Feature Analysis on Software Effort Datasets

Ekrem Kocaguneli
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505, USA
ekocagun@mix.wvu.edu

Katerina D. Goseva
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505, USA
katerina.goseva@mail.wvu.edu

ABSTRACT
Software related metrics can be grouped into three categories: Product, process and resource (PPR) metrics. For a successful productivity model the datasets are supposed to cover all these 3 categories [6]. However, publicly available software effort datasets were collected according to the needs of specific organizations, where data specification is highly dependent on the expert opinions. Furthermore, with feature selection methods it has been observed that not all the attributes are necessary for high accuracy values in software effort estimation. This may be due to noise in certain features, their information content or the fact that some are more correlated with the effort than the others.

In this study, we are analyzing how effort datasets are distributed into PPR metrics. We are also investigating the linear correlation between features to the effort through R-square analysis and finally we are conducting feature selection on each dataset to see whether R-square analysis is in agreement with feature selection. We believe that it is worth the effort to investigate the dataset features in terms of PPR metrics, linear correlation and feature selection due to at least 2 reasons: 1) Such an analysis will give us better understanding in terms of feature characteristics of datasets and 2) the relation between different analysis may provide insight into how and why different characteristics relate to each other.

1. INTRODUCTION
Although there are many publicly available datasets for software effort estimation, the problem of large variances in terms of estimation performance exists for these datasets [15]. We can talk about variance not only in terms of estimation performance, but also in terms of model variables. For example it was shown that even calibrated COCOMO model paremeters may vary drastically when they are re-learned from random sub-sampling of the same datasets that generated the actual model parameters [17]. Of course the variances in models and estimation results may be due to noise that is in the data. However, another explanation of this phenomenon may be hidden behind how we define our features inside datasets or behind our assumptions of these features.

Fenton proposes that for a successful productivity model the datasets are supposed to cover product, process and resource metrics [6]. In this research, we will use 3 well known software effort datasets to question this argument and will observe how features are distributed into PPR-metrics. Furthermore, we will obsreve the similarities and differences between datasets and present our opinions regarding our observations.

One assumption regarding the datasets used in this study is that the features defining these datasets are linearly correlated with the effort. In particular the attributes that define a project in terms of size (lines of code or function points) are supposed to have a strong linear relationship with the effort. To understand to what extent this proposition is true, we will use statistical R-Square method.

Another well known fact with the datasets used in this research is that feature subset selection has the high possibility of increasing the estimation accuracy. We will use wrapper subset selection algorithm to select the subset of features that yields the lowest error rate. Furthermore, we will observe if there is an obvious relationship between the features selected by wrapper and the features that gain high R-Square values.

In summary we can group our objectives in this research into two main groups:

1. To observe the distribution of features into PPR metrics
2. Investigate the existence and reasons of similarities or differences between high R-Square valued features and wrapper selected features

2. BACKGROUND
We can group software effort estimation studies basically into two categories [18]: Expert judgment and model-based techniques.

Expert judgment methods have a wide range of application and are one of the most preferred effort estimation techniques [8]. The application of an expert judgment method may follow either explicit guidelines (following a method like Delphi [2]) or may be conducted through implicit techniques (informal discussions among such experts). One of the foreseeable problems with expert-based estimates is that they may be affected negatively due to competing interests among the staff. For example, a faulty estimation of a senior expert may be taken over the more rational estimation made by a junior expert working for the same company. Another problem, which was indicated by Jorgensen et. al., is the poor capability of humans in terms of improving their own expert judgment [9].

Model-based techniques are the methods generated by using algorithmic and parametric approaches or by induced prediction systems. The former approach (algorithmic and parametric) fundamentally relies on the adaptation of an expert-proposed model to local data. A very well known example to such an approach is Boehm's COCOMO method [3]. The latter approach (induced prediction systems) is useful in the case where local data do not conform to the specifications of the expert's method. A few examples of induced prediction systems are linear regression, neural nets, model trees and analogies [13, 16, 19].

