Journal of Software Engineering and Applications, 2012, 5, 270-276
http://dx.doi.org/10.4236/jsea.2012.54032 Published Online April 2012 (http://www.SciRP.org/journal/jsea)

Comparative Study of the Performance of M5-Rules Algorithm with Different Algorithms

Heetika Duggal, Parminder Singh

Department of Information Technology, Chandigarh Engineering College, Mohali, India.
Email: hitikaduggal@gmail.com, singh.parminder06@gmail.com

Received September 28th, 2011; revised February 5th, 2012; accepted March 1st, 2012

ABSTRACT

The effort invested in a software project is probably one of the most important and most analyzed variables in recent years in the process of project management. The determination of the value of this variable when initiating software projects allows us to plan adequately any forthcoming 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. In this study, performance of M5-Rules Algorithm, single conjunctive rule learner and decision table majority classifier are experimented for modeling of Effort Estimation of Software Projects and performance of developed models is compared with the existing algorithms namely Halstead, Walston-Felix, Bailey-Basili, Doty in terms of MAE and RMSE. The proposed techniques are run in the WEKA environment for building the model structure for software effort and the formulae of existing models are calculated in the MATLAB environment. The performance evaluation criteria are based on MAE and RMSE. The result shows that the M5-Rules have the best performance and can be used for the effort estimation of all types of software projects.

Keywords: Software Cost Estimation; Effort Estimation; Effort Estimation Models; Rule Generation; COCOMO Model; Conjunctive Rule Learner; Decision Table; M5-Rules Learner

1. Introduction

Software effort estimation is the critical part of software projects. Effective development of software is based on accurate effort estimation. Many quantitative software cost estimation models have been developed and implemented by practitioners in the past three decades. These include predictive parametric models such as Boehm's COCOMO models [1], Price S [2] and analytical models such as those introduced in [3-5]. An empirical model uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular database available. An analytical model, on the other hand, uses formulae based on global assumptions, such as the rate at which developer solves problems and the number of problems available [6]. A good software cost estimate should be conceived and supported by the project manager and the development team. It is accepted by all stakeholders as realizable. It is based on a well-defined software cost model with a credible basis. It is based on a database of relevant project experience and it should be defined in enough detail so that its key risk areas are understood and the probability of success is objectively assessed [7].

In this paper, the performance of single conjunctive rule learner, M5-Rules Algorithm and decision table majority classifier is compared for Modeling of Effort Estimation of Software Projects. The dataset is based on the cost factors in COCOMO II. The performance of the developed model was tested on NASA software project dataset and compared to the models presented in [8-11]. The developed models were able to provide good estimation capabilities as compared to other models provided in the literature.

The remainder of this paper can be described as follows:

Section 2 outlines the literature review about the techniques that are used for effort and cost estimation. Section 3 discusses the methodology adopted for generating and comparing a number of models. Section 4 highlights results of implementation. It discusses the results of the various models used for the effort estimation and Section 5 is all about conclusions of this research work.

2. Related Work

One of the most important problems faced by software developers and users is the prediction of the size of a programming system and its development effort. Software effort estimation stands as the oldest and most mature aspect of software metrics towards rigorous software measurement. Considerable research had been carried out in the past, to come up with a variety of effort prediction models. The background information of various software effort and estimation models to be used in this research work is discussed as follows:

- M. H. Halstead [11] in 1977 proposed the model which predicts the rate of error and do not require the in-depth analysis of programming structure. It proposed the code length and volume metrics. *Code length* is used to measure the source code program and *volume* corresponds to the amount of required storage space. Numerous industry studies support the use of Halstead in predicting programming effort and mean number of programming bugs. However it depends on completed code and has little or no use as a predictive estimating model.
- Walston-Felix Model developed by C. E. Walston and C. P. Felix in 1977 at IBM provides the relationship between delivered lines of source code (L in thousands of lines) and effort E (E in person-month).
- Doty Model [12] published in 1977, is used to estimate efforts for Kilo lines of code (KLOC). This model constitutes various aspects of the software development environment such as user participation, customer-oriented changes, memory constraints etc.
- Bailey and Basily [13] in 1981 described a meta-model which allows the development of effort estimation equations which are best adapted to a given development environment. The resultant estimation model will be similar to that of IBM and COCOMO is based on data collected by organization which captures its environmental factors and the differences among given projects.
- Albrecht has developed a methodology to estimate the amount of the "function" the software is to perform, in terms of the data it is to use (absorb) and to generate (produce). The "function" is quantified as "function points," essentially, a weighted sum of the numbers of "inputs", "outputs", "master files", "inquiries" provided to, or generated by, the software. Albrecht-Gaffney model established by IBM DP Services Organization, uses function point to estimate efforts.

