Comparing Software Prediction Techniques Using Simulation

Martin Shepperd and Gada Kadoda

Abstract—The need for accurate software prediction systems increases as software becomes much larger and more complex. A variety of techniques have been proposed; however, none has proven consistently accurate and there is still much uncertainty as to what technique suits which type of prediction problem. We believe that the underlying characteristics—size, number of features, type of distribution, etc.—of the data set influence the choice of the prediction system to be used. For this reason, we would like to control the characteristics of such data sets in order to systematically explore the relationship between accuracy, choice of prediction system, and data set characteristic. Also, in previous work, it has proven difficult to obtain significant results over small data sets. Consequently, it would be useful to have a large validation data set. Our solution is to simulate data allowing both control and the possibility of large (1,000) validation cases. In this paper, we compared four prediction techniques: regression, rule induction, nearest neighbor (a form of case-based reasoning), and neural nets. The results suggest that there are significant differences depending upon the characteristics of the data set. Consequently, researchers should consider prediction context when evaluating competing prediction systems. We also observed that the more "messy" the data and the more complex the relationship with the dependent variable, the more variability in the results. In the more complex cases, we observed significantly different results depending upon the particular training set that has been sampled from the underlying data set. This suggests that researchers will need to exercise caution when comparing different approaches and utilize procedures such as bootstrapping in order to generate multiple samples for training purposes. However, our most important result is that it is more fruitful to ask which is the best prediction system in a particular context rather than which is the "best" prediction system.

Index Terms—Prediction system, simulation, machine learning, data set characteristic.

1 Introduction

The ability to build prediction systems for software engineers remains an important but largely unsolved problem. One area that is proving to be particularly intractable is project prediction. Here, we would like to predict aspects such as duration and effort at an early stage in the development. A wide range of techniques have been proposed. These include statistical methods, parametric models, and machine learning (ML) methods. Obviously, this then raises the question which technique, or techniques, are "best." Unfortunately, and perhaps unsurprisingly, there is no simple answer.

The problem of selecting a technique for building prediction systems is well illustrated by the following example. The authors have been associated with the development of a technique for building project effort prediction systems based upon the idea of analogies or case-based reasoning which we implemented as a software tool known as ANGEL [1]. In order to evaluate the technique, we compared predictions derived from ANGEL with those from a regression model built from a stepwise regression procedure (SWR). Effectively, we treated SWR as a benchmark. We utilized a number of data sets that were in the public domain and three data sets that we ourselves collected. Using a Mean Magnitude of Relative Error (MMRE) accuracy indicator,[1] we found that, for all nine data sets, ANGEL had a better or equal level of accuracy to the SWR model. Can we therefore conclude that regression-based approaches should no longer be pursued and that all estimators should use ANGEL? The answer is no. There are at least three reasons why this is so.

First, subsequent studies have reported more mixed experiences. Niessink and van Vliet [2] found similar results using a different data set. Unfortunately other researchers, most notably Briand et al. [3] and Mrytveit and Stensrud [4] obtained conflicting results where the regression model generated significantly better results than an analogy-based approach. While there are some differences in approach, in particular, Briand et al. [3] used a different procedure to select the best feature subset, due to the fact that the problem is NP-hard, this does not fully explain differences in the results.

Second, we have shown elsewhere that the choice of error indicator can have a significant effect in ranking candidate prediction systems in order of accuracy [5]. Put simply, different accuracy indicators conflict. While MMRE is a widely used indicator with the advantage that comparisons can be made across data sets, it suffers from being asymmetric in that under estimates are bounded while over estimates are not. We believe this may contribute—at least in a small measure—to the unreliability of comparing different techniques for building prediction systems.

• *The authors are with the Empirical Software Engineering Research Group, School of Design, Engineering, and Computing, Bournemouth University, Poole, UK. E-mail: {mshepper, gkadoda}@bournemouth.ac.uk.*

Manuscript received 01 Jan. 2001; Accepted 15 Mar. 2001.
Recommended for acceptance by S. Pfleeger.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 114700.

1. MMRE is defined as $\frac{100}{n}\sum_{i=1}^{i=1}\frac{|e_i-\hat{e}_i|}{e_i}$, where $n \in N$.

Third, the underlying characteristics of the problem data set is likely to exert a strong influence upon the relative effectiveness of different prediction systems. For example, the two data sets Briand et al. used both appear to contain well-defined hyperplanes such that the regression procedures are able to generate models with good explanatory power as evidenced by the high R squared values. One would not expect case-based reasoning (CBR) to perform well since, instead of interpolating or extrapolating, it endeavors to draw datapoints to the nearest cluster. Clearly, this is not an effective strategy where data falls upon or close to a hyperplane. That is, a linear function exists that "explains" the relationship between the dependent variable and the independent variables.

