

# Exploring Classification Algorithms for Early Mission Formulation Cost Estimation

Marc Sanchez Net  
 Massachusetts Institute of Technology  
 77 Massachusetts Ave 33-409  
 Cambridge, MA 02139  
 617-953-7977  
 msnet@mit.edu

Daniel Selva  
 Cornell University  
 Upson Hall, Campus Rd  
 Ithaca, NY 14853  
 617-682-6521  
 ds925@cornell.edu

Alessandro Golkar  
 Skolkovo Institute of Science and Technology  
 100 Novaya Ulitsa  
 Skolkovo, Moscow Region, 143025, Russian Federation  
 golkar@skolkovotech.ru

**Abstract**—Current cost estimation methods for early mission formulation typically use parametric regressions and analogies based on historical data sets. This approach, while widely spread in industry, is also subject to critique due to large parameter uncertainty and due to the reliance on small data sets on which regressions are based. Issues are accentuated in early mission formulation efforts, due to the immaturity of the mission concept and technical data available in preliminary design phases. In other words, conventional cost estimation methods sometimes have too high ambitions for the quantity of the information available in early mission formulation. Yet, cost estimation is of primary importance to determine feasibility of a mission within a space program. In this paper, we explore less ambitious approaches based on machine learning algorithms that are better suited to cost estimation for early mission formulation. In particular, we use classification algorithms to categorize missions into a predefined number of cost classes, e.g. Discovery or Flagship mission class. We compare different classification algorithms, study the performance and the utility of different levels of granularity in class definition, and test the proposed approaches on selected Earth Observation missions for which public information on cost is available. The methodology proposed in this paper provides an alternative approach for early cost estimation of new missions to cost and systems engineers.

from early mission phases (during mission formulation and program management) until late operational phases of the lifecycle (project control). Proposing teams go through costing exercises to come up with budgets for their missions and relate expected cost with estimates of delivered benefits, and attempt to demonstrate that proposed mission concepts are cost effective, thus worthwhile to be considered for funding. The Government Accountability Office (GAO) of the United States requires cost estimates from federal agencies as part of their planning and budgeting process. The importance of cost estimating is stressed in GAO’s Cost Estimating and Assessment Guide: “the ability to generate reliable cost estimates is a critical function, necessary to support the Office of Management and Budget’s (OMB) capital programming process.” [1] Similar processes can be found in Europe; as stated in the European ECSS Cost and schedule management standard, “the main objectives of cost and schedule management are to 1) plan accurately the phasing of procurements, expenses and resources for the project; 2) highlight any deviations and hence propose remedial actions, with the aim of completing the project within the given time and financial constraints.” [2]

## TABLE OF CONTENTS

1	INTRODUCTION .....	1
2	RELATED WORK.....	2
3	DATASET DESCRIPTION .....	2
4	METHODS.....	3
5	RESULTS .....	6
6	CONCLUSION .....	8
	APPENDIX .....	10
	ACKNOWLEDGMENTS .....	10
	REFERENCES .....	13
	BIOGRAPHY .....	13

## 1. INTRODUCTION

### Motivation

Cost estimating is an integral element of space mission design, as it plays fundamental roles in the development process

Analogy-based costing and parametric cost models are often employed for cost estimation during the first phases of the design. However, while the importance of cost estimating is globally recognized as an important need, the accuracy of cost estimates in early mission formulation is often subject to challenge and debate. A more realistic goal for this phase is to assess the probability that the mission cost and other programmatic parameters will stay within a certain range.

The goal is thus to estimate a “class” for the mission. These classes can be holistic agglomerations of information about cost, risk, or other programmatic parameters. For instance, these classes can match existing programs such as Discovery[3], New Frontiers[4], and so forth. Alternatively, they can be understood as a generalization of the concept of mission categories introduced in the NASA Cost Engineering Handbook [5] (Category 1:  $LCC > 1000\$M$ , Category 2:  $250\$M < LCC < 1000\$M$  or  $LCC < 250\$M$  and high risk, Category 3:  $LCC < 250\$M$  and low risk, where  $LCC$  indicates Lifecycle Cost).

### Research Goals

The focus of this work is on cost classifiers, leaving more complex classifiers (e.g. cost-risk) for future work. The paper analyzes the development of cost classification algorithms,

978-1-4799-1622-1/14/\$310 ©2014 IEEE.

<sup>1</sup> IEEEAC Paper #2304, Version 2, Updated 01/12/2014.

derived from machine learning theory, and discusses their relevant trade-offs. This is different from the formulation of cost estimation as a parametric regression problem, where a point cost estimate with confidence intervals is sought. The goal of the research is to explore tools that are able to provide cost estimates in a form that is better suited to the needs of program managers during early mission formulation.

This goal is achieved by comparing the performance of different classification methods as cost classifying tools, and characterizing their trade-offs in classifying accuracy versus granularity of their estimates, where granularity is defined as cost range associated to each mission class (hence providing an indication of value of the proposed tool to the mission designer).

This paper answers to the following research questions:

- Which classification tools (classifiers) are most suitable for cost estimating in early mission formulation?
- What trade-offs are found in the development of cost classifiers?
- What input variables are driving predictors to develop cost classifiers for space missions?

### Paper Structure

The remainder of this paper is structured as follows. Section 2 provides an overview of related work in the literature, discussing advantages and limitations of existing cost estimating methods. Section 3 presents the dataset on which classifiers have been developed and compared, derived from a set of 36 Earth Observation missions. Section 4 compares performance and explores classifying tradeoffs of four different methods, namely logistic regression, additive trees, k-nearest neighbors, and neural networks. Lastly, section 5 derives conclusions from the research, and outlines avenues of future work in this area.

## 2. RELATED WORK

This section describes previous work in cost estimating as applied to early space mission formulation, providing an overview of the state-of-the-art in the field. Mission lifecycle cost estimates in early formulation studies have been chiefly developed using analogy-based costing and parametric regression cost models. Bottom-up cost estimation approaches are not appropriate for early mission cost estimation since they systematically require data that is not available in that phase.

Analogy-based costing provides cost estimates relying on the experience of the cost engineer, whom is able to adjust cost figures associated to other missions to provide quick estimates when regression cost models cannot be developed [6]. Such approach relies on the experience of the estimator and its ability to comprehensively assess programmatic, performance, and other differences between programs under exam. As such, analogy-based costing is potentially subject to biases and errors from the estimator, and cannot therefore guarantee an objective basis of the estimate.

On the other hand, parametric models attempt to increase objectivity of their results by providing estimates based on statistical analysis of previous missions cost data. Such models perform regression analysis on previous mission cost datasets, in order to derive statistically significant Cost Estimating Relationships (CERs). The goal of CERs is to provide

cost estimates as a function of selected input parameters that are found to be representative cost drivers. Examples of regression cost models in the space industry abound, where such models have been developed to estimate lifecycle cost of space missions [7], and instruments [8], to name two relevant examples. In general, accuracy has found to be relatively low (around 30%) and variance relatively high, due to lack of detail available in mission requirements and specifications, and insufficient availability of previous cost data [9]. This fact is exacerbated by the unprecedented nature of many missions. Furthermore, parametric models are sometimes poorly understood and improperly used. Ranges of validity of regressions are sometimes ignored leading to inappropriate model extrapolations, and point cost estimates are provided without the corresponding confidence intervals.

