Semi-supervised learning applied to performance indicators in software engineering processes

Leandro Bodo[1,3],
Hilda Carvalho de Oliveira[1,3], Fabricio Aparecido Breve[1,3], Danilo Medeiros Eler[2,4]
[1]Dep. of Statistics, Applied Mathematics and Computer Science
[2]Dep. of Mathematics and Computer Science

Universidade Estadual Paulista, UNESP
[3]Rio Claro, [4]Presidente Prudente - Brazil
lbodo@rc.unesp.br, hildaz@rc.unesp.br, fabricio@rc.unesp.br, daniloeler@fct.unesp.br

***Abstract*—Performance indicators are critical resources for quality control in the software development process. Over time, the data volume of the historical basis these indicators increase significantly. Moreover, the diversity of treatment (individual or group) and the frequency of data collection hamper the analysis. The time and reliability of these analyzes are important to support a faster and more effective decision. Thus, this paper proposes the use of artificial neural networks with semi-supervised learning to analyze the historical basis of performance indicators. In order to support the sample labeling process it is recommended to use information visualization techniques. An indicator reference model was defined based on the software process reference model MPS for Software (MPS-SW) to be used before the labeling process.**

***Keywords*—Software Processes, performance indicators, machine learning, artificial neural network, MPS-SW.**

I. INTRODUCTION

During a software development, procedures for quality control aim to identify ways to mitigate causes of unsatisfactory results. Furthermore, these procedures also allow to evaluate the performance of the processes and to indicate the need for changes and corrections on the process [1], [2]. Specific project results must be monitored to determine if they are in compliance with the quality criteria previously defined and which are relevant to the development of the software. Statistical techniques have helped the evaluation of the results of quality control.

The performance indicators are important tools to the quality control in general. These indicators aid to quantify the performance of activities, processes and products, making it possible to analyze the results and compare them to the planned goals [3]. The indicators provide numerical relations that reflect the current state of the processes. These relations provide information for decision-making related to the processes, and consequently, to their own business. The performance indicators can present variations that require the management's attention when making a decision. It is possible to evaluate the variations that occurred and generate prognostic (projections) through historical comparisons of an indicator.

The performance indicators might be analyzed either individually or in groups, depending on the defined specifications to support the decision-making. These groupings can also consider distinct indicators from different projects or products, or consider the same indicator that comes from different projects or products. The analyses are performed continuously during the software development process, in order to create a historical basis and judge the quality of the software during the whole process [2].

However, the historical basis of the performance indicators tends to become large and complex That happens because of the large amount of data that is stored at the same time and the intrinsic diversity of the indicators (different types, granularity and frequency of data collection). Furthermore, the data volume that is produced by these indicators tends to increase dramatically over the monitoring time. All these factors require a solution that enables the analysis of performance indicators, improving the use of these indicators in the software development process.

Within this context, this paper proposes the use of semi-supervised machine learning techniques for analysis of performance indicators in software development processes. This type of learning reduces the cost with labeling process of supervised learning and does not despise the labels of samples, it could happen in the unsupervised learning.

The goal is to "teach" (to train) an Artificial Neural Network (ANN) to recognize the existing patterns in the large volume of the historical data generated by the indicators. Thus, the ANN will be able to analyze the indicators in different development processes and provide results that indicate the status of a particular situation, similar to a traffic light. Overall, an ANN can analyze indicator groups simultaneously, with different complexities. This allows the indicator groups to be controlled in dashboards. The same goes for individual indicators. The same goes for individual indicators. This technique is able to provide a decision-making more effective

and faster.

Therefore, the *section II* presents an overview of machine learning and ANN. The *section II* outlines the semi-supervised learning algorithm for analysis of performance indicators: Particle Competition and Cooperation (PCC).

ANNs have been used for treatment of performance indicators in different applications of Software Engineering. The section *IV* presents a few comments about it and discusses some criteria for the use of ANNs in software development processes.

On the other hand, the *section V* outlines an indicator model based on processes levels G and F of reference model MPS for software (MPS-SW). This model will help to guide the grouping of indicators in software development organizations.