Returning to our main argument, we can see that it is proving problematic to establish whether CBR is a useful technique or not. One possibility is some form of meta-analysis, such as a simple vote counting procedure. See Pickard et al. [6] or Miller [7] for a discussion of some of the issues. Another possibility is try to understand what conditions favor a particular technique. There is evidence that this may be a fruitful approach. We therefore believe that the question, which technique, or techniques, are "best" should be modified to include a notion of context. In other words, what is the relationship between different properties of the data set and the accuracy of a prediction system? Once we can start to answer this question, not only will we be able to understand why we get conflicting results from different research teams evaluating the same technique for building prediction system, but also we will be able to better advise practitioners.

The idea of artificially generated data with known properties to explore software engineering data set modelling techniques was first proposed by Pickard et al. [8]. This is an attractive idea for a number of reasons. First, it provides the researcher with a great deal more control over the characteristics of a data set. In particular, it enables the researcher to vary one property at a time, thereby allowing a more systematic exploration of the relationship between data set characteristics, type of prediction system, and accuracy. By contrast, especially with smaller real data sets, the true properties may not be fully known. For example, it may be difficult to differentiate between types of distribution in the presence of extreme outliers. Second, small data sets necessitate small validation sets. As we have shown elsewhere [5], this can be extremely problematic if we would like to show that there are significant differences between competing prediction systems. Statistical testing is essential if we would like to argue that the difference in accuracy between Prediction System A and Prediction System B is of interest. Otherwise, it is something of a moot point as to whether a difference between, say $\mathrm{MMRE} = \mathrm{x}\%$ and $\mathrm{MMRE} = \mathrm{x} + 5\%$, is important or not. With simulated data, it is possible to have both a small training set—a common real world occurrence—and a large validation set with which to assess our findings.

The remainder of this paper goes onto describe our experimental procedure to explore the relationship between data set characteristics and prediction system performance. We then describe our results from the simulated data sets. This is followed by a discussion of the relative performance of the techniques under scrutiny—stepwise regression, rule induction, artifical neural nets, and case-based reasoning—and the circumstances that favor each technique. We then reflect upon our procedure and make some suggestions as to how it might be improved for future investigations. Lastly, we conclude by considering the need for this type of research and the importance of our findings.

2 Prediction Techniques

This section describes the four prediction techniques that were compared in our simulation experiment:

- stepwise regression (SWR),
- rule induction (RI),
- case-based reasoning (CBR), and
- artificial neural nets (ANN).

Our choices were influenced by two factors. First, each technique adopts a highly contrasting approach to generate a prediction. Second, they represent areas of significant research activity by the software metrics community.

Stepwise Regression. This approach uses linear regression analysis to determine the relationship between two or more quantitative variables so that one variable (the Y or dependent variable) can be predicted from the others (the X or independent variables). The general form of such a prediction system is $Y = \beta_0 + \beta_1 X_1, \ldots, \beta_n X_n$. This then involves finding suitable independent variables and values for the β coefficients. Many researchers have followed this approach including [9], [10], [11].

For our analysis, we used SPSS for Windows, version 9. The procedure we used for the selection of variables to construct the regression model was stepwise selection [12]. In stepwise selection, the first independent variable that is considered for entry into the equation is the one with the largest positive or negative correlation with the dependent variable. The F test for the hypothesis that the coefficient of the entered variable is zero is then calculated. If the variable passes the criterion (the probability associated with the F statistics is less than .05), the second variable is selected based on the highest partial correlation. If it passes the entry criteria, it also enters the equation. If the first variable fails to meet entry requirements, the procedure terminates with no independent variables in the equation. After the first variable is entered, it is examined for removal according to removal criterion.[2] After each step, variables already in the equation are examined for removal, until none remains that meets the removal criterion. There is often no unique subset of the independent variables that best predict the dependent variable, which is considered one of the limitations of the stepwise regression search approach since it seeks to identify just that [13]. However, it is a widely used regression procedure in software prediction related research.

2. The maximum probability (*F-to-remove or POUT*) is the default removal criteria that is used in SPSS, where variables with a t value less than 0.01 are removed.

Rule Induction. As the name suggests this approach derives more general rules from specific cases which form the training set. The rules are organized into trees, sometimes referred to as regression trees. There are many different algorithms for achieving this, such as ID3 and C4.5 [14], [15]. We used the unsupervised learning algorithm C5.0 and implemented in the data mining toolkit Clementine [16]. An advantage of using Clementine is that it handles issues such as tree pruning automatically—pruning is important to prevent the rule tree over adapting to the training set and being unable to effectively generalize to new problems. Examples of using RI to develop prediction systems include [17], [18].

Case-Based Reasoning. Next, we used our in-house developed CBR shell, ANGEL [1]. CBR is considered to be fundamentally different from rule induction and regression approaches in that it utilizes specific knowledge of previously solved cases to solve future ones, while the former use generalized knowledge or relationships, respectively. In other words, CBR is model free. The idea of using analogies as a basis for estimating software project effort is not new. Boehm [19] suggested the informal use of analogies as a possible technique almost 20 years ago. The idea was reiterated by Cowderoy and Jenkins [20] in 1988, but again, with no formal mechanism for selecting analogies. The next development was from Vicinanza et al. [21], [22] who suggested that developments from the machine learning community in the form of CBR might be usefully adapted to help make better software project predictions. Case-based reasoning has four distinct aspects:

- characterization of cases,
- storage of past cases,
- retrieval of similar cases to use as analogies, and
- utilizing the retrieved case to solve the target case problem, sometimes known as case adaptation.

