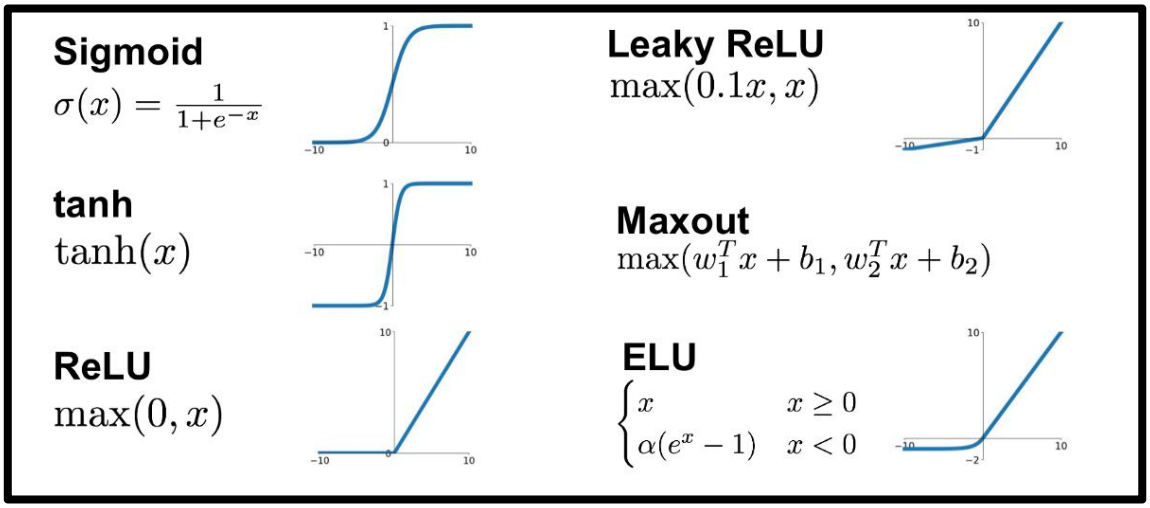

Fig 3: Activations Functions

CHAPTER 3
RESULTS AND CONCLUSION

	PRECISION	RECALL	F1-SCORE	SUPPORT	ACCURACY
SVM	**1.00**	**0.99**	**0.99**	**75**	**99.1228%**
NAIVE BAYES	**0.59**	**0.66**	**0.62**	**67**	90.3509%
DECISION TREE	**0.59**	**0.66**	**0.62**	**67**	92.9825%
DT BAGGING	**0.59**	**0.66**	**0.62**	**67**	92.9825%
EXTRA TREE	**0.59**	**0.66**	**0.62**	**67**	92.9825%
RANDOM FOREST	**0.59**	**0.66**	**0.62**	**67**	97.3684%
GRADIENT DESCENT	**0.59**	**0.66**	**0.62**	**67**	97.3684%
ADABOOST	**0.97**	**0.99**	**0.98**	**75**	97.3684%
XGBOOST	**0.97**	**1.00**	**0.99**	**75**	**98.2456%**
ANN	**0.96**	**1.00**	**0.98**	**75**	97.3684%
DEEP LEARNING	**0.96**	**1.00**	**0.98**	**75**	97.3684%

Table 1: Performance Matrix based on Breast Cancer Classification Under Various Classification Algorithms

From the results (Table 1) obtained using various Machine Learning Classification Algorithms, it is observed that SVM gives the maximum accuracy of 99.1228% with 1.00 precision and XGBOOST Classifier with 98.2456% accuracy and 1.00 recall. Then, comes ADABOOST classifier, Artificial Neural Network (Layer 1 : 30 units, Layer 2: 15 units, Layer 3: 1 unit , Batch size: 30, Epoch: 1000) and Deep Neural Network (Layer 1 : 455 units, Layer 2: 200 units, Layer 3: 100 units, Layer 4: 50 units, Layer 5: 1 unit, Batch size: 30, Epoch: 1000) with 97.3684% of accuracy. Thus, it is clearly evident that Support Vector Classifier works the best for classification under conditions provided.

