Learning to Change Projects

Raymond Borges
Lane Department of CS&EE
West Virginia University, USA
borgesraymond@gmail.com

Tim Menzies
Lane Department of CS&EE
West Virginia University, USA
tim@menzies.us

ABSTRACT

Background: Most software effort estimation research focuses on methods that produce the most accurate models but very little focuses on methods of mapping those models to business needs.
Aim: In our experience, once a manager knows a software effort estimate, their next question is how to *change* that estimate. We propose a combination of inference + visualization to let managers quickly discover the important changes to their project.
Method: (1) We remove superfluous details from project data using dimensionality reduction, column reduction and feature reduction. (2) We visualize the reduced space of project data. In this reduced space, it is simple to see what project changes need to be taken, or avoided.
Results: Standard software engineering effort estimation data sets in the PROMISE repository reduce to a handful of rows and just a few columns. Our experiments show that there is little information loss in this reduction: in 20 datasets from the PROMISE repository, we find that there is little performance difference between inference over all the data and inference over our reduced space.
Conclusion: Managers can be offered a succinct representation of project data, within which it is simple to find critical the decisions that most impact project effort.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Management—Time Estimation; I.2.6 [**Artificial Intelligence**]: Learning—Analogies, induction

General Terms

Algorithms, experimentation

Keywords

Effort Estimation, Optimization

1. INTRODUCTION

Software engineering projects produce data. Strangely, despite the abundance of such data, it is still difficult to offer *useful* insight about projects to project managers (and by "useful", we mean lists of critical decisions that managers should either take or avoid). Menzies & Shull [16], followed by Menzies & Shepperd [15] discuss the difficulties associated with reaching clear conclusions using current data mining methods.

PROMISE '12, September 21-22, 2012 Lund, Sweden

A likely contributing factor to these software project failures is that there continues to be a substantial disconnect between the information and insights needed by project managers to make good decisions and that which is typically available to them [21]. Much research in software effort estimation focuses on introduction and evaluation of software estimation methods (as much as 61% of topics collected in a study of over 300 papers from 76 journals [17]). Very little research has focused on connecting the methods and models to business needs.

When project managers cannot understand method results or do not have access to data mining tools they have to fall back on intuition. Such intuition-based decisions can sometimes work out well, but often they are unnecessarily suboptimal or even destructive [10]. This may also be the case when managers are overloaded with results too complex, which, although may be very accurate, are not straightforward and the manager does not understand how to correctly apply them.

In this paper we intend to focus on simplifying the task of reviewing project data. The goal of this task is to read project data and find the critical decisions that most change a project's estimate. We find that after applying three data *reduction operators* and one *visualization operator* (dimensionality reduction, row reduction, column reduction and rule reduction) to project data from the PROMISE repository, the reduced data is small enough for manual browsing. These reductions are implemented within our IDEA tool (**IDEA** is short for **I**terative **D**ichotomization on **E**very **A**ttribute).

Our pre-experimental concern was that the reductions performed by IDEA might remove important data. Hence, this paper compares leave-one-out predictions using:

1) a nearest neighbor method in our reduced set of project data
2) the predictions made by 90 other methods on 20, PROMISE, effort estimation data sets. Note that the predictions in this second study could use all the project data.

As discussed below, there is little difference between the predictions found using (1) the reduced data or using (2) all the data, that is, our reductions were not discarded information essential to learning significant project changes. Hence, we can recommend IDEA as a tool for generating project recommendations.

This paper is structured as follows. §2 discusses the motivation behind the research. §3 summarizes the main algorithm supporting the research called IDEA. §4 talks about the practical application by giving worked examples of IDEA. In §5 we discuss our results when compared over 20 datasets against 90 methods commonly found in SE effort literature [1, 2, 5, 9, 11, 12, 17, 19, 20]. In §6 we summarize our results and discuss their significance. §7 are some acknowledgements. §8 are references.

2. MOTIVATION

The ability to explain how a conclusion was reached is a very powerful tool for helping users to understand and accept the conclusions of a data miner. Despite this, sometimes explanatory power must be decreased in order to increase the efficacy of the predictor [14].

Industrial practitioners of data mining might need a fast algorithm but may lack the skill to understand how it works and therefore be unsure of its trustworthiness. We propose there is a method to have both a sufficient understanding and a fast algorithm by using accurate visualization of the results. In this paper there are many visualizations, but, the simplicity of IDEA result's makes it much easier to comprehend its predictions straight out of the method.

Data mining methods grow more and more complex as well as faster and more accurate. Combining learners may produce even more accurate results. This combination of learners is known as multi-methods. Given the state of the art in data mining, we can now arrive at predictions which prove accurate but we cannot comprehend their meaning because the complexity necessary to reach those conclusions seems too great.

