Journal of Network and Information Security
Volume 05 Issue 01, 2017
ISSN.: 2321-6859

Software Effort Prediction - A Datamining Approach

V. Gopinath[1], R. R. Menon[2]
[1]Assistant Professor, Dept. of Computer Science & Engineering, Adi Shankara College of Engineering & Technology Ernakulam, India. Email: vidya.cs@adishankara.ac.in
[2]Assistant Professor, Dept. Computer Science & Engineering, Adi Shankara College of Engineering & Technology, Ernakulam, India. Email: raghi.cs@adishankara.ac.in

Abstract: Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. As a whole, the software industry doesn't estimate projects well and doesn't use estimates appropriately. We suffer far more than we should as a result and we need to focus some effort on improving the situation. Effort estimation is important to minimize the cost of a software project.

The existing situation may lead to serious consequences to the company as because of poor effort estimation a major percentage of the project turns out to be either more expensive than expected, late on deliver and many more issues. Not properly giving importance to the effort estimation task by under-staffing it, running the task of low quality deliverables and setting too short schedule resulting in loss of credibility as deadlines are missed always lead to problems.

The current system available for effort estimation produces non-comprehensible results. Hence the purpose of this project is to produce a software system which produces a more accurate and comprehensible results using modern tools and make it easier for the project manager to easily identify the effort needed to complete a software project in terms size of project, cost etc. The various algorithm used are Support vector machine(SVM) which are best for both classification and regression and an Active Learning Based Approach (ALBA)for rule extraction from the output of SVM to produce a comprehensible output for rule.

Keywords: Classification, Datamining, Regression.

I. Introduction

Data mining is search large stores of data automatically to find patterns and trends that go beyond simple analysis process. To segment data and evaluate the probability of future events data mining uses sophisticated large algorithms. One of the major research topics in Knowledge Discovery in Data (KDD) is effort prediction.

Classification is a datamining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each class in the data. Classification models predict categorical class labels, and prediction models continuous valued functions.

There are several algorithms are in data mining used for the rule extraction. If use data mining algorithms for predicting effort it may effective. In our proposed methodology, we apply active learning algorithm over the input data on which it learns and more specifically, focusing on attributes, which for rule extraction are those areas in the input space where the effort is needed.

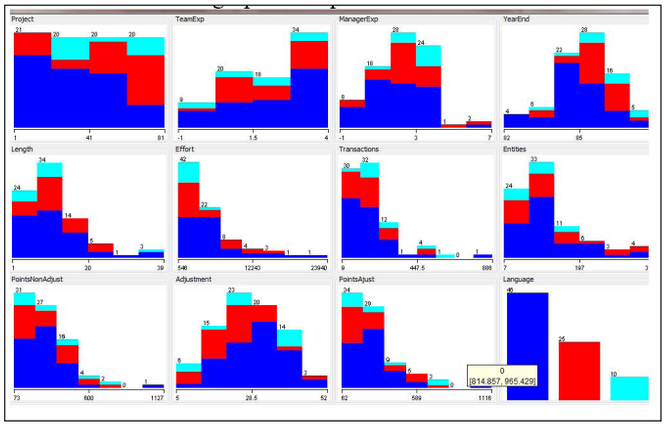
From the rule extraction we get run information for attributes and REP tree. A representative tree is created which will more precisely classify the attributes. Regression trees can be used to model functions, through each end point will result in same predicted value, a constant for that end point can be achieved. Thus regression trees like classification trees except that the end point will be predicted function value rather than a predicted classification.

ALBA rule extraction is used to find out the data values in the region determined. It avoids the data generating in outliers. Using uncertainty function the data located next to the most confidently predicted vectors are considered. ALBA extracts rules from the trained SVM model by explicitly making ALBA use of key extracts rules from the trained SVM model by explicitly making use of key concepts of the SVM: support vectors, and it will be close to decision boundary.

A. Preprocessing

The data is converted to ARFF (Attribute Relation File Format) format to process in WEKA. An ARFF file is a ASCII text file that describes a list of instances sharing a set of attributes. The processed data in weak can be analyzed using data mining techniques like visualization.

The figure shows processed attributes which are visualized into a 2 dimensional graphical representation



The information can be extracted with respect to two or more associative relation of data set. In this process we have made an attempt to visualize the impact of attribute on each other. Knowledge extraction from database is become one of the key process of each and every organization for their development issues.

B. Classification

Classification consists of assigning a class label to a set of unclassified cases.

1. *Supervised Classification*

   The set of possible classes is known in advance.

2. *Unsupervised Classification*

   Set of possible classes is not known. After classification we can try to assign a name to that class. Unsupervised classification is called clustering.

Here we are using SVM algorithm for classification.

The support vector machine is a learning procedure based on statistical learning theory. SVMs have originally being developed to solve classification problems but can be extended to regression problems as well.