Cost engineering practitioners mostly rely on grassroots estimates, analogy-based cost modeling, and parametric regression analysis, due to widespread adoptions of these methods in the community [10]. Nevertheless, researchers in academia have attempted to study and characterize new methods and tools for this purpose. The application of machine learning techniques to cost estimation was previously attempted in software cost estimation, with the application of neural networks for cost estimating purposes and their comparison with regression models [11]. Similar tools were applied to cost estimating in other industries besides aerospace, such as in building construction and software engineering, with comparisons done between neural networks, nearest neighbors, support vector machines, regression trees, and step-wise regression with optimized set reduction [12], [13].

## 3. DATASET DESCRIPTION

A dataset of past Earth Observation satellite missions derived from previous research [14] has been used to train and benchmark classifiers in this study. The dataset includes 7 different mission parameters (Table 1) and total lifecycle costs for 36 Earth Observation missions. Cost data was reported in million Euros, fiscal year 2012.

Note that the total instrument cost used in the dataset was derived from the instrument mass, power, and data rate by using Equations 1 and 2. These Cost Estimating Relationships are taken from the the NASA Instrument Cost Model (NICM):

$$C_{opt} = 979.9m^{0.328}PW^{0.357}R_b^{0.092} \quad (1)$$

$$C_{RF} = 19899m^{0.284}PW^{0.325}R_b^{0.09}TRL^{-1.296} \quad (2)$$

where  $C_{opt}$  is the cost of an optical instrument,  $C_{RF}$  is the cost of an RF instrument,  $m$  is the instrument mass in kg,  $PW$  is the instrument peak power in W, and  $R_b$  is the instrument average data rate in kbps.

The cost distribution characteristics of the available dataset are shown in Table 2. Positive skewness and kurtosis values highlight the asymmetric nature of the dataset cost distribution, featuring fewer high-cost (500+ MEUR FY12) missions than “low” cost missions for which information is available. This precludes the utilization of methods based on the normality of the data, such as typical computations of confidence intervals (e.g. t-test).

**Table 1:** Dataset Input Variables

<b>Technical Parameters</b>	
Total instrument cost estimated using NASA’s Instrument Cost Model (NICM)	
Total number of instruments	
Total number of instruments with TRL $\leq 5$	
Total number of instruments with TRL $\leq 6$	
Total instrument mass (kg)	
Total instrument power (W)	
Total instrument data rate (dB-bps)	

**Table 2:** Dataset Characteristics

<b>Cost distribution (MEUR FY12)</b>	
Minimum	11
25th Percentile	187
50th Percentile	288
75th Percentile	577
Maximum	1750
<b>Distribution Characteristics</b>	
Standard deviation	386
Skewness	1.59
Kurtosis	3.21

## 4. METHODS

This section presents an overview of the general methodology followed in order to compare multiple classification algorithms. In particular, it first describes the candidate classification algorithms along with the set of classes to be used while comparing them. Next, it explains how predictor variables are selected. Finally, it shows how the trade-off between classification granularity and performance is analyzed.

### *Candidate Classification Algorithms*

Four classification algorithms were identified as potential candidates for cost estimation: logistic regression, additive trees, nearest neighbors, and neural networks [15]. Other well known algorithms such as Support Vector Machines (SVMs) or Discriminant Analysis were deliberately left out of this study because, at least in their simplest formulation, were considered to be unsuited for the problem at hand. For instance, SVMs are typically used to distinguish between two classes, a major limitation for an application where the range of costs can span from tens of millions of dollars to more than one billion. Similarly, Discriminant Analysis expects multivariate normal distributions for input variables, an assumption that does not hold for the current dataset. Furthermore, the four algorithms selected span a large set of possibilities in terms of strategy (estimate a function vs proceed by analogy) and complexity (number of parameters).

*Logistic Regression*—Logistic regression lies within the set of linear classifiers that compute the posterior probability of belonging to a class based on the predictor inputs. Let  $G$  be a random variable that can take values from a discrete set of classes  $\Omega$  and let  $\underline{X}$  be a vector of random variables, one for each predictor. Then, the logistic regression will estimate the probability of pertaining to class  $g$  given a set of predictors  $\underline{x}$

as:

$$Pr(G = g | \underline{X} = \underline{x}) = \frac{\exp(\beta_0 + \underline{\beta}^T \underline{x})}{1 + \exp(\beta_0 + \underline{\beta}^T \underline{x})} \quad (3)$$

where  $\beta_0$  and  $\underline{\beta}$  are coefficients fitted to minimize training error. The decision boundaries for the different classes correspond to the values of  $\underline{x}$  such that  $Pr(G = g_1 | \underline{x}) = Pr(G = g_2 | \underline{x})$ . This can be reformulated using the *logit* function as:

$$\log \frac{Pr(G = g_1 | \underline{x})}{Pr(G = g_2 | \underline{x})} = \beta_0 + \underline{\beta}^T \underline{x} = 0 \quad (4)$$

thus yielding to a linear boundary (line, plane or hyperplane depending on the number of predictors).

*Additive Trees*—Linear models (either regression-based or classifiers) assume that the relationship between inputs and outputs is linear. This relationship can be generally expressed as

$$g[Y] = \beta_0 + \beta_1 X_1 + \dots + \beta_N X_N = \beta_0 + \underline{\beta}^T \underline{X} \quad (5)$$

If  $g[Y]$  is set to the identity ( $g[Y] = Y$ ) then the resulting model is a multivariate linear regression, while  $g[Y]$  equal to the *logit* function yields the logistic regression.

Generalized additive models extend linear models by allowing linear combinations of transformations on the predictor variables.

$$g[Y] = \alpha_0 + f_1(X_1) + \dots + f_N(X_N) \quad (6)$$

In this case, the decision boundaries for classifying are no longer linear but are shaped according to the set of functions  $f_i$  utilized.

Additive trees are a subset of additive models where the decision boundaries recursively partition the input space (i.e., the set of allowable values for each of the predictor variables  $X_i$ ) into multiple rectangles and then assign a particular value or class for each of them. For classification purposes, this partitioning process is automatically done through a measure of node impurity, i.e. how many samples are misclassified in any given rectangle.

The main advantage of additive trees is as two-fold: First, they can be graphically represented to convey the classification criteria in an easy and understandable way. Second, they identify the predictor variables that have main effects in the classification process and clearly specify their threshold values. On the other hand, additive trees tend to be very unstable and create completely different tree structures when the input data is modified. This problem is particularly important in the problem at hand since the number of data points is limited. As a result, two random partitions between training and test sets might result in different trees, different splitting variables and different thresholds for them.

*Neural Networks*—For the purpose of this study, only Neural Networks (NN) with a single hidden layer network fitted using a back-propagation algorithm (also known as single layer perceptron) were considered. In this sense, a Neural Network can be viewed as a two-stage classification or regression model that, when combined, yields to non-linear decision

boundaries. For classification purposes, a NN consists of: an input layer with  $M$  nodes,  $M$  being the number of predictor variables; a hidden layer with  $N$  nodes where  $N$  is a free parameter that has to be determined through cross-validation; and an output layer with  $P$  nodes,  $P$  being the number of classes.

