Software Effort Estimation: Harmonizing Algorithms and Domain Knowledge in an Integrated Data Mining Approach

Jeremiah D. Deng, *Member, IEEE,* Martin K. Purvis, *Member, IEEE,*
and Maryam A. Purvis

Abstract

Software development effort estimation is important for quality management in the software development industry, yet its automation still remains a challenging issue. Applying machine learning algorithms alone often can not achieve satisfactory results. In this paper, we present an integrated data mining framework that incorporates domain knowledge into a series of data analysis and modeling processes, including visualization, feature selection, and model validation. An empirical study on the software effort estimation problem using a benchmark dataset shows the effectiveness of the proposed approach.

Keywords: software effort estimation, machine learning

I. Introduction

Data mining research has made prominent advances in recent years, with various computational methods contributed from fields such as statistics and machine learning being explored and applied in many application domains. On the other hand researchers have also realized the lack of incorporating domain knowledge in data mining solutions [14], [21]. Without proper and sufficient use of domain knowledge, data mining applications may risk the danger of choosing the wrong or suboptimal algorithms or models. Apart from this lacking of the incorporation of domain knowledge, an automated data mining process is prone to lead to the misinterpretation of data analysis outcomes, and therefore reduce confidence in the usage of these methods.

J. D. Deng is with the Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand. E-mail: ddeng@infoscience.otago.ac.nz.

M. K. Purvis is with the the Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand. E-mail: mpurvis@infoscience.otago.ac.nz.

M. A. Purvis is with the the Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand. E-mail: tehrany@infoscience.otago.ac.nz.

Despite these widely accepted notions, integrating domain knowledge into data mining solutions is challenging. Some domain knowledge is difficult to put into explicit form, such as rules. Often, there are inconsistencies between what computer algorithms suggest and what human experts understand.

Software metric Making realistic estimates of software development effort has been desired by the software industry for a long time. An accurate effort prediction can benefit project planning, management, and better guarantee the service quality of software development. The importance of software effort modeling is obvious and people have spent considerable effort in collecting project development data in large quantities. While expert judgment remains widely used, there is also increasing interest in applying statistics and machine learning techniques to predict software project effort. In recent years, with the release of software development data into the public domain such as those from ISBSG [2], the prospects of building powerful, practical estimation tools seem more favourable, even when formidable challenges still remain.

In this article, we present a case study of software effort estimation that combines the use of data mining algorithms and domain knowledge. While various regression techniques have previously been proposed in software effort estimation, few have paid attention to such important issues as feature selection, model selection and validation. In this paper, we have adopted an integrated data mining approach, where data visualization, feature selection, and modeling have been conducted, and all these processes interact with relevant domain knowledge. Through such interaction, data analysis processes can often be assisted by domain knowledge, and the results confirmed. When there are discrepancies between algorithm outcome and domain knowledge, this can help either to draw attention to the corresponding issues (which can be further analyzed in latter data processing), or even correct possible misunderstandings on the human side. In the end, domain knowledge also helps to validate the prediction model as well as assist in the interpretation of its final outcome, making the entire data analysis and modeling process both relevant and effective.

The remainder of the paper is organized as follows. In Section II we briefly review a few relevant works on software effort estimation. Section III presents the basic computational methods we have adopted. The case study is presented in Section IV. Finally, we conclude the paper with a short discussion.

II. Relevant work

Jeffery *et al.* studied using public domain metrics for software development effort [6]. They investigated the ISBSG dataset but warned of the use of it for prediction purposes, suggesting the use of company-specific data instead. Mendes *et al.* [10] looked at the comparison of some effort estimation techniques for web hypermedia applications, including case-based reasoning and linear and stepwise regression. Two machine learning methods were used in Srinivasan and Fisher [22] to build estimators from historical data and the model sensitivity of these methods was discussed. A notable work by Shepperd and Schofield [18] investigated the use of case-based reasoning (CBR). Oligny *et al.* [12] explored the relation between effort and duration in software engineering projects.

A number of machine learning approaches for software effort estimation were investigated by Mair *et al.* [9], including CBR, artificial neural networks (ANNs) and rule induction. It was found that although ANNs have superior accuracy, their configuration is problematic and it is necessary to explore further the interaction between the end user and the prediction system. Preliminary data analysis for effort estimation data has also been explored by Liu and Mintram [8].

