AN ESTÍMATE OF THE NECESSARY EFFORT IN THE DEVELOPMENT OF SOFTWARE PROJECTS.

Martín Rodríguez Inés Galván Julio Hernández Pedro Isasi

Abstract

The estímate ofthe effort in the development of software projects has already been studied in the field of software engineering. For this purpose dijferent ways of measurement such as Unes of code andfunction points, generally addressed to relate software size with project cost (effort) have been used. In this work we are presenting a research project that deals with this field, us'mg machine learning techniques to predict the software project cost. Several public set of data are used. The analysed sets of data only relate the effort invested in the development of software projects and the size ofthe resultant code. For this reason, we can say that the data used are poor. Despite that, the results obtained are good, because they improve the ones obtained in previous analyses. In order to get results closer to reality we shouldfind data sets of a bigger size that take into account more variables, thus offering more possibilities to obtain solutions in a more ejficient way.

1. Introduction

The effort invested in a software project is probably one of the most important and most analysed variables in recent years in the process of project management. The determination of the valué of this variable when initiating software projects allows us to plan adequately any forthcoraing activities. As far as estimation and prediction is concerned there is still a number of unsolved problems and errors. To obtain good results it is essential to take into consideration any previous projects. Estimating the effort with a high grade of reliability is a problem which has not yet been solved and even the project manager has to deal with it since the beginning.

Several methods have been used to analyse data, but the reference technique has always been the classic regression method. Therefore, it becomes necessary to use some other techniques that search in the space of non linear relationships. This work presents a study of machine learning techniques in the task of predicting project cost from Unes of code and function points, using a set of examples.

These examples are real data measured in projects. The projects are of different types and cannot be mixed. There are general rules in order to estímate the effort, but they cannot be used in a general way, because each software development company, even of the same business field, has its own particular casuistry.

To solve this problem, it is interesting to use machine learning methods with inductive learning. This way it is possible to build up a prediction system to fit each company characteristics, taking into account real examples of projects developed by the firm. These prediction systems will adapt automatically to the company's idiosyncrasy.

This theme has been considered in the past, by means of neural networks (NN), fuzzy logic, case-based reasoning (CBR) and genetic programming (GP) [8], [12], [16], [19], but the results obtained are not quite convincing. For this reason, in this work, we developed a careful study of different techniques in order to analyse if it is possible to solve the problem in a more efficient way.

-

2. Framework

Some of the previous works in the field have built up models (through equations) according to the size, which is the factor that affects the cost (effort) of the project the most [8], [14]. The equation that relates size and effort can be adjusted due to different environmental factors such as productivity, tools, complexity of the product and other ones. The equations are usually adjusted by the analyst to fit the real data of the projects.

From this perspective, different equaüon patterns have come out [8], [11], but none of them has produced enough evidence to be considered the defínitive cost function, in case there is one.

Nevertheless, the characteristic that has to be satisfied by the estimation equation is: the model should be capable of doing its best on estimating reliabiy the majority of the real valúes»

As we mentioned above, it has not been possible until now to obtain an equation, set of equations or patterns of equations that can satisfy this premise, and therefore there is no reference or comparison parameter. Then it can be assumed that the equations are not a good tool to obtain an optimum prediction.

For this reason, this work aims more at the prediction of the effort without considering cost or production functions. Statistical and artificial intelligence methods (regression, neuronal networks, instance based learning and some other procedures) have also been used due to their capability to predict.

3. Machine Learning Methods Used

In this work, different methods based on instance selections have been used. In the following paragraphs the methods are briefly described.

3.1. Neural Networks

Human beings have a deep desire to reproduce cognitive skills by artificial means. The appearance of a new field of study called artificial intelligence proves the fascination of human beings for the understanding of intelligence.

One of the most developed fields in this área has been the "neural networks". Neural network could be described as the computerized simulation of the behaviour of parís of the human brain by replicating in a low scale the patterns that it performs to obtain results from perceived experiences.

Specifically, we are dealing with the analysis and reproduction of the learning and acknowledgement mechanism which some of the most evoived species possess. Artificial neural networks can be characterized as computational models involving functions such as capability to adapt and learn, clustering and parallel processing.

3.2. K*

The distance between instances can be defíned as the complexity of transforming one instance into another. Calculation of the complexity is done in two steps. First a finite set of transformations that map instances to instances is defíned, A program to transform one instance *a* to another *b* is a finite sequence of transformations starting at *a* and ending at *b*.