Typical major models that are being used as benchmarks for software effort estimation are:

- Halstead
- Walston-Felix
- Doty (for KLOC > 9)
- Bailey-Basili

All these models have been derived by studying large number of completed software projects from various organizations and applications to explore how project sizes mapped into project effort. But still these models are not able to predict the effort estimation accurately.

As the exact relationship between the attributes of the effort estimation is difficult to establish, so machine learning approaches could serve as an automatic tool to generate model by formulating the relationship based on its training. In this proposed study, it is tried to build a more accurate model that can provide accurate estimates of effort required to build a software system when compared with the other models provided in the literature.

Cost Estimation Model

COnstructive COst MOdel (COCOMO) [14,15] is used to estimate the software cost. It was first published in 1981 (COCOMO 81) and in 1997 (COCOMO II). Some differences between COCOMO 81 and COCOMO II are as follows: COCOMO 81 has 63 data points, uses Kilo Deliverable Source Instructions (KDSI) to measure the project size and three development modes to be represented by scale factors. In contrast, COCOMO II has 161 data points, uses KSLOC project size, and five scale factors.

The COCOMO software cost model measures effort in calendar months of 152 hours (and includes development and management hours). COCOMO assumes that the effort grows more than linearly on software size; *i.e.* months = a*KSLOC^b*c. Here, "a" and "b" are domain-specific parameters; "KSLOC" is estimated directly or computed from a function point analysis; and "c" is the product of over a dozen "effort multipliers" *i.e.* months = a*(KSLOC^b)*(EM1* EM2 * EM3 * ...). In COCOMO I, the exponent on KSLOC was a single value ranging from 1.05 to 1.2.

In COCOMO II, the exponent "b" was divided into a constant, plus the sum of five "scale factors" which modeled issues such as "have we built this kind of system before?". The COCOMO I, effort multipliers are similar but COCOMO II dropped one of the effort multiplier parameters; renamed some others; and added a few more (for "required level of reuse", "multiple-site development", and "schedule pressure"). The effort multipliers fall into three groups: those that are positively correlated to more effort; those that are negatively correlated to more effort; and a third group containing just schedule information. In COCOMO I, "sced" have a U-shaped correlation to effort; *i.e.* giving programmers either too much or too little time to develop a system can be detrimental. The actual development effort is expressed in months (one month = 152 hours and includes development and management hours). The cost factors are shown in **Table 1**.

3. Used Methodology

The following steps are used for the comparative study:

Table 1. Cost factors in COCOMO II.

Cost Factors	Description
	Product
RELY	required software reliability
DATA	database size
CPLX	product complexity
	Computer
TIME	execution time constraint
STOR	main storage constraint
VIRT	virtual machine volatility
TURN	computer turnaround time
	Personnel
ACAP	analyst capability
AEXP	application experience
PCAP	programmer capability
VEXP	virtual machine experience
LEXP	language experience
	Project
MODP	modern programming practice
TOOL	software tools
SCEP	development schedule

3.1. Preliminary Study

First, Survey of the existing Models of Effort Estimation is to be performed.

3.2. Data Collection

Secondly, Historical Data being used by various existing models for the cost estimation is collected.

3.3. Effort Calculation Using Different Models

The following models are used for the data collected in the previous step and the effort for each developed approach is calculated.

- M5-Rules Algorithm
- Decision Table Majority Classifier
- Single Conjunctive Rule Learner
- Halstead Model
- Walston-Felix Model
- Bailey-Basili Model
- Doty Model

In addition to single conjunctive rule learner, M5-Rules Algorithm and decision table majority classifier, the different existing models: Halstead Models, Walston-Felix Model, Bailey-Basili Model and Doty Model are also used for the comparison of results. The equations for the existing models are as under: (**Table 2**)

Table 2. Existing effort estimation models.