Hence, for this paper, we explore simplifications of the data sets rather than elaborations of algorithms. Surprisingly, in a field called "data mining", there are few reports about invariants in the data being explored. Rather, most papers (e.g. at PROMISE) discuss details of how different algorithms might explore this data. Here, we show that the information content of the PROMISE effort estimation data sets can be represented in a handful of rows and columns. We hope this paper prompts other researchers to spend more time exploring features of the data rather than minor details about their algorithms.

3. DATA REDUCTION OPERTORS

This section is an overview of IDEA's data reduction operators.

The premise of IDEA is that we study the data to find insights beneath any irrelevancies. To do this, we must first remove these irrelevancies and use what lies beneath to make good predictions.

Our method IDEA is a divisive hierarchical clustering algorithm based on FastMap [7]. IDEA performs *dimensionality reduction* by recursive clustering decent where at each level it computes a new derived dimension. Note that this generates a dendrogram (a tree of clusters) with leaf clusters. The original data can then be tagged with the identifier of the leaf cluster than contains them.

In the following discussion, we will use the following definition. We will say that a project's *context* is its nearest leaf cluster. Note that a context is characterized by the centroid of that cluster.

After dimensionality reduction comes *row reduction*. All rows in one leaf cluster (found via FastMap) are replaced with their centroid. To generate this centroid, a new row is synthesized using the median/mode of all numeric/symbolic columns.

After row reduction comes *column reduction*. We apply an adaption of the Fayyad-Irani discretizer. All rows are tagged with the context that contains them. All columns are then ranked via the probability that the values in that column will select for the fewest contexts. In this way, we can select the columns whose values are only found in a few contexts.

After column reduction comes *rule reduction*. Rather that generate complex summaries of the reduced data, we merely report paired contrast sets between contexts. This generates very simple rules since (1) we focus the learning on contrasts between some current context and one other; (2) we only use the reduced set of attributes found via column reduction and (3) we only report the differences between pairs of centroids generated via row reduction.

We divide these reduction operators into three *inference operators* (dimensionality reduction, column reduction, and row reduction) and one *visualization operator* (rule reduction). The reason for this division is as follows. If rule reduction runs last, then it executes over a very small set of rows and columns. In that reduced space, "inference" can be replaced by a simple visual comparison between either (*now* and *feared)* or (*now* and *envied)*.

- *Now*: the user's current context.
- *Feared*: a close context with much worse effort values.
- *Envied*: a close context with much better effort value.

Examples of these four data reduction operators are given below. Before showing those examples, we offer some details on how we defined distance between rows (this inter-row measure is used extensively in IDEA). In Aha's et. al. [6]'s work on instance-based reasoning, they use a Euclidean measure to compute the distance between two rows. Within that measure is a function to compute the *difference* between two values from the same column. Their *difference* function, which is used in IDEA, is as follows. Note that this approach handles numeric, symbolic, and missing values:

```
function difference (X , Y)
        if both are missing then
                return "1" (max value)
        else if non-numeric values then
                if one is missing then return 1
                else return X == Y
                end if
        else if numeric then
                normalize each one via 1-min max-min
                if none are missing then return (X - Y )^2
                else if present <0.5 then return (1-present)^2
                else return (present)^2
                end if
        end if
```

3.1 Dimensionality reduction with FastMap

FastMap [7] is a linear-time algorithm time that maps objects in n-dimensions to points in some lower-dimensionality space. Formally, FastMap is a Nystrom algorithm [18] for finding an approximation to the components of PCA. We use it here since it is simple to code, runs in linear time, and has proven useful on other software engineering data [13].

For this work IDEA recursively map project data down to one dimension; then divides the data at the median point of that mapping; then recurses on each half. It works as follows:

1. First choose any row X at random then find the farthest row from this initial point and call it *East*.

2. Locate the farthest row from *East*, call it *West*. Now we can draw an imaginary line between rows *East* and *West*. Examples of these lines are shown in Figure 1.

3. The line from *East* to *West* approximates the dimensionality of greatest variance since these are the farthest points from each other in the group. All other rows have a distance *(a, b)* to *(East, West)*.

4. Let the distance *East* to *West* be *c*. All other rows can be projected into the *East, West* line at some distance from *East*, using the cosine rule: $(a^2 + c^2 - b^2) / (2c)$.

5. Once all points are plotted we find the median point and spit the rows into two groups either side of the median.

In Figure 2 we see dimensionality reduction of an N-dimensional data set by using the median value to split each cluster into two smaller clusters. In IDEA, this recursion terminates when too few examples fall into each split (for N projects, we stop at less than 10 examples or *sqrt(N)* whichever is satisfied first).