In the situation of effort prediction, CBR might be deployed as follows: We have n projects or cases, each of which needs to be characterized in terms of a set of p features. In addition, we must also know the feature that is to be predicted. Features can either be continuous (e.g., experience of the project manager), discrete (e.g., the number of interfaces), or categorical (e.g., development environment). In practice, many approaches treat discrete features as if they were continuous. Historical project data is collected and added to the case base. When a prediction is required for a new project it is referred to as the target case. The target case is also characterized in terms of the p features. Incidentally, this imposes a constraint on the feature set in that it should only contain features for which the values will be known at prediction time. The next step is to measure similarity between the target case and other cases in the p-dimensional feature space. Possibly, the most similar cases or projects could then be used, with adaptation to generate a prediction for the target case. Once the target case has completed, it can be added to the case base. A more general account of CBR may be found in [23].

ANGEL is an analogy-based estimation tool that searches for similarities between a target entity, such as a proposed software project and a set of historical entities of the same class. Each entity is characterized by a number of attributes that are available at the point when estimates are required. Similarity is measured as Euclidean distance in p-dimensional space. In this simulation, we used a single analogy, without adaptation, and utilized the entire feature set. Effectively, this turns the technique into a nearest neighbor (NN) method. While it is unlikely that NN would be the most effective form of CBR it has the merit of being very straightforward for this analysis.

TABLE 1
Summary of Software Effort Data Set Characteristics

Dataset	Cases (n)	Features (p)	Categorical Features	Outliers?		Collinearity?	
Albrecht	24	6	0	$^4/_7$	57%	$^5/_5$	100%
Atkinson	21	16	0	$^{11}/_{17}$	65%	$^{15}/_{16}$	94 %
Desharnais	77	9	1	$^5/_{10}$	50%	$^9/_9$	100%
Finnish	38	29	2	$^4/_{30}$	13%	$^{24}/_{29}$	83 %
Kemerer	15	4	0	$^1/_5$	20%	¾	75 %
Mermaid	28	17	1	$^2/_{18}$	11%	$^{11}/_{17}$	65 %
Min	15	4	0		11%		65%
Max	77	29	2		65%		100%
Median	26	12.5	0.5		35%		88.5%

Note that the Desharnais data set contains four additional cases with missing values so these are excluded from our analysis.

Artificial Neural Nets. Finally, we looked at ANNs as a means of predicting. A number of researchers have applied neural nets to the problem of predicting software effort, often with good results, see for example [24]. Although there has been a great deal of work in this area, we focused on a simple multilayer perceptron with a back propagation learning algorithm as implemented in the Clementine data mining tool [16]. This had the advantage of reducing user interaction in terms of configuring the ANN—which was a significant factor given the amount of simulation work—but may well have had an impact upon the results, a point that we will return to later in this paper.

It is apparent from the above discussion that the theoretical foundation of the four chosen techniques are very different; however, we could of course extend the list of techniques. We chose not to do so partly on pragmatic grounds since the simulation work is quite time consuming. The next section describes our method for simulating data set characteristics and analyzing prediction system performance.

3 Method

The aim of this investigation was to explore what data set properties favor which particular techniques for building prediction systems.

Table 1 summarizes a range of characteristics of a number of software data sets that are available in the public domain. The purpose of this is to identify how these data sets vary and also what data set characteristics might be considered typical. This is to help inform our investigation of the relationship between data set characteristics and prediction system performance. The columns in the table list the number of cases in the data set, total number of features (independent variables), the number of features that are categorical, and measures for outliers and collinearity. The first figure in the outliers columns represents the number of variables with extreme values out of the total number of variables (including the dependent), while the other figure is the percentage of that. The first section of the collinearity column is the number of variables that exhibit significant correlations with other independent variables out of the total number of independent variables. It can be observed that categorical features are not used very often and outliers and multicollinearity exists in every single data set.

Our technique was to artificially generate data sets. This was done blind so that the researcher building the prediction systems was unaware of the built in characteristics of the data set or of the underlying models. As previously indicated, we used simulated data sets. There are a number of problems with the real data sets that might have hindered our analysis. First, it is unclear to what degree they possess the various characteristics that we are interested in. For example, it is quite difficult to differentiate between heteroscedasticity and outliers. This is particularly acute for small data sets. Second, we cannot control how we combine the factors we would like to explore. Third, we do not know what the "true" model is so again assessing prediction systems on small data sets is fraught with difficulty. Finally, we could generate large validation sets. With a real data set, the small number of cases (even using a jack knife or bootstrap [25]) results in a small validation set with all kinds of unpredictable consequences.

