

Predictive Models in Software Engineering: State-of-the-Art, Needs, and Challenges

Lionel C. Briand, P. Eng.



Canada Research Chair in Software Quality Engineering
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada



PROMISE, September 24th, 2006

Problem definition

- Predict relevant quality and productivity variables based on information (explanatory variables) available at the time where the prediction is needed
 - Effort
 - Fault-proneness
 - Change impact
- A relevant prediction model supports important decision making
 - Predict effort based on requirements information: planning
 - Predict fault-proneness based on design information: drive V&V
 - Predict impact based on change data and design information: Impact analysis

What does it take?

- Good data
 - Relevant to your objectives
 - Complete
 - Reliable
 - Precise
- Adequate analysis / modeling technique
 - Type of data (scales)
 - Type of relationships
 - Context usage of the model

Modeling techniques

- There is a wealth of modeling techniques
 - Regression (least-squares, logistic, Poisson, etc.)
 - Machine learning (decision trees, coverage rules)
 - Neural-networks (back propagation ...)
 - And many others
- Plus a number of useful, related techniques
 - Outlier detection (e.g., Mahalanobis Distance)
 - Multicollinearity analysis (e.g., VIF)
 - Scale transformations (non-linearities)
 - Factor analysis (e.g., PCA)
 - MARS (multivariate adaptive regression splines)

Selection criteria

- Accuracy, but difficult to know a priori which will perform best
- Usability by non-experts (parameters, overfitting)
- Are the results interpretable?
- Theory: exploratory versus confirmatory analysis
- Number and type of variables
- Computation time for model building and application

Guidelines

- Factors that favor decision/regression tree and rule algorithms:
 - discrete explanatory variables,
 - numerous, interrelated explanatory variables
 - need for interpretable models,
 - lack of theory (exploratory), e.g., interactions
- Factors that favor regression
 - statistical skills that help use and interpret models
 - good understanding of expected relationships, interactions ... (though had a certain success with MARS)
 - manageable number of variables (selection techniques not that effective for large number of interrelated variables, PCA can help but there is usually a loss ...)

Guidelines (2)

- Factors that favor neural networks
 - no need for model interpretation
 - little theory and many explanatory variables
 - strong non-linearities and interactions
 - lengthy computations for model construction on large datasets are acceptable

Experience summary

- Decision trees (CART) and covering algorithms for rules (Ripper) yields highly intuitive models for non-statisticians (which represent most users)
- Often people need to understand why they get a prediction to trust it
- However, I have seen many cases where multivariate regression or neural networks yielded better accuracy (though not always large differences)
- Never was very lucky with nearest neighbors-type of techniques (lack efficient data fitting)
- Combining techniques has often been helpful – but again it is hard to determine a priori what will work best

