Exploiting the Effort Estimation Data Structure for Active Learning

Ekrem Kocaguneli, *Student Member, IEEE,* and Tim Menzies, *Member, IEEE,*

Abstract—
Background: It is assumed that *all* instances of the software effort estimation (SEE) datasets are useful in an analogy-based estimation (ABE) context. Hence, all instances are labeled. Also SEE datasets favor certain algorithms. In [1] CART and ABE have been identified as the best performing algorithms.
Motivation: Labeling all projects is a costly and time-consuming activity. If datasets have suitable structure/topology, active learning can guide learners to reduce the labeling cost and time.
Aim: We aim to understand the topology of the effort datasets and define an active learning scheme to guide labeling of instances.
Method: We define popularity-based E(k) matrices that identify the order of instances to be labeled. We augment a standard ABE method (*passiveNN*) with this guiding system (*activeNN*). Then we compare the performance of *activeNN* to that of *passiveNN* and CART.
Results: There is no-point in labeling all the instances in a dataset. *ActiveNN* can attain results comparable to *passiveNN* and CART with orders of magnitude less labels.
Conclusion: Actual topology of the instance space is different than the expected: Some instances are popular than the others. This knowledge is successfully exploited to build an active-learning based guiding system in effort estimation for the first time.

Index Terms—Software Cost Estimation, Analogy, *k*-NN

1 Introduction

Software effort estimation (SEE) experiments that use all the instances of a dataset assume that all instances are useful for estimation. We would like to call that approach *"assumption-all"*, which states that:

"All instances are useful in estimation."

Another possible assumption is that some of the instances are redundant or even disruptive (noise). Similarly, some others are more popular, i.e. used more frequently during estimation. We would like to call that popularity based approach as *"assumption-pop"*. It states that:

"Only popular instances are useful."

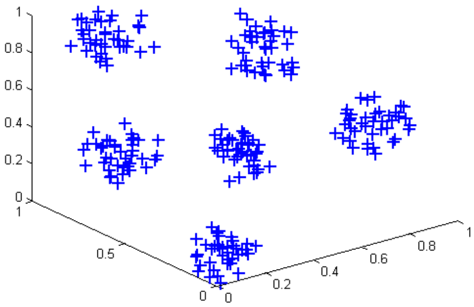
Data collection is a difficult process and in software engineering (SE) domain data it is not easy to access high-quality data. The so called "data-drought" is in fact quite commonly recognized by researchers [2]. The first author has the experience that more than half the effort of building an effort estimation model is related with data collection [2]. The second author also acknowledges that fact: After two years of effort only 7 projects could be added to the NASA-wide software cost metrics project [3]. Even Boehm shares the same experience, after more than 20 years there are less than 200 projects in COCOMO datasets [4]. After years of research on data collection [5], software-metrics expert Norman Fenton also agrees by saying: "...much of the current software metrics research is inherently irrelevant to industrial mix... any software metrics program that depends on some extensive metrics collection is doomed to failure [6]."

Relation between *data-draught* and the SEE data assumptions (*assumption-all* and *assumption-pop*) is that *data-draught* favors *assumption-pop*, which ensures less data collection. However, utilization of *assumption-pop* requires clever guidance systems that understand the underlying structure of the data so as to guide a learner towards *popular* instances.

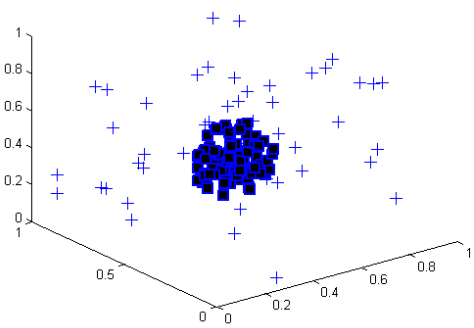
The proposed guidance system in this paper is a product of the active learning paradigm. Active learning is an answer to *data-draught*. It is based on the motivation that collecting labeled data is expensive [7]–[9]. Data collection activities can be immensely reduced, if labeling efforts are concentrated on certain instances that the learner considers most useful for estimation . Unlike *passive* supervised learning that assumes the learning problem is defined by an unknown function producing the training examples; active learning suggests labeling only the instances that are useful for the learning process [7].

In this study we investigate SEE data assumptions in the context of analogy-based estimation (ABE) by utilizing an active learning-based guidance system. This investigation gives critical information regarding the topology/structure of SEE datasets. It is shown that effort datasets behave in accordance with *assumption-pop*: Using *popular* instances attains as good performance as

• *Ekrem Kocaguneli, Tim Menzies and Kel Cecil are with the Lane Department of Computer Science and Electrical Engineering, West Virginia University. E-mail: ekocagun@mix.wvu.edu, tim@menzies.us*



(a) Expected



(b) Actual

Fig. 1: Expected and actual topologies for the purpose of demonstration. Assumption-all assumes that *all* instances are used in estimation, hence topology would look like a). Assumption-pop states that only the *popular* instances (filled squares) are used for estimation.

using *all* instances. Furthermore, this structural information is used to guide a nearest-neighbor based (only use the closest neighbor for estimation) ABE. It is shown that with as few as 17.3% of the entire training set (desharnais dataset of Figure 13), it is possible to attain the same performance values.

Figure 1 illustrates the implicit topologies suggested by *assumption-all* and *assumption-pop* in the context of nearest-neighbor based ABE. In Figure 1 a hypothetical dataset of 3 dimensions (with values between 0 and 1) is depicted. *Assumption-all* favors a topology as in Figure 1a, i.e. it states that *all* instances are useful for estimation (see cluster where all instances are closest neighbor of some other instance). On the other hand *assumption-pop* states that some of the instances are more *"popular"* (squares) than the others and a majority of the instances are redundant (plusses) for estimation in a closest-neighbor setting. Note that Figure 1 only demonstrates a concept, actual SEE datasets have much higher dimensions.

1.1 Research Questions

The research questions that guided our study are as follows:

- RQ1: Which dataset assumptions are supported by SEE datasets and what is the implication of these assumptions on the dataset topology?
- RQ2: How does the dataset topology affect the data collection/labeling effort?
- RQ3: What is the performance of active learning-based ABE methods to other algorithms?

1.2 Contributions

The contributions of this study are:

- An investigation of software effort dataset characteristics
- The first application of active learning on software effort estimation
- An active-learning guidance system based on dataset characteristics
  - Reduction in data collection effort

2 Motivation

The intuition of this work came from a quirk during the experiments of another study in our laboratory. We were interested in the effect of injecting noise to the datasets in the context of ABE. To our surprise, when a portion of the labels were shuffled (after shuffling instances become noise), the ABE performances before and after noise injection were statistically the same. This was a hint that the datasets had a different topology than predicted by *assumption-all*.