The *section VI* shows the application of the PCC algorithm on a real set of performance indicator samples. These indicators belong to the historical basis of a company that develops software, that is a project partnership and certified in the level G on the model MPS-SW. The results of PCC application are compared with the results that have been obtained in two supervised ANN techniques. There was no comparison with unsupervised methods, because this type of learning ignores the information of the sample labels - and this information is fundamental for the type of problems presented. The final considerations of this article are presented in *section VII*.

II. Machine Learning and Artificial Neural Network

Within the context of artificial intelligence, machine learning is an area that involves the construction of systems capable of acquiring knowledge automatically. This area considers the development of algorithms that use the acquired "experience" to produce results without human intervention. A machine learning algorithm can make decisions based on examples of input data [4].

Overall, machine learning algorithms require the analysis of a large amount of samples for learning. The idea is to teach the algorithms to solve different problems in a given context. This context may have characteristics that cause changes over time and/or the type of application and use [5].

ANN is a type of machine learning techniques, which is discussed in *subsection A*. There are different machine learning categories, each one recommended for a particular type of problem. The *subsection B* presents three categories: supervised, unsupervised and semi-supervised.

A. Artificial Neural Network (ANN)

ANNs are computational techniques based on mathematical models inspired by the neural structure of intelligent organisms. The acquisition of knowledge in ANNs is performed through experience [6]. They have the ability to learn by examples, making inferences from that learning and improving their performance gradually.

ANN has a behavior based on groups of neurons in the human brain, which receive and transmit information through the dendrites and axons, respectively [7]. When a specific set of data inputs and their outputs are presented with an ANN, it is able to auto adjust its synaptic weights. The adjustment of the connections is obtained by learning adopting as training criterion (and subsequent analysis) a specific activation function. The training phase of ANNs consists in a functional relationship mapping that exists between the inputs and outputs. Following training, the network should be able to generalize the behavior of the process at the time when other inputs are presented to it (different inputs from those used during training) [8].

ANNs can be used for performance evaluation indicator groups of software engineering processes, as proposed in this paper. Indicators are considered the network attributes and have different measurement value and goals, without a pattern in data types defined (may contain integer, boolean or real values). Therefore, a software factory can "teach" how certain indicator groups can express control targets, adjusting its parameters on demand.

B. Machine Learning categories

A supervised learning algorithm requires that an expert (external entity) introduce some sets of patterns for the inputs and the corresponding standards for the outputs (results). An output can be a numeric value or can predict a class label for the input object. In the training phase of an ANN, for example, the expert indicates explicitly for each input if the output response is good or bad (data labeling process). Thus, the result provided by the network is compared to the expected answer. If the result is different than expected, an error is reported to the network and adjustments need to be made in order to improve future responses.

Unsupervised learning algorithms do not require an external entity to perform the training process. They aim to determine how the data is organized only based on the input patterns, without labels or output values. These algorithms process the inputs available and try to establish internal representations to encode features and classify them automatically, by detecting the singularity in the input samples.

Semi-supervised learning algorithms use both labeled and unlabeled data for training. In many cases, the use of few labeled data with many unlabeled data considerably improves accuracy of the learning [9].

Due to the large amount of existing databases, label data for supervised learning algorithms have become an increasingly unworkable process. Usually, the labeling process is expensive and time consuming, requiring intense involvement of human experts. On the other side, the unsupervised algorithms ignore valuable information label of the data items. Semi-supervised algorithms can mitigate these problems: a few labeled data items are combined with a large amount of unlabeled data, producing better classifiers [10].

III. Algorithm ANN semi-supervised adopted

The selected algorithm to propose a treatment for performance indicators in this paper is called Particle Competition and Cooperation (PCC) [9].This semi-supervised ANN algorithm was chosen because it requires little human effort and consequently little financial cost.

The PCC algorithm consists of a graph of related networks, which has various particles moving through the network. These particles are organized in the form of teams, where particles of the same team go over the network in a cooperative way in order to propagate their labels. Meanwhile, particles of different teams compete with each other to determine the borders of the class, and reject intrusive particles.