Here confusion matrix terms are defined as:

TP=true positive: number of examples predicted positive that are actually positive.

FP=false positives: number of examples predicted positive that are actually negative.

TN=true negatives: number of examples predicted negative that are actually negative.

FN=false negative: number of examples predicted negative that are actually positive.

Weka confusion matrix if a is taken to be positive class.

	a	b
Actual a=0	TP	FN
Actual b=1	FP	TN

```
Classifier output

LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 0.11 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          46               56.7901 %
Incorrectly Classified Instances        35               43.2099 %
Kappa statistic                          0
Mean absolute error                      0.2881
Root mean squared error                  0.5367
Relative absolute error                 75.5865 %
Root relative squared error            123.2731 %
Total Number of Instances               81

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         1          0.568     1         0.724      0.5      1
                 0         0          0         0         0          0.5      2
                 0         0          0         0         0          0.5      3
Weighted Avg.    0.568     0.568      0.323     0.568     0.411      0.5

=== Confusion Matrix ===

  a  b  c   <-- classified as
 46  0  0 |  a = 1
 25  0  0 |  b = 2
 10  0  0 |  c = 3
```

Fig. 1: Illustrate SVM Classification from a Given Data Set

Weka confusion matrix if a is taken to be negative class.

	a	b
Actual a=0	TN	FP
Actual b=1	FN	TP

Recall is the TP rate what fraction of those that are actually positive were predicted positive?

Precision is the TP / predicted positive what fraction of those predicted positive are actually positive.

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$\text{recision} = \frac{tp}{tp + fp}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

F-measure is the measure that combines precision and recall is the harmonic mean of precision and recall.

$$F = 2.(\text{precision.recall}/\text{precision} + \text{recall})$$

Mean absolute error is a linear score which means that all the individual differences are weighted equally in the average.

Root mean squared error is a quadratic scoring rule which measures the average magnitude of the error. The formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample.

Kappa statistic is a measure of agreement normalized for chance agreement.

II. Rule Extraction

Rule extraction is performed for understand the classification in order to analyze the attribute values in tree like format.

There are two reasons why rule extraction is often used. Firstly to gain better insight into how complex models make their decisions. By building a set of rules mimicking the performance of a complex model, we are able to better comprehend the inner workings of model. A second reason for using rule-extraction is because we want to improve the performance of rule induction techniques.

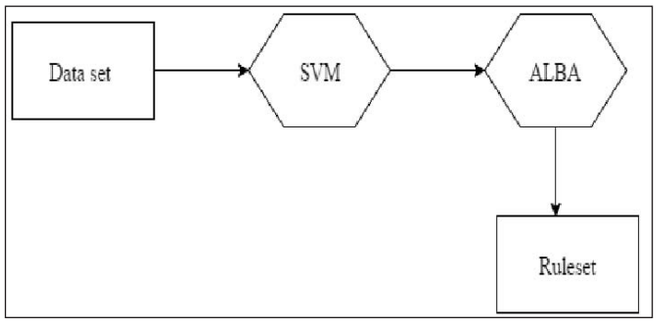


Fig. 2: Set-Up of the Rule Extraction Technique

There are several algorithms are in data mining used for the rule extraction. If use data mining algorithms for predicting effort it may effective. In our proposed methodology, we apply active learning algorithm over the input data on which it learns and more specifically, focusing on attributes, which for rule extraction are those areas in the input space where the effort is needed.

From the rule extraction we get run information for attributes and REP tree. A representative tree is created which will more precisely classify the attributes. Regression trees can be used to model functions, through each end point will result in same predicted value, a constant for that end point can be achieved. Thus regression trees like classification trees except that the end point will be predicted function value rather than a predicted classification.

It has been argued that models which are both accurate and comprehensible are to be preferred in the domain of software engineering. The problem is that by using traditional techniques these two requirements often collide. For example, complex, non-linear techniques often perform very well in terms of predictive performance but mostly yield un-interpretable models. Rule induction techniques on the other hand construct very comprehensible result but have the disadvantage of reduced predictive power.

C. Active Learning Based Algoritham

ALBA uses the support vectors as proxies for the decision boundaries, seeing that these specific training points are typically located near the decision boundary. By using ALBA, we can predict the effort needed for software completion.