The initial experiments of this paper questioned the differences between the expected and the actual topology of the instance space. Our expectation was that the whole instance space was useful, i.e. topology is governed by *assumption-all*. However, when we generated the so called $E(k)$ matrices (see §4 for details), we were surprised to see that the median of useful instances during estimation in a *closest-neighbor* setting had a median value of 25%. In other words, the effort values of more than half the instances in the datasets were redundant. The exact percentage of instance that are closest-neighbor to any other instance are given in Figure 2. We used that information to build an active-learning based guidance system for ABE.

Hassan et al. envision the active learning solutions on data collection as a future direction in SE [9]. The experience of various researchers (including the authors of this paper) [2]–[6] concerning *data-drought* also points in that direction. In other words a solution to aid the expensive data collection activities is urgent. This urgency is even more serious for SEE as collection/labeling of new instances (software projects) can mean many years. This paper is an answer to that urgent problem and is the first application of active learning in SEE.

Dataset	Percentage
kemerer	67%
desharnaisL2	64%
desharnaisL3	60%
cocomo81s	55%
cocomo81o	50%
nasa93_center_1	50%
cocomo81e	36%
telecom	33%
sdr	25%
desharnais	25%
nasa93	22%
desharnaisL1	17%
nasa93_center_2	16%
cocomo81	16%
finnish	16%
nasa93_center_5	15%
maxwell	15%
miyazaki94	13%

Fig. 2: Ratio of the instances used for prediction in a closest-neighbor setting to the dataset size. Note that the median percentage value is 25%, meaning that only a limited amount of instances are the closest neighbor of other instances and are useful in estimation.

3 BACKGROUND

3.1 Software Effort Estimation

Software effort estimation (SEE) can be defined as the process/activity of estimating the total effort necessary to complete a software project [10]. Various different methods used for effort estimation are grouped under two main categories:

- algorithmic methods
- non-algorithmic methods

Algorithmic methods require labeled historical data so as to learn a model. Their estimations are generated by passing new projects through the learned model. There is a very high number of proposed models in SEE. Figure 3 of [1] shows that for analogy-based effort estimation alone, likely combinations are more than 6000. Furthermore model building comprise an important portion of SEE research. The biggest research topic in SEE since 1980s is the introduction and comparison of new methods [11]. Some examples to algorithmic methods are: various kinds of regression (simple, partial least square, stepwise, regression trees), neural networks and instance-based algorithms, just to name a few. The common property of all these algorithmic methods is that they all require *labeled* data. Therefore, the experiments of this study, which show considerable reductions in labelling effort concern an important portion of SEE literature.

Non-algorithmic methods makes use of experienced human experts. Non-algorithmic methods, a.k.a. expert-based estimation, is defined to be a human intensive approach that is most commonly adopted in practice [12]. On one hand, these methods are flexible and intuitive as they can be applied in a variety of circumstances where other estimating techniques do not work. For example, when there is no historical data or the requirements of a project are unavailable at the initial stages, a rough estimate in a very short period of time can be provided by expert estimates. On the other hand -regardless of the efforts to establish guidelines for expert-based methods [12]- there are still many ad-hoc methods used in practice.

3.2 Active Learning

Active learning is based on the assumption that some instances are more informative than the others when building a learner [7], [8]. Its motivation comes from the fact that labeling all the instances is very costly and it can be significantly reduced by active learning heuristics [8].

Active learning differs from passive supervised learning. The heuristics assume that learner has some control over the training examples [13]. Through so called experts (human or algorithmic), learner can choose which instances are to be labeled.

In machine learning literature there is a significant amount of active learning studies. Dasgupta et al. ask for generalizability guarantees in active learning [14]. They use a greedy active learning heuristic and show that it can attain the same performance as good as any other heuristic in terms of reducing the number of required labels [14]. In [7], Kaariainen et al. investigate the noise injection effect on the active learning performance in terms of classification performance. Balcan et al. show in [15] that -given the samples are i.i.d.- active learning can attain the same performance as a supervised learner with exponentially less samples. In [16], Wallace et al. use active learning for a deployed practical application. They propose a citation screening model based on active learning augmented with a priori expert knowledge.

In software engineering, the practical applications of active learning can be seen in software testing [17], [18]. In [17] active learning is used to augment learners for automatic classification of program behavior. Bowring et al. show that learners augmented with active learning yield significant reduction in data labeling effort and they can generate comparable results to those of supervised learning. Xie et al. use human inspection as an active learning strategy for effective test generation and specification inference [18]. In their experiments, the amount of selected tests for the human inspection were feasible, i.e. labeling required much less effort than screening all the tests. Hassan et al. points out active learning as part of the future of software engineering data mining [9]. However, to the best of our knowledge this promising direction for data analysis has not been exploited in software effort domain. Hence, this paper is the first step in that direction.

4 METHODOLOGY

4.1 Algorithms

The algorithms used in this study are the combination of a pre-processor and a learner. The pre-processors are *logging (log)* and *normalization (norm).* With the **norm** preprocessor, numeric values are normalized to a 0-1 interval using Equation 1. Normalization means that no variable has a greater influence than any other.

$$normalizedValue = \frac{(actualValue - min(allValues))}{(max(allValues) - min(allValues))} \quad (1)$$

With the **log** preprocessor, all numerics are replaced with their natural logarithm value. This **log**ging procedure minimizes the effects of the occasional very large numeric values.

The learners are:

- *An instance-based learner:* ABE0-*x*NN and
- *An iterative dichotomizer :* **C**lassification **A**nd **R**egression **T**rees (*CART*).

ABE0 is our name for a very basic type of ABE that we derived from various ABE studies [19]–[21]. In **ABE0-*x*NN**, features are firstly normalized to 0-1 interval, then the distance between test and train instances is measured according to Euclidean distance function, *x* nearest neighbors are chosen from the training set and finally for finding estimated value (a.k.a adaptation procedure) the median of *x* nearest neighbors is calculated. We adopted a single *x* value in this study:

> **ABE0-1NN:** Only the closest analogy is used. Since the median of a single value is itself, the estimated value in **ABE0-1NN** is the actual effort value of the closest analogy.

The two pre-processors and the learners are combined to form two different learners:

- log&ABE0-1NN
- norm&CART

The reason for the selection of these particular algorithms is a prior work of the authors [1], where 90 algorithms are evaluated on the datasets of this study. As a result of this extensive study, norm&CART as well as log&ABE0-1NN turned out to be superior to other algorithms.