Traditionally, labels are spread over the network in a global way in other semi-supervised learning algorithms based on graphs. This means that the information is propagated from all nodes to all other nodes for each iteration of the algorithm, accordingly to the respective edge weights. On the other hand, the spread of the label occurs locally at the PCC algorithm. Therefore, each step of the algorithm, every particle propagates its label to a selected neighbor by the rule "random[1]-greedy[2]". Thus, each particle visits only a portion of the nodes that potentially belongs to its team (subnet), preventing from redundant operations are carried out [9].

Breve [10] presents a metaphor of particles based on ant behavior (**Fig. 1**) to explain the PCC algorithm.

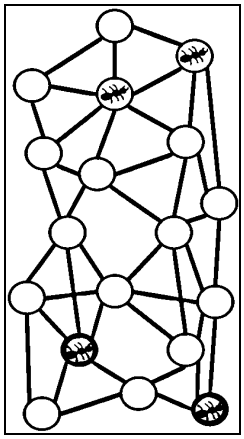

Fig 1. Metaphor based on behavior ant [10].

Every node on the network has an array of elements, responsible for representing a level of particles each team. Initially, the input data set is transformed into a non weights and undirected network, where each node corresponding to a non-labeled sample will have its field level configured with the same value $\frac{1}{c}$, where c = number of classes / teams. The **Fig. 2 (a)** explains the initial situation with four classes, ie, nodes not labeled with domain levels 0.25: [0.25 0.25 0.25 0.25]. Next, a set of ants are placed on the network. Each of them is a labeled data item that is set to the highest value. The **Fig. 2 (b)** illustrates this situation to the four classes labeled as Class A: [1 0 0 0]. The subset of particles with the same label is called "team" [9].

1*Random walk*: each particle chooses any neighbor to visit at random, without worrying about the domination levels or distance from it home node. It is a movement for acquisition of new node and exploration [9].

2*Greedy walk*: each particle visits the nodes that are closest to their home nodes, especially those who are already dominated by its own team. It is a movement to defend the territory of his team [9].

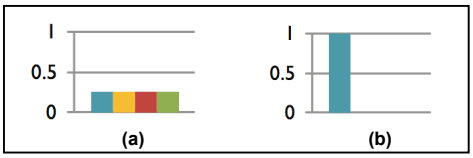

Fig. 2. Initial domination level: (a) unlabeled sample; (b) labeled sample.

Every ant chooses a neighboring node to visit each iteration of the algorithm, using the "random-greedy" rule. The domain level of their team is increased when an ant selects a neighboring node. Meanwhile, domain levels of other teams are decreased. If the node to be visited is in control of its own team, the ant gets stronger, increasing their domain levels. However, the ant weakens if the visited node is domain of the other team. Each ant works prioritizing the domain of their respective neighborhoods (neighboring nodes). For this, they have a particular node as home and each ant keeps information of the distance between their respective homes and other network nodes. This method includes cooperation between the ants. Thus they prioritize helping their teammates with their neighborhoods, and eventually try to invade opponents' territories. Each team tries to dominate the largest possible number of nodes, and simultaneously try to avoid the invasion of ants from other teams. At the end of the iterative process, each unlabeled data item will be labeled according to the team's label which contains the highest domain level.

IV. Artificial Neural Network applied to performance indicators

Machine learning techniques have been used with performance indicators in different areas of knowledge. Melo et al. [11] used an ANN Multilayer Perceptron (MLP) to predict the variation in the flow of vehicles on roads, helping drivers to selecting the best routes. Neto, Nagano and Moraes [12] used an unsupervised ANN to classify agricultural cooperatives based on their socioeconomic indicators. Cattinelli et al. [13] used an ANN to analyze groups of performance indicators in 109 hemodialysis clinics in Italy, Portugal and Turkey. Within the context of Software Engineering, Kutlubay et al. [14] used ANN techniques for detecting defects in software products.

In this way, this paper proposes the use of PPC algorithm (see *section III*) for an ANN with semi-supervised learning for quality control during software development. This proposal has a cost of labeling data up to 20% - which is considered low cost. However, it will allow the development of tools that will automate the monitoring processes effectively and efficiently. The reliability of the labeling process can be aided with information visualization techniques. It is recommended the LSP techniques (Least Squares Projection) and parallel coordinates.