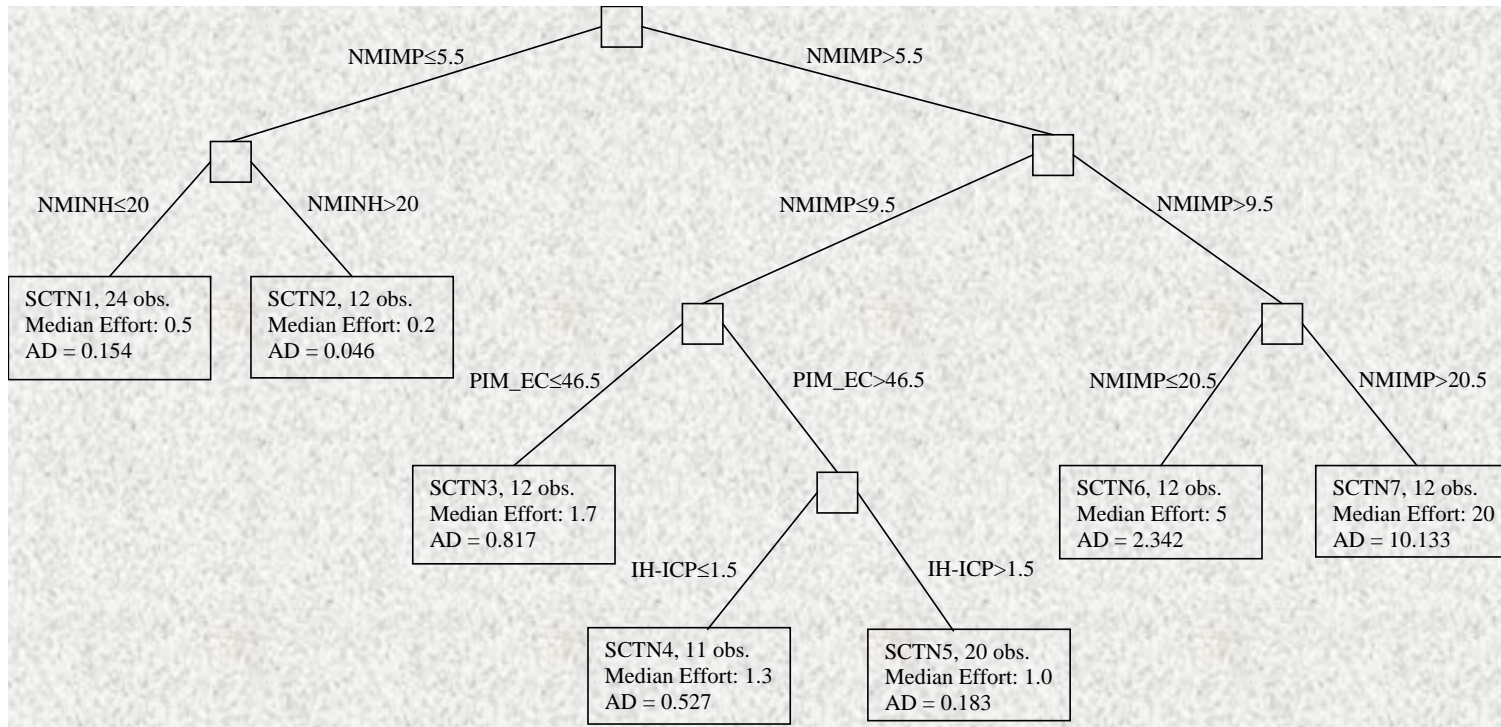
Experience summary

- ML algorithms (e.g., Ripper, C4.5) are based on combinations of numerous heuristics and often require parameters that are not easy to set
- No strong theoretical framework and explicit set of assumptions
- But it is easier to understand why you get a certain prediction, which is often useful in practice

Combination of techniques (example 1)

- Briand & Wuest (2001), cost prediction model based on size and design measures in OO systems (e.g., class coupling)
- Procedure to combine (Poisson) regression and regression trees (to uncover interactions)
 1. Run regression trees by specifying that terminal nodes should contain at least 10 observations .
 2. Add dummy variables (binary) to the data set by assigning observations to terminal nodes in the regression trees, i.e., assign 1 to the dummy variable for observations falling in its corresponding terminal node.
 3. Run stepwise Poisson regression using both design measures and the dummy variables as potential covariates.

Regression tree and Poisson regression

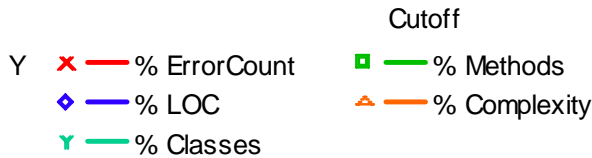
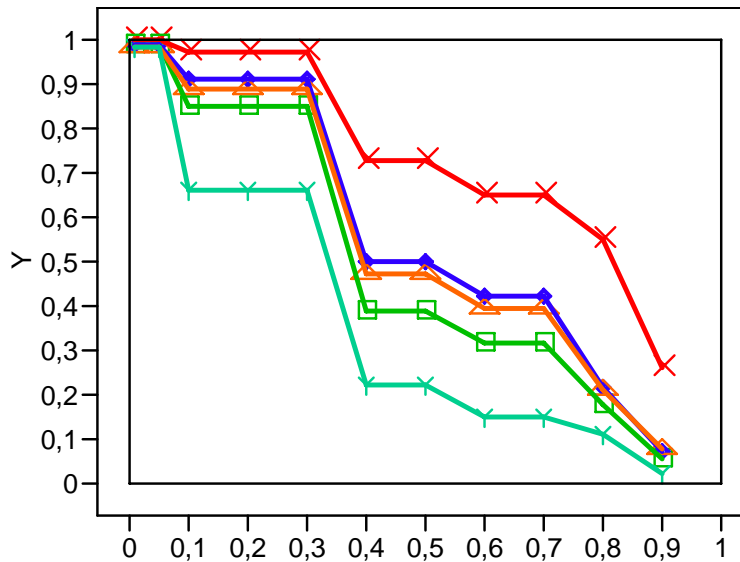