There are two different versions of *log&ABE0-1NN*: the one working on the so called *"active-pool"* (the pool that contains only the instances labeled by the active learning-based guiding system) and the one working on a training set with all instances labeled. For convenience we will name the former as *"activeNN"* and the latter as *"passiveNN"*. Since we have only one CART based algorithm (*norm&CART*) the learner name (CART) and the algorithm name (*norm&CART*) will be used interchangeably from now on.

4.2 Building a Guidance System

Generate distance-matrices: For a dataset D of size N, the associated distance-matrix (DM) is an $N \times N$ matrix that keeps the distances between every possible instance-tuple. For example, a cell located at i^{th} row and j^{th} column ($DM(i,j)$) keeps the distance between i^{th} and j^{th} instances of the dataset D. By its definition, matrix DM has certain properties:

- Symmetric: Since the distance between the instances *i and j* is equal to the distance between the instances *j and i*.
- Zero Diagonals: As cells on the diagonal ($DM(i,i)$) represent the distances of the instances to themselves, diagonal entries are zero.

Generate E(k) matrices: "**E**veryones **k**-th nearest matrix" (E(k)) can be defined as the static analysis of the instance space. E(k)[i,j] is *true* if "j" appears in the *k* nearest neighbors of "i" and *false* otherwise. The trivial case where i=j is ignored, i.e. an instance's nearest neighbor does not include itself. In this study the nearest-neighbor based ABE is consdiered: E(1) describes just the single nearest-neighbor. The "popularity" of instance "x" is defined to be $\sum_{j=1}^{n} E(1)[x,j]$, i.e. how often is "x" someone else's nearest-neighbor?

Calculate popularity index based on $E(1)$ and determine the sort order for labeling: Popularity index of an instance is the sum of its occurrences as the nearest-neighbor to another instance. We have observed that the popular instances with $E(1)[x,j] = 1$ have a median percentage of 25% among all datasets, meaning that more than half the data is unpopular with $E(1)[x,j] = 0$. Our speculation based on that fact is that popularity offers an active learning scheme:

- Sort training instances descending by their popularity index
- Label instances following that sort order
- Place labelled instances in the active-pool to be used for estimation
- Compare results of different size active-pools with one another as well as with passive-ABE0-1NN and CART.

Find Stopping Point and Halt: The idea behind an active learning-based the guidance system is to provide a clever ordering for the labeling of instances. Thereby, the amount of instances to be labeled will be reduced. In our implementation, after a certain amount of instances are labeled and added to the active pool, the guiding system halts. The stopping point is determined by a set of rules:

- If there is no estimation accuracy improvement in the active-pool for n consecutive times.
  - In our experiments $n = 3$ yielded the best results.
- If the Δ between the best and the worst MRE of the last n instances in the active-pool is infinitesimal
  - In our experiments $n = 3$ and $\Delta < 0.1$ yielded the best results.

4.2.1 Toy Example

Here we provide a toy example to illustrate the step-by-step execution of the active learning guidance system. Assume that the training set of the toy example consists of 3 instances/projects: P_1, P_2 and P_3. Also assume that these projects have 1 dependent and one independent variable. In that case our toy dataset would look like Figure 3.

Project	KLOC	Effort
P_1	20	3
P_2	10	4
P_3	40	7

Fig. 3: The projects of the toy example. Our hypothetical dataset consists of 3 projects described 1 independent variable (KLOC) and 1 dependent variable (effort in man-months).

Since our data has only 1 independent variable, we can visualize it on a linear scale as in Figure 4.

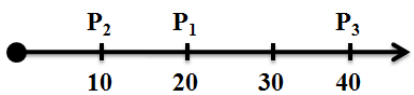


Fig. 4: Visualization of projects on a linear scale, where the axis shows KLOC values.

The first step of the guidance system is to *build the distance matrix* from our training set. Since projects are described by a single attribute (KLOC), the Euclidean distance between two projects will be the difference between the normalized KLOC values. The resulting *distance-matrix* is given in Figure 5.

	P_1	P_2	P_3
P_1	0	0.34	0.66
P_2	0.34	0	1
P_3	0.66	1	0

Fig. 5: The distance matrix of the projects P_1, P_2 and P_3.

Creating the $E(k)$ matrix based on the distance matrix is the second step. As we are creating the $E(k)$ matrix we traverse the distance matrix row-by-row and label the instances depending on their distance order: closest neighbor is labeled 1, the second closest neighbor is labeled 2 and so on. Note that diagonal entries with the distance values of 0 are ignored, as they represent the distance of the instance to itself, not to a neighbor. After this traversal, the resulting $E(k)$ matrix is given in Figure 6.

	P_1	P_2	P_3
P_1	na	1	2
P_2	1	na	2
P_3	1	2	na

Fig. 6: The $E(k)$ matrix resulting from the distance matrix of Figure 5. The cells with a value of na mean that ordering for that cell is *not-applicable*.

Calculating the popularity index based on $E(1)$ and determining the labeling order is the final step of the guidance system. $E(1)$ is a special case of $E(k)$, where cells with $k = 1$ are marked with 1's and the others are marked with 0's. The popularity index associated with each instance is calculated by summing the values in every column, i.e. the sum of the 1^{st} column is the popularity index of the 1^{st} instance, the sum of the 2^{nd} column is the popularity index of the 2^{nd} instance and so on. The $E(1)$ matrix and the popularity indices of our toy example is given in Figure 7.

		P_1	P_2	P_3
	P_1	0	1	0
	P_2	1	0	0
+	P_3	1	0	0
	Popularity :	2	1	0

Fig. 7: The $E(1)$ matrix and the popularity indices of the toy example. Note that popularity index is the sum of the columns of the $E(1)$ matrix.

According to Figure 7 the labeling order of the instances will be: P_1, P_2 and then P_3. In other words, in the first round we will ask our hypothetical expert to label P_1 and place that label in the active pool. In that round, since active-pool contains only 1 *labeled-instance* it will be the closest neighbor of every test instance and the estimates for all the test instances will be the same (the label of P_1). In the second round, P_2 will be labeled by the expert and placed into the active-pool. This time test instances will have 2 alternatives to choose their closest-neighbor from, hence the estimates will be either the label of P_1 or the label of P_2. Finally expert will label P_3 and place it into the active-pool. The change of the active pool is shown in Figure 8. Note that the transition from $Round_i$ to $Round_{i+1}$ in an actual setting is governed by the stopping rules. Therefore, in an actual setting -unlike the toy example- the expert labels only a small portion of the unlabeled instances.