The following data set characteristics—typical of software data sets—were considered:

- number of cases in the training set (n)[3]
  - small (n = 20)
  - large (n = 100)
- number of features in the training set (p)
  - small (p = 4)
  - large (p = 15) (deferred)
- distribution of values
  - normal
  - positive skew
  - Gamma (deferred)
- independence of features
  - independent
  - multicollinearity
  - heteroscedasticity (deferred)
- feature type
  - categorical (deferred)
  - discrete (deferred)
  - continuous

Note that some aspects have been deferred for future investigation. In particular, while we accept that the ability to deal with categorical and discrete features is a significant aspect of a prediction system, we focused only upon continuous features. The number of features in the simulated data sets were kept to the minimum of $p = 4$ independent variables. Although increasing the number of features will be considered for future work but, given the labor-intensive simulation work requires, it is sufficient to test our ideas and to provide results representative of small naturally occurring data sets.

3. The sizes of 20 and 100 cases were selected to be representative of software project effort data sets which tend to be relatively small (see Table 1). There are very few publicly available data sets in excess of this size.

The independent variables were generated as follows:

- *normal*. The mean was 500 and the standard deviation was chosen as 25 percent of the mean, in this case, 125. Values were generated by randomly sampling from the above Normal probability distribution. Had negative values been generated we would have replaced them with 1 however, this eventuality did not arise.
- *outliers*. Outliers were generated by positively skewing the distribution. This is achieved by an increasing scaling function that was applied to values falling above the mean of a normal distribution. We then tested that the Skewness coefficient (which measures the degree of asymmetry of the distribution) exceeded 5.0 for all variables.
- *collinearity*. This is achieved by ensuring that the three of the independent variables were functionally related to the fourth variable, so, $X_n = f(X_1 + \varepsilon_1)$ where $n = 2 \ldots 4$. The functions were randomly generated using a simple grammar. We then tested to ensure minimum cross correlation values of $r = 0.7$. With hindsight we should have used an external variable in place of X_1.
- *outliers plus collinearity*. Here, both the outlier and collinearity procedures were applied.

In addition to the training set—either 20 or 100 cases—we also generated a large validation set of 1,000 cases. These were used to assess the accuracy of the prediction system, but, of course, are not used to generate it. A large training set was useful since, especially in the presence of outliers, small training sets can lead to wildly fluctuating results. We repeated the process of sampling a training set twice in order to establish how reliable the results were. In the past, many researchers, including ourselves, have relied upon a single partitioning of a data set into training and validation sets. We wanted to see how safe this procedure was.

Furthermore, we need to consider the "true model" which was built into the data set:

- continuous (Y1),
- discontinuous (Y2), and
- random.

The random model was included, as a form of control. We believe that information about the likelihood of false positives, that is erroneously believing that we have a prediction system, also warrants investigation.

The two models were randomly generated using a simple grammar which enabled models of arbitrary length, using arbitrary subsets of the available independent variables $X_1 \ldots X_n$ and arbitrary operators available from a spreadsheet package (e.g., arithmetic, trigonometric, logarithms and decisions). We imposed one restriction, however, that we wanted a continuous and a discontinuous model. One reason for this choice is that it is not obvious

TABLE 2
Analysis of Accuracy (MMRE) for Continuous Model (Y1)

Dataset	Small training set (20)				Large training set (100)			
	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal	9.90 10.32	23.00 20.70	20.03 22.14	15.25 17.63	9.77 9.31	13.41 15.67	17.90 17.80	38.96 9.09
Normal + outliers	36.57 63.95	166.68 256.05	205.04 57.64	162.39 160.30	51.33 40.19	96.16 88.11	37.62 34.39	28.12 32.57
Normal + multicollinearity	11.11 20.65	27.95 28.23	26.03 34.07	36.71 38.36	17.87 12.04	27.17 19.96	24.65 21.33	46.63 14.16
Normal + outliers + multicollinearity	285.73 140.59	139.51 149.36	26.14 22.50	232.65 261.63	172.14 148.22	17.70 18.52	13.71 14.55	62.08 52.57

TABLE 3
Analysis of Accuracy (MMRE) for Discontinuous Model (Y2)

Dataset	Small training set (20)				Large training set (100)			
	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal	711.05 484.59	476.36 197.53	640.74 198.01	553.96 134.13	447.37 404.95	223.43 64.21	212.03 157.43	449421.2 949.81
Normal + outliers	295.67 313.54	226.36 162.12	572.68 206.13	254.13 562.15	296.17 258.12	106.80 187.70	245.99 166.38	319.06 307.55
Normal + multicollinearity	645.29 1087.44	624.23 855.00	960.78 770.92	617.17 614.92	437.33 427.40	173.50 193.59	180.29 147.14	826.98 404.21
Normal + outliers + multicollinearity	* *	* *	361.49 405.00	* *	498.66 248.30	593.60 423.56	417.84 277.13	498.09 378.08

that all data sets contain hyperplanes that form the basis for useful predictions. Where such a situation is not the case, one would anticipate this to favor the ML techniques.