$$\hat{\mu} = e^{\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n}$$

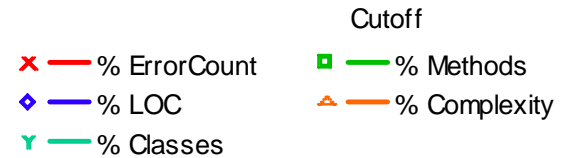
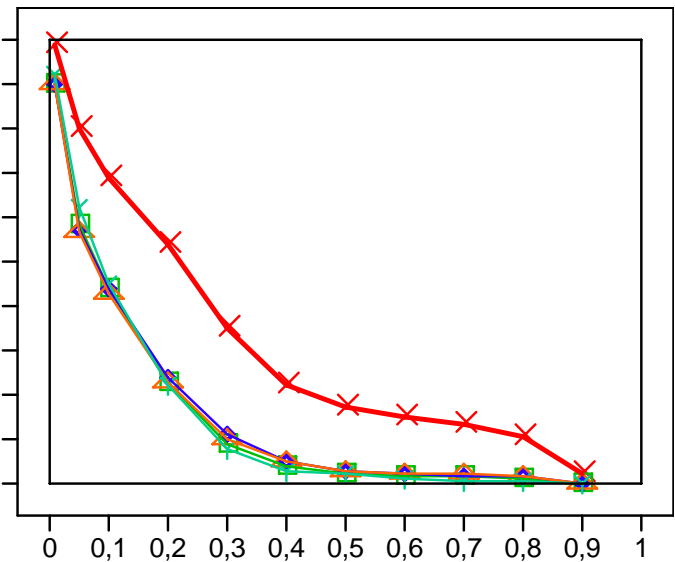
$$\Pr(y | x_1, \dots, x_n) = \frac{e^{-\hat{\mu}} \hat{\mu}^y}{y!}$$

Combination of techniques (example 2)

Decision tree



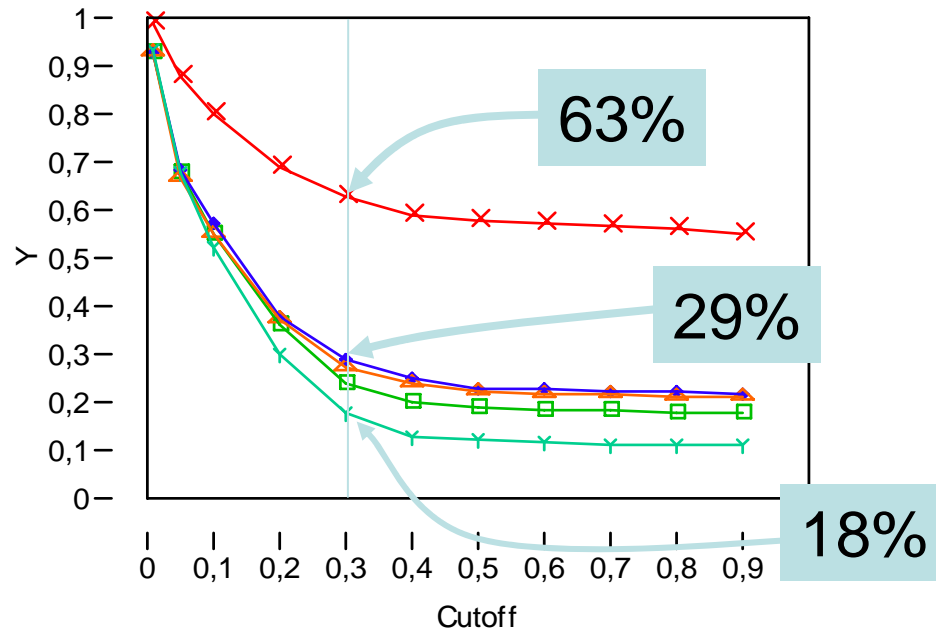
NN



Combination of techniques (example 2)