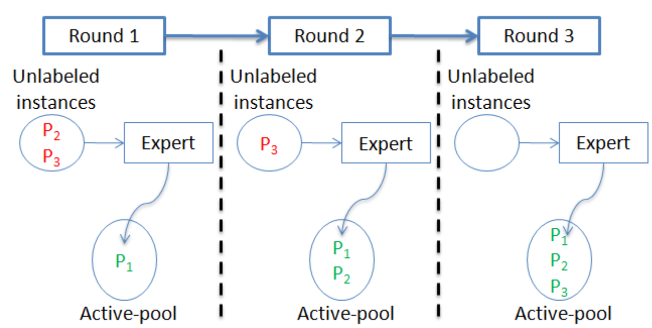


Fig. 8: The change of active pool for the toy example. Note that in an actual setting transition between $Round_i$ to $Round_{i+1}$ is governed by the stopping rules.

4.3 Experiments

Run CART and passiveNN on entire training set: The algorithms are run on the entire training set and their estimations are stored. As for the sampling method 10Way cross-validation is used. 10Way works in the following manner:

- Randomize the order of instances in the dataset
- Divide dataset into 10 bins

Dataset	Used by
telecom	[22], [23]
kemerer	[22]–[24]
cocomo81o	[25]–[27] ,
desharnaisL1	[27] ,
cocomo81s	[25]–[27] ,
desharnaisL3	[27] ,
albrecht	[20], [22]–[24], [28], [29]
cocomo81e	[25], [27], [30]
nasa93_center_5	[25]–[27]
desharnaisL2	[27]
desharnais	[10], [20]–[23], [27], [28], [31]–[33]
maxwell	[28], [34]
sdr	[35], [36]
nasa93_center_1	[25]–[27]
miyazaki94	[37]
nasa93_center_2	[25]–[27]
finnish	[23], [38]
cocomo81	[4], [25]–[27],
nasa93	[25]–[27]

Fig. 9: A sample of effort estimation papers that use the data sets explored in this paper.

- Choose 1 bin at a time as the test set and use the remaining bins as the training set
- Repeat above procedure 10 times

Run activeNN on active-pool: At each iteration active-pool is populated with training instances on the order of their popularity. The assumption with the active-pool approach is that:

- all the training instances outside the active-pool are considered unlabeled
- *activeNN* is only allowed to use instances in the active pool.

Before a training instance is allowed to join the active pool, a hypothetical-expert labels that instance, i.e. the effort value is revealed to the algorithm. At first active-pool only contains 1 instance: the most popular instance, so the estimates based on a single-instance active pool are all the same. As the population of the active-pool increases, *activeNN* has more labeled training instances to estimate from.

Compare algorithms: Once the execution of the algorithms is over, the performance of *activeNN*, *passiveNN* and CART are compared under different performance measures. Note that *activeNN* with different active-pool sizes have different estimates for the test instances. The *activeNN* estimates used for comparison are the ones generated by the active-pool at the stopping point.

4.4 Performance Measures

Performance measures comment on the success of a prediction. For example, the absolute residual (AR) is the difference between the predicted and the actual:

$$AR_i = x_i - \hat{x}_i \tag{2}$$

(where $x_i, \hat{x}_i$ are the actual and predicted value for test instance i).

The Magnitude of Relative Error measure a.k.a. MRE is a very widely used evaluation criterion for selecting the best effort estimator from a number of competing software prediction models [23], [39]. MRE measures the error ratio between the actual effort and the predicted effort and can be expressed as the following equation:

$$MRE_i = \frac{\mid x_i - \hat{x}_i \mid}{x_i} = \frac{\mid AR_i \mid}{x_i} \tag{3}$$

A related measure is MER (Magnitude of Error Relative to the estimate [39]):

$$MER_i = \frac{\mid x_i - \hat{x}_i \mid}{\hat{x}_i} = \frac{\mid AR_i \mid}{\hat{x}_i} \tag{4}$$

The overall average error of MRE can be derived as the Mean or Median Magnitude of Relative Error measure (MMRE, or MdMRE respectively), can be calculated as:

$$MMRE = \frac{\sum_{i=1}^{n} MRE_i}{n} \tag{5}$$

$$MdMRE = median(allMRE_i) \tag{6}$$

A common alternative to MMRE is PRED(25), and defined as the percentage of predictions falling within 25% of the actual values, and can be expressed as:

$$PRED(25) = \frac{100}{N} \sum_{i=1}^{N} \begin{cases} 1 \text{ if } MRE_i \leq \frac{25}{100} \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

For example, PRED(25)=50% implies that half of the estimates are failing within 25% of the actual values [23].

There are many other performance measures including Mean Balanced Relative Error (MBRE) and the Mean Inverted Balanced Relative Error (MIBRE) studied by Foss et al. [39]:

$$MBRE_i = \frac{\hat{x}_i - x_i}{min(\hat{x}_i, x_i)} \tag{8}$$

$$MIBRE_i = \frac{\hat{x}_i - x_i}{max(\hat{x}_i, x_i)} \tag{9}$$

```
if Mann-Whitney(P_i, P_j, 95) says they are the same then
   tie_i = tie_i + 1;
   tie_j = tie_j + 1;
else
   if better( median(P_i), median(P_j)) then
      win_i = win_i + 1
      loss_j = loss_j + 1
   else
      win_j = win_j + 1
      loss_i = loss_i + 1
   end if
end if
```

Fig. 10: Comparing algorithms (*i,j*) on performance (P_i,P_j). The "better" predicate changes according to P. For error measures like MRE, "better" means lower medians. However, for PRED(25), "better" means higher medians.

Performance measures should be supplemented with appropriate statistical checks. Otherwise, they may lead to biased or even false conclusions [39]. In this study so called $win, tie, loss$ statistics are used to aid the performance measures with Mann-Whitney Rank-Sum test

(95% confidence). The pseudo-code of the $win, tie, loss$ statistics is given in Figure 10: We first check if two distributions i, j are statistically different (Mann-Whitney rank-sum test, 95% confidence); otherwise we increment tie_i and tie_j. If the distributions are statistically different, we update win_i, win_j and $loss_i, loss_j$ after comparing the performance measures so as to see which one is better.

4.5 Datasets

There is at least one study in SEE using one or more of the 19 datasets used in our study (see Figure 9). Therefore, the results presented here are based on a large corpus and concern a number of previously published SEE studies. The description of 20 datasets used in this study are provided in Figure 11. These datasets are available at http://promisedata.org/data.

As described in Figure 11, the datasets were collected in different parts of the world:

- The desharnais dataset includes Canadian software projects,
- cocomo81 and nasa93 include projects developed in the United States,
- sdr, contains projects of various software companies in Turkey [30].