The nodes of the input layer are directly connected to those of the hidden layer and these, in turn, are connected to the nodes of the output layer. Each connection has a weight associated with it, i.e. a real number that defines the importance of that connection on the node. Inside any neuron, the weighted inputs are added and run through an activation function that defines the output value of the network. Multiple activation functions are possible, although classifiers are commonly based on the *sigmoid* function [15]. This is the setup that will be used in this study.

**Nearest Neighbors**—Nearest neighbor algorithms take a completely different approach to classification. They do not incorporate any mathematical model to delineate the decision boundaries, nor they are designed to provide insight into what predictors and relationships dictate the outcome. Nevertheless, they are widely used and have proven to be useful classifiers.

**K-Nearest Neighbor (K-NN)** classifiers estimate input sample  $x_0$  class based on the classes of its  $K$  closest training points  $x_i, i = 1, \dots, K$  for which the true class is known. In particular, the algorithm looks at the class for each of these  $K$  points, counts their occurrences, and assigns to  $x_0$  the class that is most frequent. As a result, the shape of the decision boundary is arbitrary and will generally not correspond to any simple mathematical model.

Two pieces of information are required as inputs to this process: the number of neighbors to consider ( $K$ ), and the distance metric. The value of  $K$  has to be optimized to obtain good accuracy while avoiding overfitting the training data. This is usually done through trial and error over a cross-validation subset of samples. Euclidean distance between normalized points in the dataset was used as distance metric.

### Benchmark Methodology

Figure 1 shows the generic methodology followed in order to compare the performance of the four candidate classification algorithms. The process begins by setting the extreme values for the mission cost (e.g., \$10-2,000M), the allowed range for the classification algorithm parameters (e.g.,  $L = 1 - 5$  for *Additive Trees*), and the allowed range for the number of classes (e.g., 4-20). For each number of classes (e.g.,  $N = 5$ ), the class boundaries are computed (e.g., class 1 between \$10-50M, class 2 between \$50-100M, and so forth).

Once these have been computed, the evaluation of a classification algorithm can start. A particular value for the classification algorithm parameter (e.g.  $L = 5$ ) and the number of classes (e.g.  $N=10$ ) is selected. The algorithm is then trained  $N_{it} = 100$  times using all possible subsets of predictor variables. At each iteration, random partitions for the training and test sets (80% vs. 20% of the total samples) are used to train the algorithm. Performance is assessed through the mean and standard deviation of the percentage of misclassifications. The obtained results indicate the performance of the classification algorithm (both bias and variance) for each subset of predictors. The best subset of predictor variables is selected using a combination of mean

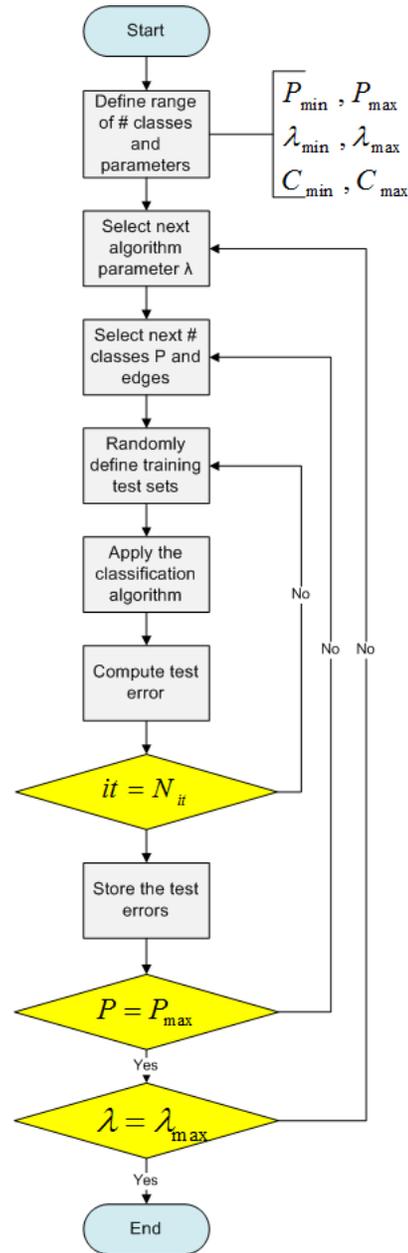


Figure 1: Benchmark Methodology

and standard deviation of the test error.

This evaluation process is repeated for all the candidate values of the algorithm parameter and all the possible numbers of classes. The obtained results can be used in two ways: First, the optimal value of the algorithm parameter can be chosen by comparing the performance of its best subset of predictor variables for different numbers of classes. Second, the performance of the different algorithms tuned with their best parameters, and their best predictor subsets, can be directly compared for the different number of classes.

**Definition of Classes**—Applying classification algorithms requires the definition of cost "classes". The number and range of these classes is important because it directly affects both the performance of the algorithm and the utility to program managers. Indeed, if the number of classes is very large, the

utility to the program manager will arguably increase at the expense of performance of the classification algorithm, and vice-versa.

There are several strategies to define the group of cost classes in the context of early space mission cost estimation. The first one consists in defining this group as close as possible to the classes loosely defined by major NASA programs, namely flagship missions, Explorer missions (MIDEX, SMEX, EX), Opportunity missions, Earth Ventures, and so forth. This strategy provides a useful input to program managers who can assess the likelihood that a certain project will meet the cost requirements to be in a certain program (e.g. SMEX). On the other hand, this leads to a classification scheme with ranges that differ very much in size (e.g., the class of SMEX would have a much smaller range than the class for flagship missions). While this is not a drawback *per se*, the fact that these classes are defined ad-hoc makes it hard to parameterize and is therefore not appropriate for our automated benchmark study.

A second strategy consists in simply taking evenly spaced classes in the cost axis. Given a desired number of classes  $P$ , this defines the  $\Delta_{cost}$  between classes:

$$\Delta_{cost} = \frac{C_{max} - C_{min}}{P} \quad (7)$$

where  $C_{max}$  and  $C_{min}$  are some maximum and minimum mission cost, that can for example be taken from the actual dataset of missions. Alternatively,  $C_{min}$  can be set to 0 and  $C_{max}$  can be set to some round number larger than the maximum database cost, in order to yield “nice” cost categories. The problem with this second strategy is that the linear distribution of classes in cost may not be very useful, because it effectively provides very low resolution for small missions, and too high resolution for large missions (see Table 3).

A variant of this second strategy is to take evenly spaced classes in a logarithmic cost space. In this case,  $\Delta_{cost}$  is no longer constant, but increases with cost. What stays constant is a cost “multiplier”  $\pi_{cost}$  that is the ratio between consecutive edges of the cost classification scheme. Given a desired number of classes  $P$ , this defines the  $\pi_{cost}$  between classes

$$\pi_{cost} = 10^{\frac{\log C_{max} - \log C_{min}}{P}} \quad (8)$$

$$c_{i+1} = \pi_{cost} c_i \quad (9)$$

where the boundaries of the  $i$ th class are  $[c_i, c_{i+1})$ . Examples of these three strategies are provided in Table 5 for  $C_{min} = 10$  and  $C_{max} = 2000$ . Note that  $C_{min}$  is set to 10 instead of 0 to avoid the region of the log function that is too steep.