On the other hand, all of these systems are built on inherent assumptions. In the case where data violates such assumptions, patches are applied. A simple example of a patch is taking the logarithm of exponential distributions before linear regression [3, 11]. However, choosing the appropriate patch is also problematic to some extent and requires qualified experts.

Depending on the particular needs of an organization, different organizations are free to follow an expert judgment, a model-based approach, or some combination of the two in different settings. However, the goal of any estimation model is common: To attain high estimation accuracy. On the other hand, estimation accuracy does not depend only on the choice of estimation model. Another critical factor is the well defined and collected historical data that is kept up to date. In this research, we will take a look at particular characteristics of common software effort datasets and will explain the similarities as well as differences between these datasets.

3. DATASETS

In this study, we used three of the most commonly used datasets in software effort estimation domain: Cocomo81 and Nasa93 [3] as well as Desharnais [4]. Cocomo81 and Nasa93 datasets are made up of projects developed in NASA by both NASA itself or by subcontractors, whereas Desharnais dataset contains projects developed by a Canadian software house. Therefore, there is a huge amount of variety in the datasets used in this research in terms of their content, their size and their development locations.

Another criteria that we have considered while selecting these datasets was the size of the datasets. Since we question the linearity relationship of project features with effort in this research, the size of the datasets is an important factor. In order to evaluate the goodness of datasets in terms of size, Kitchenham and Mendes propose a quality scoring that consists of four values: poor (less than ten projects), fair (between ten to twenty projects), good (between twenty to forty projects) and excellent (more than forty projects) [12]. If we base our quality criteria for size by following the quality criteria proposed by Kitchenham and Mendes all the datasets we selected to investigate linearity assumptions rank as excellent quality. We provide the details such as instance and feature size regarding these datasets in Figure 1.

Dataset	**Features**	$T = \|Projects\|$	**Units**
Cocomo81	17	63	months
Nasa93	17	93	months
Desharnais	11	81	hours
		Total: 237	

Figure 1: 3 datasets used in this research contain a total of 237 projects. Datasets have different characteristics in terms of the number of attributes as well as the measures of these attributes.

4. ANALYSIS OF DATASETS

In this research we will analyze 3 very commonly used datasets according to different perspectives. The first perspective we will adopt while analyzing these datasets will be the distribution of features in each dataset into PPR-metrics [6]. After PPR-metrics, we will try to discover how much linear correlation exists between each individual feature and the independent variable by means of R-square analysis [1]. Finally we will take a look at feature selection on the datasets used in this research on the basis of linear regression and comment on the similarities and differences between individual linear analysis of features and their collective predictive power through linear regression (results of feature selection).

4.1 PPR-Metrics Analysis

In this section we will provide brief descriptions of process, product and resource metrics as they are provided by Fenton et. al. [6]. Furthermore, we will divide features of datasets used in this research (Cocomo81, Nasa93 and Desharnais) into these categories and comment on the percentage distribution of each dataset into these features.

The brief descriptions of PPR-metrics are as follows:

- Process Metrics: Intention of process metrics is to capture the software-related *activities*. For example number of requirement changes, effort and number of specification or coding faults are among process metrics.
- Product Metrics: The artifacts that are produced by process activities are regarded as product metrics. As examples for product metrics we can name modularity, reuse and algorithmic complexity.
- Resource Metrics: Entities that process activities require are basically resource metrics. Some of the resource metrics are size and communication level of teams as well as age of personnel.

Both Cocomo81 and Nasa93 contain datasets collected in accordance with the COCOMO method proposed by Barry Boehm [3]. According to COCOMO, each of of Cocomo81 and Nasa93 contains 17 attributes among which 16 are independent features and one is the dependent feature. 16 of the attributes that are independent variables define particular characteristics of software projects and 1 attribute that is a dependent variable indicates how long it took to complete a particular project in terms of man-months. The distribution of these attributes into PPR-metrics is provided in Figure 2. As we can see from Figure 2, 3 out of 17 attributes define product metrics, 7 attributes explain for process metrics and another 7 are used as resource metrics.