Note that three of these data sets (nasa93_center_1, nasa93_center_2, nasa93_center_5) come from different development centers around the United States. Another three of these data sets (cocomo81e, cocomo81o, cocomo81s) represent different kinds of projects (embedded, organic and semi-detached respectively) developed by different team sizes and under different constraints [4].

Note also in Figure 11, the skewness of the effort values (up to 6.06): The datasets are extremely heterogeneous with as much as 60-fold variation. There is also some divergence in the features used to describe the datasets:

- While data sets have some effort values in common (measured in terms of man-months or man-hours), no other feature is shared by all data sets.
- The cocomo* and nasa* data sets use the features defined by Boehm [4]; e.g. analyst capability, required software reliability, memory constraints, and use of software tools.
- The other data sets use a wide variety of features including, number of entities in the data model, number of basic logical transactions, query count and number of distinct business units serviced.

5 Results

We will interpret our results with regards to two different concerns: Reduction in labeling effort and performance. Reduction in labeling effort will question how much effort we can save via an active learning-based guidance system. Note that the effort reduction makes sense only if *activeNN* has a successful performance. Therefore, in the performance part we compare *activeNN* to *passiveNN* and CART.

5.1 Labeling Effort Reduction

The reduction in the labeling effort is related to how many labels *activeNN* requested from the expert. This number is given by the so called *stopping-point* of the guidance system.

In Figure 12 one plot for each performance-category is provided (see §5.2 for performance-categories). These plots show the *MdMRE* values of the learners used in this study. The stopping-point is shown with a vertical line that is parallel to the y-axis. In all cases *activeNN* stops before asking all the labels of the training set.

Note that the *MdMRE* values of CART and *passiveNN* are stable (horizontal line), whereas the *MdMRE* values *activeNN* change at every point in x-axis. This is due to the fact that for *activeNN* the instance numbers in x-axis show the size of the *active-pool* and for different *active-pool* sizes *activeNN* yields different performance values.

Dataset	# Instances	Perc. Labeled
cocomo81s	11	81.8%
desharnaisL3	10	70%
cocomo81e	28	64.3%
cocomo81o	24	62.5%
desharnaisL1	46	56.5%
nasa93_center_1	12	50%
sdr	24	50%
nasa93_center_2	37	48.6%
kemerer	15	46.7%
telecom	18	38.9%
albrecht	24	33.4%
nasa93_center_5	40	33.4%
desharnaisL2	25	24%
finnish	38	23.7%
miyazaki	48	18.7%
cocomo81	63	17.4%
desharnais	81	17.3%
maxwell	62	12.9%
nasa93	93	8.6%

Fig. 13: The percentage of instances that are labeled at the stopping point. The median percentage value is 38.8%. The implication of this table is that it is possible reduce the effort of labeling activities by orders of magnitude.

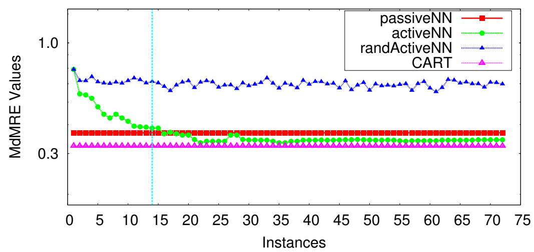
Note also that there is an additional 4^{th} line in the plots of Figure 12: *randActiveNN*. The purpose of *randActiveNN* is to provide a baseline for the *activeNN* to make sure that the results of *activeNN* are premeditated. *randActiveNN* works exactly the same as *activeNN*. The only the difference is that instead of selecting the closest instance from the *active-pool* on the basis of popularity, it randomly picks an instance from the *active-pool*. In other words it violates the core assumption (*assumption-pop*) of the guidance system.

We see in Figure 12 that *randActiveNN* performs much worse and different than *activeNN*. This shows that violation of *assumption-pop* destroys the benefits of the guidance system. It also shows that the performance results of *activeNN* are far from being coincidental.

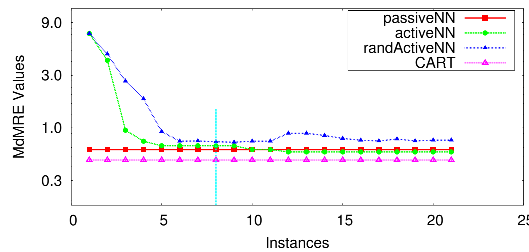
Since this result repeats itself for every dataset (random behavior of *randActiveNN*) we suffice to provide

				Historical Effort Data					
Dataset	**Features**	**Size**	**Description**	**Units**	**Min**	**Median**	**Mean**	**Max**	**Skewness**
cocomo81	17	63	NASA projects	months	6	98	683	11400	4.4
cocomo81e	17	28	Cocomo81 embedded projects	months	9	354	1153	11400	3.4
cocomo81o	17	24	Cocomo81 organic projects	months	6	46	60	240	1.7
cocomo81s	17	11	Cocomo81 semi-detached projects	months	5.9	156	849.65	6400	2.64
nasa93	17	93	NASA projects	months	8	252	624	8211	4.2
nasa93_center_1	17	12	Nasa93 projects from center 1	months	24	66	139.92	360	0.86
nasa93_center_2	17	37	Nasa93 projects from center 2	months	8	82	223	1350	2.4
nasa93_center_5	17	40	Nasa93 projects from center 5	months	72	571	1011	8211	3.4
desharnais	12	81	Canadian software projects	hours	546	3647	5046	23940	2.0
desharnaisL1	11	46	Projects in Desharnais that are developed with Language1	hours	805	4035.5	5738.9	23940	2.09
desharnaisL2	11	25	Projects in Desharnais that are developed with Language2	hours	1155	3472	5116.7	14973	1.16
desharnaisL3	11	10	Projects in Desharnais that are developed with Language3	hours	546	1123.5	1684.5	5880	1.86
sdr	22	24	Turkish software projects	months	2	12	32	342	3.9
albrecht	7	24	Projects from IBM	months	1	12	22	105	2.2
finnish	8	38	Software projects developed in Finland	hours	460	5430	7678.3	26670	0.95
kemerer	7	15	Large business applications	months	23.2	130.3	219.24	1107.3	2.76
maxwell	27	62	Projects from commercial banks in Finland	hours	583	5189.5	8223.2	63694	3.26
miyazaki94	8	48	Japanese software projects developed in COBOL	months	5.6	38.1	87.47	1586	6.06
telecom	3	18	Maintenance projects for telecom companies	months	23.54	222.53	284.33	1115.5	1.78
		Total: 699							

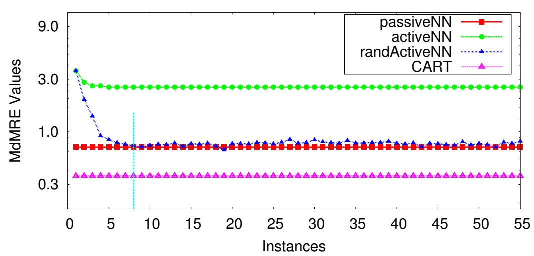
Fig. 11: The 699 projects used in this study come from 19 datasets. Indentation in column one denotes that indented dataset is a subset of its non-indented parent.