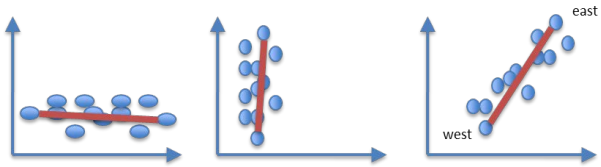

Figure 1. Inferring a dimension of greatest variability by joining two distant points.

3.2 Row reduction by clustering centroids

After clustering rows via recursive FastMap we condense clusters and replace numeric attributes by their mean value, including the class attribute, and nominal attributes are replaced by the median, mode of the numeric, symbolic attributes. A similar but slower approach is Principle Direction Divisive Partitioning (PDDP) [3].

Row reduction is shown in Figure 3. Here, the dendrogram of clusters found by recursive FastMap is shown as a set of grey boxes. Each leaf cluster contains one example synthesized from the leaf clusters of Figure 2.

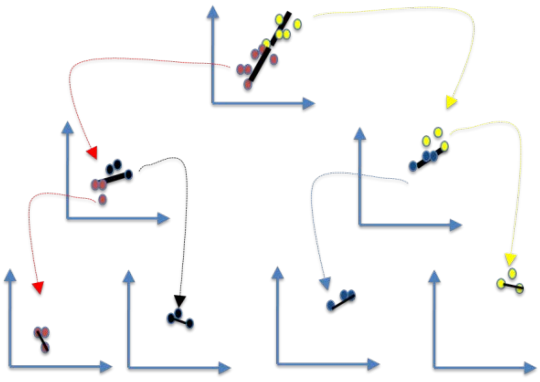

Figure 2 FastMap recursive clustering. The black line shows the inferred FastMap dimension. Arrows indicate half the data being moved to a sub-region. Note that this process works for data of arbitrary initial dimensionality. This example starts with N=12 examples and stops when sub-regions contain N < sqrt(12)=3.46 examples.

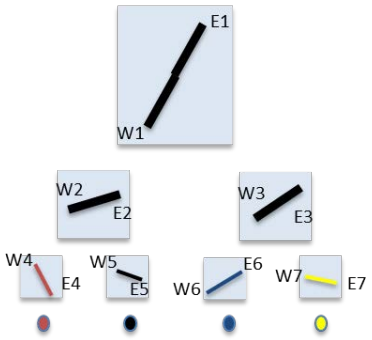

Figure 3 Centroids chosen by median

For example the Miyazaki94 data set in the PROMISE repository contains 48 projects. After dimensionality and row reduction, IDEA would represent that data as 5 contexts; i.e. that data set clusters into five regions. In a similar analysis, the 93 projects of PROMISE's nasa93 data set clusters into 13 contexts.

3.3 Column reduction via entropy

IDEA sorts columns by the probability they select for fewer contexts then remove the more ambiguous ones. We use the following heuristic: choose the attribute that produces the "purest" nodes. We use the popular impurity criterion: information gain. Information gain is the information content before a split compared to the information content after splitting on the attribute. We can calculate it using the measure of Shannon entropy [4], where Shannon entropy is given by equation 1.

$$\text{Entropy} = \sum(-p \log(p)) \quad (1)$$

This process lets us focus on columns that are better at selecting for a smaller number of contexts. For example, Figure 4 is some data from the PROMISE data set NASA93. For illustrative purposes, we assume that the data contains 9 projects described by analyst capability and programmer capability. We further assume that dimensionality and row reduction found two clusters: c1 and c2.

Figures 5 and 6 show the entropy calculation for analyst capability and programmer capability. In Figure 5 we see how a split on the values of analyst capability usually selects for a single cluster either 75% to 80% of the time. By contrast, in Figure 6, we see that a split on the values of programmer capability selects for a single cluster at most 66% of the time. Those figures show the calculation of the expected value for entropy: 0.239 and 0.273 for analyst and programmer capability (respectively). Based on this calculation, we find analyst capability most informative since it selects for a more specific (smaller) set of clusters.

Analyst Capability	Programmer Capability	Leaf Cluster
2	3	c1
3	3	c1
3	4	c1
3	5	c1
3	5	c1
2	3	c2
2	4	c2
2	4	c2
3	5	c2

Figure 4 Sorted columns

Analyst Capability	Leaf Cluster	Entropy = sum(r* sum(-p*log(p)))
2	c1	
2	c2	4/9 * (-1/4*log(1/4) - 3/4*log(3/4))
2	c2	= 0.107
2	c2	
3	c1	
3	c1	5/9* (-1/5*log(1/5) - 4/5 *log(4/5))
3	c1	= 0.132
3	c1	
3	c2	**entropy = 0.107+0.132 = 0.239**

Figure 5 Entropy calculation on analyst capability

Prog. Capability	Leaf Cluster	Entropy = sum(r* sum(-p*log(p)))
3	c1	3/9* (-1/3*log(1/3) -2/3*log(2/3))
3	c1	= 0.091
3	c2	
4	c1	3/9* (-1/3*log(1/3) - 2/3*log(2/3))
4	c2	= 0.091
4	c2	
5	c1	3/9* (-1/3*log(1/3) -2/3*log(2/3))
5	c1	= 0.091
5	c2	**entropy = 0.091*3 = 0.273**

Figure 6 Entropy calculation on programmer capability

Table 1 Miyazaki94 centroid results. The cluster ids (in column one) come from an internal numbering scheme that is not related to (say) row number in the original file.