Following the usual development of the complexity theory such programs (sequences) are made "prefix free" by appending a termination symbol to each string. The usual definition of the (Kolmogorov [15]) complexity of a program is the Iength of the shortest string representing the program. Using this approach a Kolmogorov distance between two instances can be defined as the Iength of the shortest string connecting the two instances. This approach focuses on a single transformation (the shortest one), out of many possible transformations. The result is a measure of distance which is very sensitive to small changes in the instance space and which does not solve in a satisfactory way the smoothness problem. The K* distance tries to deal with this problem by summing up all possible transformations between two instances.

On real datasets it performs well against a range of both rule-based and instance-based learning schemes. The technique of summing up the probabilities of all possible transformation paths solves the smoothness problem and contributes strongíy to its good overall performance. The underlying theory also allows clean integration of both symbolic and real valué attributes and a principled way of dealing with missing valúes.

The implementation and results showed in [6] are a first implementation of K* method to predict real valué attributes.

33. Instance-Based Learner

Instance-based learner (IBL) classifíes an instance by comparing it to a database of pre-classified examples. The fundamental assumption is that similar instances will have similar classifications. The corresponding components of an instance-based learner are the distance function which determines how similar two instances are, and the classification function which specifies how instance similarities yield a final classification for the new instance. In addition to these two components, IBL algorithms have a concept description updater that determines whether new instances should be added to the instance data base and which instances from the database should be used for the classification. In simple IBL algorithms, after an instance has been classified, it is always moved to the instance database along with the correct classification. More complex algorithms may filter which instances are added to the instance database to reduce storage requirements and improve tolerance to noisy data.

The nearest neighbour algorithms are the simplest instance-based learners. They use some domain specific distance function to retrieve the single most similar instance from the training set. The classification of the retrieved instance is given as the classification for the new instance. Edited nearest neighbour algorithms are selective, in which instances are stored in the database and used in classification. *K-nearest neighbour* (KNN) algorithms are only slightly more complex. The k nearest neighbours of the new instance are retrieved and whichever class is predominant among them is given as the new instances classification. A standard nearest neighbour classification is the same as a k-nearest neighbour classifier for which k=L

1B1 is an implementation of a nearest neighbour algorithm with a specific distance function. Real valued attributes are normalised to a common scale so all attributes have equal weight and the missing values are assumed to be maximally different than the present value. *1B2* contains extensions to reduce storage requirements; only misclassified instances are saved.

IB3 is a further extension to improve tolerance to noisy data; instances that have a sufficiently bad classification history are forgotten and only instances that have a good classification history are used for classification.

4. Used Data

Eleven sets of data have been used. Each set shows information about certain amount of software development projects. For each project, there are two variables: one, (independent variable) that refers to the size of the generated code -measured in lines of code or function points-, and the other (dependant variable) that indicates the effort (time) invested in the development of projects. The data sets used in this work are the following:

1st Set: Abran and Robillard [1], Projects: 21. It is a subset out of a total of 36 projects. The code size is measured in function points and the effort in person-days.

2nd Set: Albrecht and Gaffney [2], Projects: 24. This data set corresponds to projects carried out by IBM and analysed by Shepperd and Schofield [20].

3rd Set: Bailey and Basili [3]. Projects: 18. The code size is indicated in thousands of lines of code (KLOC) and the effort is indicated in man-months.

4th Set: Belady and Lehman [4]. Projects: 33. The code size is indicated in lines of code (LOC) and the effort is indicated in man-months.

5th Set: Boehm [5]. Projects: 63. One of the most analysed sets.

6th Set: Heiat and Heiat [10]. Projects: 35. Small scale projects. The size of the code is indicated in LOC and the effort is indicated in person-hours.

7th Set: Kemerer [13]. Projects: 15. It was used to test different estimation methods. The independent variable used is function points.

8th Set: Miyazaki [18]. Projects: 47. The code size is indicated in KLOC.

9th Set: Shepperd and Schofield [20]. This data set has been used to test the method of estimation by analogy. The independent variable is the number of files.

10 Set: Deshamais [7]. Projects: 61. This data set relates function points to effort, using the concepts of *entity* and *transaction* to identify the function points.

11th Set: Kitchenham and Taylor [14]. Projects: 33. This data set is composed of 33 projects developed in the same language (S3, a high level language). The data relates LOC to effort (man-months).