Overall, the software processes and the performance indicators are defined for any different software developer organization. Thus, an indicator model was created to guide the grouping of performance indicators before using an ANN. This

model will serve as a reference to the initial work of selection and categorization of the indicators in the organization before the labeling process, as shown in *section V*.

V. Software Process Reference Model for Performance Indicators

As software organizations have specific indicators and processes, the processes included in the MPS-SW quality model were considered as reference for this work. Initially, seven processes are being considered as levels G and F models. The largest numbers of certified software organizations are in these two maturity levels (almost 90% of the certifications). An overview of this model is presented in the *subsection A*.

The *subsection B* presents an indicator model based on the processes of the maturity levels G and F of the MPS-SW model, to support the selection and clustering of indicators before the application of the ANN. The company's real indicators may be converted to the MPS-SW model indicator. However, not all model indicators can be converted into real indicators.

A. MPS-SW model

The software MPS-SW model is part of Brazilian Software Process Improvement Program (MPS.BR) created in 2003. This model is based on ISO / IEC 12207 (software lifecycle) and ISO / IEC 15504 (software evaluation). Currently there is an agreement between the Software Engineering Institute (SEI) and the Association for Promoting the Brazilian Software Excellence (SOFTEX) for joint assessment and certification of MPS-SW models and CMMI-DEV [17]. The MPS-SW model is designed to benefit mainly micro, small and medium software enterprises (MSME).

The MPS-SW model has seven maturity levels aiming a gradual implementation and certification from the first level: G. This maturity level includes two critical software processes for MSME: Requirements Management and Project Management. On each level are added new processes, as shown in **TABLE I**. This means that a higher level involves its processes more the processes from the lower levels. The highest level, A, involves all processes from the lower levels and emphasizes the continuity of the improvement in processes [18].

TABLE I. Processes added to each Maturity Level (ML) of the MPS-SW [18].

ML	PROCESSES
A	(*no new processes are added*)
B	Project Management (*new outcomes*)
C	Decision Management; Risk Management; Development for Reuse
D	Requirements Development; Product Design and Construction; Product Integration; Verification and Validation
E	Human Resources Management; Process Establishment; Process Assessment and Improvement; Project Management (*new outcomes*); Reuse Management
F	Measurement; Configuration Management; Acquisition; Quality Assurance; Project Portfolio Management
G	Requirements Management; Project Management

The utilization of performance indicators is required for the process from level F - that is kept up to level A. However, a good practice is to adopt performance indicators from level G.

B. Reference model for indicators

A business ontology was developed by Pizzoleto [16] organizing the levels G and F of the MPS-SW model. This ontology proposed performance indicators for almost all the expected results in the processes. **Fig. 3** shows the Protégé system screen with part of the measurements process of the F level, as described in the ontology.

Using this ontology, interviews in MPS-SW certified software companies were held. Based on literatures on performance indicators in Software Engineering, such as [19], [20], [21] and [22], an indicator model has been developed for the processes of the levels G and F. **Fig. 4** shows the indicator model with the following attributes: description, purpose, calculation method, measure unit, collection frequency, results presentation frequency and scope application (project, product and business).

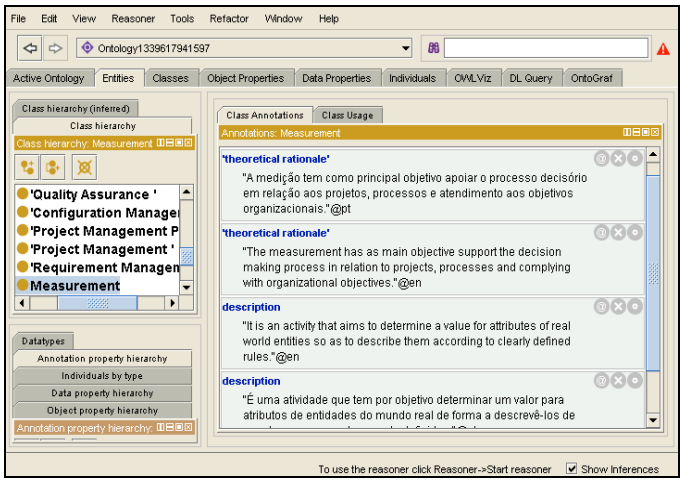

Fig.3. Measurement process from the MPS-SW ontology in the Protégé system [19].

