ISSN: 2278-4632
Vol-13, Issue-05, May 2023

CASE STUDIES ON SOFTWARE REUSE USING MACHINE LEARNING FOR SOFTWARE ENGINEERING

Chennaiah Kate, Assistant Professor, IT, St. Peter's Engineering College(A), Telangana, India

Dr. Anjaiah Adepu, Professor, CSE, St. Peter's Engineering College(A), Telangana, India

Sreekanth Kottu, Assistant Professor IT, St. Peter's Engineering College(A), Telangana, India

Chinni Krishnaiah.G. Assistant Professor CSE-AIML,Mallareddy University,Telangana,India

[1]chennaiah@stpetershyd.com , [2]anjaiah@stpetershyd.com,[3]sreekanthkottu@gmail.com

[4]chinnikrishnaiahg@mallareddyuniversity.ac.in

ABSTRACT

There are several machine learning algorithms accessible right now. In the twenty-first century, selecting appropriate learners to run on a given data collection instead than writing the learner itself is the challenge. In this essay, we propose that the final learner selection shouldn't be exclusive because there are certain benefits to subjecting data sets to different learners.

We conduct a case study on a reusable data set utilizing three distinct learning styles—association rule, decision tree induction, and treatment—to demonstrate our claim. Software reuse is a subject of intense discussion in the academic and professional worlds; experience has shown that it can be both a benefit and a burden. Our students discovered various techniques which should considerably increase the chances of a reuse programme working, despite the fact that there is much disagreement over where and when reuse should be incorporated into a project.

KEYWORDS: Artificial intelligence algorithms, AI in software engineering, AI in data mining, reuse, empirical research, treatment learning, association rule learning, decision tree learning, C4.5, J4.8, J4.8 PART, TAR2, and APRIORI.

INTRODUCTION

There are just too many machine learning technologies available now. Users are faced with a plethora of options, making it difficult to decide which learners to apply to their data. The difficulty currently is not in creating or developing a learner, but rather in sifting through the vast number of capable and mature learners to identify a fraction that will be beneficial.

Making sure that you have run enough learners to establish a solid and credible conclusion is a difficulty that is equally prevalent and significant as selecting which learners to run. We think that researchers stop

ISSN: 2278-4632
Vol-13, Issue-05, May 2023

too soon when faced with the challenge of selecting a suitable subset of learners, i.e., they draw their findings after testing just one learner.

However, there is no reason why those outcomes cannot be cross-checked with other methods given the relative simplicity of running numerous learners. In reality, applying a variety of learning algorithms to a data collection can provide a wide range of insightful outcomes. We utilize the data set from "Success and Failure Factors in Software Engineering" by Morisio et al. to demonstrate this claim [18]. This information is a compilation of interview findings with industry projects introducing reuse. Morisio, Ezran, and Tully were in charge of conducting the interviews [18]. The 288 Process Improvement Experiments financed by the European Commission are broken down into the industrial initiatives that are being presented. Each time, the project was picked because it was an honest attempt to put a reuse method into practice.

The researchers (Morisio, et al.) used the output of the CART [5] machine learner to draw their findings in their study. Their data set consisted of a list of successful and unsuccessful instances of the deployment of a software reuse programme in various organizations. Here, we demonstrate that several distinct (and crucial) outcomes may be obtained by applying additional learning to the data set (using alternative learning methods). The distinctions between our findings and those of Morisio et al. are shown in Figure 1. Please see the original article [18] for a complete list of all the traits gathered. The J4.8 [13], J4.8 PART [13], APRIORI [2], and TAR2.2 [22] machine learners were used to reevaluate the data from the aforementioned study. J4.8 is a decision tree induction learner that divides data into subsets for each unique value of an attribute. It is based on C4.5. J4.8 PART uses partial decision trees—more particular, the decision trees created by J4.8—to determine categorization rules. APRIORI is an association rule learner since it looks for connections between different data set properties. TAR2.2 is a treatment learner, to sum up. An algorithm used by a treatment learner looks for attribute ranges rather than classifications or correlations in an effort to forecast the occurrence of the best class more frequently and/or the worst class less frequently [22]. This is especially helpful when a better and worse class can be quickly identified, such when determining whether a software reuse programme was successful or unsuccessful.