The latter strategy was chosen for this study, as it is closer to Strategy 1 in terms of utility for program managers, while being easy to parameterize.

*Selection of Algorithm Parameters*—Three of the four candidate classification algorithms have a parameter that needs to be estimated: the minimum number of observations per leaf  $L$  in additive trees, the number of neighbors  $K$  in K-NN and the number of neurons in the hidden layer  $N$  for

NN. The classical approach for obtaining the value of these free parameters is through cross-validation (CV). The original data set is partitioned into three subsets, one for training the algorithm, another one for estimating the CV error and a third one to compute the test error. Then, a plot showing the evolution of the CV error as a function of the free parameter is used to assess its optimal value.

This study has followed a slightly different approach to compute these optimal values. The main rationale for this change is the highly limited amount of samples in the original data set. When that is the case, dividing the samples into three non-overlapping subsets questions their validity since all of them have so few samples that they can become unrepresentative. This problem is especially important for classification purposes since the number of samples determines the minimal delta in misclassification percentage. For instance, if the test set only has five samples, then the granularity of misclassification error is 20%.

In order to avoid these problems, the approach followed in the benchmark methodology only divides the original data set into two partitions, one for training and another one for validation. A set of typical values for the algorithmic parameters is identified through a combination of experiments and literature review, thus restricting the number of options that need to be evaluated and compared. Finally, once the value of the parameter is fixed, the performance of the algorithm is evaluated taking into account the optimal selection of the predictor variables.

*Selection of Predictor Variables*—As explained in Section 3, the dataset contains seven parameters that can potentially be used as predictor variables. This yields  $2^7 - 1 = 127$  subsets of parameters that can constitute the predictor variable set.

The optimal set of predictor variables is found for each algorithm (e.g. NN) and parameter value (e.g.,  $N = 16$ ) by brute force, by comparing the performance of the 127 sets and picking the best one among them. The performance of a predictor set can be summarized by two metrics: the mean and the variance of the test error. The former is a measure of the bias, while the latter is a proxy for the stability of the predictor set.

Since we have two objectives (bias and stability), a single optimal predictor set cannot be objectively chosen unless preferences between the two objectives are provided. It is however possible to discard predictor sets that are dominated by others, i.e., that have higher mean error *and* higher variance. This yields a Pareto frontier containing the non-dominated predictor sets in the test error mean-variance space. Such Pareto frontiers will be illustrated and discussed in the results section.

Choosing a particular set within the Pareto front requires the use of subjective information. One way of doing this is combining mean  $\bar{e}$  and variance  $\sigma_e$  into a single objective by using a weight  $\alpha$ , as in:

$$\min_{\underline{X}} Z(\underline{X}, \lambda^*) = \bar{e}(\underline{X}, \lambda^*) + \alpha \sigma_e(\underline{X}, \lambda^*) \quad (10)$$

where  $\lambda^*$  represents the optimal value for the algorithm parameter (e.g.,  $L^* = 1$ ). For  $\alpha = 0$ , this is effectively minimizing bias; for  $\alpha \gg 1$ , this is effectively minimizing variance; and for any intermediate  $0 < \alpha < \alpha_{max} \approx 3$ , this

**Table 3: Comparison of class definition strategies**

Strategy	#classes	Ranges in \$M
1. Based on NASA programs	5	[0-50, 50-200, 200-500, 500-1000, 1000-2000]
2. Evenly spaced linear cost	5	[0-400, 400-800, 800-1200, 1200-1600, 1600-2000]
3. Evenly spaced log cost	5	[0-30, 30-80, 80-240, 240-690, 690-2000]

is effectively minimizing a certain conservative percentile of the test error. For example,  $\alpha = 1$  effectively minimizes the 68% percentile.

*Trade-off between Classification Granularity and Performance*—The trade-off between classification granularity (as defined by the number of classes  $P$ ), and the performance of the classification algorithm has already been introduced in Section 4. Higher  $P$  is more useful to program managers, but inevitably yields poorer performance.

In order to study this trade-off in a systematic manner, it is useful to plot these two metrics for each algorithm (granularity, represented by  $\pi_{cost}$ , and performance, represented by  $Z(\underline{X}^*, \lambda^*)$  for the best predictor set). This allows the objective elimination of combinations of parameters that are dominated in the performance-granularity space. Again, this yields a Pareto frontier of non-dominated solutions. Subjective preferences need to be applied in order to further select a model within this Pareto frontier.

The final model can be chosen from the granularity-performance Pareto frontier by taking into account several considerations:

- An error larger than 30-40% is probably not acceptable
- Fewer than 4 classes is probably not acceptable

## 5. RESULTS

This section presents the results of applying the tools and methods presented in Section 4 to the dataset described in Section 3. Results are shown for each algorithm separately. First, results concerning the selection of algorithm parameters are presented, followed by the selection of predictor variables and the trade-off between granularity and performance.

### *Selection of Algorithm Parameters*

Figure 2 presents a summary of the performance for each classification algorithm for different values of the parameters. In order to facilitate the plot readability, only a subset of the allowed parameters values are shown, always including the optimal value and ensuring that the trend in the evolution of algorithmic performance versus parameter variation is adequately captured. Note also that *logistic regression* is not included in this discussion since no parameters need to be selected.

Selecting the optimal value for  $L$ ,  $N$  and  $K$  based on the plots from Figure 2 is not a straightforward decision. In particular, two main questions arise: what is the right choice for the value of the parameter if the optimal value varies with the number of classes? And second, what delta in performance between two points is significant from a statistical standpoint? The first question was answered by determining what is the most frequent value that obtains the lower mean test errors. Concerning the second question, it

**Table 4: Optimal Algorithm Parameters**

Algorithm	Parameter $\lambda$	Optimal Value $\lambda^*$
Additive Trees	$L$	1
Neural Network	$N$	16
Nearest Neighbors	$K$	1

was considered that two points with difference in Mean Test Error  $\Delta MTE \leq \pm 5\%$  have the same level of performance.

The results of applying these criteria to Figure 2 are shown in Table 4. They indicate that both *additive trees* and *neural networks* achieve their best levels of performance when highly complex models are used. For the first algorithm, this translates into single sample leaves ( $L^* = 1$ ) that create complex, deeply branched tree structures and hinder the comprehensibility of the input space partitions and branching thresholds.