Model Name	Effort
Halstead Model	Effort = $5.2(\text{KLOC})^{1.50}$
Walston-Felix Model	Effort = $0.7(\text{KLOC})^{0.91}$
Bailey-Basili Model	Effort = $5.5 + 0.73(\text{KLOC})^{1.16}$
Doty (for KLOC > 9)	Effort = $5.288(\text{KLOC})^{1.047}$

3.4. Performance Evaluation Criteria for Comparison of Models

The following performance criteria's are adapted to access and evaluate the performance of effort estimation models.

- **Mean absolute error (MAE)**

$$\text{MAE} = \frac{|a_1 - c_1| + |a_2 - c_2| + \cdots + |a_n - c_n|}{n}$$

where actual output is a, expected output is c.

Mean absolute error, MAE, is the average of the difference between predicted and actual value in all test cases; it is the average prediction error [16].

- **Root Mean-Squared Error (RMSE)**

$$\text{RMSE} = \sqrt{\frac{(a_1 - c_1)^2 + (a_2 - c_2)^2 + \cdots + (a_n - c_n)^2}{2}}$$

where actual output is *a*, expected output is *c*.

Root Mean Square Error, RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated [16]. It is just the square root of the mean square error.

The mean-squared error, MSE is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value.

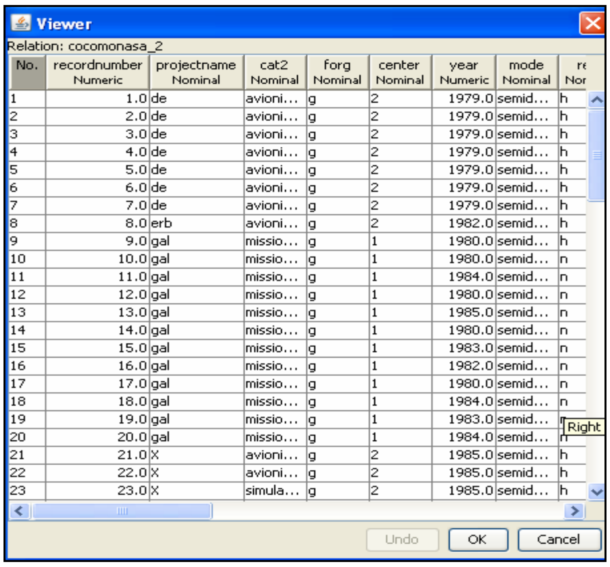
The root mean-squared error is simply the square root of the mean-squared-error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values. The mean absolute error and root mean squared error is calculated for each machine learning algorithm.

4. Results & Discussion

The implementation of used methodology is done in WEKA open source software [17], and certain calculations are performed in the MATLAB environment. Different steps discussed in the methodology are implemented and the comparative analysis of various models is done in terms of MAE and RMSE values.

Table 3 shows the publicly available PROMISE Software Engineering Repository data set which is used for the experimentation. It consists of 93 instances each with 23 input attributes and one output attribute named as effort. **Figures 1-3** describes the statistical analysis of different input attributes.

Table 3. COCOCMO NASA 2 data set.