Level	MPS-SW PROCESS	Description	Goal	Calculation	Application
	GQA4	Variation of planning	Shows the percentage of variation on sprint project planning	Total implementation of requirements planned on project - Total requirements implemented on project) * 100	Project
F	GRE1	Open requirements	Evaluate the number of requirements which were open on day	Sum of total open requirements which were open on day	Product
	GRE5	Impediment requirements	Shows the percentage of requirements responsible for project offside	(Total Impediment requirements / Total open requirements early in the project) * 100	Project
G	GRE4	Deployed requirements	Shows the number of requirements that have been implemented on day	Sum of all the implemented requirements on day	Product
	GPR9	Project Planning	Present the percentage of total time the project coordinator acted with planning	(Total hours that pointed only at the project planning / Total hours pointed out by project coordinator) * 100	Project

Fig. 4. Representation of the reference model of performance indicators.

A group indicator model was defined based on four perspectives of the Balanced Scorecard strategy (BSC): Financial, Customer, Internal business processes, and Learning

and growth. These perspectives reflect the vision and organizational strategy of a company [23]. The proposal was to associate the indicators used in the model presented in **Fig. 4** with the categories of BSC perspectives. **Fig. 5** illustrates some of these indicators associated with each view. This model supports the software organization to form sets of indicators to the process of labeling and training of ANN. The output result for each perspective will be the "status" of the analyzed set of indicators.

BSC	INDICATOR	TRAFIC LIGHT
Financial	...	
	Rework during implementation of the requirement	
	Efficiency of project planning	
	Impediment requirements	
	...	
Customer	...	
	Customer service policy	
	Number of notifications generated by customers	
	Cumulative requirements	
	...	
Internal business processes	...	
	Variation of planned	
	Project planning	
	Adherence to the processes	
	...	
Learning and growth	...	
	Training of programmers	
	Training of analysts	
	Training of project team	
	...	

Fig. 5. Indicators groups based on the BSC perspectives with three possible outputs.

VI. RESULTS

In order to evaluate the use of ANNs on analysis of performance indicators in software processes, experiments were performed with two different types of learning: supervised and unsupervised. Two algorithms of ANN supervised that are well-known in the literature were selected: Multilayer Perceptron (MLP) and K-Nearest Neighbor (KNN). The intention was to compare the results with the PCC algorithm, presented in *section III*.

A real indicators database used in software processes was applied as an input to the ANN. These indicators were chosen from the reference model of the proposed indicators in *section V*. The data were obtained from a company certified in level G of MPS-SW model. This company, a partner of the project, develops software for public management. The *subsection A* provides more information about the database used.

The output classification was made using very simple criteria, similar to a traffic light, as shown in **Fig. 6**. In this metaphor, the green light indicates that it is to continue the process execution because the results of the analysis in the indicators group report that the situation is satisfactory. The yellow warning signs to pay attention, because the level of satisfaction at the previously set target is regular. Already the red light indicates that the process should be stopped, to be unsatisfactory - too far from the established pattern.

Thus for ANNs, the "Green" label obtained "satisfactory", the "Yellow" label "Regular" and the "Red" label "Unsatisfactory". Therefore, if the output provided by the ANN obtained labeled "satisfactory" (Green), then the group of the indicators is analyzed in accordance with the desired - the processes are controlled. If the result of the output is "Regular" (Yellow), then the process has breaks, so greater attention is needed in case management. Finally, if the output is "unsatisfactory", the processes are not effective, requiring corrective actions.

Although the applications of the results of ANNs algorithms are presented in *subsection C*, some relevant information about the experimental procedure is presented in *subsection B*.