- Combining NN and decision tree

Fault = $(\text{Prob}_{\text{DT}} \geq 0.8)$ or $(\text{Prob}_{\text{NN}} \geq \text{cutoff})$



Main Challenges

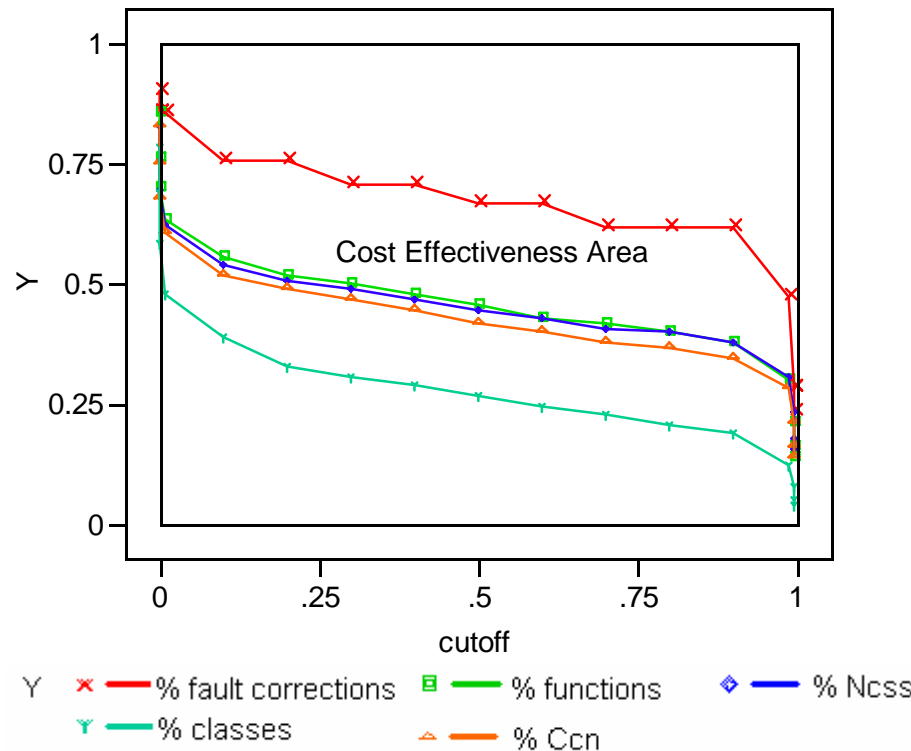
- Data modeling is NOT the main difficulty – many techniques available, though they need to be used with care and skill.
- How to efficiently collect relevant and precise data for the typical prediction problems in software engineering?
- We need to go beyond model accuracy and find ways to exploit such models in a cost-effective way. How do we measure the cost-effectiveness of models?
- Visualization of model predictions?

Accuracy versus cost-effectiveness

- Is a given model accuracy worth the cost of data collection and modeling?
 - Misclassification rates in fault-proneness models
- Is a model based on more expensive data worth the additional cost?
- Assumptions need to be made about the application of the model and its alternatives
- Cost-effectiveness is context dependent