It is important to highlight that, as it can be appreciated in the previous descriptions, the number of sets of projects is limited and also the number of projects in each set is small (the biggest set has 63 projects). These data sets have been obtained through the analysis made of some software development companies. These analyses, in most cases, have been performed along several years,

5, Results

As it can be appreciated, the size of the data sets is different, this is, the amount of projects is variable: For example, the biggest set (Boehm) has 63 and the smallest (Kemerer) has 15. In this work, a data analysis tool called WEKA [21] has been used. It includes methods such as: KNN, linear regression, neural networks and K* (previously described). It is necessary to notice that the results published in [8] were used as a reference in order to measure the efficiency of the previously mentioned methods. The methods used in [8] were approximation to square, cubic and logarithmic functions and genetic programming, indicated in this section as Curve and GP, respectively.

In *Table l* and *Table 2* are shown the results obtained using the previously described methods as prediction tools applied to the eleven data sets.

In the first two rows the results of the reference methods (GP and Curve) are shown. Next, the results obtained by the KNN method with k=3 (KNN-3) y k=4 (KNN-4). Experiments varying the value of parameter k have been carried out, but the best results were obtained using these values.

In the row number five, the level of prediction and error obtained with NN are shown. In this case, different architecture of back propagation of NN have been proved (different number of hidden layers and nodes per layer were used, but the best results were obtained using one hidden layer and 20 neurons in this layer). The NNs have been trained until they reach convergence.

Results obtained with linear and arithmetic regressions are shown in rows number 6 and 7.

Finally, the results provided by K* method, previously described, are shown.

The analyzed data set is indicated in each one of the columns through the name of the set.

5.1 Prediction capacity

One of the most important factors to take into account by a project leader is the effort required in the development of a software project. Thus, it is necessary to have a prediction method that generates appropriate predictions with the smallest possible error.

To measure the prediction capacity of the methods, the reliability level (in percentage) of the obtained results can be used. This will reflect how good (resemblance to the actual values) are the results produced by a given method.

Another measure to be considered is the error. The error indicates the difference between the predicted value and the real one. It can be considered of similar importance to the reliability level, but with certain caution. When making the cumulative of all the errors for a certain data set, if there was one value whose error was much bigger than the errors of the remaining ones and the error of these is minimal, it can slant the result by reflecting too big error, being opposed therefore, to the previous parameter.

To measure the prediction capacity of the methods, two well-known measures have been used: PRED and MMRE that are described next.

5.1.1. PRED (/)

Level / prediction (PRED(/)) can be defined as the quotient between the number of cases in which the estimated values are within the absolute *I* limit of the real values and the total number of cases:

Let

$$A=\{x\in\Re^n \quad / \quad x_i\neq 0\}$$
$$A=\{(x_1,x_2,\dots,x_n)\in\Re^n \ / \ x_i\neq 0,\ \forall i=1,2,\dots,n\}$$
$$f: A\times A\rightarrow\{0,1\}^n$$
$$f(x,y)=z$$
$$f((x_1,x_2,\dots,x_n),(y_1,y_2,\dots,y_n))=(z_1,z_2,\dots,z_n)$$

where

$$z_i\begin{cases}1 & \text{if } 1-l\leq\left|\dfrac{x_i}{y_i}\right|\leq 1+l \qquad \forall\ i=1,2,\dots,n\\ 0 & \text{otherwise}\end{cases}$$
$$g:\{0,1\}^n\rightarrow\mathbf{Z}$$
$$g(x)=z$$
$$g((x_1,x_2,\dots,x_n))=\sum_{i=1}^{n}x_i$$

Let $\hat{e},e\in A$

Prediction:

$$PRED(l)=\frac{g(f(\hat{e},e))}{n}$$

For this work a level of prediction 0,25 was considered

5.1.2. MMRE

Media Magnitude of Relative Error (MMRE) can be defined as:

$$MMRE=\frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i-\hat{e}_i}{e_i}\right|$$

Where e; is a real value of the variable in the project, ej is its estimate and n is the number of projects. It can be stated then that if the MMRE is low, we will be able to make a good number of predictions. The criterion to consider a model as good is that it has an MMRE < 0,25,

5.2 Analysis of Results

As it has already been explained, to measure the efficiency of the methods under study, the methods Curve and PG have been taken as reference. It is necessary to mention that, neither in the methods under study nor in the reference ones, testing instances have been used. All the elements of each one of the analysed sets have been used for training.