Viewer

Relation: cocomonasa_2

No.	recordnumber Numeric	projectname Nominal	cat2 Nominal	forg Nominal	center Nominal	year Numeric	mode Nominal	re Nor
1	1.0	de	avioni...	g	2	1979.0	semid...	h
2	2.0	de	avioni...	g	2	1979.0	semid...	h
3	3.0	de	avioni...	g	2	1979.0	semid...	h
4	4.0	de	avioni...	g	2	1979.0	semid...	h
5	5.0	de	avioni...	g	2	1979.0	semid...	h
6	6.0	de	avioni...	g	2	1979.0	semid...	h
7	7.0	de	avioni...	g	2	1979.0	semid...	h
8	8.0	erb	avioni...	g	2	1982.0	semid...	h
9	9.0	gal	missio...	g	1	1980.0	semid...	h
10	10.0	gal	missio...	g	1	1980.0	semid...	n
11	11.0	gal	missio...	g	1	1984.0	semid...	n
12	12.0	gal	missio...	g	1	1980.0	semid...	n
13	13.0	gal	missio...	g	1	1985.0	semid...	n
14	14.0	gal	missio...	g	1	1980.0	semid...	n
15	15.0	gal	missio...	g	1	1983.0	semid...	n
16	16.0	gal	missio...	g	1	1982.0	semid...	n
17	17.0	gal	missio...	g	1	1980.0	semid...	n
18	18.0	gal	missio...	g	1	1984.0	semid...	n
19	19.0	gal	missio...	g	1	1983.0	semid...	n
20	20.0	gal	missio...	g	1	1984.0	semid...	n
21	21.0	X	avioni...	g	2	1985.0	semid...	h
22	22.0	X	avioni...	g	2	1985.0	semid...	h
23	23.0	X	simula...	g	2	1985.0	semid...	h

Name: mode Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (%)

No.	Label	Count
1	embedded	21
2	organic	3
3	semidetached	69

Figure 1. Statistical analysis of input attribute (mode) used in dataset.

Name: equivphyskloc Type: Numeric
Missing: 0 (0%) Distinct: 79 Unique: 69 (74%)

Statistic	Value
Minimum	0.9
Maximum	980
Mean	94.022
StdDev	133.598

Figure 2. Statistical analysis of input attribute (equivphysloc) in dataset.

Name: act_effort Type: Numeric
Missing: 0 (0%) Distinct: 74 Unique: 61 (66%)

Statistic	Value
Minimum	8.4
Maximum	8211
Mean	624.412
StdDev	1135.928

Figure 3. Statistical analysis of input attribute (act_effort) in dataset.

COCOMO attributes expressed in terms of classes {vl, l, n, h, vh, xh} is described in **Figure 4**.

Experimental Results of Machine Learning Algorithms

Historical COCOMO NASA 2/Software cost estimation dataset for the effort estimation is collected and used for the modeling in WEKA environment. The dataset consists of 93 NASA projects from different centers. The single conjunctive rule learner, M5-Rules Algorithm and decision table majority classifier are run in the WEKA environment and are evaluated by the cross validation using the 10 number of folds.

```
  increase  | acap | analysts capability
  these to  | pcap | programmers capability
 decrease   | aexp | application experience
   effort   | modp | modern programing practices
            | tool | use of  software tools
            | vexp | virtual machine experience
            | lexp | language experience
 -----------+------+---------------------------
            | sced | schedule constraint
 -----------+------+---------------------------
  decrease  | stor | main memory constraint
  these to  | data | data base size
  decrease  | time | time constraint for cpu
    effort  | turn | turnaround time
            | virt | machine volatility
            | cplx | process complexity
            | rely | required software reliability
```

Figure 4. Effort multipliers of COCOMO II model.

The Mean Absolute Error is taken as the average of the difference between predicted and actual value. Root Mean Square Error is taken as the measure of the differences between values predicted by a model and values actually observed from the thing being modeled. It is the average of the squared differences. The performance of the models is tested on the NASA software project data shown in **Table 3**.

Table 4 shows that the M5-Rules learner has the least MAE and RMSE value in comparison to Conjunctive Rule Learner and Decision table classifier. Hence the M5-Rules algorithm is the best methodology for classification as shown in **Figure 5**.

The 2d plot between the actual effort and the predicted actual effort shown in **Figure 6** gives the classifier errors. It gives the result of classification. Crosses represent the correctly classified instances.

The existing effort estimation models namely Halstead Model, Waltson-Felix Model, Bailey-Basili Model, Doty (for KLOC > 9) are run in the MATLAB environment. Effort Estimation for these models are evaluated by using the formulas mentioned in the **Table 2**. The Historical COCOMO NASA 2 dataset is used for effort estimation by existing models. **Table 5** describes the KLOC and actual effort pair used for the effort estimation. The KLOC is the Kilo lines of Code. E is effort in man-months. The performance of the machine learning algorithms and existing algorithms measured in MAE and RMSE values is shown in **Table 6**. **Figure 7** depicts the

Table 4. Performance of machine learning algorithms

Models	Performance Criteria	
	MAE	RMSE
M5 Rules	377.35	801.09
Decision Table	536.26	1127.37
Conjunctive Rule	695.31	1246.63

Table 5. KLOC and actual effort pair for effort estimation.