Using ALBA is always a two stepprocess. First you must identify a well performing black-box model set-up. Next you choose a desired white-box algorithm to explain your black box and then run ALBA on both. Steps are:

1. Load the data
2. Select and test the black box
3. Select and test a white box
4. Predict the effort

In order to improve a rule set in terms of either predictive power or fidelity, we can use one of the previously trained rule induction techniques to imitate the output of the more complex model that performs better. The way in which this limitation is realized differs between different rule extraction methods, but is crucial to the performance of the techniques. There are three key insights being exploited in ALBA. First,

By presenting the predicted target values of the training set to the white-box algorithm instead of the original target values associated with the training set, we can improve the similarity between the black-box and the white-box substantially. By doing so, the black box effectively becomes an oracle for predictions. Second, since the oracle is only dependent on the black-box, we do not have to sample any new target labels, and thus, we are free to generate new artificial data points and their predictions without restrictions. This is the active learning component of the algorithm, at any given point, the algorithm can choose which ever input data vector it wants to get label for Third, by choosing the right data vectors, further improvements of fidelity can be achieved as we generate incrementally more data.

The tree view from ALBA implementation is shown below in the fig.

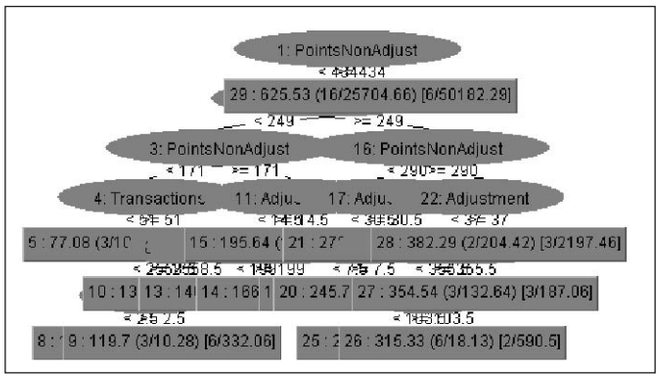


Fig. 3: Tree View of ALBA

III. Related works

Lessmann *et al* [1] proposed metric based classification as software fault prediction technique to improve software quality by identifying fault prone modules. Arisholm *et al* [2] use and compare many data mining and machine learning techniques to build fault-proneness models based mostly

on source code measures and change/fault history data. Dejaeger *et al* [3] techniques inducing tree/rule-based models like M5 and CART, linear models such as various types of linear regression, nonlinear models (MARS, multilayered perceptron neural networks, radial basis function networks, and least squares support vector machines), and estimation techniques that do not explicitly induce a model (e.g., a case-based reasoning approach. Martens *et al* [4] present a new pedagogical rule extraction algorithm for regression, based on active learning, which can be combined with any existing rule induction technique. Liu S *et al* [5] proposed an approach that simultaneously extracts a small number of rules from generated random forests and eliminates unimportant features, and hence is able to provide a simple interpretation. Junqué de Fortuny *et al* [6] proposed method is that it uses a pedagogical approach without making any architectural assumptions of the underlying model. T Verbraken, *et al.* [7] investigates the predictive power of `a number of Bayesian Network algorithms, ranging from the Naive Bayes classifier to General Bayesian Network classifiers.

A. Predictive Modeling Method

Predictive modeling leverages statistics to predict outcomes. Most often the event one wants to predict is in future, but predictive modeling can be applied to any type of unknown event, regardless of when it occurred. The main principle is the interaction between descriptive modeling guide the predictive modeling. Evaluate the appropriateness of existing dependency modeling, clustering and classification methods for educational technology, and give special instructions for their applications. Finally, we propose general principles for implementing adaptivity in learning environments. A predictive model is created from variables that are thought to provide information of the behavior of the dependent (predicted) variable. These variables are called independent or causal variables and are assumed to have known constant values. Predictive modeling techniques try to create models that maximize a score function which most often is designed to optimize the performance metric used for evaluation. For predictive modeling the primary performance goal is to achieve high accuracy, i.e. a low error between the predicted value and the real value, when the model is applied to novel data. For certain problems, a simple model, eg: linear regression model, is enough to achieve sufficient level of accuracy. Other problems require a more complicated model, e.g. a non-linear neural network oran ensemble. It is well known that ensembles,

B. Rule Induction Technique

Rule induction is one of the most important techniques of machine learning. Since regularities hidden in data are frequently expressed in terms of rules, rule induction is one of the fundamental tools of data mining at the same time.

Some rule induction systems induce more complex rules, in which values of attributes may be expressed by negation of some values or by a value subset of the attribute domain.

Data from which rules are induced are usually presented in a form similar to a table in which cases are labels for rows and variables are labeled as attributes and a decision. Rule induction algorithms may be categorized as global and local. In global rule induction algorithms the search space is the set of all attribute values, while in local rule induction algorithms the search space is the set of attribute-value pairs.

In Table show the some popular techniques existed in today.