The independent variables also all have error terms associated with them. The independent variables were generated as follows:

$$Y_1 = f(X_1, \ldots, X_n)$$
$$Y_2 = f(X_1, \ldots, X_n).$$

However, supplied to the estimator was:

$$Y_1, Y_2, X'_1, \ldots, X'_n,$$

where $X'_1 = X_1 + \varepsilon_1, \ldots X'_n + \varepsilon_n$. The error terms were generated as normally distributed with a mean of zero and $\sigma = 10\%$ of the standard deviation of the sample of X_i. With the benefit of hindsight, we feel we should have considered a different distribution such as Gamma.

As already stated, the prediction techniques under investigation were:

- stepwise regression (SWR),
- rule induction (RI) using the Clementine data mining package which uses the C5.0 algorithm,
- case-based reasoning (CBR) using ANGEL, and
- artificial neural net (ANN) again using the Clementine data mining package which provides a feed-forward multilayer perceptron and back propagation learning algorithm.

The estimator was provided with five unlabeled data sets each with different properties, e.g., normal, random, etc. Furthermore the estimator was unaware of the "true model." The estimator then developed prediction systems using the four techniques listed above for the two dependent variables Y1 and Y2 and then validated them on the validation set of 1,000 cases for that particular data set.

We compared the accuracy of different prediction systems and data sets in two ways. First, we computed the MMRE which allowed comparisons between data sets and is generally easier to interpret. Second, we computed the absolute residuals and used these for tests of significance when we wanted to consider differences between prediction systems or data set characteristics. Since the absolute residuals were inevitably heavily skewed—confirmed by the Kolmogorov-Smirnov test for non-Normality—we used robust tests. This meant the Kruskal-Wallis test for comparing variance and to compare the difference in location between two populations the Wilcoxon Signed Rank test when the data was naturally paired and the Mann-Whitney U test otherwise. Given the large number of tests being performed and that the validation set contained a 1,000 cases, we set our confidence limit at $\alpha = 0.01$.

4 RESULTS

We now consider the results from our simulation study. These are grouped by the type of "true model". We then address a number of specific research questions.

Tables 2 and 3 provide MMRE percentage accuracy figures for all four types of data sets, by small and large training sets using the three types of prediction systems. Each cell contains two values representing results from the two different training sets sampled from the data set. The shaded values indicate the prediction system (SWR, RI, CBR, or ANN) that yielded the most accurate prediction in terms of the MMRE indicator. Note that, for statistical testing, we used absolute residuals which are less vulnerable to bias than the asymmetric MMRE. Nevertheless, we provide MMRE in Tables 2, 3, and 4 since it is generally easier for the reader to interpret and also allows comparison between data sets. Table 2 contains the results for the Y1, Table 3 results from the Y2 discontinuous "true model." Note also, in Table 3, * denotes a prediction system where terms would have had to have been forced into the prediction system procedure since the procedure rejected all features. We do not provide accuracy figures in such circumstances.

While of course the true models were artificially generated, it is striking how much deterioration there was between the Y1 (Table 2) and Y2 (Table 3) predictions. This effect tended to be far larger than that between prediction

TABLE 4
Predicting with Random Data

Dataset	Small (20)			Large (100)		
	SWR	RI	ANN	SWR	RI	ANN
Random	45.38	49.09	47.70	*	57.49	43.07
	619.72	503.69	38.01	10101.5	2874.3	39.21
	65.56	59.39	40.71	*	53.50	42.38
	1386.0	1246.3	42.70	*	3191.8	42.46

TABLE 5
Significantly Different Results Using Two Training Sets for the Continuous Model (Y1)

Dataset	Small training set (20)				Large training set (100)			
	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal	Y		Y	Y				Y
Normal + outliers	Y	Y	Y			Y		Y
Normal + multicollinearity	Y		Y	Y	Y	Y	Y	Y
Normal + outliers + multicollinearity	Y	Y		Y				Y
Totals	4	2	3	3	1	2	1	4

TABLE 6
Significantly Different Results Using Two Training Sets for the Discontinuous Model (Y2)

Dataset	Small training set (20)				Large training set (100)			
	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal	Y	Y	Y	Y			Y	Y
Normal + outliers	Y	Y	Y	Y				
Normal + multicollinearity	Y	Y	Y	Y				Y
Normal + outliers + multicollinearity	Y		Y	Y	Y	Y		Y
Totals	4	3	4	4	1	1	1	3

systems or other data set characteristics. Such a circumstance could arise when dealing with very heterogeneous data, in which case partitioning the data into smaller more homogeneous data sets might be an effective strategy.

Table 4 shows the results generated when the data is random and there is no underlying model. Sampling the training set was repeated four times for this part of the analysis since we were particularly interested in random effects. Clearly, in this case, we did not wish to make predictions since there was no basis for any prediction. Asterisks denote when the technique was—correctly—unable to build a prediction system. Note that we do not include CBR since the technique has no means of identifying random data and will always endeavor to predict, whatever the circumstances. This could be seen as a disadvantage. More surprising is how vulnerable SWR was to finding "models" when none exist. This is particularly true for the small training set containing only 20 cases.