Product Metrics	**Abbreviation**
Lines of Code	LOC
Database size	DATA
Product complexity	CPLX
Process Metrics	**Abbreviation**
Modern programming practices	MODP
Use of software tools	TOOL
Required development schedule	SCED
Software Effort	EFF
Execution time constraint	TIME
Main storage constraint	STOR
Required software reliability	RELY
Resource Metrics	**Abbreviation**
Analyst capabilities	ACAP
Applications experience	AEXP
Programmer capabilities	PCAP
Virtual machine experience	VEXP
Programming language experience	LEXP
Virtual machine volatility	VIRT
Computer turnaround time	TURN

Figure 2: The PPR-metrics for Cocomo81 and Nasa93 datasets.

Unlike Cocomo81 and Nasa93, Desharnais dataset was collected in accordance with function points (FP) [10]. Therefore, Desharnais dataset contains completely different attributes. There are 11 attributes that make up the project description in Desharnais dataset. Among 11 attributes 10 can be defined as independent variables whereas 1 attribute (effort) is the dependent variable that is explained by the independent variables. Another difference between Desharnais dataset and the Cocomo81 as well as Nasa93 dataset is that the effort value in Desharnais dataset is man-hours, instead of man-months, which can yield better precision. The variable names, their abbreviations as well as their distributions into PPR-metrics are provided in Figure 3. From Figure 3 we see that 5 of 11 attributes make up the process metrics, 3 of the rest constitute the product metrics and another 3 explain for the resource metrics.

Product Metrics	**Abbreviation**
Scheduled time in months	Length
Count of logic transactions	Transactions
Number of entities in systems data model	Entities
Function Points	Points non adjust
Adjusted Function Points	Points Adjust
Process Metrics	**Abbreviation**
The year project ended	YearEnd
Transforms points function points	AdjustmentFactor
Effort spent in hours	Effort
Resource Metrics	**Abbreviation**
Team Experience	TeamExp
Manager Experience	ManagerExp
Language Type	Language

Figure 3: The PPR-metrics for Desharnais dataset.

After briefly commenting on individual datasets, we wanted to provide an overall view of the distribution of features into PPR-metrics. Figure 4 summarizes the percentage distribution of all attributes in each one of the 3 datasets used in this research. Since the attributes for Cocomo81 and Nasa93 are same to each other, their percentage values are exactly the same. As we can see from Figure 4, Cocomo81 and Nasa93 dataset separate more than 40% of their attributes in process metrics and resource metrics. The product attributes on the other hand get a share of only 17.65%, which is less than half of process or resource metrics. The distribution of features in Desharnais dataset on the other hand are much different than Cocomo81 and Nasa93. 45.45% of all the features in Desharnais dataset is separated for product metrics, whereas process and resource metrics get a share of 27.27%.

When we compare the percentage distribution of features in Cocomo81 and Nasa93 datasets to that of Desharnais, we see that the two methods (COCOMO and FP) focus on different parts of software production. When we consider the percentage values of PPR-metrics given in Figure 4 we can say that the focus of features in Cocomo81 and Nasa93 are on process and resource metrics whereas the product metrics have a relatively less importance. On the other hand, unlike COCOMO method Desharnais dataset places more importance on the product metrics (45.45%) and less importance on process and resource metrics (27.27% for each).

	Cocomo81	Nasa93	Desharnais
Product Metrics	17.65%	17.65%	45.45%
Process Metrics	41.18%	41.18%	27.27%
Resource Metrics	41.18%	41.18%	27.27%

Figure 4: The percentage distribution of all features to PPR-metrics for Cocomo81, Nasa93 and Desharnais dataset.

4.2 R-Square Analysis

R-Square analysis is used for statistical models whose purpose is to predict an outcome depending on the basis of other related information. It is fundamentally the proportion of variability in the data that is accounted by the statistical model [5]. In this research we are using R-square analysis in the context of linear regression, in which case R-square is the square of sample correlation coefficient between the outcome and the values that are used for prediction. In our case, outcome is the recorded effort value of a project and the values used for prediction are each single attribute that are recorded in the dataset.