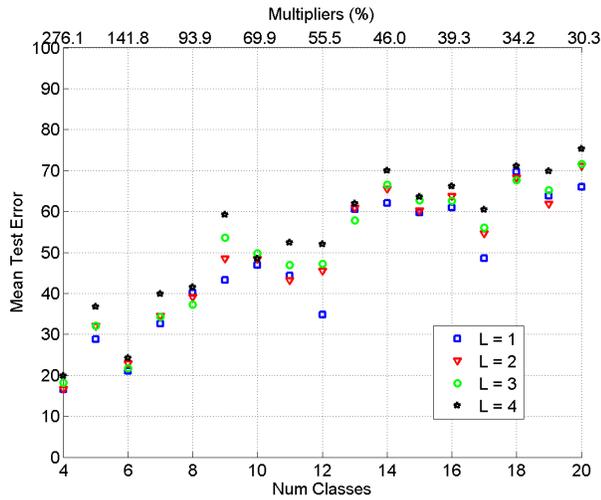
Similarly, *neural networks* improve performance as the number of neurons in the hidden layer is increased, thus yielding a more complex network. Nevertheless, it has also been noted that once the number of hidden layers exceeds  $N^* = 16$  then the  $\Delta MTE$  is generally less than 5%, thus not yielding significant performance improvements. Therefore, this value has been set as the optimal compromise between model performance and model simplicity.

Finally, one could argue that the *nearest neighbor* algorithm exhibits the opposite behavior of performance versus complexity, that is, the simpler the classification pattern the better. Indeed, a K-NN algorithm that only uses the closest sample to classify ( $K = 1$ ) almost always obtains the best performance. Note however that this simplicity in the model description yields the most complex decision boundary.

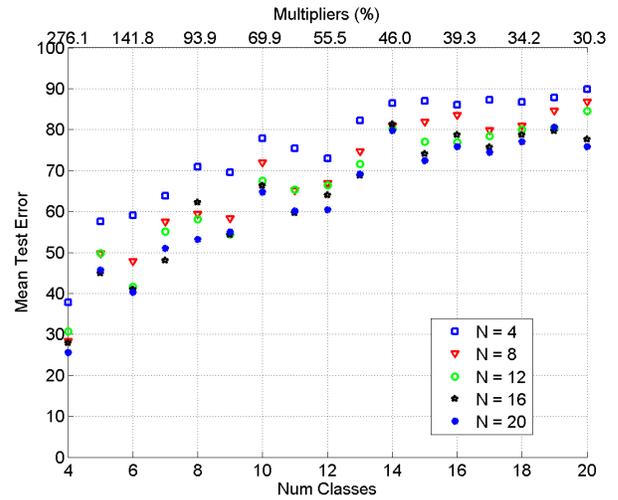
### *Selection of Predictor Variables*

Each candidate algorithm was applied 100 times to the dataset for each combination of predictor variables, and the test error for each trial was computed. Mean and standard deviations were gathered. Figure 3 shows, for each algorithm and for a range of number of classes, the mean vs the standard deviation of the test error for different combinations of predictor variables. Note that only non-dominated combinations of predictor variables are shown; combinations with higher mean and higher standard deviation than the ones shown on Figure 3 were eliminated.

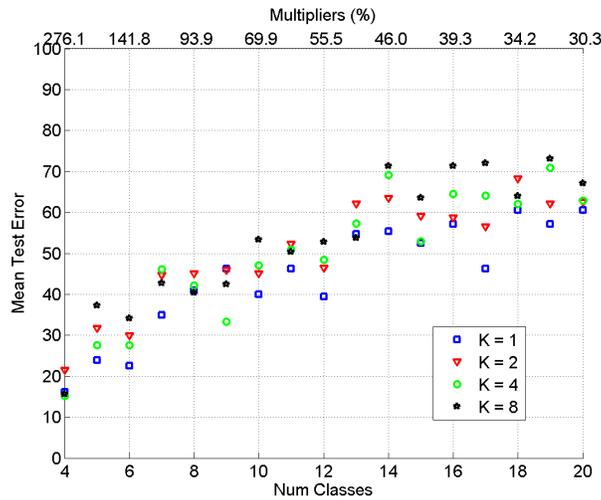
Given the relative small range in standard deviation compared to the range of mean error apparent on Figure 3, combinations of predictor variables minimizing mean test error were chosen as optimal. In other words, the methodology presented in Section 4 was applied with  $\alpha = 0$ . These optimal combinations are highlighted in red.



(a) Additive Tree - Minimum number of samples per leaf  $L$



(b) Neural Network - Number of hidden layer neurons  $N$



(c) K-Nearest Neighbors - Number of neighbors  $K$

**Figure 2:** Mean Test Error vs. Num of Classes for multiple parameter values

The details of these optimal combinations of predictor variables are shown on Table 6 in the Appendix. Figure 4 contains a summary of the frequency of occurrence of each predictor variable for the different algorithms. The figure shows that the total NICM cost is clearly the best predictor variable, as it appears most often in the best predictor sets for all algorithms. Figure 4 also shows an important difference between logistic regression and neural networks ( $N^* = 16$ ) on one side, and additive trees ( $L^* = 1$ ) and nearest neighbors ( $K^* = 1$ ) on the other side. Logistic regression and neural networks rely almost exclusively on the total instrument cost (as predicted by the NASA Instrument Cost Model) in their best classification models, whereas additive trees and nearest neighbors use a more diverse set of predictor variables. Note that this effects disappears almost completely for larger values of  $K$  in nearest neighbors. It also decreases, albeit to a lesser extent, for larger values of  $L$  in additive trees.

#### Trade-off between Classification Granularity and Performance

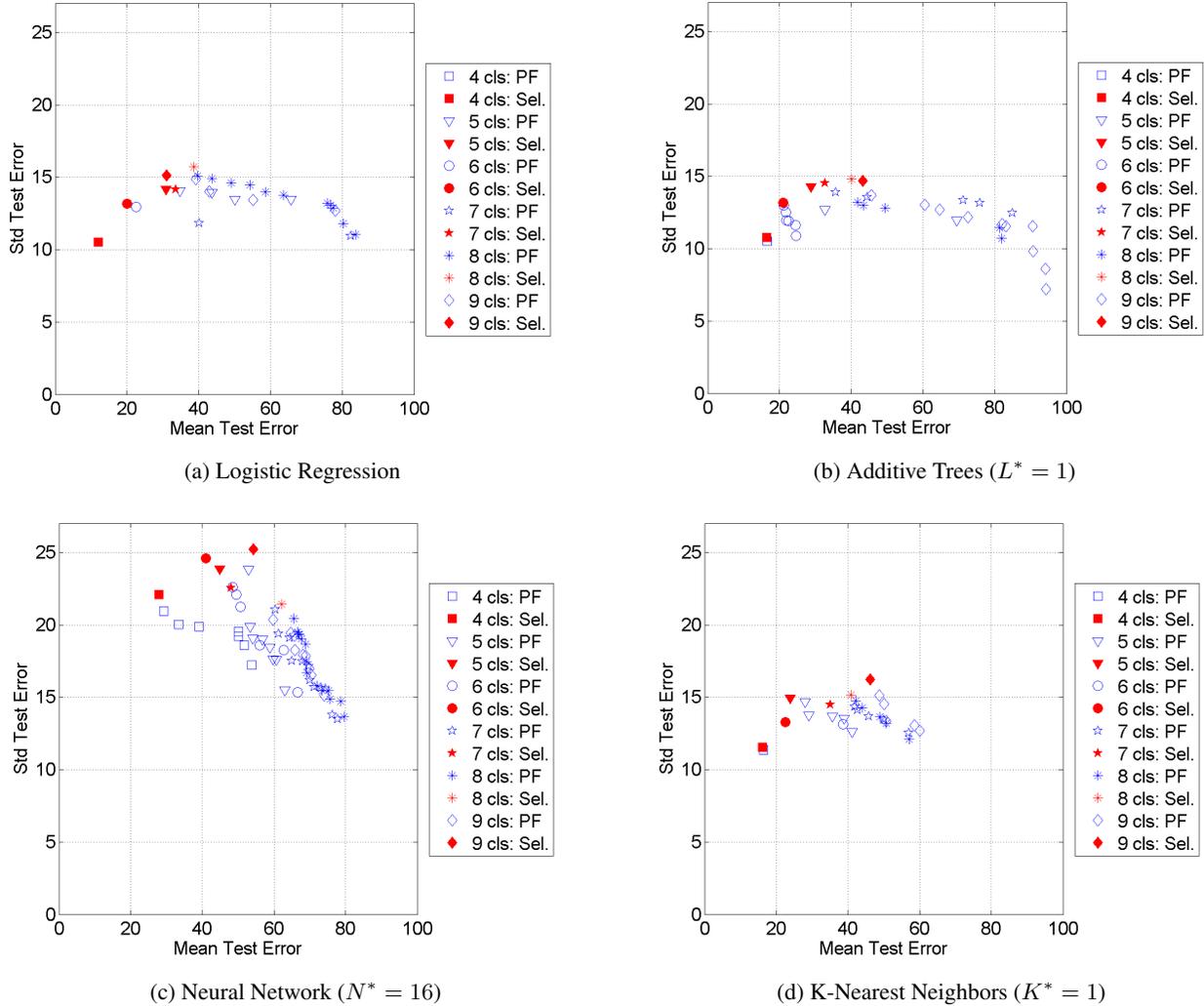
This section discusses the classification performance of each candidate algorithm with respect to the classification granularity. Figure 5 presents the mean test error as a function of