We now turn to more specific research questions.

4.1 Do Data Set Characteristics Matter?

The short answer is yes, very much. Even a cursory inspection of Tables 2 and 3 reveals great disparity of performance and, more importantly, relative performance of the four different prediction techniques. Broadly speaking, SWR produced the most accurate predictions for the Normal and Normal + Outlier data sets, while the "messier" data sets favored the ML techniques. The impact of the data set characteristic was confirmed using the Kruskal-Wallis test on the absolute residuals. This enabled us to confirm that 1) the interaction between data set characteristic and technique makes a difference for all cases ($\alpha = 0.01$) and 2) that the differences between prediction systems for a *given* data set are significant. While this is not surprising, it is important to know that the choice of prediction system does matter. It also rather undermines the view of seeking the "best" prediction technique. "Best" depends upon context or data set characteristics. This suggests our ultimate research goal is to provide information so that an estimator can answer the question: Inasmuch as I believe my data set is characterized by the following, what would be the most suitable prediction technique?

4.2 How Repeatable are the Results?

Here, we considered the question of sampling the training set from the overall data set. Conventionally, researchers, including ourselves, have partitioned the overall data set into a training set and validation set often using a two thirds to one third split. This is done randomly. The question is, supposing the training set is comprised different cases would we still obtain the same results; for example, is Prediction System A still to be preferred over Prediction System B? We endeavored to answer this question by performing each prediction twice using different training sets sampled from the same underlying data set. We then compared the resulting pairs of predictions using a Wilcoxon Signed Rank test.

Tables 5 and 6 show where there was a significant ($\alpha = 0.01$) difference between the results from the two training sets. Differences were a frequent occurrence, especially for small training sets where, as one might suppose, individual values could exert considerable leverage. For the small training set, 27 out 32 tests resulted in significantly different predictions when the prediction procedure was repeated. Unfortunately, Table 1 indicates that small data sets are something of a norm. Encouragingly, the large training sets were far less vulnerable to this effect with only 14 out of 32 tests resulting in different values. Nevertheless, from our analysis, these differences led to a rank reversal problem—where the order of preferment was changed—on five occasions. It is noteworthy that the ANN technique was particularly vulnerable to changes in training set. As a consequence, researchers need to be extremely careful that their results are not an artifact of a particular training set sample. We therefore recommend that several training sets are sampled from a data set whenever researchers wish to compare different prediction techniques.

4.3 How Much Does Training Set Size Matter?

Here, we investigate the effect of using larger training sets. Do they always improve prediction accuracy and under what circumstances is it most influential? To answer this question, we compared the absolute residuals of predictions made using the small (20 cases) with the large (100 cases) training set. We tested for differences

TABLE 7
Training Set Size Matters for the Continuous Model (Y1)

Dataset	Continuous Model (Y1)				Discontinuous Model (Y2)			
	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal		Y	Y	Y	Y	Y	Y	Y
Normal + outliers		Y	Y	Y	Y	Y	Y	Y
Normal + multicollinearity		Y	Y	Y		Y	Y	Y
Normal + outliers + multicollinearity	Y	Y	Y	Y	Y	Y	Y	Y
Totals	1	4	4	4	3	4	4	4

using the Wilcoxon Signed Rank test once again setting the confidence limit at $\alpha = 0.01$.

Table 7 shows the circumstances in which a larger training set size significantly reduced prediction errors measured as absolute residuals. Note that increasing the training set never had a detrimental effect. Interestingly, SWR benefited least from a larger training set, particularly, for the better behaved data sets and the continuous true model. The probable reason was that, in such circumstances, SWR could identify a useful hyperplane in the data even with only 20 datapoints. By contrast, the ML approaches always benefited from having larger training sets. Extra cases were most valuable where there was a complex "cost" function (Y2) and/or the data sets were "messier" (outliers and outliers + collinearity). For well behaved data with a near linear "cost" function (Y1) 20 or 100 cases makes little difference for SWR in particular. The availability of data could therefore be an influential factor when choosing a prediction technique.

4.4 Which Prediction Technique is "Best?"

Here, we turn back to Tables 2 and 3. SWR tended to do best with the continuous "cost" function (Y1), while the ML approaches performed better when the "cost" function was discontinuous (Y2).

Table 8 contains counts of when a technique yielded the most accurate predictions. For example, SWR was the most accurate technique on a total of 10 occasions. On the basis of this data, if you had no other information, this analysis would favor CBR since it appeared to be the best all round predictor by a small margin. On the other hand, if you believed the "cost" function to be approximately continuous, even if not linear, then SWR would be a better candidate. By contrast, RI only produced the most accurate predictions for the Y2 data and even then was outperformed by CBR.