To briefly summarize how R-square analysis works assume that the dependent variables in a dataset are called y_i, which represent the actual or observed values. Each actual or observed variable is associated with a modeled value, which we will represent with $\hat{y}_i$. We will measure the variability of dataset by sums of squares. In R-square analysis we will use 3 different sums of squares:

1. Total Variation: Total Sum of Squares (SST)
2. Explained Sum of Squares: Regression Sum of Squares (SSR)
3. Explained Variation: Residual Sum of Squares (SSE)

The formulas of SST, SSR and SSE are given in Equation 1, 2 and 3 respectively. In Equations 1, 2 and 3 n represents the size of the dataset, whereas bar over a variable (e.g. $\bar{y}$) represents the mean of all variable values [1].

[1] $\bar{y} = \sum_{i=1}^{n} y_i$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{1}$$

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \tag{2}$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{3}$$

The formulation defining the realationship between the aforementioned sum of squares is given in Equation 4.

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{4}$$

Given the sum of square formualtions and their relationships, now we can define R-Square. R-square is the ratio of explained sum of squares (SSR) to total variation (SST) as in Equation 5.

$$R^2 = \frac{SSR}{SST} \tag{5}$$

In this research, we have used the R-square analysis whose explanation was given above and we used it in the context of linear regression. In that respect, the R-square values can range from 0 to 1 and its value gives us information regarding how well the model fits to data. The values closer to 1 means that the model fits the data points well and an R-square value of 1 means a perfect fit of regression line to the data. As the R-square value gets closer to 0, then this means that there is not a good fit of model to dataset instances.

We have computed the R-square values for each attribute in every dataset. The R-square values for Cocomo81 and Nasa93 are given in Figure 5 in a descending order. Although these features are supposed to be linearly correlated with effort, the highest R-Square value for Cocomo81 is 0.43 and 0.35 for Nasa93, which are both very low to be considered important. For some other attributes, the R-Square value goes down to 0. Therefore, we cannot say that either one of 16 independent features have a high correlation and is responsible for explaining the variation in the recorded effort values. However, the R-Square values can still give us some idea regarding which attributes have more explanatory power in terms of variation in the effort values. As we can see from Figure 5, the highest R-Square value belongs to LOC attribute in both of the datasets, meaning that LOC is better than any other attribute in terms of explaining the variation in effort value. Furthermore, TIME and RELY attributes have R-Square values that are relatively higher than the rest of the attributes. We will compare the results of R-Square analysis with the feature subset selection results in Section 5.

Attributes	**Cocomo81**	**Nasa93**
LOC	0.43	0.35
TIME	0.02	0.29
RELY	0.04	0.23
STOR	0.01	0.14
CPLX	0	0.1
DATA	0.2	0.04
PCAP	0.02	0.03
TOOL	0	0.02
VEXP	0	0.02
VIRT	0	0.01
ACAP	0.02	0.01
AEXP	0	0.01
LEXP	0.01	0
MODP	0.07	0
SCED	0	0
TURN	0.04	0

Figure 5: The R-square values for Cocomo81 and Nasa93 datasets. R-square values are sorted in a descending order. All the R-Square values are low to suggest a significant meaning. However, the highest R-Square value belongs to LOC, which means that LOC can better explain the variations in the dependent variable when compared to other attributes.

In Figure 6 we see the R-Square values for the attributes of Desharnais dataset. The results of Figure 6 are somewhat similar to the results of Cocomo81 and Nasa93 datasets, in the sense that all the R-Square values are quite low. The rule of tumb is that we expect to observe an R-Square value of higher than 75% or even 80% to claim that the attribute is really successful in terms of explaining the independent variable. However, as we see in Figure 6, none of the features can attain such high R-Square values. The highest R-Square value belongs to PointsAdjust feature with a value of 0.55 and it is followed by PointsNonAdjust and Length with R-Square values of 0.5 and 0.48 respectively. Therefore, we cannot say that any of the features can substantially account for the variation of dependent variable effort. On the other hand, we can still use these R-Square values to give us a hint regarding which features are more important than the others in terms of explaining the effort value variation.