the number of classes. For each classification algorithm, the set of points displayed represents the obtained performance with the best combinations of predictor variables. Additionally, the expected performance of a random classifier has been introduced as a generic benchmark reference.

Results indicate that when few classes are considered ( $P < 10$ ), the percentage of misclassifications can be improved by as much as 40% compared to what a random classifier would achieve. Nevertheless, this margin of improvement is progressively reduced as the number of classes increases, from 40% for  $P = 4$  to 25-35% for  $15 \leq P \leq 20$  (15% for NN).

On the other hand, Figure 5 enables straightforward comparison of the classification algorithms. The first conclusion that can be drawn is that *Neural Networks* consistently achieve the worst performance, whereas *logistic regression*, *additive trees* and *nearest neighbors* do not show a significant difference in performance.

That said, when selecting an optimal classification algorithm, not only performance has to be taken into account. In



**Figure 3:** Test Error Variance vs Test Error Mean (cls: classes; Sel: Selected; PF: Pareto Front)

particular, three other factors might be relevant: (1) the level of complexity for the classification model, (2) how easy it is to understand and communicate the classification process and (3) whether the model provides any supplementary information that might be relevant to the decision maker. With these considerations in mind, the following conclusions can be drawn:

- *Neural Networks* have the lowest classification performance, with mean test errors 10% higher than the three other algorithms.
- Although *additive trees* have good classification performance, they require a high level of complexity that hinders the understandability of the tree and provide little insight into why the classification threshold values have been selected.
- *Nearest neighbors* obtains relatively good results, similar to those of logistic regression and additive trees. Furthermore, it is particularly advantageous for its simplicity and the easiness with which one can trace the rationale behind a given classification.
- *Logistic regression* obtains good results but does not offer the traceability inherent to nearest neighbors. The main advantage of this algorithm is the form of its output, i.e. the probabilities that a particular mission falls within each class.

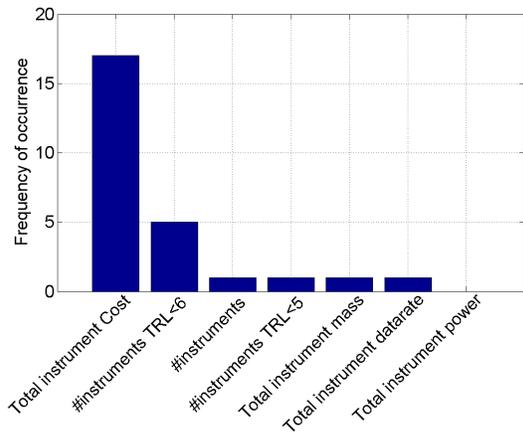
These probabilities can be then used to (1) estimate the degree of confidence for a given classification, and (2) judge the risk of the mission being more expensive than the initial class range selected through expert judgment.

## 6. CONCLUSION

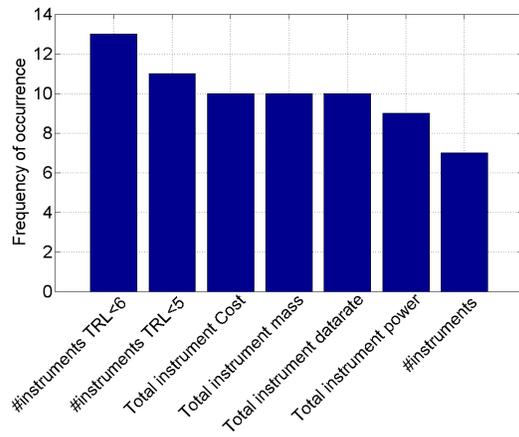
### Summary

This paper has presented a methodology for using classification algorithms in early mission cost estimation. Thus, instead of providing a numerical estimate of the expected cost of the mission for a given percentile (e.g., 50% or 70%), the mission is assigned to a cost class, taken from a predetermined set of cost classes. These classes can, for example, represent cost ranges for existing programs (e.g., Earth Venture, Earth explorer, flagship mission).

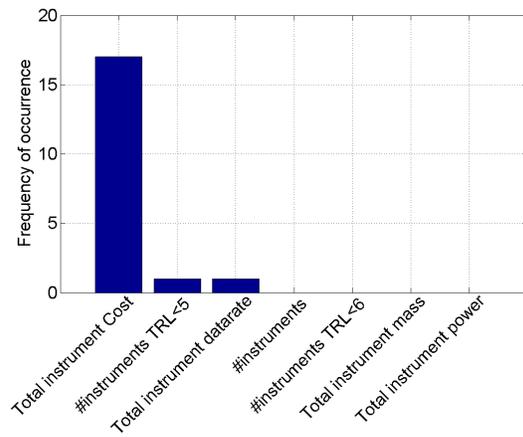
Classification provides less information to the decision maker than point estimates, but the information is more robust and better aligned with the goals of cost estimates during feasibility studies. Indeed, during early mission formulation, program managers are more interested in knowing whether their mission will fit the budgetary constraints of a certain