Cluster	Effort	ESCRN	EFILE	FORM
[62]	38.29	4	3	3
[64]	31.5	4	4	4
[57]	30	1	1	1
[58]	13.9	2	2	1
[59]	45.85	3	5	2
[60]	28.5	1	1	2

IDEA, we prune attributes which have a value of more than one half (0.5) of the maximum entropy of any column.

3.4 Rule reduction

To implement the final rule reduction operator, we collect some contextual knowledge from the user. Using that knowledge, we can find:

Now: which context is closest to their current project.

Feared/ Envied: contexts close to now that the user most wants to *avoid/ achieve.*

Given this information we can generate very small rules that list recommendations of what to do or what to avoid. These recommendations are the contrast sets between the centroid values of the different contexts:

$$What2Do = Envied - Now$$

$$What2Avioid = Feared - Now$$

IDEA runs rule reduction last. Hence, these contrast sets come from the reduced data space found by the above reduction operators. Consequently, rule reduction is very simple to implement (since all the smart data selection has already been done already). The current version of IDEA just prints tables visualizing the *What2Do* and *What2Avioid* contrast sets.

What happens after that is up to managers. IDEA reports minimal rules that can drive a project from one context to another. What the manager does with that information depends on many factors that are specific to particular projects. For example:

- Some column values might be easier to change than others. Managers might heuristically elect to manipulate just these easier-to-change values.
- Sometimes a manager might ignore an improvement to a better context if (a) that improvement is only slight and (b) the changes required to reach that new context are very drastic.
- Sometimes, a manager may recognize a near-by Feared context, but realize that the project is perilously close to making some of the *What2Avoid* decisions. In this case, the manager might alert higher management of a train wreck in progress in order to discuss risk mitigation strategies.

Note that these actions require extensive business knowledge that is not currently collected in the PROMISE repository. Hence, for know, IDEA terminates when it can map out the space of options (leaving the final decisions regarding actions up to the manager).

4. Examples

4.1 Miyazaki94

The results for Miyazaki94 from IDEA were shown in Table 1. Note the effects of dimensionality reduction, row reduction, and column reduction:

- From the 9 columns, IDEA has reduced these to 3
- From the 48 rows, IDEA has reduces these to 6 contexts.

To illustrate how IDEA handles this information, assume that the *Home* context of some manager is cluster 59 of Table 1[1]. Note that these cluster ids, shown in column one of that figure, come from an internal numbering scheme that is not related to, e.g., row number in the original data set. This Home context is shown in Table 2. The contrast sets (column attribute differences) between this *Home* context and all others are shown in Table 3.

Table 2 Closest related centroid to project Miyazaki94

Cluster	Effort	ESCRN	EFILE	FORM
[59]	*45.9*	*3*	*5*	*2*

Table 3 Miyazaki94 Effort percentage changes

Cluster	Effort	ESCRN	EFILE	FORM
[62]	-16%	1	-2	1
[64]	-31%	1	-1	2
[57]	-35%	-2	-4	-1
[58]	-70%	-1	-3	-1
[60]	-38%	-2	-4	0

[1] Cluster ids, shown in column 1 of that figure, come from an internal numbering scheme unrelated to the original row number in original set.

From the differences in Table 3, we can compute a distance between *Home* and the other contexts. This is shown in Figure 7. From that figure, we note that this manager has nothing to *Fear* (since no other context contains projects with greater effort) and many things to *Envy*:

- The closest cluster is cluster 62 with a 16% reduction in effort.
- But with a few more changes, cluster 58 offers a dramatic reduction in effort (70%).

As discussed above, what happens now is up to the manager and how much control they have of their local environment. A rational manager would at least consider the contrast between 59 and 58 to consider if those changes are possible in the local context.

4.2 Nasa93

Table 4 shows the columns and rows that IDEA selects from the PROMISE Nasa93 data set.

To illustrate how IDEA handles this information, assume that:

- The *Home* context of some manager is cluster 121 of Table 4 (shown in Table 5).
- This manager most *Fears* projects of similar size but which take much longer to develop.

In order to explore the *Fears* of this manager, we isolate the contexts with similar lines of code to cluster 121: these are the four projects shown in Table 6. Note cluster 120: this project takes much greater effort than the Home context of 121.