Attributes	**Desharnais**
PointsAdjust	0.55
PointsNonAdjust	0.5
Length	0.48
Transactions	0.34
Entities	0.26
Adjustment	0.22
Language	0.07
TeamExp	0.06
ManagerExp	0.03
YearEnd	0

Figure 6: The R-square values for Desharnais dataset. R-square values are sorted in a descending order.

4.3 Feature Selection Analysis

This subsection will summarize our results of feature subset selection. Although feature subset selection has been used in literature to improve the accuracy values in software effort estimation systems [14–16], our reason for using feature subset selection will be different from those studies.

In our research, we intend to use feature subset selection on top of linear regression to observe whether the highest accuracies are attained with the features that have also attained highest R-Square values. In other words, we want to observe whether features that can individually explain the variations in the dependent variable better than other attributes come together for higher accuracy values or whether the individual performance has no relation when attributes are combined for achieving the highest estimation accuracies.

There are a number of feature subset selection algorithms in the literature such as principal component analysis (PCA), linear discriminant analysis (LDA) and wrapper [1]. Actually PCA combines the features to make better estimations rather than selecting out some features. However, since it is an unsupervised method, it had found its area of implementation in the literature. LDA and wrapper are supervised methods and they select out individual features out of a set of all features. Among these feature subset selection algorithms we will use wrapper due to 2 reasons: 1) Wrapper requires a predictive algorithm to run, which in our case will

Cocomo81		**Nasa93**		**Desharnais**	
Wrapper	R-Square	Wrapper	R-Square	Wrapper	R-Square
RELY	RELY	RELY	RELY	Length	Length
MODP	TIME	TIME	TIME	PointsNonAdjust	PointsNonAdjust
LOC	LOC	VEXP	STOR	PointsAdjust	PointsAdjust
		LEXP	CPLX	Language	Transactions
		MODP	DATA		
		LOC	LOC		

Figure 7: Wrapper results compared with R-Square analysis for all datasets. Starting from the highest R-Square-scored features, we selected out as many features as those that were selected by wrapper. The dark-highlighted cells are the ones that are common in R-square analysis and wrapper.

be the linear regression and 2) wrapper has been reported to yield better results in comparison to other methods [15].

The fact that wrapper needs a predictive algorithm actually fits perfectly to our case, because we are willing to see the performance of features when linear regression was applied. Wrapper basically works by trying out every possible combinations of all features subject to the predictive algorithm and it selects out the subset of features that has yielded the highest accuracy [1]. Therefore, for a dataset of size n, wrapper searches a set of "$2^n - 1$" combinations[2]. We have used the WEKA toolkit for our experimentation that included linear regression as well as wrapper feature subset selection [7].

Figure 7 summarizes the results of feature subset selection via wrapper as well as the the comparison of wrapper to R-square results for all datasets. As we can see the number of features selected from Cocomo81, Nasa93 and Desharnais datasets are 3, 6 and 4 respectively. What we can call interesting in Figure 7 is that although Cocomo81 and Nasa93 are both NASA projects and share the same features, wrapper selects twice as more features for Nasa93 in comparison to Cocomo81.

Figure 7 also gives us information regarding the common features that have the highest R-Square values and that are also selected by wrapper. When reporting the features in Figure 7, we restricted the number of features to be shown to that of features selected by the wrapper, i.e. if wrapper has selected n features out of dataset x, then we select out $n - many$ features that received the highest R-Square values. The dark-shaded cells in Figure 7 are the features that are the common to top n R-Square features and to wrapper selected features. In Section 5, we provide the detailed comments for each dataset.