TABLE 8
Counts of Best Technique by "Cost" Function

Prediction Technique	Y1	Y2	Total
SWR	9	1	10
RI	0	7	7
CBR	5	6	11
ANN	2	2	4

TABLE 9
Counts of Best Technique by Data Set Characteristics

Prediction Technique	Normal	Normal+ outliers	Normal+ collinearity	Normal+ outliers+ collinearity
SWR	4	1	4	1
RI	3	3	1	0
CBR	1	2	1	7
ANN	0	2	0	0

Table 9 is similar to Table 8 except the counts are organized by data set characteristics. We see for example, if it is believed that the data set is approximately Normal, then SWR is to be preferred while, if it has outliers and collinearity, CBR is strongly to be preferred.

We believe this type of systematic exploration of the relationship between prediction system accuracy and data set characteristics will lead to a better understanding of when to use a particular technique for deriving a prediction system and that this is to be preferred to rather naive debates of which technique is "best." However, to make progress, many more experiments will be required.

5 Discussion and Future Improvements

This paper has described the use of simulation to generate data sets with which to evaluate different techniques for building prediction systems. This builds on the ideas of Pickard et al. [8] who first proposed the use of simulated data for evaluating software models. We have argued that this is very important if we are to systematically explore the impact of different data set characteristics upon predictions. An analysis of six publicly available data sets indicated that salient characteristics include the presence of outliers and collinearity. Moreover, the majority of data sets were small severely restricting the size of training set and validation sets. Using this information, we tried to simulate "realistic," or naturally occurring, data.

Our procedure involved separating the data generation from evaluation so that the analysis was conducted blind. This was to try to eliminate possible bias. While the work was extremely time consuming, we were still able to compare four contrasting techniques—stepwise regression, rule induction, a form of case-based reasoning, and neural nets—on four data sets with differing characteristics and a fifth control data set which was random. We also explored the impact of training set size and the type of "cost" function. Finally, all tests were repeated with different training sets being sampled from the underlying data set.

While this work is still at an early stage, many important lessons have emerged and we would encourage other researchers to further explore this area. We believe there to be three important lessons.

The first lesson was that there was a strong relationship between the success of a particular technique and characteristics of the prediction problem, such as training set size, nature of the "cost" function, and general characteristics of the data set (e.g., presence of outliers). While this may not seem surprising, it very much militates against the idea of seeking the "best" prediction technique. For example, on a simple vote counting basis, we might argue that CBR was the best technique since it outperformed the other techniques on the most occasions (11 out of 32). However, if one adopted this perspective—on our data alone—this would lead to choosing the wrong prediction system two thirds of the time. In particular, if the data was characterized by near normality or a continuous "cost" function this would actually favor SWR which would then be a much better choice than CBR. We believe researchers should instead consider the question of what conditions most favor or inhibit various prediction techniques.

The second lesson has more to do with validation procedure. Elsewhere, [5] we have argued that it is necessary to statistically test the significance of perceived differences in accuracy between competing prediction systems. However, in this simulation study, we have found even that may not be enough since the results frequently differ between the pairs of training sets that have been sampled from the underlying data set. On five occasions, this resulted in a ranking reversal so that the preferred prediction system changed. This problem was most acute when the data was most "messy" and the "cost" function most complex. Clearly, the solution is that researchers will need to repeat their sampling and validation procedures a number of times in order to gain confidence in their results.

The third lesson relates to false positives or finding a model when none exists. We considered this problem by means of randomly generated data sets. Our results highlight a problem with all four techniques. Clearly, CBR is most vulnerable, as it is model free and, therefore, has no simple means of detecting when a prediction should *not* be made. However, the other two ML approaches (RI and ANN) were also consistently deceived and even the SWR technique can make Type I errors in finding model coefficients significantly different from zero when that is not the case. This area has not been given much consideration, yet we believe it is almost as important to consider the ability of a technique to determine when it should be trusted as its ability to give accurate results.

In addition, our results have started to provide information upon what circumstances favor which prediction technique. For example, a small training set, a Normal distribution with outliers, and an approximately continuous "cost" function could lead to the conclusion that SWR was a strong candidate for prediction purposes. At present, we feel our conclusions in this area must remain quite tentative as they depend upon various arbitrary decisions such as the random generation of "cost" functions, etc..

As we have indicated there is also need for much additional work. We would like to explore a richer set of distributions than merely the Normal distribution, for instance, the Gamma distribution which is quite prevalent in software data sets. We also need to explore other types of "cost" functions. Finally, there is a need to ensure that this type of work can be tied back to the "real world" by further research into the characteristics of naturally occurring data sets.

Acknowledgments

The authors would like to their thank professor Barbara Kitchenham for suggesting the idea of simulation in the first place and for many invaluable discussions. The research described in this paper was partly funded by grant (Grant GR/L37298) from the UK Engineering and Physical Sciences Research Council and the Defence Evaluation Research Agency. We are grateful for careful reading and useful suggestions from the anonymous referees. A version of this paper was first presented at the Seventh IEEE International Software Metrics Symposium, London, 4-6 April, 2001.