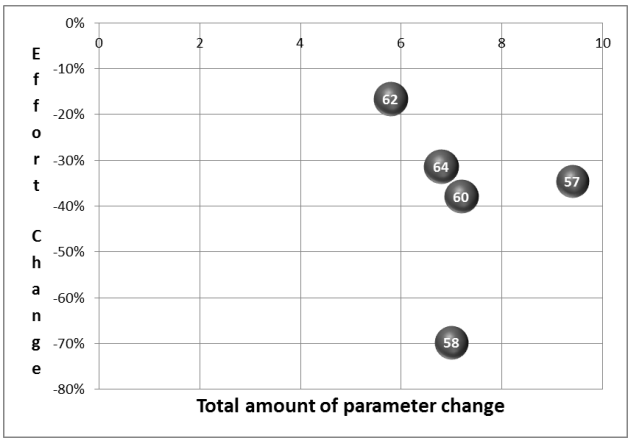

Figure 7 Possible effort changes for Miyazaki94

Table 4 Results for NASA93

Cluster	effort	apex	plex	pmat	rely	data	cplx	time	stor	kloc	acap	pcap
[113]	38	3	2	3	4	3	5	5	5	6.2	3	3
[112]	37	3	3	4	4	2	4	3	3	8.1	3	3
[110]	206	3	3	4	4	3	4	3	3	28.25	3	3
[120]	300	5	2	4	3	5	4	5	5	28.3	5	5
[115]	156	4	3	4	3	3	4	3	3	30.75	3	3
[121]	192	4	2	4	3	5	4	5	5	35.5	5	3
[118]	290	4	4	2	4	4	4	3	5	43.5	4	4
[116]	166	3	3	3	3	2	4	3	3	66.25	4	3
[125]	1645	4	4	4	5	4	5	6	6	70	4	4
[119]	300	4	4	2	3	3	4	3	5	77.5	4	4
[114]	304	4	3	3	4	3	3	3	3	84.5	4	4
[123]	342	5	3	3	3	2	4	3	3	125	4	5
[122]	144	4	3	3	3	2	4	3	3	131	4	4

Table 5 Centroid 121 Software Effort Parameters for NASA93

[121]	192	4	2	4	3	5	4	5	5	35.5	5	3

Table 6 NASA93 centroids with similar KLOC to [121]

Cluster	effort	apex	plex	pmat	rely	data	cplx	time	stor	kloc	acap	pcap
[110]	206	3	3	4	4	3	4	3	3	28.25	3	3
[120]	300	5	2	4	3	5	4	5	5	28.3	5	5
[115]	156	4	3	4	3	3	4	3	3	30.75	3	3
[121]	192	4	2	4	3	5	4	5	5	35.5	5	3
[118]	290	4	4	2	4	4	4	3	5	43.5	4	4

Table 7 Contrast set between *Home*=121 and *Feared*=120

Cluster	effort	apex	plex	pmat	rely	data	cplx	time	stor	kloc	acap	pcap
[120]	300	5	2	4	3	5	4	5	5	28.3	5	5
[121]	192	4	2	4	3	5	4	5	5	35.5	5	3
Contrast	+108	+1	0	0	0	0	0	0	0	-7.2	0	+2

Table 7 table shows the difference between cluster 121 and 120. Note that this table shows *What2Avoid* contrast set since it displays the column changes that would drive this manager's project into a Feared context where projects tale 300/192= 156% longer to build. Due to IDEA's dimensionality, column, and row reduction operators, the size of this contrast set is very small (only two columns, plus some KLOC differences):

- In *Feared*, developers have a little more application experience (apex).
- In *Feared*, developers have more programmer experience (pcap).

Our business-level interpretation of this result is as follows. In Cluster 120, the more capable programmers are using their increased application experience to implement a more complex solution. Perhaps they are building a domain-specific language for problems like Cluster120. Perhaps they are working on reusable design patterns to enhance productivity of future developers working on that kind of project. Whatever the reason, the business question must now be, is there a business case for the increased cleverness of the Cluster 120 implementation:

- If Cluster 120 is implementing core services that will be used in many future applications, then perhaps the added cost of the Cluster 120 projects will be regained in savings from faster future developments.
- But if the Cluster 120 developers are racing other vendors to get products to a highly competitive market, then perhaps the Cluster 120 developers need to be directed towards shorter incremental solutions where the base functionality is delivered quickly which, in turn, could fund their more advanced development work.

As mentioned above, IDEA cannot make that kind of business case since it is reasoning over the PROMISE data sets that lack that kind of business meta-knowledge. Nevertheless, what IDEA can usefully do is at least prune away the irrelevancies and offer a clear visualization (like Table 7) of the key business decisions.