5. RESULTS

This section presents the results we have elicited after our experimentation. We will address each dataset separately in related subsections and provide details regarding the feature selection of wrapper as well as R-Square. Probably one fact we need to clear out before continuing with the results is that R-Square analysis in fact *does not* select any features. When we refer to R-Square selected features, we use it to refer to the features that had the highest n R-Square values, where n is the number of features selected by wrapper for the same dataset.

5.1 Cocomo81

As we can see from Figure 7, 2 out of 3 features in Cocomo81 dataset are common between R-Square and wrapper. The common features in Cocomo81 dataset are RELY (required software reliability) and LOC (lines of code). LOC had received the highest R-Square values in both Cocomo81 and Nasa93, meaning that its ability to explain for the variation in effort is higher than anyone of the other features in the same datasets. Furthermore, it is common sense that more LOC will result in bigger projects in terms of project size and will ultimately result in longer time spent in finalizing this project. Therefore, selection of LOC is no surprise. RELY also has a high R-Square value and has been selected by wrapper. Although the impact of RELY is not as obvious as LOC, we can still say that the more reliable the software is required to be, the more effort has to be put into each phase of software development, particularly in the testing phase. Therefore, common selection of RELY also makes sense. The uncommon features for Cocomo81 are TIME (Execution time constraint) selected by R-Square and MODP (Modern programming practices) selected by wrapper. Selection of MODP by wrapper although it does not have a high R-Square value is a nice example of the fact that some features make sense when they are used together with other features.

5.2 Nasa93

There is a substantial difference between the results of Nasa93 and the results of Cocomo81. The two datasets have exactly the same features that define the individual projects and they both contain software projects that were developed for NASA. Therefore, the intuition is to expect similar re-

[2]All subsets of a set of size n minus the empty set.

sults, at least to some extent. However, as we can see from Figure 7 wrapper has selected out 6 features from Nasa93 dataset, whereas it was 3 datasets for Cocomo81 datasets. Although 3 of the 6 features that were selected from Nasa93 dataset are the same to those of Cocomo81, it is interesting to see additional 3 features. Furthermore, similar to Cocomo81 results, in Nasa93 RELY and LOC are common to wrapper and R-Square selection. The third feature that is common to wrapper and R-Square in Nasa93 is TIME (Execution time constraint). TIME was not a common feature for Cocomom81 dataset.

The remaining 3 features for Nasa93 dataset are not common between wrapper and Cocomo81. Since we know that wrapper has searched through all the combinations of features and has selected the combination that yielded highest accuracy value for linear regression, we again observe the effect of low R-Square valued features coming together to generate higher estimation performance.

5.3 Desharnais

Desharnais dataset has the highest similarity between R-Square and wrapper selected features. Out of 4 features that were selected by wrapper, 3 of them are common to R-Square as well. In other words, 75% of the features are common between R-Square and wrapper. Furthermore, common features are in fact all define the software in terms of its size (PointsAdjust, PointsNonAdjust) and schedule (Length). Therefore, we can say that size and schedule related features really are important to define the projects in Desharnais dataset. On the other hand, the fourth feature selected by wrapper and R-Square are Language and Transactions respectively. The different features have big differences in terms of their R-Square values: Language has an R-Square value of 0.07 and Transactions has an R-Square value of 0.34, which suggests that its not the combination of features with highest R-Square values that make up the best estimation, but the features that explain more together.

6. THREATS TO VALIDITY

Although we tried to use widely used datasets, these datasets are not the only ones that are used in software effort estimation domain. Therefore, some other datasets may yield different results in comparison to our results and this is one threat to the validity of our results.

Another possible threat to the validity of our results is the choice of linear regression. Since we are using R-Square analysis, we selected out linear regression for combining with wrapper. However, depending on the choice of the algorithm, the wrapper selected features may change.

Finally, we based our assumption on the premise that wrapper selects the best combination of features that yield the lowest error rate. The fact that wrapper selects the lowest error rate yielding features is true, however we have not calculated the actual magnitudes of error in this research. Therefore, we only know that wrapper generates better results than R-Square selected features, yet we do not know exactly how much better results it generates.