References

[1] M.J. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Trans. Software Eng.*, vol. 23, pp. 736-743, 1997.
[2] F. Niessink and H. van Vliet, "Predicting Maintenance Effort with Function Points," *Proc. Int'l Conf. Software Maintenance,* 1997.
[3] L. Briand, T. Langley, and I. Wieczorek, "Using the European Space Agency Dataset: A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," *Proc. 22nd IEEE Int'l Conf. Software Eng.*, 2000.
[4] I. Myrtveit and E. Stensrud, "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models," *IEEE Trans. Software Eng.*, vol. 25, pp. 510-525, 1999.
[5] M.J. Shepperd, M.H. Cartwright, and G.F. Kadoda, "On Building Prediction Systems for Software Engineers," *Empirical Software Eng.*, vol. 5, pp. 175-182, 2000.
[6] B.A. Kitchenham and N.R. Taylor, "Software Cost Models," *ICL Technical J.*, vol. 4, pp. 73-102, 1984.
[7] J. Miller, "Can Results from Software Engineering Experiments be Safely Combined?," *Proc. IEEE Sixth Int'l Metrics Symp.*, 1999.
[8] L. Pickard, B. Kitchenham, and S. Linkman, "An Investigation Analysis Techniques for Software Datasets," *Proc. Sixth IEEE Int'l Software Metrics Symp.*, 1999.
[9] R. Gulezian, "Reformulating and Calibrating COCOMO," *J. Systems Software,* vol. 16, pp. 235-242, 1991.
[10] P. Kok, B.A. Kitchenham, and J. Kirakowski, "The MERMAID Approach to Software Cost Estimation," *Esprit Technical Week,* 1990.
[11] S.G. MacDonell, M.J. Shepperd, and P.J. Sallis, "Metrics for Database Systems: An Empirical Study," *Proc. Fourth IEEE Int'l Metrics Symp.*, 1997.
[12] *SPSS for Windows—Base System User's Guide,* Release 6., SPSS Inc., 1993.
[13] J. Neter, W. Wasserman, and M.H. Kutner, *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Designs.* Homewood, Ill: Irwin, 1985.
[14] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[15] J.R. Quinlan, "Learning Decision Tree Classifiers," *ACM Computing Surveys,* vol. 28, pp. 71-72, 1996.
[16] *Clementine User Guide—Version 5,* Integral Solutions Limited, 1998.
[17] M.A. de Almeida, H. Lounis, and W.L. Melo, "An Investigation on the Use of Machine Learned Models for Estimating Correction Costs," *Proc. IEEE Int'l Conf. Software Eng.*, 1998.
[18] R.W. Selby and A.A. Porter, "Learning from Examples: Generation and Evaluation of Decision Trees for Software Resource Analysis," *IEEE Trans. Software Eng.*, vol. 14, pp. 743-757, 1988.
[19] B.W. Boehm, *Proc. Software Engineering Economics.* Englewood Cliffs, NJ: Prentice-Hall, 1981.
[20] A.J.C. Cowderoy and J.O. Jenkins, "Cost Estimation by Analogy as a Good Management Practice," *Software Engineering 88,* 1988.
[21] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation," *MIS Quarterly,* vol. 16, pp. 155-71, 1992.

[22] S. Vicinanza, M.J. Prietula, and T. Mukhopadhyay, "Case-Based Reasoning in Effort Estimation," *Proc. 11th Int'l Conf. Information Systems,* 1990.

[23] D. Leake, "CBR in Context: The Present and the Future," *Case Based Reasoning: Experiences, Lessons and Future Directions,* D. Leake, ed. Menlo Park: AAAI Press, pp. 1-35, 1996.

[24] G. Wittig and G. Finnie, "Estimating Software Development Effort with Connectionists Models," *Information & Software Technology,* vol. 39, pp. 469-476, 1997.

[25] B. Efron and G. Gong, "A Leisurely Look at the Bootstrap, the Jackknife and Cross-Validation," *The Am. Statistician,* vol. 37, pp. 36-48, 1983.

Martin Shepperd received the PhD degree in computer science from the Open University, in 1991. Currently, he has the chair of software engineering at Bournemouth University, UK. He has published more than 70 refereed papers and three books in the field of empirical software engineering. He has served on many program committees including the European Software Engineering Conference, the IEEE International Metrics Symposium and the Software Quality Workshop. He is the editor of the journal *Information and Software Technology* and an editorial board member for *IEEE Transactions on Software Engineering*. He is a council member of the Centre for Software Reliability and a member of the UK, Engineering and Physical Sciences Research Council Computing College. His research interests include software metrics and empirical software engineering.

Gada Kadoda received the BSc degree (honors) in computer science from the University of Khartoum (Sudan), the MSc degree in information systems and technology from City University, London, UK, and the PhD degree from Loughborough University. She is a research fellow in the Empirical Software Engineering Group, in the school of Design, Engineering, and Computing, Bournemouth University, Poole, UK. Her research interests include computer-aided teaching, empirical software engineering, and software prediction systems.