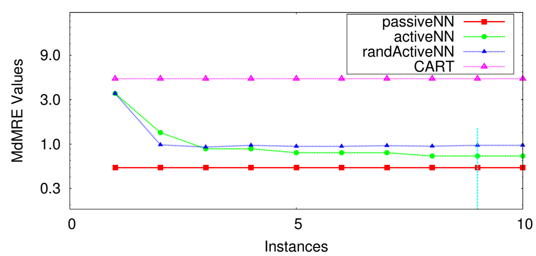
(a) Pro-Active: desharnais



(b) Pro-CART: albrecht



(c) Con-Active: maxwell



(d) Con-CART: cocomo81s

Fig. 12: Sample plots for different category of results. The line parallel to y-axis indicates the stopping point.

one plot per category. The summary of all the datasets is provided in Figure 13. Figure 13 shows the percentage of labeled instances in comparison to dataset sizes. Note that the percentage labeled instances can be as low as 8.6% with a median percentage value of 38.8%. The lowest-percentage dataset that belongs to the category of *Pro-Active* is the *desharnais* dataset with a percentage value of 17.3%. These results show that *activeNN* works with orders of magnitude less labels when compared to *passiveNN*. The implication of this finding is quite striking. It means that we can save an orders of magnitude effort from dataset labeling activities in an actual setting.

5.2 Performance

Figure 14 shows the comparison of *activeNN*, *passiveNN* and CART subject to 7 different performance measures. The table summarizes the performance measures in terms of $win - loss$ values. Figure 14 alone is difficult to interpret, therefore we use it as the basis to a more structured analysis: The results of Figure 14 is interpreted in terms of result categories. Our performance results can be grouped into 4 categories: *Pro-Active*, *Con-Active*, *Pro-CART* and *Con-CART*.

Pro-Active: In this category, the *activeNN* has comparable or superior performance (in at least 4 out of 7 error measures) to *passiveNN*. We are interested in *comparable* results as well as the better results, because in both cases *activeNN* has a considerable reduction in data labeling effort.

Pro-CART: For the datasets falling in this category, CART is dominantly superior to *activeNN* (better in terms of 3 or more performance measures).

Con-Active: For the datasets in this category *activeNN* is the worst performing algorithm, i.e. it loses to *passiveNN* AND CART (which means $win - loss$ value of -2) in 3 or more performance measures.

Con-CART: For the datasets in this category, CART is the worst performer, i.e. it is worse than *passiveNN* AND *activeNN* (which means $win - loss$ value of -2) in 3 or more performance measures.

Dataset	Method	MMRE	MAR	Pred(25)	MdMRE	MBRE	MIBRE	MMER
cocomo81	passive	1	0	1	1	0	0	1
	active	1	0	0	0	0	0	-2
	cart	-2	0	-1	-1	0	0	1
cocomo81e	passive	2	2	0	2	2	2	1
	active	0	-1	-1	-1	-1	-1	-2
	cart	-2	-1	1	-1	-1	-1	1
cocomo81o	passive	-1	-2	-1	-1	-1	-1	-1
	active	2	1	2	2	2	2	-1
	cart	-1	1	-1	-1	-1	-1	2
cocomo81s	passive	1	1	2	2	0	2	2
	active	1	1	0	0	2	0	0
	cart	-2	-2	-2	-2	-2	-2	-2
desharnais	passive	1	-1	-1	-1	-1	-1	-1
	active	1	-1	-1	-1	-1	-1	-1
	cart	-2	2	2	2	2	2	2
desharnaisL1	passive	0	2	2	2	2	2	2
	active	2	-2	-2	-2	-2	-2	-1
	cart	-2	0	0	0	0	0	-1
desharnaisL2	passive	1	0	-2	-2	0	0	0
	active	1	0	1	1	0	0	-1
	cart	-2	0	1	1	0	0	1
desharnaisL3	passive	1	1	1	1	1	1	-1
	active	1	1	1	1	1	1	0
	cart	-2	-2	-2	-2	-2	-2	1
nasa93	passive	2	0	2	2	2	2	0
	active	0	-2	-2	-2	-2	-2	-2
	cart	-2	2	0	0	0	0	2
nasa93_center_1	passive	1	-1	0	0	-1	-1	-1
	active	1	0	-1	-1	-1	-1	-1
	cart	-2	1	1	1	2	2	2
nasa93_center_2	passive	2	1	1	1	1	1	2
	active	0	-1	1	1	-1	1	-2
	cart	-2	0	-2	-2	0	-2	0
nasa93_center_5	passive	1	0	0	0	1	1	1
	active	1	0	-1	-1	-2	-2	-2
	cart	-2	0	1	1	1	1	1
sdr	passive	1	1	1	1	2	2	2
	active	1	0	1	1	-1	-1	-1
	cart	-2	-1	-2	-2	-1	-1	-1
albrecht	passive	0	-1	-1	-1	-1	-1	-2
	active	0	-1	-1	-1	-1	-1	1
	cart	0	2	2	2	2	2	1
finnish	passive	2	-1	-2	0	-1	-1	0
	active	0	-1	0	-2	-1	-1	-2
	cart	-2	2	2	2	2	2	2
kemerer	passive	0	0	0	0	0	0	-2
	active	1	0	0	0	0	0	1
	cart	-1	0	0	0	0	0	1
maxwell	passive	1	0	0	0	0	0	-2
	active	-2	-2	-2	-2	-2	-2	0
	cart	1	2	2	2	2	2	2
miyazaki94	passive	1	-1	0	0	-2	0	-1
	active	1	1	-2	-2	0	-2	-1
	cart	-2	0	2	2	2	2	2
telecom1	passive	1	0	0	0	0	0	-1
	active	1	0	0	0	0	0	0
	cart	-2	0	0	0	0	0	1

Fig. 14: The $win - loss$ values. Those results are used to form the so called *"performance categories"*.

The distribution of the datasets to these categories are given in Figure 15. The $win{-}loss$ values used to generate Figure 15 are given in Figure 14.