The role that domain knowledge plays in data mining applications has been addressed in general [1], [14] as well as in several domain applications, such as medicine [20], stock index analysis [13], and finance [21]. To our knowledge, there is little discussion in the literature addressing the use of domain knowledge in a systematic machine-learning based solution, especially for the purpose of software development effort.

III. Data analysis and modeling

1) The framework: The advances of machine learning and data mining techniques have provided powerful means of extracting useful information and knowledge from data, such as text, DNA, images and videos. For software estimation, various techniques can be used for preprocessing, analysis, and modeling. However, we are not aware of any previous work that focused on on adopting an integrated approach that combines all these aspects to cope with difficult prediction tasks. Our computational framework that undertakes this integration is shown in Figure 1.

In general a number of machine learning techniques can be employed, including feature selection, visualization, clustering, and regression for modeling. These computational procedures are not isolated from each other; rather, they share active synergy between them. For instance, visualization can help to identify outliers, which can be removed in the preprocessing procedure. Feature selection helps to produce meaningful clustered structure in visualization, and this gives clues for choosing prediction algorithms in the modeling procedure. Feature selection may not only provide better input for the predictor, but also may in turn rely on the use of predictors to evaluate feature selections. Apart from these, we emphasize that domain knowledge plays a central role that oversees and contributes to various computational processes. Domain knowledge can provide direct input into preprocessing and feature selection, for example by weeding out suspicious data or noise or by suggesting features based on best-practice. It can also provide meaningful measurements to evaluate the performance of the modeling procedure. On the other hand, findings revealed by clustering analysis and modeling practice can also provide insight that can be added into domain knowledge.

Herewith we go through a few important procedures and briefly introduce some relevant algorithms.

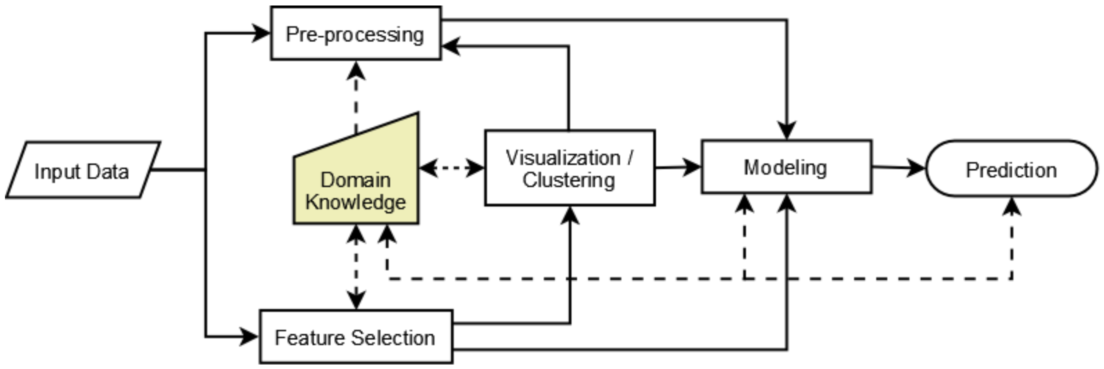

Fig. 1. Diagram of the integrated approach.

A. Visualization

Visualization can play an important part in exploratory data mining processes and provide insight on the modeling of data. Since software metrics data are typically multi-dimensional, to view them on a 2-D display would usually require a multidimensional scaling method, such as principal component analysis (PCA) or Sammon's projection [16]. On the other hand, given the numerous data entries one may have in a dataset, clustering algorithms such as k-means and spectral clustering [19] become useful in revealing the intrinsic structure of data distribution, and can also help to identify potential outliers in the data.

B. Feature selection

The Pearson correlation test can be used to assess the correlation between two variables. Denoting the two random variables as $\{x_i\}$ and $\{y_i\}$ $(i = 1, 2, ..., n)$, their means as m_x and m_y, and standard deviations as σ_x and σ_y, then the Pearson correlation coefficient is defined as

$$p = \frac{1}{n-1} \frac{\sum_{i=1}^{n} (x_i - m_x)(y_i - m_y)}{\sigma_x \sigma_y} \tag{1}$$