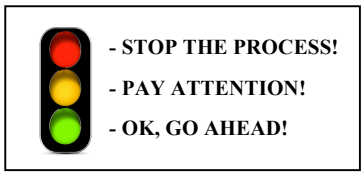

Fig.6. Traffic light metaphor to classification of ANN.

A. Indicators database used

The company that provided the performance indicator data has been working in software production for over 30 years and has hundreds of municipal governments as clients. The historical basis includes indicators of four years ago. This paper does not include the process of the data collection, but it is worth remembering that the MPS-SW certification (level G) provides some guarantees of the use of the best practices and data reliability.

The **Fig. 7** presents the statistical data related with the database used, considering three projects, identified as "A", "B" and "C". It is worth mentioning that the company's project managers collaborated with the labeling process of the samples:

- Number of instances: 300;
- Number of attributes: 3;
- Missing values: none;
- Distribution of grades: 33% for each one of the classes;
- Information about the attributes (indicators):
  a) Open requirements per daily in Project A;
  b) Open requirements per daily in Project B;
  c) Open requirements per daily in Project C;
  d) Cluster:
    - Cluster 1: Satisfactory;
    - Cluster 2: Regular;
    - Cluster 3: Unsatisfactory.

DESCRIPTION	MIN	MAX	AVG	STDEV
Project A	5	80	28.55	13.61
Project B	1	76	29.66	11.00
Project C	4	70	24.86	11.00

Fig. 7. Statistical data related on the database of the indicators used.

B. *Information about the experimental process*

Before presenting the results of the ANN techniques, it is important to present some relevant information and certain criteria adopted.

First of all, the software tool used for the application of supervised learning algorithms is called Weka, from the University of Waikato.

The Matlab system was used to implement the PCC algorithm proposed for analysis of the performance indicators (*section III*).

The time spent by the humans in labeling process of the sample was calculated from the sum of each measurement performed. This time will be important in the experiments presented in *subsection C*. Already the time spent in the review process made by the algorithms was measured by specific functions in each tool.

It is worth noting that each attribute had its normalized value before being used in the algorithms.

The training was made using ten different configurations, according to the percentage of trained sample: 2%, 4%, 6%, 8%, ..., 20%. Thus, each algorithm was executed ten times for each trained percentage, and the result presented is the average of the ten runs. This was necessary because the algorithms used are stochastic pattern and the data labeled in training were random.

C. *Application of Artificial Neural Network algorithms*

The grouping of indicators was based on the reference model presented in *section V*. The data from the same indicator were grouped but relating to three different projects (A, B and C). This kind of grouping allows a comprehensive analysis of the progress of different projects and/or the company's products, in relation to the selected requirement.

As presented in *subsection A*, 300 instances (100 of each cluster) and three attributes were used. As shown in *subsection B*, training of ANNs was carried out gradually from 2% to 2% by selecting that percentage of samples for training and the rest for validation.

The **Fig. 8** shows the result of the accuracy rate for each algorithm, according the increase in the percentage of trained samples. It can be noticed that the KNN algorithm presented good results with 2% trained samples, but the result got close to 81% accuracy when 20% of the samples were trained. The MLP algorithm achieved strong growth at the beginning and gradual growth until the end, almost reaching 85% accuracy. The PCC algorithm achieved about 76% accuracy with 2% of the trained samples. It had an increase of 6% with 4% of the trained samples and gradually increased from 82% to 84%, with 12% of the trained samples. Considering the range between 12% and 20% of trained samples, the PCC algorithm showed a significant increase of 84% to 92%. So, the best accuracy rate was achieved with the PCC algorithm for all variations of training, even with few data for training.

In relation to the accuracy by class, the **Fig. 9** shows the result of better time PCC algorithm. The algorithm correctly classified 96 samples as satisfactory and missed four, classifying them as Regular. In Cluster 2 (Regular) the PCC algorithm correctly classified 86 samples and 14 wrong, and 2 as satisfactory and 12 as cluster 3 (Unsatisfactory). The cluster 3 obtained 86 correct samples and missed 14, classifying it as Regular.