In *Table 1* the prediction results are shown. The best results in each one of the domains are indicated in bold.

In the domain Albrecht, the prediction that makes the method GP is of 64,00, being this the best result, but the difference with the following better one is small: the prediction of K* with a value of 62,50. Similar situations are given in the domain Bailey, where, although GP is the best (73,70), the result of K* (72,22) is very close.

Table 1. Obtained predictions with 25% level (best results in bold).

	DATA SET										
Method	Abran	Albrecht	Bailey	Belady	Boehm	Heiat	Kemerer	Miyaz.	Shepp.	Deshar.	Kitchen.
Curve	57,14	54,17	61,11	33,33	17,46	94,29	33,33	42,55	50,00	44,26	27,27
GP	77,30	**64,00**	**73,70**	35,30	15,60	94,40	62,50	47,90	57,90	51,60	32,40
knn-3	76,19	54,17	66,67	33,33	20,63	88,57	53,33	**57,45**	72,22	48,78	27,27
knn-4	76,19	45,83	66,67	33,33	19,05	91,43	53,33	55,32	50,00	41,46	39,39
NN	**80,95**	20,83	50,00	12,12	6,35	94,29	33,33	29,79	22,22	48,78	21,21
LR	66,67	33,33	61,11	12,12	9,52	91,43	13,33	14,89	44,44	39,02	12,12
AR	**80,95**	58,33	72,21	24,24	19,04	97,13	**73,33**	40,42	**83,33**	**58,53**	59,05
K*	**80,95**	62,50	72,22	**90,91**	**76,19**	**97,14**	66,67	44,68	61,11	46,34	**84,85**

Table2. Media Magnitude of Relative Error (best results in bold).

	DATA SET										
Method	Abran	Albrecht	Bailey	Belady	Boehm	Heiat	Kemerer	Miyaz.	Shepp.	Deshar.	Kitchen.
Curve	0,2364	**0,5313**	0,2935	0,6528	1,1336	0,0892	0,4435	0,3999	0,4489	0,5428	0,8458
GP	0,2560	0,5480	0,2690	0,7100	1,7810	**0,0870**	0,5840	0,5060	0,4560	0,6230	1,1430
knn-3	0,2510	0,6292	0,2546	0,9275	1,1651	0,1014	0,4357	**0,3980**	0,3422	0,3372	0,6662
knn-4	0,3013	0,6672	0,2587	1,0298	1,2550	0,1063	0,4838	0,3909	0,5509	0,4085	0,7267
NN	0,2178	1,7452	0,2808	1,1772	10,8029	0,1080	1,3578	0,7125	2,3270	0,3732	1,6275
LR	0,2722	0,8993	0,2665	1,3900	3,5691	0,1221	1,0583	1,4559	0,8571	0,4348	1,6900
AR	**0,1593**	0,7163	**0,2035**	1,3534	1,9786	0,0890	**0,2259**	0,5768	**0,1731**	**0,3331**	0,4545
K*	0,2511	0,6371	0,2450	**0,2352**	**0,3662**	0,0998	0,2654	0,4065	0,4364	0,3946	**0,1495**

On the other hand, in the domain Abran, certainly the best methods are K*, NN and AR (80,95), followed by GP (77,30), also with a minimal difference.

In the domain Belady, as in the previous one, the best method is K*, but here it is necessary to highlight that the difference with the following better one is considerable: K*=90,91 and GP=35,50. Important differences have also occurred in the domains Boehm, Heiat and Kitchenham where K* continues being the best method (76,19, 97,14 and 84,85 respectively) and knn-3 (20,63) and AR (97,13 and 59,05) are the next best ones. GP is lower than these ones.

In the domain Miyazaki the method that provides the better result was knn-3. The difference with the next best one (GP) is approximately of 10%.

In the domain Desharnais the best method is AR (58,53) and the next one is GP (51,60). In the domains Shepperd and Kemerer, the best method also AR (83,33 and 73,33 respectively), but it is followed by knn-3 (72,22) and K* (66,67) respectively. GP is lower than these ones.

Thus, it can be said that the best of the two reference methods -as for prediction reference- is GP. When GP presents good results, the difference with the methods studied in this work is small. On the other hand, when some of the methods under study produce more accurate predictions than GP the difference is considerable. The excellent results that K* provides must be pointed out.