Apart from correlation analysis, very few approaches in software effort estimation pay sufficient attention to the problem of feature selection using machine learning techniques. In machine learning, there are a number of attribute evaluators capable of selecting a subset of attributes for classification and regression tasks. For the latter, a common method is the so-called 'wrapper approach', which regression models built on different attribute subsets to find out what an optimal feature selection should be. Another approach is use some pre-defined evaluators, such as ReliefF, to assess the contribution of attributes to the prediction of the target attribute. The ReliefF evaluator [7] was derived from assessing the quality of attributes for classification. The difference between two values of an attribute A_k in instances i and j is defined as:

$$\text{diff}(A_k, i, j) = \frac{|A_k(i) - A_k(j)|}{\text{max}A_k - \text{min}A_k} \tag{2}$$

The quality of A_i for classification can thereby be estimated by checking the distance between instances found in the set of nearest neighbours Ω. The estimate is defined as the difference between the distance probabilities measured among neighbours either from the same class, or from different classes:

$$W(A_k) = P(\text{diff}(A_k)|(i,j) \in \Omega, \text{class}(i) \neq \text{class}(j)) - P(\text{diff}(A_k)|(i,j) \in \Omega, \text{class}(i) = \text{class}(j)). \quad (3)$$

The ReliefF algorithm can be easily extended for attribute evaluation in regression [15].

C. Modeling

Various modeling techniques have been applied in previous studies. A number of machine learning algorithms can be applied, including statistical regression, multi-layer perceptrons, k-nearest neighbour (k-NN), radial basis functions, and support vector machines. Here we first take a look on the performance indicators commonly used in the software effort estimation literature.

1) Performance evaluation: The predictor needs to go through a validation set for its performance to be evaluated, and in this connection a number of performance indicators can be calculated. For each entry of the validation set, suppose the prediction is f_i', while the actual effort value is f_i. Then the magnitude of relative error (MRE) is defined as

$$\text{MRE}_i = \frac{|f_i' - f_i|}{f_i}. \quad (4)$$

The mean magnitude of relative error (MMRE) is defined as the mean of all MREs produced by the evaluation set:

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|f_i' - f_i|}{f_i}, \quad (5)$$

where n is the number of data entries in the evaluation set.

Another commonly used indicator is the *Prediction at level l*, denoted as $\text{Pred}(l)$, which measures the percentage of estimates that fall within a range of $l\%$ difference either side of the actual value. It has been suggested that a good prediction system should satisfy $Pred(20) \geq 75\%$ [3], but this is often difficult to achieve in software metrics estimation. Pred(30) is another commonly chosen measurement [4], [11].

It appears that taking the log-transform of numeric attributes is a popular data preprocessing method applied to software metrics data, and the software effort values are represented in log-transform as well [5], [11]. Cautions need to be taken when comparing the prediction accuracy, especially when both log-transform and no-transform approaches are involved. For predictions using log-transformed effort data, MMRE should be measured by making comparison with the original effort data, which should not be ignored when using a regression tool on the transformed data.

2) Prediction algorithm: We used a simple k-nearest neighbour predictor in this study. Given a new input, the predictor will give the average target value of the first k nearest neighbours found in the training set, weighted by the reciprocal of their distances to the new input. Suppose the new input is x, its k nearest neighbours are $c_1, c_2, ..., c_k$, and their relevant target values (for prediction) are $t_1, t_2, ..., t_k$ respectively, then the k-NN estimation is calculated as follows:

$$\text{Est.}(x) = \frac{\sum_{i=1}^{k} \frac{1}{dist.(x, c_i)} t_i}{\sum_{i=1}^{k} \frac{1}{dist.(x, c_i)}}. \tag{6}$$

The choice of the k value very much depends on the actual data and will be discussed later.

IV. A CASE STUDY

A. The dataset

The Desharnais dataset is available from the PROMISE Software Engineering Repository [17]. It consists of 81 software projects commercially developed by a Canadian software house between 1983 and 1988. There are the following eight variables in the dataset:

1) Team experience in years (`ExpEqp`)
2) Experience of project manager in years (`ExpMan`)
3) Number of transactions (`Trans`)
4) Unadjusted function points (`RawFPs`)
5) Adjusted Function Points (`AdjFPs`)
6) Adjustment factor (`AdjFac`)
7) Number of entities (`Ent`)
8) Duration in months (`Dur`)
9) Development environment (1,2,3) (`DevEnv`)
10) Actual effort in hours (`Eff`)
11) Year finished (`Year`)

B. Feature selection

Two feature selection indexes were employed to value the worthiness of these attributes in predicting `Eff`: the Pearson coefficients and the ReliefF index. For ReliefF we used the Weka toolbox [23], using a neighbourhood size of 20 to match with the k-nearest neighbour predictor setting used in later experiments.