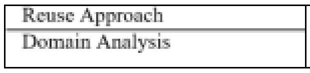
Attribute	Morisio et.al.	this paper
state:		
Application Domain	Not analyzed	×
Size of Baseline	Not analyzed	X
Production Type	X	×
high-level control:		
Top Management Commitment	X	×
low-level control:		
 Reuse Approach Domain Analysis		

Figure1.Conclusionswherewedisagreewith Moriso et.al. ×/X= no/some evidence (respectively) in this data set that this attribute is relevant to determining success or failure of a reuse project.

The remainder of this essay demonstrates how we created Figure 1. We'll demonstrate how employing a range of learners on a dataset (as opposed to focusing just on a single class of machine learner) may lead to more conclusive and beneficial outcomes.

2 BACKGROUND

2.1 Learning

Discovering significant patterns in data sets is the aim of learning. These data sets are difficult to analyze manually, and it can be time-consuming and labor-intensive. If a machine can be "taught" to look for these patterns, the process becomes both faster and simpler. Many other sorts of learners, such as decision tree learners, classification rule learners, association rule learners, and treatment learners, have developed to help with this. Each of these machine learners, which are each discussed in the following sections, was employed in our study of the data set.

2.1.1 Decision Tree Learners and J4.8

Decision Tree Learners look for routes that lead to a certain class instance. Specifically, J4.8 employs a technique known as decision tree induction. This approach splits data using a common recursive splitting method, creating a decision tree with training examples for one class at the leaf nodes [12]. This indicates that the J4.8 algorithm's output is a "decision tree" derived from the supplied data, which offers a possible route to follow in order to reach a certain class instance, or success. The C4.5 algorithm has been ported to Java as J4.8 [21].

C4.5 constructs its trees using a heuristic entropy measurement of the information content. The root of a tree is chosen based on the property that provides the most information gain. The J4.8 [13], J4.8 PART [13], APRIORI [2], and TAR2.2 [22] machine learners were used to reevaluate the data from the aforementioned study. J4.8 is a decision tree induction learner that divides data into subsets for each unique value of an attribute. It is based on C4.5. J4.8 PART uses partial decision trees—more particular, the decision trees created by J4.8—to determine categorization rules. APRIORI is an association rule learner since it looks for connections between different data set properties. TAR2.2 is a treatment learner, to sum up.

$$I(p,n) = -\left(\frac{p}{p+n}\right) log_2 \left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right) log_2 \left(\frac{n}{p+n}\right)$$

Consider an attribute A with values A1, A2,... Av. When we choose Ai as the root of a new sub-tree inside of C, a new sub-tree Ci will be added, including all of the items in C that have Ai. The weighted average may then be used to define the anticipated value of the information needed for that tree.

ISSN: 2278-4632
Vol-13, Issue-05, May 2023

$$E(A) = \sum_{i=1}^{v} \left(\frac{p_i + n_i}{p + n} \right) I(p_i, n_i)$$

The information gain of branching on *A* is therefore:

$$gain(A) = I(p,n) - E(A)$$

2.1.2 Classification Rule Learners and J4.8 PART

A classification rule learner finds a rule that applies to instances in a certain class (and excludes examples that are not in the class), separates those instances out, and then continues learning on the remaining instances [13]. J4.8 PART is a partial decision tree rule learner, to be more precise. The programme analyses the J4.8 algorithm's trimmed decision trees and looks for rules that may be inferred from the tree. In some circumstances, categorization rules might be far more readable and compact than decision trees. Furthermore, rules are frequently chosen over decision trees since each rule appears to represent a distinct "nugget" of information [13].

2.1.2 Association Learners and APRIORI

A generalization of classification rule learners, association rule learners may forecast for any property in the data set, not only the class [13]. Learners of association rules+63 may identify important correlations between qualities.

For instance, APRIORI often demonstrates that some features forecast for other attributes. In certain situations, it would be beneficial to eliminate the dependent attribute(s), as they are not essential for evaluation and often impede and confound the learning process.

2.1.3 Treatment Learners and TAR2

Instead of looking for a predictor for a class, treatment learners seek out a treatment that anticipates an increase in the frequency of the best class and a drop in the frequency of the worst. This distinguishes them from most other learners. When analyzing a data set with two clearly separated classes (i.e., success and failure), this kind of learner is quite helpful.

The TAR2 algorithm searches for attribute ranges that appear more frequently in classes with high scores than in classes with lower scores[22] and generates various rules depending on those attributes. Additionally, one may choose how many attributes to employ to create a rule. This enables the creation of a diverse and comprehensive collection of rules that can include either a range for each characteristic included in the data set or a range for just one attribute. The mining algorithm for TAR2 is described in depth below.