Another important point is the performance of PCC algorithm for analysis of performance indicators compared to a human (HUM). The **Fig. 10** shows that as the amount of data increases the difficulty of the human being also increases significantly. The measurement obtained by the analysis of a human being had an expert in the field during the data labeling step. This specialist visually analyzed the indicator results on each project and compared with its expected result.

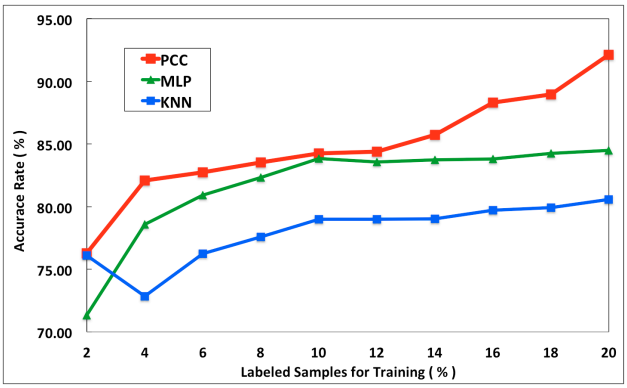

Fig. 8.Accuracy rate by technique.

		PREDICTION		
		Satisfatory	**Regular**	**Unsatisfactory**
ACTUAL	**Satisfatory**	**96**	4	0
	Regular	2	**86**	12
	Unsatisfactory	0	14	**86**

Fig. 9. Accuracy rate with cluster: actual vs prediction.

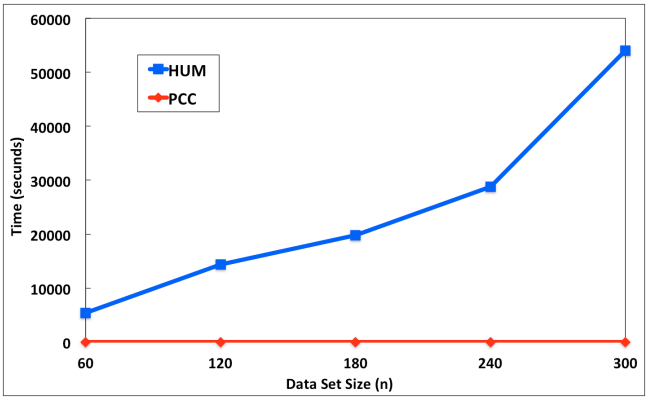

Fig10. Runtime: algorithm PC vs human (HUM).

VII. CONCLUSIONS

This paper proposes the use of semi-supervised ANN technique called Cooperation and Competition between particles (PCC) to support the performance indicator analysis

in processes of software engineering. Results with real data bases showed that the PCC algorithm achieved excellent accuracy rate, higher than the two supervised learning algorithms used.

The use of an ANN allows composed groupings indicators of different processes, products and designs and enables analysis of individual indicators. Tools that implement ANNs provide a differentiated overview of what is being measured, regardless of the complexity of the data. This contributes to the use of performance indicators in software development companies with different sizes, including micro, small and medium enterprises. Reference models proposed for selection and grouping of indicators contribute to assist organizations in selecting the appropriate settings to quality control processes.

REFERENCES

[1] PMI - Project Management Institute INC (Pennsylvania). Um guia do conhecimento em gerenciamento de projetos: Guia PMBOK. 4. ed. Newtown Square: PMI, 2008. 459 p.

[2] I. Sommerville, Engenharia de Software. 9. ed. São Paulo: Pearson, 2011. 529 p. Tradução de: Kalinka Oliveira; Ivan Bosnic.

[3] L. H. Boyd, J. F. A. Cox, "Cause-and-effect approach to analyzing performance measures". Production and Inventory Management Journal, v. 38, n. 3, p. 25-32, 1997.

[4] T. Mitchell, "Machine Learning". [s.i]: McGraw Hill, 1997.

[5] E. Alpaydin, "Introduction to machine learning". Cambridge, Ma: The MIT Press, 2004.

[6] S. Haykin. Neural Networks: a comprehensive foundation. Upper Saddle River, Nj: Prentice Hall, 1994. 768 p.