TABLE I

FEATURE SELECTION INDEXES OF THE ATTRIBUTES FOR PREDICTING Eff.

Attributes	ReliefF	Pearson coeff.
Dur	0.061	0.65
Trans	0.058	0.58
AdjFPs	0.054	0.70
RawFPs	0.050	0.72
DevEnv	0.041	-0.24
Ent	0.038	0.50
AdjFac	0.012	0.41
ExpEqp	-0.003	0.26
ExpMan	-0.004	0.16
Year	-0.022	-0.03

Upon first look of the feature selection results in Table I, it is tempting to include Dur into the prediction model, since both indexes report the highest on Dur. The Pearson coefficient between Dur and Eff almost indicates a significant correlation. In fact, other software effort data also demonstrate a strong connection between Dur and Eff [12]. However, from domain knowledge, we knew both duration and effort are the *consequence* of software development, not control factors. Therefore, we chose not to include Dur into our model consideration, since it can not play a role in real-world software development effort prediction. From domain knowledge we also identified the redundancy between RawFPs and AdjFPs, since the latter is only linearly scaled from the former by AdjFac. The relatively high values for AdjFPs also suggest that it is sufficient to include it alone and ignore the other two attributes.

The Year attribute is interesting. Although experience in software development may intuitively suggest a little speed-up in project development as years progress, the feature analysis results denied the connection between Year and Eff. Hence we decided not to include Year. While industry experience would suggest both ExpEqp and ExpMan should be relevant to development effort, it is interesting to note that both feature selection indexes suggest either minor or even negative contribution to a prediction model. In the case of this disagreement, we decided to reconcile it by using domain knowledge. Since these are considered important factors in the software industry, we decided that their inclusion in the prediction model would likely be beneficial.

On the other hand, both our indexes agree that Trans, AdjFPs, and Ent are the most important features. This is well aligned with domain knowledge, since the number of transactions, the number of function points, and the number of entities are all good indicators to the scale and complexity of 4GL software projects. The treatment on DevEnv is a bit tricky. Although ReliefF suggests its relevance in effort prediction, the Pearson index would suggest it has poor direct contribution to the prediction result.

We decided to keep this attribute within the dataset for investigation in later prediction experiments.

Dealing with missing values in a dataset is another topic, but is not our focus in this study. By removing data items with missing values, we have in total 77 data entries. In the reduced dataset, the following 6 attributes are included: `ExpEqp`, `ExpMan`, `DevEnv`, `Trans`, `AdjFPs` and `Ent`, plus a target `Eff` value. Besides this 6-attribute set, we also considered a further reduced 4-attribute set that does not include `ExpEqp` and `ExpMan`. From now on, we refer to these as the 'full datasets' since they are not further split into subsets.

C. *Visualization*

We used the normal-cut algorithm for spectral clustering [19] to cluster the data into four possible clusters, and we projected the labeled data onto a 2-D display, as shown in Fig. 2. The cluster structure is not easily perceived even after 2-D projection, however. Hence we decided to use nearest neighbour-based algorithm for the modeling of effort data.

Fig. 2 also reveals an outlier in the upper right corner of the graph. It is actually the last data entry, which lies quite far away from other data entries. In practice, outliers should be removed from the dataset so that the final prediction model is built free of 'noises'. Here, however, we keep this outlier for benchmarking purpose.

D. *Effort prediction*

1) Working with entire data set: First we considered the entire data set with 77 samples after removing samples with missing values. k-NN predictors are built using both the original dataset and a transformed dataset with log-transformed `Eff` values. Unless otherwise stated, all results are reported by a 10-fold cross validation over the entire data set with $k = 20$. These performance indexes are reported: average MMRE, minimal MMRE, maximal MMRE, and Pred(30). The results are shown in Table II.

TABLE II

PERFORMANCE ON THE FULL DATASETS.