In Figure 15 12 datasets fall into the category of *Pro-Active* and 5 fall into *Con-Active*. A total of $12 + 5 = 17$ datasets are able to differentiate *activeNN* from *passiveNN* as better or worse. Among 17 datasets, for 70% of the datasets *activeNN* is a substitute for *passiveNN*. In other words, in 70% of the datasets *activeNN* is comparable to or better than *passiveNN* for much less data labeling effort.

In the category of *Con-Active* there are 5 datasets. In other words, only in $5/19 = 26\%$ of the datasets was *activeNN* worse than both competitors at the same time.

Upon that information we can say that for the majority of the datasets *assumption-pop* works. In other words, for less labeling effort we can gain the same performance as *assumption-all.*

Category	Datasets	#
Pro-Active	cocomo81, cocomo81o, cocomo81s desharnais, desharnaisL2, desharnaisL3 nasa93_center_1 , albrecht, finnish kemerer , miyazaki94, telecom	12
Con-Active	desharnaisL1, nasa93, nasa93_center_5 maxwell, miyazaki94	5
Pro-CART	desharnais, nasa93_center_1, nasa93_center_5 albrecht, finnish, maxwell miyazaki94	7
Con-CART	cocomo81s, desharnaisL3, nasa93_center_2 sdr	4

Fig. 15: The distribution of datasets into result-categories. Last column shows the number of datasets in each category. Note that 12 datasets fall into the category of *Pro-Active* and 5 fall into *Con-Active*. Among $12 + 5 = 17$ datasets that differentiate *activeNN* from *passiveNN*, for 70% of the datasets *activeNN* is a substitute for *passiveNN*.

As for the performance of CART in comparison to *activeNN*, we look at the category of *Pro-CART*. There are 7 datasets in *Pro-CART*, which shows that for 37% of the datasets CART is dominantly superior to *activeNN*. This shows that for the majority of the datasets ($100\% - 37\% = 63\%$), *activeNN* is comparable to the most successful algorithm reported in [1]. Therefore, an *assumption-pop* based guidance system does not only outperform standard passive ABE methods, but also is comparable to the state-of-the-art learners.

The category of *Con-CART* shows the failure of CART and contains only 4 datasets. For only 21% of the datasets CART performs worse than both ABE methods. The results of this last category support the findings of [1] in an ABE context.

6 Threats To Validity

Internal validity asks to what extent the cause-effect relationship between dependent and independent variables holds [40]. The ideal case to observe that relationship would be to learn a theory on the available data and apply the *learned* theory on completely new and unseen data. However, considering the *data-drought* in SEE, the ideal case is unfeasible. Therefore, the ideal scenario is simulated by sampling methods to separate the available data into training and test sets. In this study we used 10Way sampling method to simulate the ideal case.

External validity is the question of how widely the results can be generalized [41]. So as to observe the applicability of our results to a wide spectrum of SEE datasets, we use a total of 19 datasets that have very different characteristics. Although the analysis on 19 datasets is more extensive than an average SEE study, we acknowledge that our experiments need to be replicated on new datasets.

Construct validity (i.e. face validity) asks if we are in fact measuring what we intend to measure [42]. A beneficial discussion can be found in [43], where Kitchenham et al. state that different performance measures evaluate different aspects of the prediction accuracy. So as to evaluate our results in terms of different aspects, we use 7 different performance measures. Another point made by Kitchenham et al. is that sole usage of performance measures is wrong and they need to be supported with statistical checks. To address that validity issue we use $win - tie - loss$ statistics, where we make use of Mann-Whitney U test at a significance level of 95%.

Another validity issue is the use of experts in active learning guidance system. The assumption with the experts is that they are able to provide the labels of the training set upon a request from the guidance system. As the labels in the training set is also collected by the experts this assumption is fairly appropriate. On the other hand an interesting application would be to introduce a white-noise on the data labels to introduce the expert-judgment error. This is currently left as a future work.

7 Conclusions

Our conclusions are three-fold following the research questions that guided this study.

RQ1: Which dataset assumptions are supported by SEE datasets and what is the implication of these assumptions on the dataset topology? In this study two different assumptions are investigated: *assumption-all* and *assumption-pop*. The former assumes a random topology and states that any instance may be the closest instance to another instance; hence, all instances of the training set are useful. The latter assumes a more structured topology where popular instances are central and are the closest-neighbor to multiple other instances; hence a limited number of labels is sufficient. At the end of our analysis, we have seen that *assumption-all* does not hold and SEE datasets have a topology indicated by *assumption-pop*.

RQ2: How does the dataset topology affect the data collection/labeling effort? We have seen the assumptions regarding SEE dataset topology can be successfully used as a guidance system. Such a guiding removes the need to label all the instances in a dataset. In fact we observed that it is possible to reduce the number of instances to be labeled by orders of magnitude. Therefore, our conclusion is that immense amount of effort spent in data collection/labeling can be saved with the proposed guiding system.

RQ3: What is the performance of active learning-based ABE methods to other algorithms? Based on the topology foreseen by *assumption-pop* we defined an active learning-based guidance system to be used with ABE0-1NN. The combination (*activeNN*) proved to be comparable to standard passive-learning based ABE0-1NN (*passiveNN*) as well as more complex learners like CART.

8 Future Work

This initial study uses a single popularity metric: so called E(k) matrix. A plausible future direction would be to device new popularity metrics based on the data topology. Those new popularity metrics may as well be used to augment the E(k) matrix. For example E(k) matrices only cares about the popularity index, i.e. two instances with the same popularity index are treated the same. However, this index can be augmented with a total-distance metric meaning that the instance with the lower total-distance is preferable to the other one.

Another point of future research is the stopping rules: When should the guidance-system stop asking for new labels? Currently we make use of multiple rules and they offer favorable performance values. However, current rules are heuristics and can only approximate the optimum stopping point.

One final and fairly easy-to-do future direction to this research is to introduce the human-error into the experimentation. Current version of the paper assumes that experts can reveal the actual effort values in the training set. However, a company with novice experts may end up labels that are far from being perfect. Addition of a white-noise to the actual effort values as the human-error can simulate such an experimentation.

References

[1] J. Keung, E. Kocaguneli, and T. Menzies, “A Ranking Stability Indicator for Selecting the Best Effort Estimator in Software Cost Estimation,” *Automated Software Engineering (submitted)*, 2011. [Online]. Available: http://menzies.us/pdf/11draftranking.pdf

[2] T. Menzies, S. Williams, O. Elrawas, D. Baker, B. Boehm, J. Hihn, K. Lum, and R. Madachy, “Accurate Estimates Without Local Data?” *Software Process Improvement and Practice*, vol. 14, no. 4, pp. 213–225, Jul. 2009.