[7] Z. Kovacs, Redes Neurais Artificiais: Fundamentos e Aplicações. São Paulo: Edição Acadêmica, 1996.

[8] A. P. Braga, A. C. P. L. F. Carvalho, T. B. Ludemir, Redes Neurais Artificiais: teoria e aplicações. Rio de Janeiro: LTC, 2000.

[9] F. A Brave, L. Zhao, M. G. Quiles, W. Pedrycz, J. Liu, "Particle competition and cooperation in networks for semi-supervised learning". IEEE transactions on knowledge and data engineering, [s.i], 2009.

[10] F. A. Breve, Aprendizado de máquina utilizando dinâmica espaço-temporal em redes complexas. 2010. 165 f. Tese (Doutorado) - Curso de Ciências de Computação e Matemática Computacional, Departamento de ICMC-USP, Universidade de São Paulo, São Carlos, 2010.

[11] R. G. Mello, D. A. P. Junior, J. F. G., Oliveira, C. F. Bremer, Avaliação de desempenho para o gerenciamento estratégico do chão de fábrica. ANPAD. 14p, 2000.

[12] S.B. Neto, M. S. Nagano, M. B. C. Moraes, Utilização de redes neurais artificiais para avaliação socioeconômica: uma aplicação em cooperativas. R. Adm, São Paulo, v.41, n.1, p.59-68, 2006.

[13] I. Cattinelli, E. Bolzoni, M. Chermisi, F. Bellocchio, C. Barbieri, F. Mari, C. Amato, M. Menzer, A. Stopper, E. Gatti. Computational intelligence for the Balanced Scorecard: Studying performance trends of hemodialysis clinics. Artificial Intelligence in Medicine archive, v 58, I3, p. 165-173, 2013.

[14] O. Kutlubay, M. Balman, D. Gül, A. B. Bener, A Machine Learning Based Model for Software Defect Prediction, working paper, Bogazici University, Computer Engineering Department, 2005.

[15] SOFTEX - Associação para Promoção da Excelência do Software Brasileiro. MPS. BR Melhoria de processo do software brasileiro: Guia de Implementação – Parte 9: Implementação do MR-MPS em organizações do tipo Fábrica de Software, 2012a. Available at: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_9_20111.pdf>. Acesso em: 01 jun. 2015.

[16] A. V. Pizzoleto, Ontologia Empresarial no modelo MPS.BR visando modelagem de processos de negócios, com foco nos níveis G e F. Dissertação (Mestrado) – Universidade Estadual Júlio de Mesquita Filho, 2013.

[17] M. Kalinowski, G. Santos, S. Reinehr, M. Montoni, A.R. Rocha, K.C. Weber, G.H. Travassos, "MPS.BR: promovendo a adoção de boas práticas de engenharia de software pela indústria brasileira". XIII Congreso Ibero americano en "Software Engineering" (CIBSE), Cuenca, Ecuador, 2010.

[18] M. Kalinowski, K. Weber, N. Franco, E. Barroso, V. Duarte, D. Zanetti, G. Santos, "Results of 10 Years of Software Process Improvement in Brazil Based on the MPS-SW Model". 9th International Conference on the Quality of Information and Communications Technology, IEEE, 2014.

[19] A. V. Pizzoleto, H. C. Oliveira, Methodology for ontology development in support to the MPS model for software. In: International Conference on Software Engineering Research And Practice (Serp), 11. 2013, Las Vegas-Nevada, 7p, 2013.

[20] T. R. Ojha, Analysis of hey performance indicators in software development. Master on Science Thesis. Tampere University of Technology, 2014.

[21] G. Santos, M. Montoni, R. C. S. Filho, A. E. Katsurayama, D. Zanetti, A. O. S. Barreto, A. R. Rocha, Indicadores da Implementação do Nível E do MR-MPS em uma Instituição de Pesquisa. VIII Simpósio Brasileiro de Qualidade de Software.

[22] R. T. Moreira, G. N. Lima, B. B. Machado, W. T. Marinho, A. Vasconcelos, A. C. Rouiller, Uma abordagem para melhoria do processo de software baseada em medição. VIII Simpósio Brasileiro de Qualidade de Software.