Table 8: Data used in the Assessment study

Dataset	Cols	Rows	Description
cocomo81	17	63	NASA projects
cocomo81e	17	28	Cocomo81 embedded projects
cocomo81o	17	24	Cocomo81 organic projects
cocomo81s	17	11	Cocomo81 semi-detached projects
nasa93	17	93	NASA projects
nasa93center_1	17	12	Nasa93 projects from center 1
nasa93center_2	17	37	Nasa93 projects from center 2
nasa93center_5	17	40	Nasa93 projects from center 5
desharnais	12	81	Canadian software projects
desharnaisL1	11	46	Language1 desharnais projects
desharnaisL2	11	25	Language2 desharnais projects
desharnaisL3	11	10	Language3 desharnais projects
sdr	22	24	Turkish software projects
albrecht	7	24	Projects from IBM
finnish	8	38	inland software projects
kemerer	7	15	Large business applications
maxwell	27	62	Finland commercial-bank projects
miyazaki94	8	48	Japanese COBOL projects
telecom	3	18	Telecom maintenance projects
china	18	499	Chinese software company projects

5. ASSESSMENT

The examples of the last section demonstrate the benefits of an IDEA-style analysis (business users can be focused on a very small number of key issues that most effect their domain). But how accurate are the recommendations of IDEA? The data reductions reported above for Miyazaki94 and Nasa93 are quite drastic (most rows removed, over half the columns removed). If such a drastic reduction deleted important project data, the rules generated by IDEA (e.g. Table 7) would be spurious.

To test this, we explored the accuracy of the effort estimates made by a k=1 nearest neighbor algorithm that ran over IDEA's cluster centroids[2]. For the Miyazaki94 and Nasa93 data sets, those centroids were shown in Tables 3 and 4. In that study, "nearest" was defined using:

- just the columns selected by column reduction and
- distance calculation of Section 3.

IDEA's recommendations were compared to combinations of ten learners and nine pre-processors (so 10*9 = 90) methods in all. To pick those learners and pre-processors, we reviewed the effort estimation literature and selected methods with some support in that literature. For full details of those 90 methods, see [8]. In summary, the nine pre-processors were:

1. norm: normalize numerics 0..1, min..max
2. log: replace numerics of the non-class columns with their logarithms
3. PCA: replace non-class columns with principle components
4. SWReg: cull uninformative columns with stepwise regression
5. Width3bin: divide numerics into 3 bins with boundaries (max-min)/3
6. Wdith5bin: divide numerics into 5 bins with boundaries (max-min)/5
7. Freq3bins: split numerics into 3 equal size percentiles.
8. Freq5bins: split numerics into 5 equal size percentiles.
9. None: no pre-processor.

Also, the 10 learners were:

1. INN: simple one nearest neighbor
2. ABE0-1nn: analogy-based estimation using nearest neighbor.
3. ABE0-5nn: analogy-based estimation using the median of the five nearest neighbors.
4. CART(yes): regression trees, with sub-tree post-pruning.
5. CART(no): regression trees, no post-pruning.
6. NNet: two-layered neural net.
7. LReg: linear regression
8. PLSR: partial least squares regression.
9. PCR: principle components regression
10. SWReg: Stepwise regressions.

Our performance statistics was MRE; i.e. the magnitude of relative error:

$$MREi = |actual - predicted| / actual \quad (3)$$

These MRE numbers were collected in leave-one-experiments. 20 data sets were used from the PROMSE repository (see Table 8). The results from k=1 nearest neighbor were sorted into the other 90 and a Mann-Whitney test was applied to test if there was a significant difference between (1) predictions generated from IDEA's reduced data; (2) predictions generated from all the methods with better median performance scores; and (3) predictions generated from all methods with worse median performance scores.

Figure 8 shows all the MRE results for the 20 data sets; the positions of the predictions from IDEA's reduced data are shown as blue dots (one per data set). From left to right MRE scores rank from better to worst. In each line we see that some methods just do very poorly on some datasets. IDEA's score always tends to be in the range comparable to the better methods and always stays in the lower area of the graph.

Cocomo81 is the third best dataset IDEA performs well on. If we follow the purple line that represents the 91 method's MRE values we can see that, at a certain point, methods just start performing worse; this is when predictions get statistically worse, and it's about halfway through. Each line in the graph shows a similar

[2] The centroids, for the Miyazaki94 and Nasa93 data sets, were shown in Tables 3 and 4.

behavior. The blue dots representing IDEA's performance, we can see, are only statistically different a few times.