In this report, I have illustrated the performance of tabulated classification algorithm using various evaluations metrics. Understanding how well a machine learning model is going to perform on unseen data is the ultimate purpose behind working with these evaluation metrics. Metrics like accuracy, precision, recall are good ways to evaluate classification models for balanced datasets, but if the data is imbalanced and there's class disparity, then other methods like ROC/AUC perform better in evaluating the model performance.

As we've seen, the ROC curve isn't just a single number; it's a whole curve. It provides nuanced details about the behavior of the classifier, but it's also hard to quickly compare many ROC curves to each other. The AUC is one way to summarize the ROC curve into a

single number so that it can be compared easily and automatically. A good ROC curve has a lot of space under it (because the true positive rate shoots up to 100% very quickly). A bad ROC curve covers very little area. So high AUC is good, and low AUC is not so good.

One last important point to keep in mind, since we focused largely on classification tasks in this guide: Any machine learning model should be optimized and evaluated according to the task it's built to address.

REFERENCES

[1]. N.V. Chawla, N. Japkowicz and A. Kolcz, “Editorial: Special issue on learning from imbalanced data
sets”, SIGKDD Explorations, 6 (2004) 1-6.

[2]. S. Garcia and F. Herrera, “Evolutionary training set selection to optimize C4.5 in imbalance
problems”, in Proc. of 8th Int. Conference on Hybrid Intelligent Systems (HIS 2008), Washington,
DC, USA, IEEE Computer Society, 2008, pp.567-572.

[3]. D. J. Hand and R. J. Till, “A simple generalization of the area under the ROC curve to multiple class
classification problems”, Machine Learning, 45 (2001) 171-186.

[4]. M. Hossin, M. N. Sulaiman, A. Mustapha, N. Mustapha and R. W. Rahmat, “A Hybrid Evaluation
Metric for Optimizing Classifier”, in Data Mining and Optimization (DMO), 2011 3rd Conference on,
2011, pp. 165-170.

[5]. J. Huang and C. X. Ling, “Using AUC and accuracy in evaluating learning algorithms”, IEEE
Transactions on Knowledge Data Engineering, 17 (2005) 299-310.

[6]. P. Lingras, and C. J. Butz, “Precision and recall in rough support vector machines”, in Proc. of the 2007 IEEE Int. Conference on Granular Computing (GRC 2007), Washington, DC, USA: IEEE Computer Society, 2007, pp.654-654.

[7]. D. J. C. MacKay, Information, Theory, Inference and Learning Algorithms. Cambridge, UK: Cambridge University Press, 2003.

[8]. F. Provost, and P. Domingos, “Tree induction for probability-based ranking”. Machine Learning, 52
(2003) 199-215.

[9]. A. Rakotomamonyj, “Optimizing area under ROC with SVMs”, in J. Hernandez-Orallo, C. Ferri, N. Lachiche and P. A. Flach (Eds.) 1st Int. Workshop on ROC Analysis in Artificial Intelligence (ROCAI 2004), Valencia, Spain, 2004, pp. 71-80.

[10]. R. Ranawana, and V. Palade, “Optimized precision-A new measure for classifier performance evaluation”, in Proc. of the IEEE World Congress on Evolutionary Computation (CEC 2006), 2006, pp. 2254-2261.

[11]. S. Rosset, “Model selection via AUC”, in C. E. Brodley (Ed.) Proc. of the 21st Int. Conference on Machine Learning (ICML 2004), New York, NY, USA: ACM, 2004, pp. 89. 9-108.

[12]. H. Wallach, “Evaluation metrics for hard classifiers”. Technical Report. (Ed.: Wallach, 2006)
http://www.inference.phy.cam.ac.uk/hmw26/papers

[13]. S. W. Wilson, “Mining oblique data with XCS”, in P. L. Lanzi, W. Stolzmann and S. W. Wilson (Eds.) Advances in Learning Classifier Systems: Third Int. Workshop (IWLCS 2000), Berlin, Heidelberg: Springer-Verlag, 2001, pp. 283-290.