Dataset	avg. MMRE	min MMRE	max MMRE	Pred(30)
D=6	0.61	0.25	1.35	0.40
D=6(Log)	0.54	0.11	1.06	0.40
D=4	0.61	0.24	1.35	0.40
D=4(Log)	0.54	0.11	1.06	0.40

From the results, it can be seen that using log-transformed effort data can significantly improve the MMRE indexes. The two *experience* attributes, `ExpMan` and `ExpDev`, included in the $D = 6$ set but

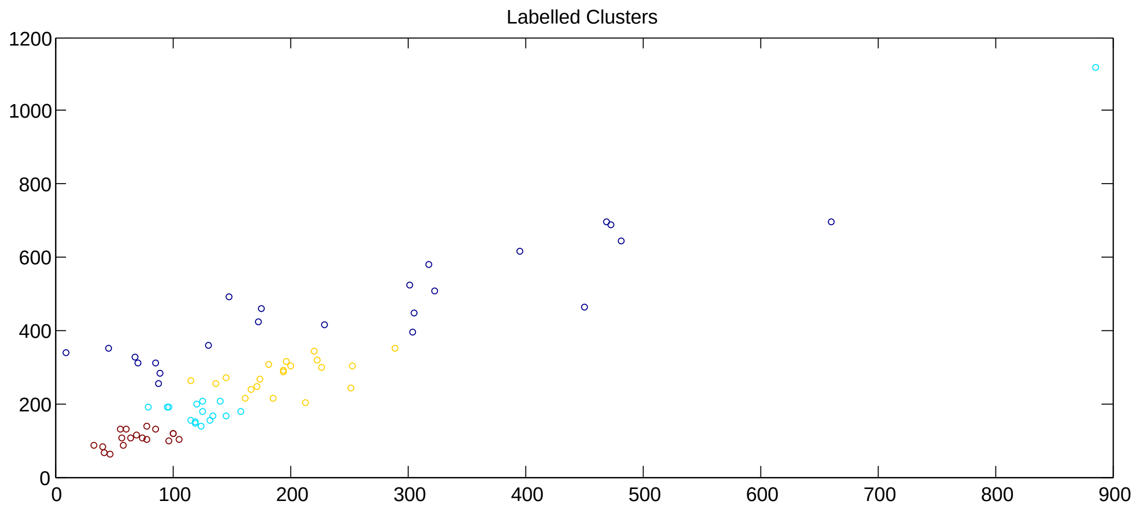

Fig. 2. Data clusters visualized by normal-cut with 2-D PCA projection.

TABLE III

Performance on the homogeneous subsets.

Subset	MMRE	Pred(30)
`DevEnv`=1, k=24	0.36	0.60
`DevEnv`=2, k=11	0.28	0.55
`DevEnv`=3, k=4	0.31	0.70

not in $D = 4$, do not contribute anything to the performance. We decided to drop these two attributes in further experiments. On the other hand, the Pred(30) value is rather low for all cases (only 40$\%$ predictions fall within 30$\%$ difference range), meaning that the prediction is not stable and the model needs to be modified.

2) Working with subsets: As indicated by the feature selection indexes, we suspected it is the `DevEnv` attribute that causes trouble in prediction experiments based on the entire dataset. Again, from domain knowledge, one would favour the use of homogeneous data for modeling. Indeed, data about projects using different program languages often have little relevance to each other. Hence, it makes sense to extract that attribute and split the full set into three homogeneous subsets according to the `DevEnv` value. The experiments are shown in Table III. Again, 10-fold cross validation was conducted, and the effort values were also log-transformed. The results obtained from using homogeneous subsets are much improved in two aspects: lower average MMRE (prediction accuracy) and Pred(30) (prediction stability). This also conforms with the earlier findings during feature selection and aligns well with real world practices, where divide-and-conquer is often a preferred approach in dealing with complex data.

3) Model selection: We considered a k-NN estimator in this study, however, the optimal setting of k is still an open question. In order to find a good setting of this parameter, we not only considered the average MMRE, but also Pred(30), and the standard deviation of MMRE during cross validation. A good estimation model should give a small average MMRE as well as a small MMRE standard deviation, along with a relatively big Pred(30).

Here we report the results obtained from the homogeneous dataset Subset 1 only, but other results were obtained using similar analysis processes. Figure 3 shows the performance evolution for k to increase from 1 to 40, all with 10-fold cross validation involved. The performance data can be summarized as follows:

1) The average MMRE first smoothly decreases until k=25. Then it becomes flat, and the starts to gradually climb up from k=30.
2) The MMRE standard deviation initially decreases, becomes quite steady for k between 10 and 30, and then starts to rise slowly.