Table 9 comments on IDEA's results compared to everything to the right and left of each blue dot: Group1, Group3 are all the results with a median MRE less, more than IDEA. We perform statistical Mann-Whitney U-test with a 95% difference measure. The Win-Tie-Loss numbers in Table 9 come from the following MWU (Mann-Whitney U-test) comparison:

if MWU(E_i, E_j , 95) statistically same **then**
    $tie_i = tie_i + 1$;
    $tie_j = tie_j + 1$;
else
  if (E_i, E_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

Table 9 Win-Tie-Loss statistics for IDEA against 90 methods

Dataset	Group 1	Group 2 (IDEA)	Group 3
Albrecht	1	1	1
Cocomo81	1	2	3
China	1	2	3
Cocomo81e	1	2	2
Cocomo81o	1	1	1
Cocomo81s	1	1	1
DesharnaisL1	1	1	1
DesharnaisL2	1	2	2
DesharnaisL3	1	2	2
Desharnais	1	1	2
Finnish	1	2	2
Kermerer	1	1	1
Maxwell	1	2	2
Miyazaki94	1	2	3
NasaCenter1	1	1	1
NasaCenter2	1	1	2
NasaCenter5	1	2	2
Nasa 93	1	2	3
SDR	1	1	2
Telecom1	1	2	2

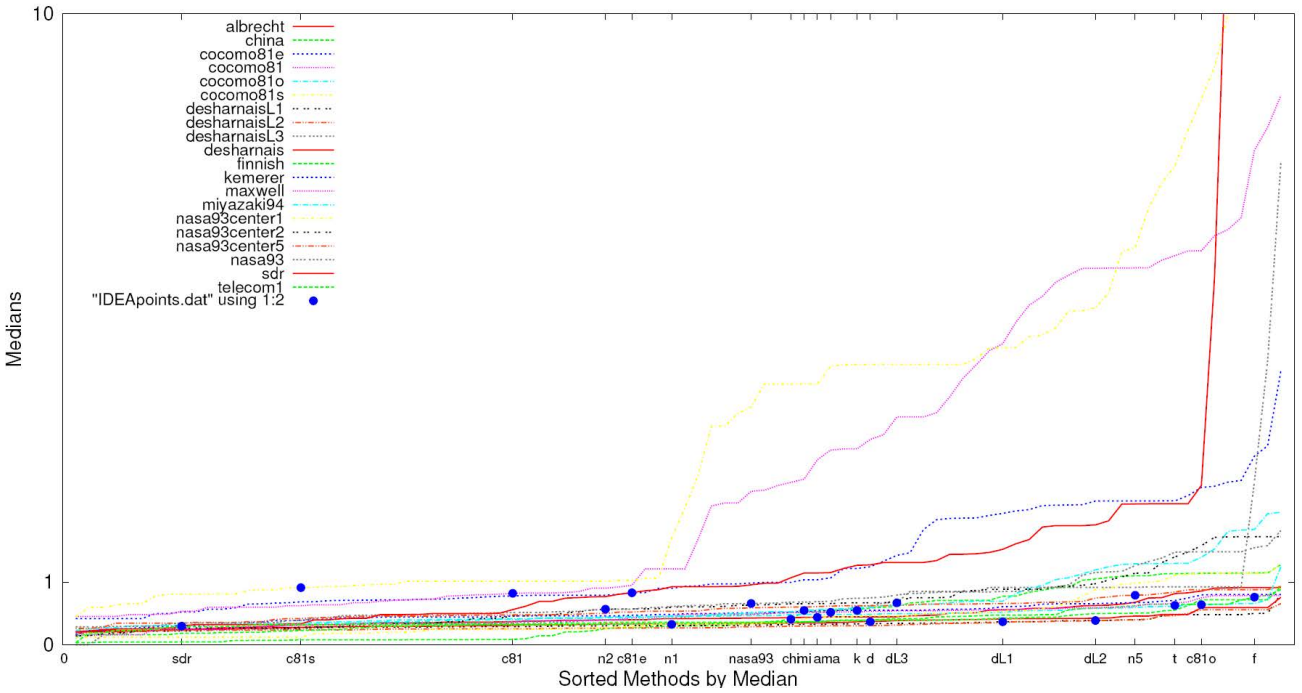

Figure 8 Sorted methods by Median MRE, IDEA score below for each abbreviated dataset

As shown in Table 9 several data sets generated results were the best and worst results were statistically indistinguishable (e.g. the Albrecht results on line one). This is not a new finding- we have seen and reported on this effect before [8].

At first glance, Table 9 seems somewhat negative regarding IDEA; observe how in Cocomo81, DesharnaisL2, DesharnaisL3, Finnish, Maxwell, and NasaCenter5, IDEA's results are indistinguishable from the worst result. However, a closer inspection of the raw data reveals a different conclusion.