Simple cost-effectiveness analysis

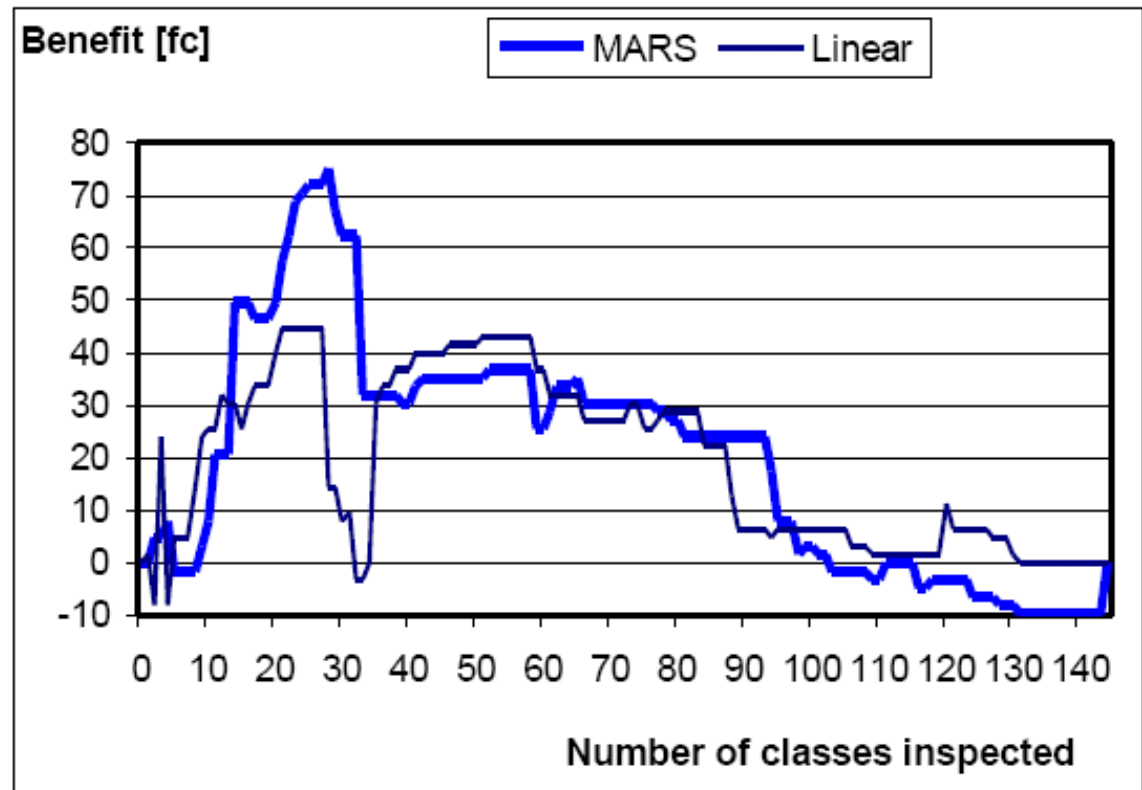
- Fault-proneness model for legacy Java system model (Arisholm and Briand, 2006)



Complex, context-dependent cost-effectiveness

- Assume that fault-proneness model drives inspections
- Many assumptions and parameters
- All classes predicted fault-prone are inspected.
- Inspections find a certain percentage of faults
- We assume an average cost of letting a fault through when not detected during inspection.
- The cost of inspecting a class is assumed to be proportional to the size of the class.
- For computing a benefit, we need a baseline of comparison. As a baseline, we assume a simple model that ranks the classes by their size.