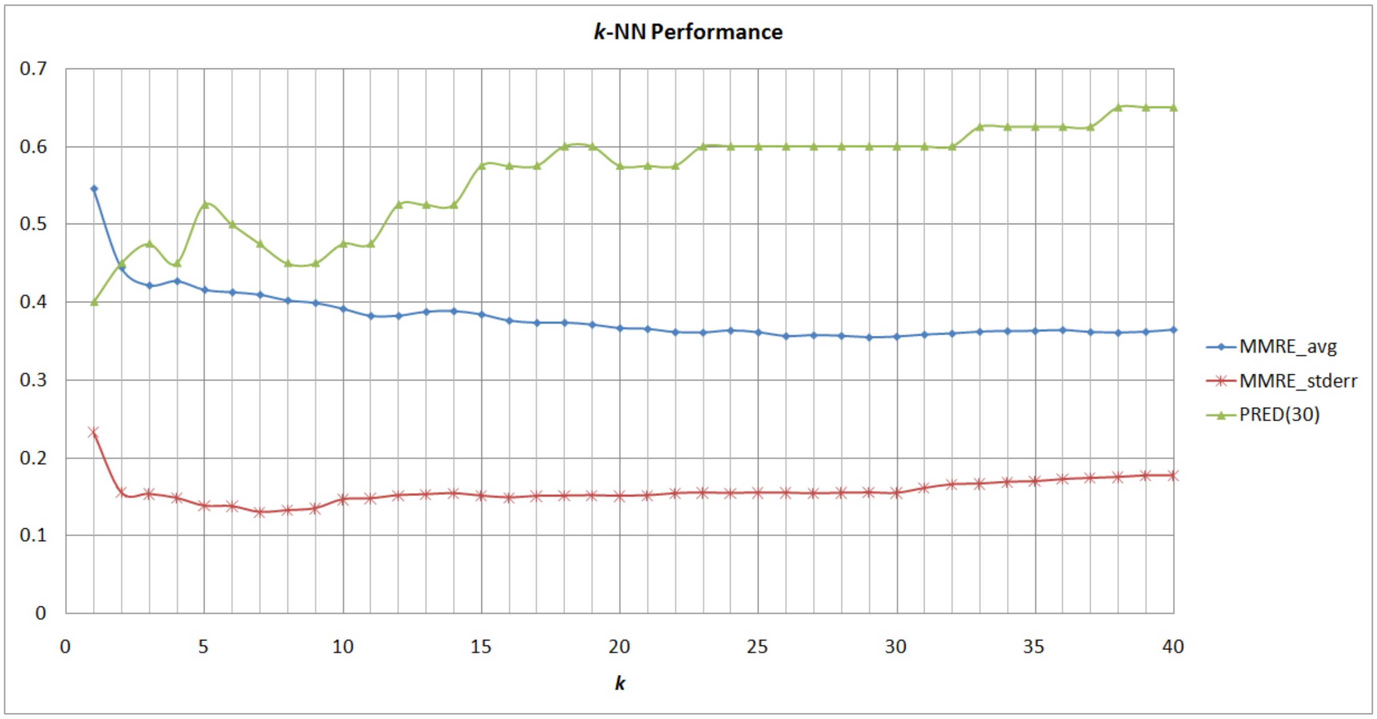

Fig. 3. Performance evolution of k-NN classifiers against the increasing k value.

3) The Pred(30) curve climbs up with minor fluctuation, becomes steady when k=23, and then gradually climbs up again when k=33.

This can be explained by the fact that the weighted averaging operation between multiple nearest neighbours can help to reduce prediction errors due to noises, but it can also in the meantime possibly damage the prediction accuracy if k keeps on increasing. A joint assessment of all these tendencies would lead to an optimal choice around $k = 26$.

V. Conclusion

Software effort estimation based on experiences with past projects is difficult, because there are unique factors for each project that can have a significant impact on the overall effort. Those factors that are measurable and that are common across multiple projects need to be treated carefully and combined into an appropriate model in order for useful predictions to be made. In this paper, we have conducted a case study and attempted to investigate the software effort estimation problem using an integrated data mining approach that incorporates data visualization, feature selection, model selection, and evaluation. There are multiple interactions between these computational procedures, and it is found that domain knowledge can be employed to enhance, evaluate and confirm the computational outcomes.