TAR2 looks for attribute ranges that are more prevalent in the classes with high scores than in the classes with lower scores.

If Domain Analysis = yes and a.r. is some attribute range, then a.r. is a heuristic assessment of the value of a.r. to increase the frequency of the best class. The following definitions are used by a.r.

X(*a.r.*) : is the number of occurrences of that attribute range in class X;
e.g. *success*(*Domain Analysis. Yes*) = 9.

all(*a.r.*) : is the total number of occurrences of that attribute range in all classes; e.g. *all*(*Domain Analysis.yes*) = 9 *best* : the highest scoring class; e.g. *best* = *success*. *rest* : the non-best class; e.g. *rest* = *failure*. *score* : the score of a class X is $X.

Using these definitions, $\Delta_{a.r.}$ is calculated as follows:

$$\Delta_{a.r.} = \frac{\sum_{X \in rest} (\$best - \$X) * (best(a.r.) - X(a.r.))}{all(a.r.)}$$

A treatment is a portion of the attribute ranges with a high a.r. value. All example entries that deviate from the conjunction of the attribute ranges in the treatment are rejected by TAR2 before the treatment can be applied. Comparison is made between the ratio of classes in the remaining examples and the ratio of classes in the initial example set. The remedy that boosts the relative proportion of favored classes the greatest is the most effective [22].

2.1.4 Comparison

Finding a paper, article, or research that contrasts several learning algorithms is challenging. The numerous advantages and issues with diverse learners have been discussed widely, although often such studies only apply to students in the same class (such as categorization learners).

There are, however, certain exceptions.

Research on the application of machine learning models for predicting rectification costs was conducted by Almeida and Lounis [16]. In their article, they contrast the machine learning techniques for NewID [4], CN2 [6], C4.5 [21], and FOIL [20]. They draw the conclusion that the inductive logic programming algorithms [FOIL] are superior to top-down induction attribute value rules, top-down induction decision tree, and covering algorithms...

A treatment is a portion of the attribute ranges with a high a.r. value. All example entries that deviate from the conjunction of the attribute ranges in the treatment are rejected by TAR2 before the treatment can be applied. Comparison is made between the ratio of classes in the remaining examples and the ratio of classes in the initial example set. The remedy that boosts the relative proportion of favoured classes the greatest is the most effective [22].

3. THE TESTS

We break out our testing procedures and outcomes, as well as those from Morisio et al.'s study, in the sections below.

3.1 Morisio et.al.'s Test

Morisio et al. divided the attributes from the data set into the three categories of state variables, high-level control variables, and low-level control variables for their analysis. They specify the various classes in the following manner:

State variables are characteristics that a corporation has no control over.

High-level management choices on a reuse programme are represented by high-level control variables.

• Low-Level Control Variables - Particular methods for applying reuse.

Furthermore, Morisio et al. decided to focus only on the state and high-level control variables for their analysis. They argue that the high-level control variables come before the low-level control variables both chronologically and/or logically. For example, they claim ...a decision about whether to do a domain study and when to generate assets comes before a decision to implement reuse procedures, both logically and chronologically.

Although Morisio et. al.'s justification for segmenting the qualities is legitimate, their decision to exclude the low-level control variables has tainted their analysis. One of their key findings was that failure resulted from failing to address two or more high-level control variables [18]. Having said that, they make no mention of which specific approach should be used to achieve reuse (for instance, which processes should be used specifically for reuse). In essence, their findings point a corporation in the right direction without outlining the specifics of how to get there.

It's crucial to thoroughly analyze each characteristic while analyzing a data set. Otherwise, crucial information and patterns might be overlooked. This was the issue with the data set analysis performed by Morisio et al. They have improperly restrained their learner and constrained the breadth of their conclusions by selecting to omit specific features.

3.2 Our Tests

With the exception of the Repository attribute, which was disregarded while running APRIORI since it contains the value "yes" in every case, we kept all the variables unclassified and intact when analyzing this data set. In order to make sure that all potential patterns are found, we have also ran a variety of machine learners.

We contrast our findings for each machine learner with those from the original publication (by Morisio et al.) [18]. The students are shown in the order that they were ran.

The findings from the learners were validated using a standard procedure known as 10-way cross validation.

The data set is divided into 10 sections in this plan, each with the identical class distribution. The algorithms then use nine of the sets for learning before testing the new model on set ten. The learnt model can be regarded as legitimate if the error rate during cross validation is low (the precise meaning of low might vary depending on the domain and data set you are dealing with).