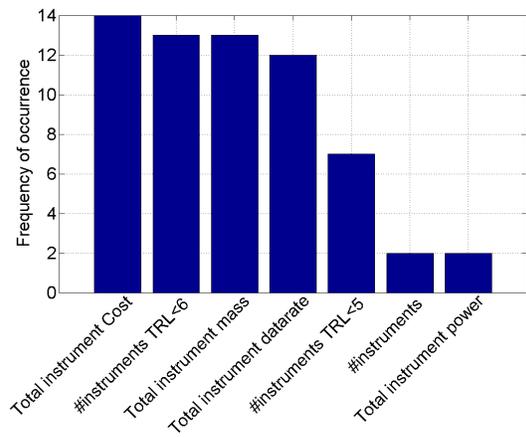
(a) Logistic regression



(b) Additive Tree

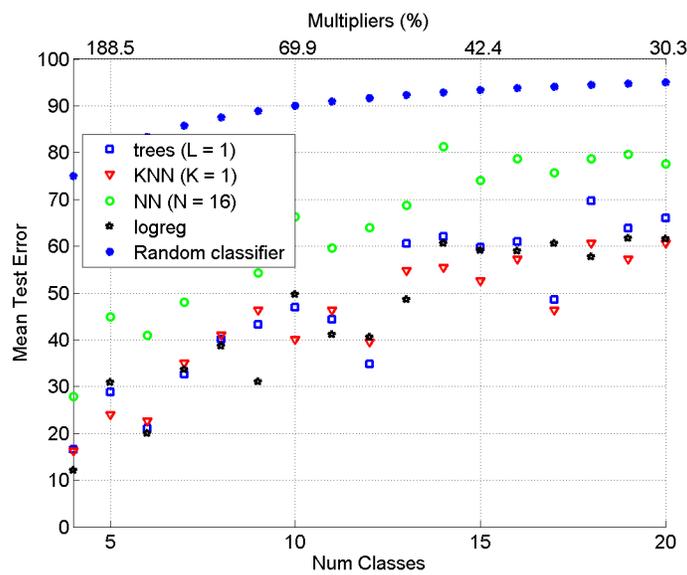


(c) Neural Network



(d) K-Nearest Neighbors

**Figure 4:** Frequency of occurrence of predictor variables for each algorithm



**Figure 5:** Classification performance vs. Number of classes

program than in obtaining a precise cost estimate.

The dataset used for this work contained mission cost as well as 5 instrument-level parameters for 36 Earth observing missions: mass, power, data rate, cost (as predicted by the NASA Instrument Cost Model) and TRL. Subsets of these parameters (and slight variations of them) were used as predictor variables.

It was immediately noted that the problem requires pre-defining a number of cost classes, and that the classification problem reduces to the estimation problem when the number of classes grows to infinity (in other words, the exact cost of a mission is never estimated correctly). It was then shown how defining evenly spaced cost classes in the log-cost space yields more useful classes and better results than defining evenly spaced classes in the linear cost space.

Four classification algorithms were chosen for comparison, namely logistic regression, additive trees, neural networks, and nearest neighbors. Their performance was compared for a number of classes ranging from 4 to 20. The performance benchmarking methodology consisted in applying each algorithm to each classification scheme 100 times for each subset of candidate predictor variables, each time with a different partition of the dataset into training and testing. The metric used for comparison was mean test error, as it was observed that most algorithms were similar in terms of the standard deviation of the test error. For algorithms with parameters, the simulations were run for a range of parameter values and the best performing values were obtained ( $K^* = 1$  for nearest-neighbors,  $L^* = 1$  for additive trees,  $N^* = 16$  for neural networks).

The results showed that for a reasonable number of classes around 6, all algorithms except for neural networks were able to achieve mean test errors around 20%. Errors fall to 10% for 4 classes, and go as high as 60-80% for 20 classes, but consistently improve random estimates. The total instrument cost as predicted by the NASA Instrument Cost Model is the most frequently used predictor variable, followed by the number of instruments with  $TRL < 6$ .

#### *Future Work*

This work has focused on the development of cost classifiers. However, it was noted in Section 1 that the same concept could be applied to classify missions in higher dimensional programmatic categories (e.g., cost-risk). The power of the classification formulation lies on the fact that it does not require to estimate individual parameters separately. Consequently, it can replace tools that estimate the joint cost-schedule-risk probability distribution when the information required to use these tools is highly uncertain.

Multiple areas of improvement can be identified as potential future work. On the one hand, the study has been highly limited by the amount of available data, which has led to some adjustments to the benchmark methodology (no cross-validation). Furthermore, this fact was aggravated for cases where many predictor variables were used, due to the curse of dimensionality. Gathering a larger data set, with a broader variety of space missions, would ensure that the classification patterns that define each mission class are better captured by the classification algorithms, thus potentially yielding better classification performance.

On the other hand, a deeper analysis of *neural networks* and *additive trees* would also be interesting. The authors

recognize that these two algorithms have a higher degree of complexity than the others, and therefore merit a deeper understanding of the tuning process. In particular, multiple parameters such as the type of activation function, number of hidden layers, branching strategy and so forth can be varied in order to achieve optimal classification results. Performing such analyses has not been the focus of this work.

Finally, this study has stated some opportunities of incorporating the outputs of classification algorithms into early cost estimation. In particular, it has been noted that some of the algorithms compute class likelihood ratios that can be used to make risk assessments for cost overruns. It has also been indicated that they can provide insight into the main factors that influence cost in space missions, and provide threshold values that help decision makers have simple rules of thumb to produce cost estimates. Extending this work to formally address these opportunities could be very valuable.

## APPENDIX

Table 5 shows the class boundaries for evenly spaced classes in the logarithmic cost space.

Table 6 shows the details of the optimal combinations of predictor variables for each algorithm and number of classes.

## ACKNOWLEDGMENTS

The authors would like to acknowledge Giancarlo Filipazzo from the European Space Agency for providing the data set for this work.

**Table 5: Class boundaries for evenly spaced classes in log space**

<b>num classes</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>
multiplier	276	189	142	113	94	80	70	62	56
logmultiplier	1.3	1.1	0.9	0.8	0.7	0.6	0.5	0.5	0.4
$c_0$ [\$M]	10	10	10	10	10	10	10	10	10
$c_1$ [\$M]	38	29	24	21	19	18	17	16	16
$c_2$ [\$M]	141	83	58	45	38	32	29	26	24
$c_3$ [\$M]	532	240	141	97	73	58	49	42	38
$c_4$ [\$M]	2000	693	342	206	141	105	83	69	58
$c_5$ [\$M]		2000	827	440	274	190	141	111	91
$c_6$ [\$M]			2000	938	532	342	240	180	141
$c_7$ [\$M]				2000	1031	616	408	291	220
$c_8$ [\$M]					2000	1110	693	471	342
$c_9$ [\$M]						2000	1177	763	532
$c_{10}$ [\$M]							2000	1236	827
$c_{11}$ [\$M]								2000	1286
$c_{12}$ [\$M]									2000