In *Table 2* the results of the generated errors are shown. The best results in each one of the domains or data sets are indicated in bold.

Only in two of the 11 domains, the reference methods are better than the method studied in this work, but in these cases with minimal differences. For example: for the domain Heiat, the best is GP with an error of 0,0870, followed by AR with 0,0890 and Curve with 0,0892; for the domain Albrecht, the best is Curve (0,5313).

Knn-3 and AR also provide good results in different domains, but as in the previous case, the difference with the following best results is minimal: in the domain Miyazaki, the errors for knn-3=0,3980 and Curve=0,3999; in the domain Desharnais, AR=0,3331, knn-3=0,3372; in the domain Abran, AR=0,1593, NN=0,2178.

In three of the eleven domains, K* was the best and in all of them the difference of results between the two is considerable: in the domain Boehm the MMRE for K* is of 0,3662 and for Curve is of 1,1336; in the domain Kitchenham the MMRE for K* is of 0,1495 and for knn-3 is of 0,6662; in the domain Belady the MMRE for K* is 0,2352 and for Curve is 0,6528.

In *Table 3* and *Table 4* the prediction results and errors, respectively, are shown, corresponding to K* and K* with crossed validation. It can be appreciated that the difference of results among the two methods is important. This is because in the crossed validation were used some of the instances for testing, that way diminishing the efficiency of the method, due to the poverty of the model (reduced number of instances and variables).

Tabic 3. Prediction for K* and K* with cross validation.

		)Al \ ii									
Method	;XBnui::	Albrcphf	Bailey:	: BciSiJy :	M m	***mm***	**Kcmr-er**	Mijrafc'	:; Shepp,	DysllW.	Kitchen
*K**	80,95	62,50	72,22	90,91	76,19	97,14	66,67	44,68	61.11	46,34	84,85
K*CV	57,14	25,00	66,67	12,12	6,35	91,43	40,00	23,40	38,89	41,46	15,15

Table 4. MMRE for K* and K* with cross validation.

	DATA SET										
Method	•Ahiwrj;;:	Altirepht	Bailey	Belady	•Boe.bni	/llefai:	Kemerec	Miyaz,	Sliepp.	Deshsr.	Kitchen.
K*	0,2511	0,6371	0,2450	0,2352	0,3662	0,0998	0,2654	0,4065	0,4364	0,3946	0,1495
K*CV	0,3800	1,6991	0,3325	3,5386	9,4863	0,1222	0,7631	1,3663	0,8145	0,4433	2,1594

In order to visualize the behaviour of each one of the methods in different domains, as in prediction and error, some graphics are shown *(Figures 1, 2, 3, 4, 5, 6)*. The discontinuous line represents the value obtained by the reference method for the different domains. This value is indicated by y. To compare the level prediction, GP was considered as reference method, since it provides better results than Curve method. And for the error, Curve was considered as a reference method, given that it presents better results than GP.

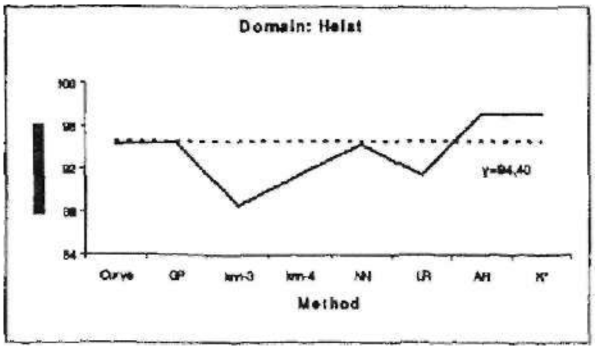

Figure 1. Prediction in Heiat domain.

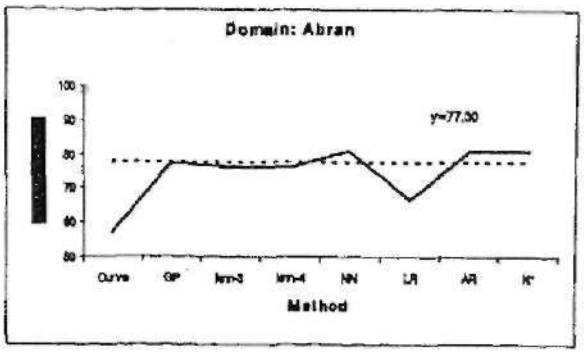

Figure 2. Prediction in Abran domain.