3.2.1 APRIORI

Since it is helpful to be aware of any dependencies in a data set before executing other learning algorithms, APRIORI was ran first. APRIORI results must be carefully analyzed, though, as certain relationships may emerge that have little "real-world" significance. APRIORI's output should have a dependency that may be connected to a scenario in real life, such as task management. Rewards Policy =

ISSN: 2278-4632
Vol-13, Issue-05, May 2023

no because commitment = no. However, no such conclusions were discovered in this data set. APRIORI did not provide a useful dependence, hence its findings did not support or refute those of Morisio et al.

3.2.2 J4.8

In this test, the J4.8 machine learner was used to process the data without any changes. The program's success or failure served as the only focus of the lesson. Human Factors was the sole attribute employed in the resultant tree. The tree is displayed below.

Human Factors = yes: success (16.0/1.0) Human Factors = no: failure (8.0)

In a 10-way cross-validation, this tree's error rate was just 4.2%.

After doing the aforementioned test, we took the characteristic Human Factors out of the data and reran J4.8. As a result, the tree is displayed below.

Reuse Processes Introduced = yes: success (15.0/1.0)
Reuse Processes Introduced = no: failure (8.0/1.0) Reuse Processes Introduced = NA: failure (1.0)

After doing a 10-way cross-validation, the error rate for this decision tree is 20.8%. As you can see, this tree likewise only makes use of Reuse Processes Introduced as one property.

The initial evaluation carried out by Morisio et al. made use of a CART machine learner. CART creates a decision tree and is quite similar to J4.8. The following decision tree is displayed.

Human Factors = yes:
Type of software production = product-family: success
  Type of software production = isolated: failure Human Factors = no: failure

The results of the two algorithms show how they differ from one another. While CART seeks the tree with the lowest failure rate, regardless of simplicity, J4.8 is motivated to discover the simplest tree with a manageable failure rate. By using two characteristics in this instance, the CART algorithm was effective in locating a tree with a 0% failure rate.

3.2.3 J4.8 PART

Complex decision trees can be made simpler by rule learners like J4.8 PART. The decision trees from 3.2.2 are quite straightforward in this investigation. The result of J4.8 PART is therefore nothing more than a straightforward repetition of those trees. The J4.8 PART algorithm's output for both of the decision trees from 3.2.2 is presented here for the purpose of completeness.

FIRST TREE:
Human Factors = yes: success (16.0/1.0)
: failure (8.0)
SECOND TREE:
Reuse Processes Introduced = yes: success (15.0/1.0)
: failure (9.0/1.0)

3.2.4 TAR2

As was said above, TAR2 is a "Treatment Learner" and as such, it has the potential to provide outcomes that are substantially dissimilar from those of conventional classification and decision tree learners. Three reliable and practical findings were produced by running this learner (under 10 way cross-validation). The outcomes from TAR2 are attribute ranges that choose for the best class (in this case, success), as was previously mentioned. The following three attribute ranges are the most helpful in choosing a reuse project's success:

Size of Baseline = L
Domain Analysis = yes
Reuse Approach = tight

None of these characteristics were included in Morisio et al.'s first review. Because only one learner was used, the depth of the study was limited as a result of Morisio et al.'s decision to omit the low-level control variables.

4. CONCLUSIONS

4.1 Software Reuse

Several inferences about the potential success of a reuse programme can be made after using multiple machine learners. According to Morisio et al.'s findings, "Human Factors" are by far the most important determining factor for success. This result makes sense since if employees are adequately taught and supported, they are more likely to adopt reuse successfully.

Next, and in no particular order, the additional factors affecting reuse are "Reuse Processes Introduced", "Size of Baseline", "Reuse Approach", and "Domain Analysis".

We give a simple rule table in two sections to summarize the fundamental actions that should be followed in order to improve the likelihood that a software reuse programme will succeed.

The first section consists of points where we concur with Morisio et al.'s results, while the second section consists of points where our conclusions diverge from theirs.

IN CONSENT:

1. Encourage, assist, and train individuals to build reusable code.

2. Introduce certain methods for reuse.

DIFFERING:

3. Don't waste your time attempting to extract reusable code from tiny projects.

4. Conducted a domain analysis.

5. Ensure that reusable work items are securely fastened.

Any company should find these steps helpful when starting a reuse programme. In fact, according to the Morisio et. al. statistics, our students are telling us that following these 5 easy actions will significantly boost their odds of success.