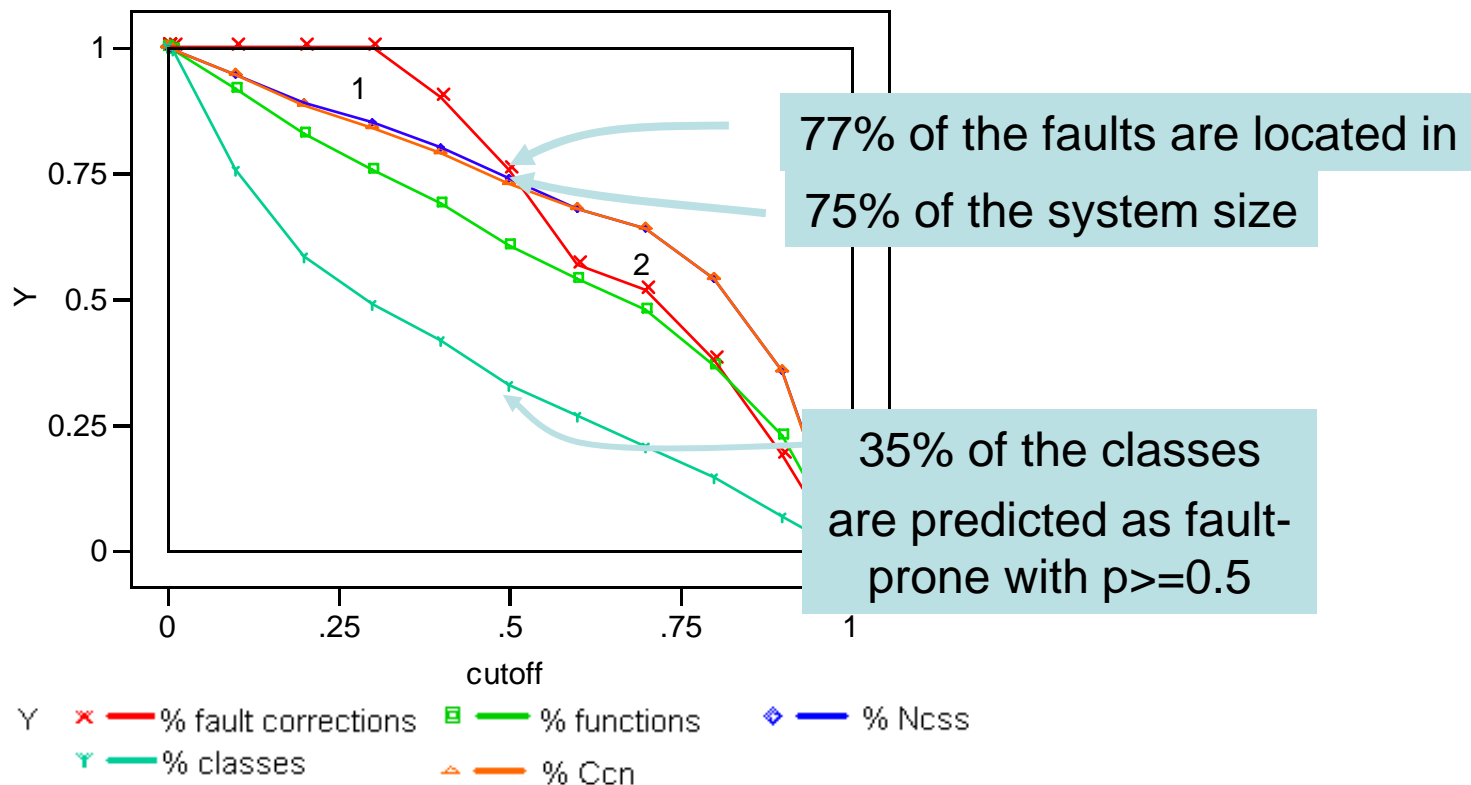
Complex, context-dependent cost-effectiveness



Briand, Melo, Wuest, 2002

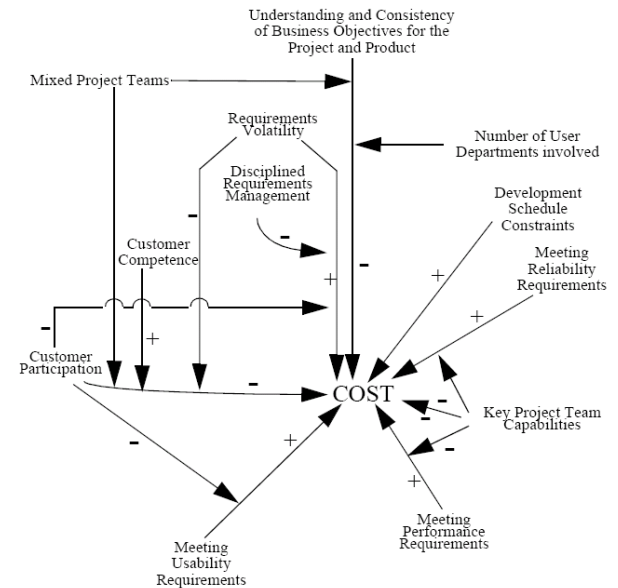
Data collection is a trade-off

- Excluding history data from legacy Java system model



Expert opinion-based prediction models

- COBRA (Briand et al., 1998)
- Cost estimation mixing project data and expert opinion
- Usually not enough projects for multivariate analysis
- Size / effort relationships based on data analysis
- Cost factors' impact modeled through expert elicitation to quantify a cost-overhead



Visualization: Tree Map – Package Level



Conclusions

- Applying models for decision making is not always obvious in practice
- Benefit is not obvious – the cost-effectiveness of models must be assessed in specific contexts
- Collecting the right data is hard – what really matters cannot always be easily measured, we cannot always afford to collect it, and we do not always know about it
- Data analysis and modeling is not really an issue in most cases – it requires care and skills though, and patience trying lots of alternatives
- However, there is a wide variation in terms of amount of data, type of data, context, etc. Not one method fits all.
- Users would rather visualize predictions in a way that facilitate decision making