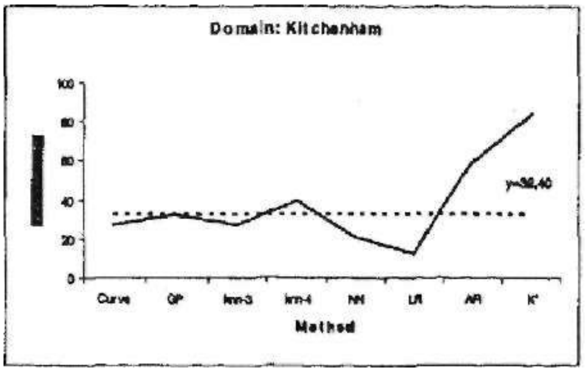

Figure 3. Prediction in Kitchenharo domain.

It can be seen that the prediction is not homogeneous, given that in the Heiat domain (see *Figure I)* only two methods improve (and just by a little) the reference method y = 94,40 (K* = 97,14; AR == 97,13), and in Kitchenham domain (see *Figure 3)* the K* method has gone beyond by fan y = 32,40, K*= 84,85.

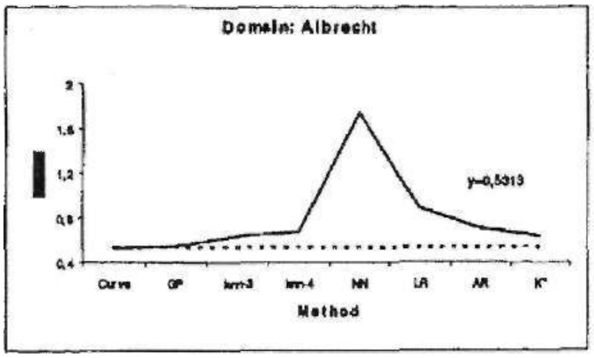

Figure 4. MMRE in Albrecht domain.

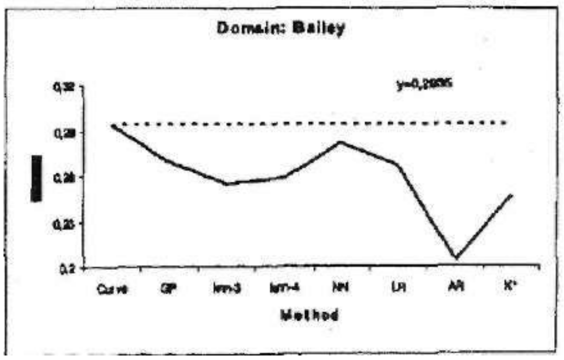

Figure 5. MMRE in Bailey domain.

Figure 4 (corresponding to Albrecht domain) shows that none of the methods enhanced the error produced by the reference one (Curve), but in *Figure 6* (corresponding to Desharnais domain) most of them have improved the results obtained by the reference method (y = 0,5428).

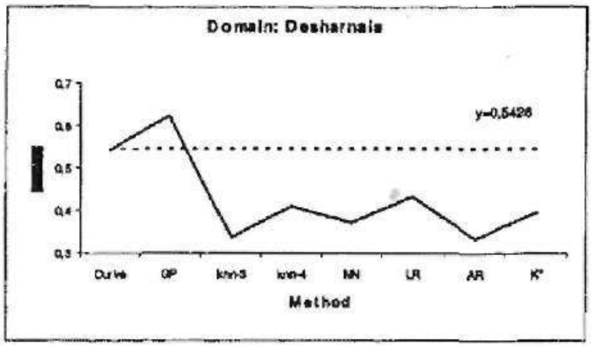

Figure 6. MMRE in Desharnais domain.

6. Conclusions

The estimation of the effort invested in the development of software projects can turn into a complicated problem to be solved if the appropriate models are not available. Unfortunately until this moment this is the situation, since there are not the necessary records in the software development companies. Years of investigation are required in order to obtain the volumes of information needed to carry out a prediction with a good level of reliability and with a low error margin.

The domains are not the most suitable, due to their size and limited number of variables, and because of the fact that they depend on the particular casuistry of each company.

The quality of the prediction can improve if more appropriate sets of data are available and a more deep study of the methods is performed.

In this work, machine learning techniques have been used for the task of predicting project cost using a set of domains that represents different types of projects. A study of the behaviour of different methods has been presented.