Table 1: Literature Survey

Author	Method	Disadvantage
Lessmann, S. *et al* [1]	Using effort prediction mechanism	Cost fixing reworking of software is high
Arisholm. E *et al* [2]	Builds effort-prone models on source code measures Change effort of history data	Prefer only smaller models
T Verbraken *et al[*7]	Investigates predictive power among other algorithms	Rule extraction contains larger added value
Junqué de Fortuny *et al*[4]	Uses rule induction technique	Predictive performance comprehensibility work in contradictory way

There are number of effort prediction methods are used in data mining but for comprehensible model use ALBA algorithm for rule extraction as well as effort prediction.

III. Methodology

For this work we use Support Vector Machine (SVM) algorithm for the classification. Use the ALBA algorithm for perform rule extraction as well as effort prediction. Firstly preprocess the dataset in to attributes and classification for rule extraction. There are three modules included in this work.

- Preprocess Module
- Classification Module
- Rule Extraction Module

A. Preprocess Module

This module is used for viewing the attributes listed in the dataset. Also project manager can eliminate the unwanted attributes from the data set. Thus the attributes essential for the current project can be extracted.

B. Classification Module

The Support Vector Machine is a learning procedure based on statistical learning theory. SVMs have originally been developed to solve classification problems but can be extended to regression problems as well. Hereto, an alternative loss function that includes a distance measure is introduced. In this case a ε-insensitive loss function is used where, This means that we do not accept any deviations that are larger than ε. In other words, the goal is to find a function that has at most ε deviation from the actual target values yi for every point xi in the training set. The optimization problem can be written with ω the weight vector in the feature space, b a threshold value i is lack variables measuring the deviation from the boundaries of the ε-insensitive zone, and C the regularization constant. The optimization criterion of Eq. penalizes training data pointswhere the distance between y and the fitted function. f(x) is largerthan ε.

C. Rule Extraction Module

From the rule extraction we get run information for attributes and REP tree. A representative tree is created which will more precisely classify the attributes. Regression trees can be used to model functions, through each end point will result in same predicted value, a constant for that end point can be achieved. Thus regression trees like classification trees except that the end point will be predicted function value rather than a predicted classification.

The results of overlap when the intersection of avg. and max. intra-cluster distance was used to detect web bots. The overlap increases when the number of suspected web bots decreases because of the smaller number of true negatives

IV. Conclusion

Software effort prediction is important tasks in order to minimize costs in a software company. The predictive model used in these cases needs to be both accurate and comprehensible. Unfortunately, to obtain predictive performance, Comprehensibility is often sacrificed and vice versa. In this paperwe illustrated that rule extraction can tackle this issue and investigated the trade-off between both requirements. By applying the rule extraction technique ALBA, we improve the rule sets in terms of fidelity and, in most cases, accuracy and recall as well. On the other hand our results validate that rule extraction allows us to get more insight in the inner workings of complex models since all extracted trees were easy to understand. This in turn improves the acceptance of the model by the end-user. We thereby hope that this methodology further facilitates the wide spread adoption od data mining in software development.

V. Future Work

Since ALBA algorithm optimizes for accuracy, it can be extended to adapt performance measures like recall, precision etc. Public domain data can be included to compare and verify with similar studies in domain.

References

[1] J. Moeyersoms, E. Fortuny, K. Dejaeger, B. Baesens, and D. Martens, "Comprehensible software fault and effort prediction: A data mining approach," *Scienc.,* 1.032, 2015.

[2] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans, Softw, Eng,* vol. 34, no. 4, pp. 485-496, 2008.

[3] E. Arisholm, L. C. Briand, M. Fuglerud, "Data mining techniques for building fault-proneness models in telecom java software," In: *ISSRE'07, The 18th IEEE International Symposium on Software Reliability, IEEE,* pp. 215-224, 2007.

[4] K. Dejaeger, W. Verbeke, Martens, D., and Baesens, B., "Data mining techniques for software effort estimation: A comparative study," *IEEE Trans. Softw. Eng.* 38 (2), 375-397.

[5] E. Junqué de Fortuny, and D. Martens, "Active learning based rule extraction for regression," In *2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW),* pp. 926-933.

[6] S. Liu, X. Dang, and Y. Chen, "Rule based regression and feature selection for biological data," 2013.

[7] E. Junqué de Fortuny, and D. Martens, "Active learning-based pedagogical rule extraction," University of Antwerp, 2014.

[8] T. Verbraken, W. Verbeke, and B. Baesens, "Profit optimizing customer churn prediction with Bayesian network classifiers," *Intelligent Data Analysis Business Analytics and Intelligent Optimization,* 2014.

[9] B. Boehm, and P. Papaccio, "Understanding and controlling software costs," *IEEE Trans. Softw. Eng.,* vol. 14, no. 10, pp. 1462-1477, 1988.

[10] P. Braga, A. Oliveira, and S. Meira, "Software effort estimation using machine learning techniques with robust confidence intervals," In *19th IEEE International Conference on Tools with Artificial Intelligence,* vol. 1, pp. 181-185, 2007.