In Figure 8, all the results have a left-hand-side valley (low errors) and a raised right-hand-side mountain (high errors). While in the valley all the methods exhibit a very similar performance, it is the mountain where we see the major performance differences. Note that none of IDEA's results (the blue spots) appear in those mountains. That is, even though in the Mann-Whitney tests, other methods do better than IDEA, qualitatively, we can recommend IDEA since in no case do its results fall into the left-hand-side region where methods most malfunction

6. CONCLUSION

We seek a "next generation" methods for empirical software engineering where humans can use the insights gained from data miners. To date, papers at PROMISE have been very weak at moving from mere estimation to offering project planning advice. We show here that a combination of data reduction operators (dimensionality, row, column, rule) can produce tiny human-readable recommendations relating to how to change a project to best, or worst, effect. We also show that those reduction operators do not remove the essential features of the data. If they did, they we would expect IDEA to perform very poorly. However, as argued above, IDEA's explanation system does not greatly compromise a data set's effectiveness.

We can conclude from this research that to learn we must cut and chip away at the data. We have to actually throw data away to be able to see the patterns more easily.

7. ACKNOWLEDGMENTS

This work was funded by NSF grant CCF:1017330 and the Qatar/West Virginia University research grant NPRP 09- 12-5-2-470.

8. REFERENCES

[1] A. Venkatachalam. 1993. Software cost estimation using artificial neural networks. *Proceedings of international joint conference on neural networks*, pp. 987–990.

[2] Boehm B., Abts C., and Chulani S. 2000. Software development cost estimation approaches \– A survey. *Ann. Softw. Eng.* 10, 1-4 (January 2000), 177-205.

[3] Boley D. 1998. Principal Direction Divisive Partitioning. *Data Min. Knowl. Discov*. 2, 4 (December 1998), 325-344. C. E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal, vol.* 27, pp. 379-423 and 623-656, July and October, 1948.

[4] Chang C. 1974. Finding Prototypes For Nearest Neighbor Classifiers. *IEEE Trans. Comput.* 23, 11 (November 1974), 1179-1184.

[5] D. Aha, D. Kibler, M. Albert. 1991. Instance-based learning algorithms. *Machine Learning 6* (1991) 37–66.

[6] Faloutsos C. and Lin K. 1994. *Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*. Technical Report. University of Maryland at College Park, College Park, MD, USA.

[7] Kocaguneli E., Menzies t., Keung J. 2011. On the Value of Ensemble Effort Estimation. *IEEE Transactions on Software Engineering, vol*. 99, no. PrePrints.

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

[9] Lorenzo Strigini L., 1996. Limiting the Dangers of Intuitive Decision Making. *IEEE Softw.* 13, 1 (January 1996), 101-103. DOI=10.1109/52.476293 http://dx.doi.org/10.1109/52.476293

[10] Lum K., Menzies T., & Baker D. 2008. 2cee, a twenty first century effort estimation methodology. *ISPA/SCEA*,p12–14.

[11] Mendes E., Watson I., Triggs C., Mosley N., and Counsell S. 2003. A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. *Empirical Softw. Engg.* 8, 2 (June 2003), 163-196. DOI=10.1023/A:1023062629183 http://dx.doi.org/10.1023/A:1023062629183

[12] Menzies, T.; Butcher, A.; Marcus, A.; Zimmermann, T.; Cok, D. 2011. Local vs. global models for effort estimation and defect prediction. *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on, vol*., no., pp.343-351, 6-10 Nov. 2011 DOI: 10.1109/ASE.2011.6100072 . URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6100072&isnumber=6100039.

[13] Menzies T., Mizuno O., Y. Takagi and T. Kikuno. 2009. Explanation vs Performance in Data Mining: A Case Study with Predicting Runaway Projects, *Journal of Software Engineering and Applications*, Vol. 2 No. 4, pp. 221-236.

[14] Menzies T. and Shepperd M. 2012. Special issue on repeatable results in software engineering prediction. *Journal of Empirical Software Engineering*, 17(1):1-17.

[15] Menzies T. and Shull F. 2010. The Quest for Convincing Evidence. *Making Software: What Really Works and We Believe It.* A. Oram, G. Wilson, eds, O'Reilly Books, 2010.

[16] M. Jorgensen and M. Shepperd. 2007. A Systematic Review of Software Develoment Cost Estimation Studies. *IEEE Trans. Softw. Eng*., vol. 33 no. 1, pp. 33-53.

[17] Platt J.C. 2005. FastMap, MetricMap, and Landmark MDS are all Nystrom Algorithms. *Microsoft Research Technical Memo*. URL: http://goo.gl/DoMzg.

[18] Shepperd M. and Schofield C. 1997. Estimating Software Project Effort Using Analogies. *IEEE Trans. Softw. Eng.* 23, 11 (November 1997), 736-743.

[19] Shepperd M., Schofield C., & Kitchenham B. 1996. Effort estimation using analogy. (ICSE '96). IEEE Computer Society, Washington, DC, USA, 170-178.

[20] Raymond P.L. Buse and Thomas Zimmermann. 2011. Information Needs for Software Development Analytics, no. MSR-TR-2011-8, 30