ISSN: 2278-4632
Vol-13, Issue-05, May 2023

4.2 Multiple Learners

We have demonstrated that a single learner (or single kind of learner) is insufficient to uncover all the relevant patterns hidden in a data set. Running an effective machine learner no longer takes days; in many situations, it just takes hours, because to computers' increased speed and memory capacity. Our premise is that a good analysis of a data set should involve many learners given the relative simplicity of operating numerous learners.

Many sophisticated machine learning tools are readily downloadable from the public domain. The J4.8, J4.8 PART, and APRIORI implementations, for instance, were taken from the WEKA [13] toolkit, which may be found at http://www.cs.waikato.ac.nz/ml/weka/.

REFERENCES

[1] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz,R. Madachy, D. Reifer, R. Selby, and B. Steece. COCOMO II model definition manual. Technical report, Center for Software Engineering, USC,, 1998. http://sunset.usc.edu/COCOMOII/cocomox.html#downloads.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, 1994.
Available from http://www.almaden.ibm.com/cs/ people/ragrawal/papers/vldb94_rj.ps.

[3] K.-D. Althoff, M. Nick, and C. Tautz. Improving organizational memories through user feedback. In F. Bomarius, editor, *Proc. of the Workshop on Learning Software Organizations (LSO) (in conjunction with the11th International Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslauten, Germany)*, pages 27–44, June 1999.

[4] R. Boswell. Manual for newid, January 1990.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J.
Stone. Classification and regression trees. Technical report, Wadsworth International, Monterey, CA, 1984.

[6] P. Clark and T. Ng. The cn2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[7] P. Coad, D. North, and M. Mayfield. *Object Models: Strategies, Patterns, and Applications*. Prentice Hall, 1997.

[8] P. Cohen, V. Chaudhri, A. Pease, and R. Schrag. Does prior knowledge facilitate the development of knowledge-based systems? In *AAAI'99*, 1999.

[9] W. Cunningham. The checks pattern language of information integrity. In J. Coplien and D. Schmidt, editors, *Pattern Languages of Program Design*. Addison-Wesley, 1995. Also avialable at *http://c2.com/ppr/checks.html*.

[10] W. Frakes and C. Fox. Sixteen questions about software reuse. *Communications of the ACM*, 38(6):75–87, June 1995.

[11] E. Guerrieri. Reuse success - when and how? In *Proceedings of WISR9: The 9th annual workshop on Institutionalizing Software Reuse*, 1999. Available from http://www/umcs.maine.edu/˜ftp/wisr/ wisr9/final-papers/Guerrieri.html.

[12] L. Holder. Intermediate decision trees, 1995. Available from http://www-cse.uta.edu/˜holder/pubs/ ijcai95.ps.

[13] I.H.Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Academic Press, 2000.

[14] N. Kerth. Caterpillar's fate: A pattern language for transformation from analysis to design. In J. Coplien and D. Schmidt, editors, *Pattern Languages of Program Design*. Addison-Wesley, 1995. Also available from *http://c2.com/ppr/catsfate.html*.

[15] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, G. Yost, and other members of the PIF working group. The PIF Process Interchange Format and framework. *Knowledge Engineering Review*, 13(1):91–120, Feb. 1998.

[16] H. L. Mauricio A. de Almeida and W. L. Melo. An investigation on the use of machine learned models for estimating correction costs. In *Proceedings of the 21st International Conference on Software Engineerin. Kyoto, Japan*, 1997.

[17] T. Menzies. Se/ke reuse research: Common themes and empirical results. In S. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering, Volume II*. WorldScientific, 2002. Available from http://tim.menzies. com/pdf/00reuse.pdf.

[18] M. Morisio, M. Ezran, and C. Tully. Success and failure factors in software reuse. *IEEE Transactions on Software Engineering*, 28(4):340–357, 2002.

[19] D. Perry. Some holes in the emperor's reused clothes. In *Proceedings of WISR9: The 9th annual workshop on Institutionalizing Software Reuse*, 1999. Available from http://www/umcs.maine.edu/˜ftp/wisr/ wisr9/final-papers/Perry.html.

[20] J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, August 1990.

[21] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1992. ISBN: 1558602380.

[22] T.Menzies and Y. Hu. The tar2 treatment learner, 2002.

Available from http://www.ece.ubc.ca/twiki/ pub/Softeng/TreatmentLearner/intro.pdf.

M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13(1), Feb. 1998