The obtained results show that the outcomes of the methods used are not homogeneous for all domains. In some cases, one method has proved to be better than the rest, and has obtained excellent results, whereas in others the predictions are far below the 25% threshold and/or the MMRE is far too big.

Nevertheless, the results obtained in this work are satisfactory since both the prediction levels and the error produced by some of the methods used can be considered excellent taking into account the limited characteristics of the models used. That is the case of the method K*.

7. References

[1] ABRAN, A., ROBILLARD, P. N., Function point analysis: an empirical study of its measurement process, in: IEEE Transactions on Software Engineering, 12 (12) (1996), pp. 895-910.

[2] ALBRETCH, A.J., GAFFNEY, J.R., Software function, source lines of code, and development effort prediction: **a** software validation, in: IEEE Transactions on Software Engineering, 9 (6) (1983), pp. 639-648.

[3] BAILEY, J. W., BASILI, V. R., A meta model for software development resource expenditures, in: Proceedings of die Fifth International Conference on Software Engineering, (1981), pp. 107-116.

[4] BELADY, L. A,, LEHMAN, M. M., The characteristics of large systems, in: P. Wegner (ed.), Research Directions in Software Technology, MIT Press, Cambridge, MA, 1979, pp. 106-138.

[5] BOEHM, B. W., Software Engineering Economics, in: Prentice Hall (ed.), Englewood Cliffs, N.L, 1981.

[6] CLEARY, J. G., TRIGG, L. E, K*: An Instance-based Learner Using an entropic Distance Measure, in: Proceedings of the IS* International Conference of Machine Learning (1995), pp. 108-114.

[7] DESHARNAIS, J. M., Analyse statistique de la productivite des projects de development en informatique a partir de la technique des points de fonction, in: Master Thesis, Univ. Du Quebec a Montreal, Decembre, 1988.

[8] DOLADO, J. J., A validation of component-based method for software size estimation, in: IEEE Transactions of Software Engineering, 26 (10) (2000), pp. 61-72.

[9] GRAY, A. R., MACDONELL, S. G., A comparison of techniques for developing predictive models of software metrics, in: Information and Software Technology, 39 (1997), pp. 425-437.

[10] HEIAT, A., HEAIT, N., A model for estimating efforts required for developing small-scale business applications, in: Journal of Systems and Software, 39 (1) (1997), pp. 7-14.

[11] HU, Q., Evaluating alternative software functions, in: IEEE Transactions on Software Engineering, 23 (6) (1997), pp. 379-387.

[12] IDRI, A., KHOSHGOFTAAR, T. M., ABRAN, A., Can Neural Networks be easily Interpreted in Software Cost Estimation?, in: 2002 World Congress on Computational Intelligence, Honolulu. Hawaii, May 2002

[13] KEMERER, C. F., An Empirical Validation of Software Cost Estimation Models, in: Communications of the Association for Computing Machinery, 30 (5) (1987), pp. 416-419.

[14] K1TCHENHAM, B. A., TAYLOR, N. R., Software project development cost estimation, in; Journal of Systems and Software, 5 (1985), pp. 267-278.

[15] LI, ML, VITANYI, P., An Introduction to Kolmogorov Complexity and Its Applications, in: Springer Verlag (ed.), New York, 1997.

[16] MACDONELL, S. G., SHEPPERD, M. J., Combining techniques to optimize effort predictions in software project management, in: Journal of System and Software, May 2003.

[17]MCKAY, B., WILLIS, M. J., BARTON, G. W., Steady-state modelling of chemical process systems using genetic programming, in: Computers and Chemical Engineering, 21 (9) (1997), pp. 981-996.

[18] MIYAZAKI, Y., TERAKADO, M., OZAKI, K,, NOZAKI, H., Robust regression for developing software estimation models, in: Journal of Systems and Software, 27 (1) (1994), pp. 3-16.

[19] QUAH, T. S., THW1N, M. M. T., Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics, in: IEEE International Conference on Software Maintenance, September 2003.

[20] SHEPPERD, M., SHOFIELD, C, Estimating software project effort using analogies, in: IEEE Transactions of Software Engineering, 23 (11) (1997), pp. 736-743.

[21] WITTEN, I. H., FRANK, E, Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations, in: Morgan Kaufmann Publishers (ed), San Francisco California, USA, 2000.