SNo.	KLOC	Actual Effort
1	25.9	117.6
2	24.6	117.6
3	7.7	31.2
4	8.2	36
5	9.7	25.2
6	2.2	8.4
7	3.5	10.8
8	66.6	352.8
9	7.5	72
10	20	72
11	6	24
12	100	360
13	11.3	36
14	100	215
15	20	48
16	100	360
17	150	324
18	31.5	60
19	15	48
20	32.5	60
21	19.7	60
22	66.6	300
23	29.5	120
24	15	90
25	38	210
26	10	48
27	15.4	70
28	48.5	239
29	16.3	82
30	12.8	62
31	32.6	170
32	35.5	192
33	5.5	18
34	10.4	50
35	14	60
36	6.5	42
37	13	60
38	90	444
39	8	42
40	16	114
41	177.9	1248
42	302	2400
43	282.1	1368
44	284.7	973
45	79	400
46	423	2400
47	190	420
48	47.5	252
49	21	107
50	78	571.4
51	11.4	98.8
52	19.3	155
53	101	750
54	219	2120
55	50	370
56	227	1181
57	70	278
58	0.9	8.4
59	980	4560
60	350	720
61	70	458
62	271	2460
63	90	162
64	40	150
65	137	636
66	150	882
67	339	444
68	240	192
69	144	576
70	151	432
71	34	72
72	98	300
73	85	300
74	20	240
75	111	600
76	162	756
77	352	1200
78	165	97
79	60	409
80	100	703
81	32	1350
82	53	480
83	41	599
84	24	430
85	165	4178.2
86	65	1772.5
87	70	1645.9
88	50	1924.5
89	7.25	648
90	233	8211
91	16.3	480
92	6.2	12
93	3	38

Table 6. Performance of machine learning algorithms along with other existing models.

Models	Performance Criteria	
	MAE	RMSE
M5 Rules	377.35	801.09
Decision Table	536.26	1127.37
Conjunctive Rule	695.31	1246.63
Halstead	6814	18963
Walston-Felix	583	1244.3
Bailey-Basili	472.2	1097.2
Doty	416.99	954.37

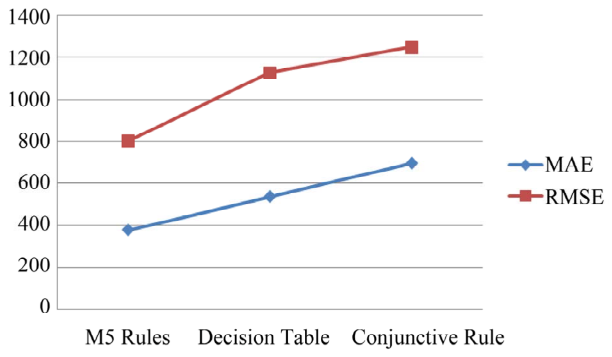


Figure 5. Comparison among machine learning models in terms of MAE and RMSE.

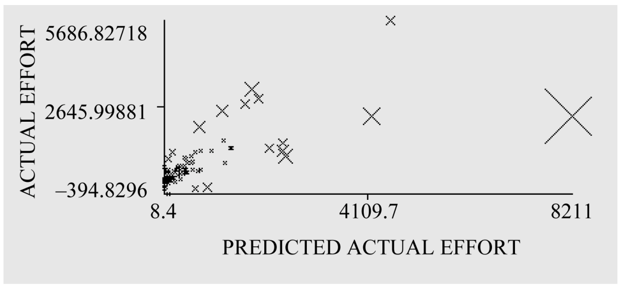


Figure 6. 2-d plot between the actual effort and the predicted actual effort for M5 rules learner.