**Table 6:** Optimal combinations of predictor variables

Algorithm	# classes	Optimal set of predictor variables
Logistic Regression	4	[Total NICM instrument cost]
Logistic Regression	5	[Total NICM instrument cost]
Logistic Regression	6	[Total NICM instrument cost]
Logistic Regression	7	[Total NICM instrument cost]
Logistic Regression	8	[Total NICM instrument cost]
Logistic Regression	9	[Total NICM instrument cost]
Additive trees ( $L = 1$ )	4	[Total NICM instrument cost, Total # instruments, Total # instruments with $TRL \leq 6$ , Total instrument power, Total instrument data rate]
Additive trees ( $L = 1$ )	5	[Total # instruments with $TRL \leq 5$ , Total # instruments with $TRL \leq 6$ , Total instrument power, Total instrument data rate]
Additive trees ( $L = 1$ )	6	[Total NICM instrument cost, Total # instruments, Total # instruments with $TRL \leq 5$ , Total # instruments with $TRL \leq 6$ , Total instrument mass, Total instrument power, Total instrument data rate]
Additive trees ( $L = 1$ )	7	[Total NICM instrument cost, Total # instruments, Total # instrument mass, Total instrument power, Total instrument data rate]
Additive trees ( $L = 1$ )	8	[Total NICM instrument cost, Total # instruments with $TRL \leq 5$ , Total # instruments with $TRL \leq 6$ , Total instrument mass, Total instrument power]
Additive trees ( $L = 1$ )	9	[Total NICM instrument cost, Total # instruments, Total # instruments with $TRL \leq 5$ , Total instrument mass, Total instrument data rate]
Neural networks ( $N = 16$ )	4	[Total NICM instrument cost]
Neural networks ( $N = 16$ )	5	[Total NICM instrument cost]
Neural networks ( $N = 16$ )	6	[Total NICM instrument cost]
Neural networks ( $N = 16$ )	7	[Total NICM instrument cost]
Neural networks ( $N = 16$ )	8	[Total NICM instrument cost, Total instrument data rate]
Neural networks ( $N = 16$ )	9	[Total NICM instrument cost]
Nearest neighbors ( $K = 1$ )	4	[Total NICM instrument cost, Total # instruments with $TRL \leq 6$ , Total instrument mass, Total instrument power, Total instrument data rate]
Nearest neighbors ( $K = 1$ )	5	[Total NICM instrument cost, Total # instruments with $TRL \leq 5$ , Total instrument mass]
Nearest neighbors ( $K = 1$ )	6	[Total NICM instrument cost, Total # instruments with $TRL \leq 6$ , Total instrument data rate]
Nearest neighbors ( $K = 1$ )	7	[Total NICM instrument cost, Total # instruments with $TRL \leq 5$ , Total # instruments with $TRL \leq 6$ , Total instrument data rate]
Nearest neighbors ( $K = 1$ )	8	[Total NICM instrument cost, Total # instruments with $TRL \leq 6$ , Total instrument mass]
Nearest neighbors ( $K = 1$ )	9	[Total NICM instrument cost, Total # instruments with $TRL \leq 5$ , Total instrument data rate]

## REFERENCES

- [1] U. G. A. Office, "Cost estimating and assessment guide: Best practices for developing and managing," U.S. Government Accountability Office, Tech. Rep. GAO-09-3SP, March 2009.
- [2] E. C. on Space Standardization, "Space project management," European Cooperation on Space Standardization, Tech. Rep. ECSS-M-60A, April 1996.
- [3] NASA, "Discovery program," 2013. [Online]. Available: <http://discovery.nasa.gov>
- [4] —, "New frontiers program," 2013. [Online]. Available: <http://newfrontiers.nasa.gov>
- [5] —, "Nasa cost estimating handbook," NASA, Tech. Rep., 2008.
- [6] R. Shishko, *Developing analogy cost estimates for space missions*. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2004.
- [7] J. Hamaker and P. Componation, "Improving space project cost estimating with engineering management variables," *EMJ - Engineering Management Journal*, vol. 17, no. 2, pp. 28–33, 2005, cited By (since 1996)4.
- [8] H. Habib-Agahi, J. Mrozinski, and G. Fox, "Nasa instrument cost/schedule model hamid habib-agahi," in *Aerospace Conference, 2011 IEEE*, 2011, pp. 1–19.
- [9] U. G. A. Office, "Nasa: Lack of disciplined cost-estimating processes hinders effective program management," U.S. Government Accountability Office, Tech. Rep. GAO-04-642, May 2004.
- [10] W. J. Larson and J. R. Wertz, "Space mission analysis and design," Torrance, CA (United States); Microcosm, Inc., Tech. Rep., 1992.
- [11] A. E. Smith and A. K. Mason, "Cost estimation predictive modeling: Regression versus neural network," *The Engineering Economist*, vol. 42, no. 2, pp. 137–161, 1997.
- [12] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *Software Engineering, IEEE Transactions on*, vol. 21, no. 2, pp. 126–137, 1995.
- [13] L. Briand, V. Basili, and W. Thomas, "A pattern recognition approach for software engineering data analysis," *Software Engineering, IEEE Transactions on*, vol. 18, no. 11, pp. 931–942, 1992.
- [14] A. Golkar and G. Filipazzo, "Development of a management support framework for space based systems of systems programs," in *AIAA Space Conference*, September 2013.
- [15] R. Hastie, Trevor; Tibshirani and J. J. H. Friedman, *The elements of statistical learning*. Springer New York, 2001.

## BIOGRAPHY



**Marc Sanchez Net** is currently a second year M.S. student in the department of Aeronautics and Astronautics at MIT. His research interests include machine learning algorithms and rule-based expert systems, and their suitability to the fields of system engineering and space communication networks. Prior to his work at MIT, Marc interned at Sener Ingenieria y Sistemas as a part of the team that develops and maintains FORAN, a CAD/CAM/CAE commercial software for shipbuilding. Marc received his degrees in both Industrial engineering and Telecommunications engineering in 2012 from Universitat Politecnica de Catalunya, Barcelona.



**Dr. Daniel Selva** received a PhD in Space Systems from MIT in 2012 and he is currently a post-doctoral associate in the department of Aeronautics and Astronautics at MIT, and an adjunct Assistant Professor in the Sibley School of Mechanical and Aerospace Engineering at Cornell University. His research interests focus on the application of multidisciplinary optimization and artificial intelligence techniques to space systems engineering and architecture. Prior to MIT, Daniel worked for four years in Kourou (French Guiana) as a member of the Ariane 5 Launch team. Daniel has a dual background in electrical engineering and aeronautical engineering, with degrees from Universitat Politecnica de Catalunya in Barcelona, Spain, and Supaero in Toulouse, France. He is a 2007 la Caixa fellow, and received the Nortel Networks prize for academic excellence in 2002.



**Dr. Alessandro Golkar** is Assistant Professor at Skolkovo Institute of Science and Technology (Skoltech) in Moscow, Russian Federation, a private university opened in collaboration with MIT. He received a Ph.D. in Aeronautics and Astronautics from MIT. He is the Director of the Strategic Innovation Research Group (SIRG) at Skoltech. His research interests lie in the areas of systems architecture, project management, systems engineering, and spacecraft design analysis and optimization. Alessandro had research and consulting experience at Caltech/NASA

*Jet Propulsion Laboratory, and at the European Space Agency. Before MIT, Alessandro received a Laurea degree in Aerospace Engineering in 2006 and a Laurea Specialistica degree in Astronautics Engineering in 2008 from University of Rome "La Sapienza", Italy.*