7. CONCLUSION

In this research we have tried to understand two facts concerning software effort datasets:

1. Distribution of features into PPR metrics
2. Existence and reasons of similarities or differences between high R-Square valued features and wrapper selected features

While trying to understand these facts, we made use of 3 common effort estimation datasets. We went through the definitions of each feature and distributed them into 3 categories of PPR-metrics. We also calculated R-Square values for each feature in these datasets and applied wrapper feature subset selection algorithm combined with linear regression to select out features.

We observed that -at least for the datasets we used in this research- the distribution of features into PPR-metrics do not follow a balanced distribution, i.e. the distribution of features into 3 different segments that define a software product appear to be random. More importantly, the percentage of PPR-metrics in different datasets show big variations. This fact may hint to two conclusions: 1) Different needs of different environments affect how the features are selected and unique development environments choose different features or 2) The selection of features merely depend on the judgment of the experts in a particular development environment and the experts do not put a strong emphasis in giving equal weight to PPR-metrics. Since we do not have much knowledge regarding the development environments for any of the 3 datasets, we cannot decide which one of the 2 conclusions should be suggested as the reason of uneven distribution of features into PPR-metrics.

As for our second goal, we are confident that there are similarities between wrapper selected and R-Square selected features. In other words, some of the features that have high R-Square values are selected by wrapper algorithm. However, there are also considerable differences between the wrapper selected features and R-Square selected features. For example in Nasa93 dataset, 3 out of 6 wrapper selected features are different than R-Square selected features. These results show that although high R-Square valued features are important for high estimation accuracy, high R-Square value is not the only factor for estimation accuracy. As we see in each one of the 3 datasets, combination of some high R-Square valued features with some other low R-Square valued features creates better estimation than choosing only the features that had high R-Square values.

8. REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.

[2] B. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches: A survey. *Annals of Software Engineering*, 10:177–205, 2000.

[3] B. W. Boehm. *Software Engineering Economics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[4] J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des

point des fonction. Master's thesis, Univ. of Montreal, 1989.

[5] R. G. Douglas and J. H. Torrie. *Principles and procedures of statistics.* New York, McGraw-Hill, 1960.

[6] N. E. Fenton. Software metrics: A rigorous approach. 1991.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[8] M. Jorgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70:37–60, February 2004.

[9] M. Jorgensen and T. Gruschke. The impact of lessons-learned sessions on effort estimation and uncertainty assessments. *IEEE Trans. Softw. Eng.*, 35(3):368–383, May-June 2009.

[10] B. Kitchenham and K. Känsälä. Inter-item correlations among function points. In *ICSE '93: Proceedings of the 15th international conference on Software Engineering*, pages 477–480, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[11] B. Kitchenham and E. Mendes. Why comparative effort prediction studies may be invalid. In *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, pages 1–5, New York, NY, USA, 2009. ACM.

[12] B. A. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Trans. Softw. Eng.*, 33(5):316–329, 2007.

[13] E. Kocaguneli. Better methods for configuring case-based reasoning systems. Master's thesis, 2010.

[14] Y. Kultur, B. Turhan, and A. B. Bener. ENNA: software effort estimation using ensemble of neural networks with associative memory. In *SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 330–338, New York, NY, USA, 2008.

[15] K. Lum, T. Menzies, and D. Baker. 2cee, a twenty first century effort estimation methodology. *ISPA / SCEA*, pages 12 – 14, 2008.

[16] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Trans. Softw. Eng.*, 32:883–895, 2006.

[17] T. Menzies, D. Port, Z. Chen, and J. Hihn. Simple software cost analysis: safe or unsafe? *SIGSOFT Softw. Eng. Notes*, 30(4):1–6, 2005.

[18] M. Shepperd. Software project economics: a roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 304–315, 2007.

[19] M. Shepperd and G. Kadoda. Comparing software prediction models using simulation. *IEEE Trans. Softw. Eng.*, pages 1014–1022, 2001.