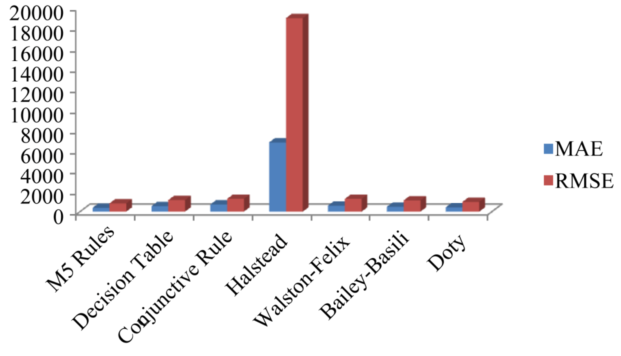


Figure 7. Comparison among different models.

comparison in terms of mean absolute error and root mean square error for different models.

5. Conclusion

In this paper, various Machine learning Algorithms, Conjunctive Rule Learner, M5-Rules algorithm and Decision Table Majority Classifier are experimented to estimate the software effort for projects. Performances of these models are tested on NASA Software Project Data and the results are compared with the Halstead, Walston-Felix, Bailey Basili, Doty Models mentioned in the literature. The proposed M5 Rule learner shows best results than among other algorithms experimented in the study with lower values of MAE and RMSE calculated as 377.35 and 801.09 respectively and able to provide good estimation capabilities as compared to other models. Hence, it is suggested to use of M5-Rules technique to build suitable model structure for the software effort.

REFERENCES

[1] B. W. Boehm, "Software Engineering Economics," 1st Edition, Prentice-Hall, Englewood Cliffs, 1981.

[2] S. Price, 2007. http://www.pricesystems.com

[3] G. Cantone, A. Cimitile and U. De Carlini, "A Comparison of Models for Software Cost Estimation and Management of Software Projects," In: *Computer Systems*: *Performance and Simulation*, Elsevier Science, Amsterdam, 1986, pp. 123-140.

[4] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering*, Vol. SE-4, No. 4, 1978, pp. 345-361. doi:10.1109/TSE.1978.231521

[5] N. A. Parr, "An Alternative to the Raleigh Curve Model for Software Development Effort," *IEEE Transactions on Software Engineering*, 1980, pp. 77-85.

[6] P. S. Sandhu, M. Prashar, P. Bassi and A. Bisht, "A Model for Estimation of Efforts in Development of Software Systems," *World Academy of Science*, *Engineering and Technology*, Vol. 56, 2009.

[7] W. Royce, "Software Project Management: A Unified Framework," Addison Wesley, Boston, 1998.

[8] B. W. Boehm, *et al*., "The COCOMO 2.0 Software Cost Estimation Model," *American Programmer*, 1996, pp. 2-17.

[9] C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation," *IBM Systems Journal*, Vol. 16, No. 1, 1977, pp. 54-73. doi:10.1147/sj.161.0054

[10] J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Codes, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, Vol. SE-9, No. 6, pp. 639-648. doi:10.1109/TSE.1983.235271

[11] M. H. Halstead, "Elements of Software Science," Elsevier,

New York, 1977.

[12] Doty Associates, Inc., "Software Cost Estimates Study," Vol. 1, 1977, pp. 77-220.

[13] J. W. Bailey and V. R. Basili, "A Meta-Model for Soft Ware Development Resource Expenditures," *Proceedings of the* 5*th International Conference on Software Engineering*, 1981, pp. 107-116.

[14] J. Baik, B. Boehm and B. Steece, "Disaggregating and Calibrating the CASE Tool Variable in COCOMO II," *IEEE Transactions on Software Engineering*, Vol. 28, No. 11, 2002, pp. 1009-1022. doi:10.1109/TSE.2002.1049401

[15] S. Devnani-Chulani, B. Clark and B. Boehm, "Calibration Results of COCOMO II.1997," 22*nd Annual Software Engineering Workshop*, NASA Goddard Space Flight Center, 1997.

[16] V. U. B. Challagulla, F. B. Bastani, I.-L. Yen and R. A. Paul, "Empirical Assessment of Machine Learning Based Software Defect Prediction Techniques," 10*th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, Sedona, 2-4 February 2005, pp. 263-270. doi:10.1109/WORDS.2005.32

[17] WEKA, 2007. http://www.cs.waikato.ac.nz/~ml/weka