[3] T. Menzies, O. Elrawas, J. Hihn, M. Feathear, B. Boehm, and R. Madachy, “The Business Case for Automated Software Engineerng,” in *ASE ’07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*. New York, NY, USA: ACM, 2007, pp. 303–312.

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

[5] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous & Practical Approach*. International Thompson Press, 1997.

[6] N. Fenton, “Keynote address, PROMISE’07,” 2007.

[7] M. Kääriäinen, “Active learning in the non-realizable case,” *Algorithmic Learning Theory*, 2006. [Online]. Available: http://www.springerlink.com/index/PKU3430515658M85.pdf

[8] S. Dasgupta and D. Hsu, “Hierarchical sampling for active learning,” *Proceedings of the 25th international conference on Machine learning - ICML ’08*, pp. 208–215, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1390156.1390183

[9] A. Hassan and T. Xie, “Software intelligence: the future of mining software engineering data,” in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 161–166. [Online]. Available: http://portal.acm.org/citation.cfm?id=1882397

[10] J. W. Keung, “Theoretical Maximum Prediction Accuracy for Analogy-Based Software Cost Estimation,” *2008 15th Asia-Pacific Software Engineering Conference*, pp. 495–502, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4724583

[11] M. Jorgensen and M. Shepperd, “A Systematic Review of Software Development Cost Estimation Studies,” *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, 2007.

[12] M. Jorgensen, “A review of studies on expert estimation of software development effort,” *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 37–60, Feb. 2004.

[13] D. Cohn, L. Atlas, and R. Ladner, “Improving generalization with active learning,” *Machine Learning*, vol. 15, no. 2, pp. 201–221, May 1994. [Online]. Available: http://www.springerlink.com/index/10.1007/BF00993277

[14] S. Dasgupta, "Analysis of a greedy active learning strategy," *in Neural Information Processing Systems 17:*, vol. 1, no. x, 2005. [Online]. Available: http://books.google.com/books?hl=en\&lr=\&id=etp-l5VrbHsC\&oi=fnd\&pg=PA337\&dq=Analysis+of+a+greedy+active+learning+strategy\&ots=_J6C5HxGBI\&sig=2oGAcWkGNbmnXf-WBblPJSDP61Y

[15] M.-F. Balcan, A. Beygelzimer, and J. Langford, "Agnostic active learning," *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pp. 65–72, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1143844.1143853

[16] B. Wallace, K. Small, C. Brodley, and T. Trikalinos, "Active learning for biomedical citation screening," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 173–182. [Online]. Available: http://portal.acm.org/citation.cfm?id=1835829

[17] J. F. Bowring, J. M. Rehg, and M. J. Harrold, "Active learning for automatic classification of software behavior," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, p. 195, Jul. 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1013886.1007539

[18] T. Xie and D. Notkin, "Mutually enhancing test generation and specification inference," *Formal Approaches to Software Testing*, pp. 1100–1101, 2004. [Online]. Available: http://www.springerlink.com/index/GJEU06J1KGQ95GYX.pdf

[19] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A comparative study of cost estimation models for web hypermedia applications," *Empirical Software Engineering*, vol. 8, no. 2, pp. 163–196, 2003. [Online]. Available: http://www.springerlink.com/index/J322342V568V429U.pdf

[20] Y. Li, M. Xie, and T. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," *Journal of Systems and Software*, vol. 82, no. 2, pp. 241–252, 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0164121208001325

[21] G. Kadoda, M. Cartwright, and M. Shepperd, "On configuring a case-based reasoning software project prediction system," in *UK CBR Workshop, Cambridge, UK*. Citeseer, 2000, pp. 1–10. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.6926\&rep=rep1\&type=pdf

[22] J. Keung and B. Kitchenham, "Experiments with Analogy-X for Software Cost Estimation," in *ASWEC '08: Proceedings of the 19th Australian Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 229–238.

[23] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, 1997.

[24] G. R. Finnie and G. E. Wittig, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models," *Performance Evaluation*, vol. 1212, no. 97, pp. 281–289, 1997.

[25] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *IEEE Transactions on Software Engineering*, vol. 32, no. 11, pp. 883–895, 2006. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/TSE.2006.114

[26] K. Lum, T. Menzies, and D. Baker, "2CEE, A TWENTY FIRST CENTURY EFFORT ESTIMATION METHODOLOGY," *ISPA / SCEA*, no. May, pp. 12–14, 2008.

[27] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. Keung, "When to use data from other projects for effort estimation," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010, pp. 321–324. [Online]. Available: http://portal.acm.org/citation.cfm?id=1859061

[28] Y. Li, M. Xie, and G. T., "A study of the non-linear adjustment for analogy based software cost estimation," *Empirical Software Engineering*, pp. 603–643, 2009.

[29] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *ICSE '96: Proceedings of the 18th international conference on Software engineering*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 170–178.

[30] A. Bakir, E. Kocaguneli, A. Tosun, A. Bener, and B. Turhan, "Xiruxe: An Intelligent Fault Tracking Tool," in *AIPR09*, Orlando, 2009.

[31] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery, "Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 471–484, 2008.

[32] J. Li and G. Ruhe, "Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA +," *Learning*, pp. 63–96, 2008.

[33] C. Kirsopp and M. Shepperd, "Making inferences with small numbers of training sets," *IEEE Proc.*, vol. 149, no. 5, 2002.

[34] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software productivity and effort prediction with ordinal regression," *Information and Software Technology*, vol. 47, no. 1, pp. 17 – 29, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0B-4CVR4MP-1/2/72e600ea322bfe4d64b1c90c3a968013

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

[36] B. Turhan, O. Kutlubay, and A. Bener, "Evaluation of Feature Extraction Methods on Software Cost Estimation," *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, vol. 290, pp. 497–497, Sep. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4343793

[37] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, "Robust regression for developing software estimation models," *J. Syst. Softw.*, vol. 27, no. 1, pp. 3–16, 1994.

[38] L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. D. Maxwell, "An assessment and comparison of common software cost estimation modeling techniques," in *ICSE '99: Proceedings of the 21st international conference on Software engineering*. New York, NY, USA: ACM, 1999, pp. 313–322.

[39] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Transactions on Software Engineering*, 2003.

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

[41] D. Milic and C. Wohlin, "Distribution Patterns of Effort Estimations," in *Euromicro*, 2004.

[42] C. Robson, *Real world research: a resource for social scientists and practitioner-researchers*. Blackwell Publisher Ltd, 2002.

[43] B. A. Kitchenham, L. M. Pickard, S. G. Macdonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEEE Proceedings-Software*, vol. 148, no. 3, pp. 101 049/ip—-sen20 010 506, 2001.