On the other hand, by engaging in a substantial data mining study, it is possible for algorithms to make some new findings that would modify or even correct existing domain knowledge. For instance, from the feature selection modeling experiments in our case study, it was found that the two experience factors are not significantly relevant to software effort. Rather, the number of function points, the number entities, and the number of transactions are more relevant. Since the scope of our study is limited, it makes sense here to merely highlight this finding and propose verifying it with more data sources.

While there still remain many interesting questions to answer on software effort estimation, especially on data collecting and the treatment of missing data, techniques such as feature selection for regression are still open research problems. This paper presents only some results obtained from a case study of software effort data. In the future we intend to experiment with more benchmark datasets, and explore further the possibility of improving software effort estimation practice using the integrated approach that combines advanced machine learning techniques and domain knowledge.

REFERENCES

[1] L. Cao and C. Zhang, "The evolution of kdd: towards domain-driven data mining," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, pp. 677–692, 2007.

[2] S. Chulani, B. Boehm, and B. Steece, "Bayesian analysis of empirical software engineering cost models," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, 1999.

[3] S. Conte, H. Dunsmore, and H. Shen, *Software engineering metrics and models*. Benjamin/Cummings, 1986.

[4] J. J. Cuadrado-Gallego, M.-A. Sicilia, M. Garre, and D. Rodriguez, "An empirical study of process-related attributes in segmented software cost-estimation relationships," *The Journal of Systems and Software*, vol. 79, pp. 353–361, 2006.

[5] J. Hale, A. Parrish, B. Dixon, and R. Smith, "Enhancing the Cocomo estimation models,," *IEEE Software*, vol. 17, pp. 45–49, 2000.

[6] R. Jeffery, M. Ruhe, and I. Wieczorek, "Using public domain metrics to estimate software development effort," in *Proc. of 7th IEEE Symposium on Software Metrics*, 2001, pp. 16–27.

[7] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of International Conference on Machine Learning*, 1992, pp. 249–256.

[8] Q. Liu and R. Mintram, "Preliminary data analysis methods in software estimation," *Software quality journal*, vol. 13, pp. 91–115, 2005.

[9] C. Mair, G. Kododa, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster, "An investigation of machine learning based prediction systems," *System and Software*, vol. 53, pp. 23–29, 2000.

[10] E. Mendes, I. Watson, T. C., N. Mosley, and S. Counsell, "A comparison of development effort estimation techniques for web hypermedia applications," in *Proc. of 8th IEEE Symposium on Software Metrics*, 2002, pp. 131–140.

[11] T. Menzies, D. Port, Z. Chen, and J. Hihn, "Simple software cost analysis: safe or unsafe?" in *PROMISE '05: Proceedings of the 2005 workshop on Predictor models in software engineering*. New York, NY, USA: ACM Press, 2005, pp. 1–6.

[12] S. Oligny, P. Bourque, A. Abran, and B. Fournier, "Exploring the relation between effort and duration in software engineering projects," in *Proceedings of World Computer Congress 2000*, 2000, pp. 175–178.

[13] Y. Ou, L. Cao, C. Luo, and C. Zhang, "Domain-driven local exceptional pattern mining for detecting stock price manipulation," in *PRICAI 2008: Trends in Artificial Intelligence*, 2008, pp. 849–858.

[14] C. Pohle, "Integrating and updating domain knowledge with data mining," in *Proceedings of the VLDB 2003 PhD Workshop (Electronic Ed.)*, M. Scholl and T. Grust, Eds., 2003.

[15] M. Robnik-Sikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 296–304.

[16] W. Sammon, "A nonlinear mapping for data analysis," *IEEE Transactions on Computers*, vol. 5, pp. 401–409, 1969.

[17] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca/SERepository., 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[18] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 12, pp. 736–743, 1997.

[19] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[20] M. Siadaty and W. Knaus, "Locating previously unknown patterns in data-mining results: a dual data- and knowledge-mining method," *BMC Medical Informatics and Decision Making*, vol. 6, no. 1, p. 13, 2006. [Online]. Available: http://www.biomedcentral.com/1472-6947/6/13

[21] A. P. Sinha and H. Zhao, "Incorporating domain knowledge into data mining classifiers: An application in indirect lending," *Decision Support Systems*, vol. 46, no. 1, pp. 287 – 299, 2008.

[22] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *IEEE Transaction on Software Engineering*, vol. 21, pp. 126–137, 1995.

[23] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, 2005.